fli4l – flexible internet router for linux
Version 3.10.4

Das fli4l-Team
E-Mail: team@fli4l.de

25. Oktober 2015

Inhaltsverzeichnis

1. Dokumentation des Basispaketes **10**
1.1. Einleitung . . . 10

2. Installation und Konfiguration **13**
2.1. Entpacken der Archive . . . 13
2.2. Konfiguration . . . 14
2.2.1. Editieren der Konfigurationsdateien . . . 14
2.2.2. Konfiguration über eine spezielle Konfigurationsdatei . . . 15
2.2.3. Variablen . . . 15
2.3. Installationsvarianten . . . 15
2.3.1. Router auf einem USB-Stick . . . 16
2.3.2. Router auf einer CD oder Netzwerkboot . . . 16
2.3.3. Typ A: Router auf Festplatte – nur eine FAT-Partition . . . 16
2.3.4. Typ B: Router auf Festplatte – je eine FAT- und ext3-Partition . . . 17

3. Basiskonfiguration **18**
3.1. Beispiel-Datei . . . 19
3.2. Allgemeine Einstellungen . . . 25
3.3. Konsolen-Einstellungen . . . 30
3.4. Hilfen zum Einkreisen von Problemen und Fehlern . . . 31
3.5. Verwendung einer eigenen /etc/inittab . . . 32
3.6. Länderspezifische Tastaturlayouts . . . 33
3.7. Ethernet-Netzwerkkarten-Treiber . . . 33
3.8. Netzwerke . . . 41
3.9. Zusätzliche Routen (optional) . . . 43
3.10. Der Paketfilter . . . 44
3.10.1. Aktionen des Paketfilters . . . 45
3.10.2. Einschränkungen in den Regeln . . . 47
3.10.3. Der Einsatz von Schablonen im Paketfilter . . . 50
3.10.4. Die Konfiguration des Paketfilters . . . 54
3.10.5. Beispiele . . . 61
3.10.6. Standardkonfigurationen . . . 64
3.10.7. DMZ – Demilitarisierte Zone . . . 68
3.10.8. Conntrack-Helfer . . . 69
3.11. Domain-Konfiguration . . . 71
3.12. imond-Konfiguration . . . 72
3.13. Allgemeine Circuit-Konfiguration . . . 75

4. Pakete **76**
4.1. Werkzeuge im Basispaket . . . 76
4.1.1. OPT_SYSLOGD – Protokollieren von Systemmeldungen . . . 76
4.1.2. OPT_KLOGD – Protokollieren von Kernelmeldungen . . . 78
4.1.3. OPT_LOGIP – Protokollieren von WAN-IP-Adressen . . . 78
4.1.4. OPT_Y2K – Datumskorrektur bei nicht Y2K-festen Rechnern . . . 78
4.1.5. OPT_PNP – Installation von isapnp tools . . . 79
4.2. Advanced Networking . . . 81
4.2.1. Broadcast Relay - Weiterleitung von IP Broadcasts . . . 81
4.2.2. Bonding - mehrere Netzwerkkarten zusammenfassen zu einem Link . . . 82
4.2.3. VLAN - 802.1Q Unterstüzung . . . 86
4.2.4. Device MTU - Anpassen der MTU . . . 87
4.2.5. BRIDGE - Ethernet Bridging für fli4l . . . 87
4.2.6. Anmerkungen . . . 91
4.2.7. EBTables - EBTables für fli4l . . . 91
4.2.8. ETHTOOL - Einstellungen für Ethernet-Netzwerkadapter . . . 92
4.2.9. Beispiel . . . 93
4.3. CHRONY - Network Time Protocol Server/Client . . . 94
4.3.1. Konfiguration des OPT_CHRONY . . . 95
4.3.2. Support . . . 96
4.3.3. Literatur . . . 96
4.4. DHCP_CLIENT - Dynamic Host Configuration Protocol . . . 96
4.4.1. OPT_DHCP_CLIENT . . . 96
4.5. DNS_DHCP - DNS- und DHCP-Server sowie DHCP-Relay und Slave DNS Server 98
4.5.1. Hostnamen . . . 98
4.5.2. DNS-Server . . . 99
4.5.3. DHCP-Server . . . 105
4.5.4. DHCP-Relay . . . 108
4.5.5. TFTP-Server . . . 109
4.5.6. YADIFA - Slave DNS Server . . . 109
4.6. DSL - DSL über PPPoE, Fritz!DSL und PPTP . . . 110
4.6.1. Allgemeine Konfigurationsvariablen . . . 111
4.6.2. OPT_PPPOE - DSL über PPPoE . . . 114
4.6.3. OPT_FRITZDSL - DSL per Fritz!Card DSL . . . 116
4.6.4. OPT_PPTP - DSL über PPTP in Österreich/den Niederlanden . . . 117
4.6.5. OPT_POESTATUS - PPPoE-Status-Monitor auf fli4l-Console . . . 119
4.7. DYNDNS - Dynamische Updates für Domain Name Services . . . 119
4.8. EASYCRON - Befehle zeitgesteuert ausführen . . . 125
4.8.1. Konfiguration . . . 125
4.8.2. Beispiele . . . 125
4.8.3. Voraussetzungen . . . 126
4.8.4. Installation . . . 126
4.9. HD - Unterstützung von Festplatten, Flash-Karten, USB-Sticks usw. . . . 126
4.9.1. OPT_HDINSTALL - Installation auf Festplatte/CompactFlash . . . 126
4.9.2. OPT_MOUNT - Automatisches Einhängen von Dateisystemen . . . 129
4.9.3. OPT_EXTMOUNT - Manuelles Einhängen von Dateisystemen . . . 129
4.9.4. OPT_HDSLEEP – automatisches Abschalten für Festplatten einstellen 130

4.9.5. OPT_RECOVER – Notfalloption . . . 130
4.9.6. OPT_HDDRV - Treiber für Festplattencontroller . . . 131
4.10. HTTPD - Status-Webserver . . . 131
4.10.1. OPT_HTTPD - Mini-Webserver als Statusmonitor . . . 131
4.10.2. Nutzerverwaltung . . . 133
4.10.3. OPT_OAC - Online Access Control . . . 134
4.11. HWSUPP - Unterstützung von Hardware . . . 135
4.11.1. Beschreibung . . . 135
4.11.2. Konfiguration des Paketes HWSUPP . . . 136
4.11.3. Experten-Einstellungen . . . 140
4.11.4. Unterstützung von VPN-Karten . . . 141
4.12. IPv6 - Internet Protokoll Version 6 . . . 142
4.12.1. Einleitung . . . 142
4.12.2. Adressformat . . . 142
4.12.3. Konfiguration . . . 143
4.12.4. Web-GUI . . . 154
4.13. ISDN - Kommunikation über aktive und passive ISDN-Karten . . . 155
4.13.1. Herstellen einer ISDN-Verbindung . . . 155
4.13.2. ISDN-Karte . . . 156
4.13.3. OPT_ISDN_COMP (EXPERIMENTAL) . . . 160
4.13.4. ISDN-Circuits . . . 160
4.13.5. OPT_TELMOND - telmond-Konfiguration . . . 169
4.13.6. OPT_RCAPID - Remote CAPI Dämon . . . 172
4.14. OpenVPN - VPN-Support . . . 173
4.14.1. OpenVPN - Einführendes Beispiel . . . 174
4.14.2. OpenVPN - Konfiguration . . . 175
4.14.3. OpenVPN - Bridgekonfiguration . . . 178
4.14.4. OpenVPN - Tunnelkonfiguration . . . 179
4.14.5. Experteneinstellungen . . . 182
4.14.6. OpenVPN - WebGUI . . . 191
4.14.7. OpenVPN - Zusammenarbeit unterschiedlicher OpenVPN Versionen . . . 194
4.14.8. OpenVPN - Beispiele . . . 195
4.14.9. Weiterführende Links zum Thema OpenVPN . . . 199
4.15. PCMCIA - PC-Card Unterstützung . . . 199
4.15.1. PCMCIA-Treiber . . . 199
4.16. PPP - Anbindung eines Rechners über serielle Schnittstelle . . . 200
4.17. PROXY - Verschiedene Proxy-Server . . . 202
4.17.1. OPT_PRIVOXY - Ein Werbung-filternder HTTP-Proxy . . . 202
4.17.2. OPT_TOR - Ein anonymes Kommunikationssystem für das Internet . . . 204
4.17.3. OPT_SS5 - Ein Socks4/5 Proxy . . . 206
4.17.4. OPT_TRANSPROXY (EXPERIMENTELL) - Transparenter HTTP-Proxy . . . 206
4.17.5. OPT_SIPPROXY (EXPERIMENTELL) - Proxy für Session Initiation Protocol . . . 207
4.17.6. OPT_IGMPPROXY - Proxy für Internet Group Management Protocol 207
4.17.7. OPT_STUNNEL - Tunneln von Verbindungen über SSL/TLS . . . 215

4.18. QoS - Quality of Service . . . 220
4.18.1. Konfiguration . . . 221
4.18.2. Anwendungsbeispiele . . . 229
4.19. SSHD - Secure Shell, Secure Copy . . . 236
4.19.1. Installation des Secure-Shell-Dienstes . . . 236
4.19.2. Installation des dbclients . . . 240
4.19.3. Installation des plink Clients . . . 240
4.19.4. Installation des sftp-server . . . 241
4.19.5. Literatur . . . 241
4.20. TOOLS - Zusätzliche Werkzeuge zum Debugging . . . 241
4.20.1. Netzwerk-Tools . . . 241
4.20.2. Die Hardware-Erkennung . . . 244
4.20.3. Dateien-Tools . . . 245
4.20.4. Entwickler-Tools . . . 246
4.21. UMTS - Anbindung mittels UMTS an das Internet . . . 246
4.21.1. Konfiguration . . . 246
4.21.2. Beispielkonfiguration für RRDTOOL . . . 249
4.22. USB - Support für USB-Geräte . . . 249
4.22.1. Probleme mit USB-Geräten . . . 251
4.22.2. Hinweise zur Benutzung . . . 251
4.22.3. Mounten von USB-Geräten . . . 251
4.23. WLAN - Wireless-LAN Unterstützung . . . 252
4.23.1. WLAN-Konfiguration . . . 252
4.23.2. Beispiele . . . 256
4.23.3. Virtual Accesspoint (VAP)(Experimentell) . . . 258
4.23.4. Zeitgesteuertes ein– und ausschalten mit easycron . . . 258
4.23.5. Spendenhinweis . . . 258
4.24. SRC - Das fli4l-Buildroot . . . 259
4.24.1. Eine Übersicht über die Quellen . . . 259
4.24.2. Übersetzen eines Programms für den fli4l . . . 260
4.24.3. Testen eines übersetzten Programms . . . 263
4.24.4. Entwanzen eines übersetzten Programms . . . 264
4.24.5. Informationen über das FBR . . . 267
4.24.6. Ändern der FBR-Konfiguration . . . 268
4.24.7. Aktualisierung des FBRs . . . 269
4.24.8. Eigene Programme ins FBR einbinden . . . 269

5. Erzeugen der fli4l Archive/Bootmedien 270
5.1. Erzeugen der fli4l Archive/Bootmedien unter Linux bzw. anderen Unix-Derivaten und Mac OS X . . . 270
5.1.1. Kommandozeilenoptionen . . . 271
5.2. Erzeugen der fli4l Archive/Bootmedien unter Windows . . . 273
5.2.1. Kommandozeilenoptionen . . . 273
5.2.2. Konfigurationsdialog – Einstellung des Konfigurationsverzeichnis . . . 274
5.2.3. Konfigurationsdialog – allgemeine Einstellungen . . . 275
5.2.4. Konfigurationsdialog – Einstellungen für Remoteupdate . . . 276
5.2.5. Konfigurationsdialog – Einstellungen für HD-pre-install . . . 277

5.3. Steuerungsdatei mkfli4l.txt . . . 278

6. Anbindung von PCs im LAN 280
6.1. IP-Adresse . . . 280
6.2. Rechnername und Domain . . . 280
6.2.1. Windows 2000 . . . 280
6.2.2. NT 4.0 . . . 281
6.2.3. Win95/98 . . . 281
6.2.4. Windows XP . . . 281
6.2.5. Windows 7 . . . 282
6.2.6. Windows 8 . . . 282
6.3. Gateway . . . 282
6.4. DNS-Server . . . 283
6.5. Verschiedenes . . . 283

7. Client-/Server-Schnittstelle imon 284
7.1. imon-Server imond . . . 284
7.1.1. Least-Cost-Routing – Funktionsweise . . . 284
7.1.2. Zur Berechnung der Onlinekosten . . . 289
7.2. Windows-Client imonc.exe . . . 289
7.2.1. Einleitung . . . 289
7.2.2. Startparameter . . . 290
7.2.3. Seite Überblick . . . 292
7.2.4. Config-Dialog . . . 293
7.2.5. Seite Anrufe . . . 299
7.2.6. Seite Verbindungen . . . 299
7.2.7. Seite Fax . . . 300
7.2.8. Seite E-Mail . . . 300
7.2.9. Admin . . . 301
7.2.10. Seiten Fehler, Syslog und Firewall . . . 302
7.2.11. Seite News . . . 302
7.3. Unix/Linux-Client imonc . . . 302

8. Entwickler-Dokumentation 305
8.1. Allgemeine Regeln . . . 305
8.2. Übersetzen von Programmen . . . 306
8.3. Modulkonzept . . . 306
8.3.1. mkfli4l . . . 306
8.3.2. Aufbau . . . 306
8.3.3. Die Konfiguration der Pakete . . . 308
8.3.4. Die Liste der zu kopierenden Dateien . . . 308
8.3.5. Die Prüfung der Konfiguration-Variablen . . . 313
8.3.6. Eigene Definitionen zum Prüfen der Konfigurationsvariablen . . . 315
8.3.7. Erweiterte Prüfungen der Konfiguration . . . 321
8.3.8. Unterstützung verschiedener Kernelversionslinien . . . 336
8.3.9. Dokumentation . . . 337
8.3.10. Dateiformate . . . 339

8.3.11. Entwickler-Dokumentation . . . 339
8.3.12. Client-Programme . . . 339
8.3.13. Quellcode . . . 339
8.3.14. Weitere Dateien . . . 339
8.4. Allgemeine Skript-Erstellung auf fli4l . . . 340
8.4.1. Aufbau . . . 340
8.4.2. Umgang mit Konfigurationsvariablen . . . 341
8.4.3. Persistente Speicherung von Daten . . . 341
8.4.4. Fehlersuche . . . 342
8.4.5. Hinweise . . . 343
8.5. Arbeit mit dem Paketfilter . . . 344
8.5.1. Hinzufügen von eigenen Ketten und Regeln . . . 344
8.5.2. Einordnen in bestehende Regeln . . . 345
8.5.3. Erweiterung der Paketfilter-Tests . . . 346
8.6. CGI-Erstellung für das *httpd*-Paket . . . 346
8.6.1. Allgemeines zum Webserver . . . 346
8.6.2. Skriptnamen . . . 347
8.6.3. Menü-Einträge . . . 347
8.6.4. Aufbau eines CGI-Skriptes . . . 348
8.6.5. Sonstiges . . . 353
8.6.6. Fehlersuche . . . 353
8.7. Hochfahren, Herunterfahren, Einwählen und Auflegen unter fli4l . . . 354
8.7.1. Bootkonzept . . . 354
8.7.2. Start- und Stopp-Skripte . . . 354
8.7.3. Hilfsfunktionen . . . 357
8.7.4. ttyI-Geräte . . . 359
8.7.5. Skripte beim Einwählen und Auflegen . . . 360
8.8. Paket „template“ . . . 361
8.9. Aufbau des Boot-Datenträgers . . . 361
8.10. Konfigurationsdateien . . . 362
8.10.1. Konfiguration Provider . . . 362
8.10.2. Konfiguration DNS . . . 363
8.10.3. Hosts-Datei . . . 363
8.10.4. imond-Konfiguration . . . 363
8.10.5. Die `/etc/.profile`-Datei . . . 364

A. Anhang zum Basispaket **365**
A.1. Nullmodemkabel . . . 365
A.2. Serielle Console . . . 365
A.3. Programme . . . 366
A.4. Andere i4l-Tools . . . 366
A.5. Fehlersuche . . . 366
A.6. Literaturhinweise . . . 367
A.7. Präfixe . . . 368
A.8. Gewähr und Haftung . . . 368
A.9. Danke . . . 368
A.9.1. Projektgründung . . . 368

A.9.2. Entwickler- und Testteam . . . 368
A.9.3. Entwickler- und Testteam (nicht mehr aktive) . . . 370
A.9.4. Sponsoren . . . 370
A.10. Feedback . . . 372

B. Anhänge der optionalen Pakete **373**
B.1. CHRONY - Benachrichtigung anderer Applikationen über Timewarps . . . 373
B.2. DSL - PPPD und Active Filter . . . 373
B.3. DYNDNS . . . 374
B.3.1. Hinzufügen von neuen Providern . . . 374
B.3.2. Dank . . . 376
B.3.3. Lizenz . . . 377
B.4. EASYCRON - Crontab in der Boot-Phase ergänzen . . . 378
B.5. HD - Fehler im Zusammenhang mit Festplatten/CompactFlashs . . . 379
B.6. HTTPD . . . 380
B.6.1. Zusätzliche Einstellungen . . . 380
B.6.2. Allgemeine Bemerkungen . . . 380
B.7. HWSUPP - Geräteabhängige Einstellungen . . . 381
B.7.1. Verfügbare LED-Devices . . . 381
B.7.2. Verfügbare Button-Devices . . . 382
B.7.3. Hinweise zu spezieller Hardware . . . 382
B.8. HWSUPP - Konfigurations-Beispiele . . . 383
B.8.1. generic-pc . . . 383
B.8.2. pcengines-apu . . . 383
B.8.3. pcengines-apu mit GPIO's . . . 383
B.9. HWSUPP - Blinkfolge . . . 384
B.10. HWSUPP - Hinweise für Paket-Entwickler . . . 385
B.10.1. LED-Erweiterungen . . . 385
B.10.2. Button-Erweiterungen . . . 386
B.10.3. Button-Aktion . . . 386
B.11. IPV6 - Anbindung ans IPv6-Internet mit Hilfe eines SixXS-Tunnels . . . 387
B.11.1. Account erstellen . . . 387
B.11.2. Tunnel konfigurieren . . . 387
B.11.3. Subnetz konfigurieren . . . 389
B.12. ISDN . . . 393
B.12.1. Technische Details zu Einwahl und Routing bei ISDN . . . 393
B.12.2. Fehlermeldungen des ISDN-Subsystems (englisch, i4l-Dokumentation) . . . 395
B.13. UMTS . . . 396
B.13.1. Unterstützte Hardware . . . 396
B.13.2. Modemschnittstelle nicht aktiviert . . . 397
B.14. Unterschiede Version 3.10.4 und 3.6.2 . . . 398
B.15. Unterschiede Version 3.10.4 und 3.10.3 . . . 403

Abbildungsverzeichnis **404**

Tabellenverzeichnis **405**

Index **406**

1. Dokumentation des Basispaketes

1.1. Einleitung

fli4l ist ein auf Linux basierender ISDN-, DSL-, UMTS- und Ethernet-Router mit geringen Anforderungen an die zugrunde liegende Hardware: Ein USB-Stick als Bootmedium, ein Intel Pentium MMX-Prozessor, 64 MiB RAM sowie (mindestens) eine Ethernet-Netzwerkkarte sind dafür vollkommen ausreichend. Das notwendige Bootmedium kann unter Linux, Mac OS X oder MS Windows erstellt werden. Dabei sind keine Linux-Kenntnisse erforderlich, aber durchaus hilfreich. Grundkenntnisse von Netzwerken, TCP/IP, DNS und Routing sollten jedoch vorhanden sein. Für eigene Erweiterungen/Entwicklungen, welche über die Standardkonfiguration hinausgehen, sind ein lauffähiges Linux-System und Linux-Kenntnisse notwendig.

fli4l unterstützt verschiedene Bootmedien, darunter USB-Sticks, Festplatten, CDs und nicht zuletzt das Booten über das Netzwerk. Ein USB-Stick ist in vielerlei Hinsicht ideal: Heutzutage kann so gut wie jeder PC von einem USB-Stick starten, er ist recht erschwinglich, er hat eine ausreichende Größe, und man kann sowohl unter Linux als auch unter MS Windows auf relativ einfache Weise eine fli4l-Installation darauf ablegen. Auch ist er im Gegensatz zu einer CD beschreibbar und kann somit nichtflüchtige Konfigurationsdaten (wie z.B. DHCP-Leases) speichern.

- Allgemeine Features
  - Erstellen von Bootmedien unter Linux (Seite 270), Mac OS X (Seite 270) und MS Windows (Seite 273)
  - Konfiguration über normale ASCII/UTF-8-Dateien
  - Unterstützung von IP-Masquerading und Portweiterleitung
  - Least-Cost-Routing (LCR): automatische Auswahl des Providers, je nach Uhrzeit
  - Anzeige/Berechnung/Protokollierung von Verbindungszeiten und -kosten
  - MS Windows/Linux-Client imonc mit Schnittstelle zu imond und telmond
  - Upload von aktualisierten Konfigurationsdateien über MS Windows-Client imonc oder via SCP unter Linux
  - Bootmedien nutzen das VFAT-Dateisystem zum dauerhaften Speichern von Dateien
  - Paketfilter: Protokollieren von Zugriffen von außen auf gesperrte Ports
  - Einheitliche Abbildung von WAN-Schnittstellen auf sogenannte Circuits
  - Paralleler Betrieb von ISDN- und DSL/UMTS-Circuits ist möglich
- Router-Basisfunktionalität
  - Linux-Kernel 3.14
  - Paketfilter und IP-Masquerading

  - DNS-Server, um die Anzahl von DNS-Abfragen an externe DNS-Server zu reduzieren
  - Netzwerkfähiger imond-Server mit Monitor- und LCR-Steuerfunktionen
  - Netzwerkfähiger telmond-Server zur Protokollierung von eingehenden Telefonanrufen

- Ethernet-Unterstützung
  - Aktuelle Netzwerkkartentreiber: Unterstützung von über 140 Kartentypen

- DSL-Unterstützung
  - Roaring Penguin PPPoE-Treiber, mit Dial-on-Demand (abschaltbar)
  - PPTP für DSL-Anbindungen in Österreich und den Niederlanden

- ISDN-Unterstützung
  - Unterstützung von knapp 60 ISDN-Kartentypen
  - Mehrere ISDN-Verbindungsmöglichkeiten: eingehend/ausgehend/Rückruf, „roh“/Punkt-zu-Punkt (ppp)
  - Kanalbündelung: automatische Bandbreitenanpassung oder manuelle Zuschaltung des zweiten Kanals über MS Windows-/Linux-Client

- Optionale Programmpakete
  - DNS-Server
  - DHCP-Server
  - SSH-Server
  - Einfache Online/Offline-Anzeige über LED
  - Serielle Konsole
  - Mini-Webserver für ISDN- und DSL-Monitoring sowie zur Rekonfiguration und/oder Aktualisierung des Routers
  - Zugangserlaubnis für bestimmte konfigurierte Netzwerke von außen
  - Unterstützung für PCMCIA-Karten (heutzutage PC-Cards genannt)
  - Protokollierung von Systemmeldungen
  - Konfiguration von ISAPnP-Karten mit den isapnp-Werkzeugen
  - Zusätzliche Werkzeuge zum Debugging
  - Konfiguration der seriellen Schnittstelle
  - Notfallsystem zur Fernwartung über ISDN
  - Software zur Anzeige konfigurierbarer Informationen auf einem LCD, z.B. von Übertragungsraten, CPU-Auslastung etc.
  - PPP-Server/Router über serielle Schnittstelle
  - ISDN-Modem-Emulator über serielle Schnittstelle

  - Druckerserver
  - Synchronisierung der Uhrzeit mit externen Zeit-Servers
  - Ausführen von benutzerdefinierten Kommandos bei eingehenden Telefonanrufen (z.B. um ein Internet-Einwahl durchzuführen)
  - Unterstützung von IP-Aliasing (mehrere IP-Adressen pro Netzwerkschnittstelle)
  - VPN-Unterstützung
  - IPv6-Unterstützung
  - WLAN-Unterstützung: fli4l kann sowohl Zugangsknoten als auch Client sein
  - RRD-Tool zum Überwachen des fli4l
  - und vieles andere mehr...

- Hardwarevoraussetzungen
  - Intel Pentium-Prozessor mit MMX Unterstützung
  - 64 MiB Speicher, besser 128 MiB
  - Ethernet-Netzwerkkarte
  - ISDN: unterstützte ISDN-Karte
  - ein USB-Stick, eine ATA-Festplatte oder eine CF-Karte (die genauso wie eine ATA-Festplatte angesprochen wird); alternativ ist auch der Start von CD möglich

- Softwarevoraussetzungen

  Unter Linux werden folgende Programme vorausgesetzt:
  - GCC und GNU make
  - syslinux
  - mtools (mcopy)

  Unter MS Windows werden keine zusätzlichen Werkzeuge benötigt, fli4l bringt alles Notwendige mit.

Zusätzlich gibt es zur Steuerung/Statusanzeige des fli4l-Routers noch den Client imonc. Dieses Programm ist für MS Windows (windows/imonc.exe) und auch für Linux (unix/gtk-imonc) vorhanden.
Und nun ...

Viel Spaß mit fli4l!

Frank Meyer und das fli4l-Team
E-Mail: team@fli4l.de

2. Installation und Konfiguration

2.1. Entpacken der Archive

Unter Linux:

```
tar xvfz fli4l-3.10.4.tar.gz
```

Funktioniert dies nicht, geht's auch so:

```
gzip -d < fli4l-3.10.4.tar.gz | tar xvf -
```

Wer die aktuelle Version in einem bereits existierenden fli4l-Verzeichnis installiert, sollte anschließend `mkfli4l.sh -c` aufrufen, also:

```
cd fli4l-3.10.4
sh mkfli4l.sh -c
```

Es wird jedoch empfohlen, ein neues Verzeichnis für eine neue Version zu benutzen – die Konfiguration kann durch ein entsprechendes Werkzeug zum Dateivergleich sehr einfach übernommen werden.

Unter MS Windows kann das komprimierte Tar-Archiv zum Beispiel mit WinZip extrahiert werden. Dabei ist jedoch zu beachten, dass die Dateien *mit* Unterverzeichnissen (Einstellung in WinZip überprüfen!) ausgepackt werden. Außerdem ist in *Optionen* ⇒*Konfiguration* die so genannte „Smart TAR CR conversion" abzuschalten. Ist diese eingeschaltet, werden einige wichtige Dateien von WinZip falsch extrahiert.

Alternativ ist das OpenSource-Programm 7-Zip (http://www.7-zip.org/) sehr zu empfehlen, welches ebenso mächtig wie WinZip ist.

Es werden folgende Dateien im Unterverzeichnis `fli4l-3.10.4/` installiert:

- Dokumentation:
  - doc/deutsch/* Deutsche Dokumentation
  - doc/english/* Englische Dokumentation
  - doc/french/* Französische Dokumentation
- Konfiguration:
  - config/*.txt Konfigurationsdateien, diese müssen bearbeitet werden
- Skripte/Prozeduren:
  - mkfli4l.sh Boot-Medium oder Dateien erzeugen: Linux/Unix-Version
  - mkfli4l.bat Boot-Medium erzeugen: Windows-Version

- Kernel/Boot-Dateien:
  - img/kernel Linux-Kernel
  - img/boot*.msg Bootscreen Texte
- Zusatzpakete:
  - opt/*.txt Diese Dateien beschreiben, was bei welchen Einstellungen in das Archiv opt.img gelangt.
  - opt/etc/* Standard-Konfigurationsdateien für viele Programme (müssen normalerweise nicht bearbeitet werden).
  - opt/files/* Optionale Kernel-Module, Dateien und Programme
- Quellcode:
  - src/* Quellcode/Werkzeuge für Linux, siehe src/README
- Programme:
  - unix/mkfli4l* Erzeugen des Bootmediums: Unix/Linux-Version
  - windows/* Erzeugen des Bootmediums: Windows-Version
  - unix/imonc* imond-Client für Unix/Linux
  - windows/imonc/* imond-Client für Windows

2.2. Konfiguration

2.2.1. Editieren der Konfigurationsdateien

Zur Konfiguration von fli4l müssen lediglich die Dateien config/*.txt angepasst werden. Um im Nachhinein die eigenen Konfiguration mit der ausgelieferten vergleichen zu können oder um mehrere Konfigurationen verwalten zu können, empfiehlt es sich, eine Kopie des config-Verzeichnisses anzulegen und die Konfiguration in dieser Kopie durchzuführen. Ein Vergleich der Konfigurationen ist dann durch Verwendung eines geeigneten Werkzeugs (z.B. “diff” unter *nix) relativ einfach möglich. Nehmen wir einmal an, die eigene config liegt in einem Verzeichnis mit Namen “meine_config” ebenfalls im fli4l-Verzeichnis dann wäre der Aufruf wie folgt:

```
~/src/fli4l> diff -u {config,meine_config}/build/full_rc.cfg | grep '^[+-]'
--- config/build/full_rc.cfg    2014-02-18 15:34:39.085103706 +0100
+++ meine_config/build/full_rc.cfg        2014-02-18 15:34:31.094317441 +0100
-PASSWORD='/P6h4iOIN5Bbc'
+PASSWORD='3P8F3KbjYgzUc'
-NET_DRV_1='ne2k-pci'
+NET_DRV_1='pcnet32'
-START_IMOND='no'
+START_IMOND='yes'
-OPT_PPPOE='no'
+OPT_PPPOE='yes'
-PPPOE_USER='anonymer'
-PPPOE_PASS='surfer'
+PPPOE_USER='ich'
+PPPOE_PASS='mein-passwd'
-OPT_SSHD='no'
+OPT_SSHD='yes'
```

Man sieht hier auch sehr schön, dass ein einfacher DSL-Router mit wenigen Handgriffen konfiguriert ist, auch wenn einen die Konfigurationsdateien auf den ersten Blick mit ihrer Fülle von Einstellungsmöglichkeiten erschlagen.

2.2.2. Konfiguration über eine spezielle Konfigurationsdatei

Da sich die Konfiguration durch das Modul-Konzept auf verschiedene Dateien verteilt, und das Bearbeiten dadurch unter Umständen etwas mühsam wird, kann man die Konfiguration auch in einer einzelnen Datei namens *<config verzeichnis>/_fli4l.txt* ablegen, deren Inhalt dann zusätzlich zu den normalen Konfigurationsdateien eingelesen wird und deren Inhalt dominiert. Um beim obigen Beispiel zu bleiben: Um einen einfachen DSL-Router zu konfigurieren, könnten wir einfach folgendes in diese Datei schreiben:

```
PASSWORD='3P8F3KbjYgzUc'
NET_DRV_N='1'
NET_DRV_1='pcnet32'
START_IMOND='yes'
OPT_PPPOE='yes'
PPPOE_USER='ich'
PPPOE_PASS='mein-passwd'
OPT_SSHD='yes'
```

Man sollte vermeiden, beide Konfigurationsvarianten zu mischen.

2.2.3. Variablen

Sie werden merken, dass einige Variablen auskommentiert sind. Wenn das der Fall ist, erhält sie eine sinnvolle Standard-Belegung. Diese Standard-Belegung ist für jede Variable dokumentiert. Wünschen Sie einen anderen Wert für diese Variable, sollten Sie das Kommentarzeichen am Anfang der Variablendefinition ('#') entfernen und den entsprechenden Wert zwischen den Hochkommata einfügen.

2.3. Installationsvarianten

In den vorhergehenden Versionen von fli4l wurde lediglich das Booten von einer Diskette unterstützt. Dies ist aus oben genannten Gründen nun nicht mehr möglich, aber die Alternative mittels eines USB-Sticks ist gegeben.

Es sind auch eine Vielzahl anderer Bootmedien (CD, HD, Netzwerk, Compact-Flash, DoC, ...) möglich und fli4l kann auch auf diversen Medien installiert (HD, Compact-Flash, DoC) werden. Dazu kann fli4l auf drei verschiedenen Wegen gebootet werden:

Single Image Der Bootloader lädt den Linux-Kern und dann fli4l als ein einziges Image — danach kann fli4l ohne weiteren Zugriff auf andere Medien booten. Beispiele dafür sind die Boottypen *integrated*, *attached*, *netboot* und *cd*.

Split Image Der Bootloader lädt den Linux-Kern und dann ein rudimentäres fli4l-Image, dass die Bootmedien einbindet und die Konfiguration und restlichen Dateien aus einem dort liegenden Archiv holt. Beispiele dafür sind diese Boottypen: *hd (Typ A)*, *ls120*, *attached* und *cd-emul*.

Installation auf einem Medium Der Bootloader lädt den Linux-Kern und dann ein rudimentäres fli4l-Image, das eine bereits vorhandene fli4l-Installation in sein Dateisystem einbindet und damit keine weiteren Archive auspacken muss. Eine HD-Installation vom Typ B ist ein Beispiel dafür.

Man sollte jedoch zunächst erst einmal fli4l in einer minimalen Version installieren und damit Erfahrungen sammeln. Möchte man später fli4l zusätzlich als Anrufbeantworter und als HTTP-Proxy einsetzen, so hat man vorher schon mal Erfahrungen mit einem grundsätzlich laufenden Router.

Für die Installation ergeben sich daraus die folgenden fünf Varianten:

USB-Stick Router auf einem USB-Stick

CD-router Router auf einer CD

Netzwerk Netzwerkboot

HD-Installation Typ A Router auf Festplatte, CF, DoC – nur eine FAT-Partition

HD-Installation Typ B Router auf Festplatte, CF, DoC – je eine FAT- und ext3-Partition

2.3.1. Router auf einem USB-Stick

USB-Sticks werden von Linux als Festplatten angesprochen, daher gelten hier die Ausführungen zur Festplatteninstallation entsprechend. Bitte beachten Sie, dass mittels des `OPT_USB` die entspechenden Treiber geladen werden müssen, damit der Stick mittels `OPT_HDINSTALL` eingebunden werden kann.

2.3.2. Router auf einer CD oder Netzwerkboot

Alle benötigten Dateien liegen auf dem Bootmedium und werden beim Booten in eine dynamische RAM-Disk entpackt. In einer Minimalkonfiguration ist damit ein Betrieb des Routers mit nur 64 MiB RAM möglich. Die maximale Konfiguration wird nur durch die Kapazität des Bootmediums und des Hauptspeichers limitiert.

2.3.3. Typ A: Router auf Festplatte – nur eine FAT-Partition

Dies entspricht der CD-version, nur dass die Dateien hierbei auf einer Festplatte liegen, wobei der Begriff „Festplatte“ hier auch Compact-Flash-Medien ab 8 MiB und andere Geräte, welche Linux als Festplatte ansprechen kann, mit einschließt. Seit fli4l 2.1.4 können auch DiskOnChip Flash-Speicher von M-Sys oder SCSI-Festplatten benutzt werden.

Die Beschränkung des Archivs opt.img durch die Diskettenkapazität wird aufgehoben, aber alle diese Dateien müssen in einer RAM-Disk mit der entsprechenden Größe beim Boot installiert werden. Dies erhöht den RAM-Bedarf beim Einsatz vieler Pakete.

Für ein Update der Softwarepakete (d.h. des Archivs opt.img und der rc.cfg über das Netzwerk) muss die FAT-Partition genügend Platz für den Kernel, das RootFS und die DOPPELTE Größe des opt.img haben! Falls auch die Notfall-Option genutzt werden soll, erhöht sich der Platzbedarf noch einmal um die Größe des opt.img.

2.3.4. Typ B: Router auf Festplatte – je eine FAT- und ext3-Partition

Im Gegensatz zum Typ A werden hier nicht alle Dateien in die Ramdisk gepackt, sondern bei dem erstmaligen Start nach der Installation oder nach einem Update aus dem Archiv opt.img direkt auf eine ext3-Partition kopiert und im späteren Betrieb von dort geladen. Bei dieser Version ist der Speicherbedarf für die RAM-Disk am geringsten und damit meist auch ein Betrieb mit sehr wenig RAM möglich.

Weitere Informationen zur Installation auf Festplatten finden Sie in der Dokumentation des Pakets HD (separat herunterzuladen)- beginnend bei der Beschreibung der Variablen `OPT_HDINSTALL`.

3. Basiskonfiguration

Ab Version 2.0 ist die fli4l-Distribution modular aufgebaut und in mehrere Pakete aufgeteilt, die extra heruntergeladen werden müssen. Im Paket `fli4l-3.10.4.tar.gz` ist lediglich die Basis-Software für einen Ethernet-Router enthalten. Für DSL, ISDN und weitere Software müssen die Pakete separat heruntergeladen werden und ausgehend vom Verzeichnis `fli4l-3.10.4/` (!) installiert werden. Durch die Auswahlmöglichkeit des Betriebssystemkerns von fli4l sind diese in die Kernel Pakete ausgelagert worden. Somit ist als Minimum Basis und ein Kernel Paket erforderlich. In Tabelle 3.1 finden Sie einen Überblick über die Zusatzpakete.

Die zur Konfiguration des fli4l-Routers verwendeten Dateien befinden sich im Verzeichnis `config/` und werden hier im Folgenden beschrieben.

Diese Dateien können mit einem *einfachen* Text-Editor oder auch mit einem speziell an fli4l angepassten Editor verändert werden. Diverse Editoren sind unter
`http://www.fli4l.de/download/zusatzpakete/addons/` zu finden.

Sind spezielle Anpassungen/Erweiterungen erforderlich, die über die unten aufgeführten Einstellungsmöglichkeiten hinausgehen, benötigt man ein lauffähiges Linux-System, um Anpassungen im RootFS vorzunehmen. In diesem Fall hilft `src/README` weiter.

Tabelle 3.1.: Übersicht über die (Zusatz-)Pakete

Download-Archiv	Paket
fli4l-3.10.4	BASIS, erforderlich!
kernel_3_14	Kernel 3.14, empfohlen
kernel_3_14_virt	Kernel 3.14-virt, alternativ, für den Einsatz in Virtualisierungsumgebungen
kernel_3_14_nonfree	Kernel 3.14, mit Unterstützung von Nicht-GPL-Treibern
kernel_3_14_virt_nonfree	Kernel 3.14-virt-nonfree, alternativ, mit Unterstützung von Nicht-GPL-Treib
fli4l-3.10.4-doc	Komplette Dokumentation
advanced_networking	Erweiterte Netzwerkkonfiguration
chrony	Time-Server/Client
dhcp_client	Verschiedene DHCP-Clients
dns_dhcp	DNS- und DHCP-Server
dsl	DSL-Router (PPPoE, PPTP)
dyndns	Unterstützung von DYNDNS-Diensten
easycron	Zeitplandienst
hd	Installation auf Festplatte
httpd	Mini-Webserver für Status-Ausgaben
imonc_windows	Der Windows-Imonc
imonc_unix	Der GTK-Unix-Imonc
ipv6	Internet Protokoll Version 6
isdn	ISDN-Router
lcd	LCD-Treiber-Software + Statusanzeige
lpdsrv	Druckerserver (ohne Spooling)
openvpn	VPN-Unterstützung
pcmcia	Unterstützung von PCMCIA-Karten
ppp	PPP-Anbindung über serielle Schnittstelle
proxy	Proxy-Server
qos	Quality of Service
sshd	SSH-Server
tools	Diverse Linux-Werkzeuge
umts	Anbindung mittels UMTS an das Internet
usb	Unterstützung der USB-Schnittstelle
wlan	Unterstützung von WLAN-Karten

3.1. Beispiel-Datei

Die Beispiel-Datei `base.txt` im Verzeichnis `config/` hat folgenden Inhalt:

```
##-----------------------------------------------------------------------------
## fli4l __FLI4LVER__ - configuration for package "base"
##
##  P L E A S E  R E A D  T H E  D O C U M E N T A T I O N !
##
##  B I T T E  U N B E D I N G T  D I E  D O K U M E N T A T I O N  L E S E N !
```

```
##
##----------------------------------------------------------------------------
## Creation:     26.06.2001  fm
## Last Update:  $Id: base.txt 42028 2015-10-19 05:32:49Z florian $
##
## Copyright (c) 2001-2014 - Frank Meyer, fli4l-Team <team@fli4l.de>
##
## This program is free software; you can redistribute it and/or modify
## it under the terms of the GNU General Public License as published by
## the Free Software Foundation; either version 2 of the License, or
## (at your option) any later version.
##----------------------------------------------------------------------------

#----------------------------------------------------------------------------
# General settings:
#----------------------------------------------------------------------------
HOSTNAME='fli4l'                # name of fli4l router
PASSWORD='fli4l'                # password for root login (console, sshd,
                                # imond)
BOOT_TYPE='hd'                  # boot device: hd, cd, ls120, integrated,
                                # attached, netboot, pxeboot
LIBATA_DMA='disabled'           # Use DMA on ATA Drives ('enabled') or not
                                # ('disabled'). The default 'disabled' allows
                                # ancient IDE CF cards to be booted from.
                                # Use 'enabled' if you boot from a VirtualBox's
                                # virtual device.
MOUNT_BOOT='rw'                 # mount boot device: ro, rw, no
BOOTMENU_TIME='5'               # waiting time of bootmenu in seconds
                                # before activating normal boot
TIME_INFO='MEZ-1MESZ,M3.5.0,M10.5.0/3'
                                # description of local time zone,
                                # don't touch without reading documentation
KERNEL_VERSION='3.14.54'        # kernel version
KERNEL_BOOT_OPTION=''           # append option to kernel command line
COMP_TYPE_OPT='xz'              # compression algorithm if compression is
                                # enabled for OPT archive;
                                # NOTE that some boot types may disallow
                                # some compression algorithms
IP_CONNTRACK_MAX=''             # override maximum limit of connection
                                # tracking entries
POWERMANAGEMENT='acpi'          # select pm interface: none, acpi, apm, apm_rm
                                # apm_rm switches to real mode before invoking
                                # apm power off

#----------------------------------------------------------------------------
# Localisation
#----------------------------------------------------------------------------
LOCALE='de'                     # defines the default language for several
                                # components, such as httpd

#----------------------------------------------------------------------------
# Console settings (serial console, blank time, beep):
#----------------------------------------------------------------------------
```

```
CONSOLE_BLANK_TIME=''           # time in minutes (1-60) to blank
                                # console; '0' = never, '' = system default
BEEP='yes'                      # enable beep after boot and shutdown
SER_CONSOLE='no'                # use serial interface instead of or as
                                # additional output device and main input
                                # device
SER_CONSOLE_IF='0'              # serial interface to use, 0 for ttyS0 (COM1)
SER_CONSOLE_RATE='9600'         # baudrate for serial console

#-------------------------------------------------------------------------
# Debug Settings:
#-------------------------------------------------------------------------
DEBUG_STARTUP='no'              # write an execution trace of the boot

#-------------------------------------------------------------------------
# Keyboard layout
#-------------------------------------------------------------------------
KEYBOARD_LOCALE='auto'          # auto: use most common keyboard layout for
                                # the language specified in 'LOCALE'
#OPT_MAKEKBL='no'               # set to 'yes' to make a new local keyboard
                                # layout map on the fli4l-router

#-------------------------------------------------------------------------
# Ethernet card drivers:
#-------------------------------------------------------------------------
#
#  please see file base_nic.list in your config-dir or read the documentation
#
#
#  If you need a dummy device, use 'dummy' as your NET_DRV
#  and IP_NET_%_DEV='dummy<number>' as your device
#
#-------------------------------------------------------------------------
NET_DRV_N='1'                   # number of ethernet drivers to load, usually 1
NET_DRV_1='ne2k-pci'            # 1st driver: name (e.g. NE2000 PCI clone)
NET_DRV_1_OPTION=''             # 1st driver: additional option
NET_DRV_2='ne'                  # 2nd driver: name (e.g. NE2000 ISA clone)
NET_DRV_2_OPTION='io=0x320'     # 2nd driver: additional option

#-------------------------------------------------------------------------
# Ether networks used with IP protocol:
#-------------------------------------------------------------------------
IP_NET_N='1'                    # number of IP ethernet networks, usually 1
IP_NET_1='192.168.6.1/24'       # IP address of your n'th ethernet card and
                                # netmask in CIDR (no. of set bits)
IP_NET_1_DEV='eth0'             # required: device name like ethX

#-------------------------------------------------------------------------
# Additional routes, optional
#-------------------------------------------------------------------------
IP_ROUTE_N='0'                  # number of additional routes
IP_ROUTE_1='192.168.7.0/24 192.168.6.99'
                                # network/netmaskbits gateway
```

```
IP_ROUTE_2='0.0.0.0/0 192.168.6.99'
                                # example for default-route

#------------------------------------------------------------------------------
# Packet filter configuration
#------------------------------------------------------------------------------

PF_INPUT_POLICY='REJECT'        # be nice and use reject as policy
PF_INPUT_ACCEPT_DEF='yes'       # use default rule set
PF_INPUT_LOG='no'               # don't log at all
PF_INPUT_LOG_LIMIT='3/minute:5' # log 3 events per minute; allow a burst of 5
                                # events
PF_INPUT_REJ_LIMIT='1/second:5' # reject 1 connection per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_INPUT_UDP_REJ_LIMIT='1/second:5'
                                # reject 1 udp packet per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_INPUT_N='1'                  # number of INPUT rules
PF_INPUT_1='IP_NET_1 ACCEPT'    # allow all hosts in the local network to
                                # access the router
PF_INPUT_2='tmpl:samba DROP NOLOG'
                                # drop (or reject) samba access
PF_INPUT_2_COMMENT='no samba traffic allowed'
                                # without logging, otherwise the log file will
                                # be filled with useless entries

PF_FORWARD_POLICY='REJECT'      # be nice and use reject as policy
PF_FORWARD_ACCEPT_DEF='yes'     # use default rule set
PF_FORWARD_LOG='no'             # don't log at all
PF_FORWARD_LOG_LIMIT='3/minute:5'
                                # log 3 events per minute; allow a burst of 5
                                # events
PF_FORWARD_REJ_LIMIT='1/second:5'
                                # reject 1 connection per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_FORWARD_UDP_REJ_LIMIT='1/second:5'
                                # reject 1 udp packet per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_FORWARD_N='2'                # number of FORWARD rules
PF_FORWARD_1='tmpl:samba DROP'  # drop samba traffic if it tries to leave the
                                # subnet
PF_FORWARD_2='IP_NET_1 ACCEPT'  # accept everything else

PF_OUTPUT_POLICY='ACCEPT'       # default policy for outgoing packets
PF_OUTPUT_ACCEPT_DEF='yes'      # use default rule set
PF_OUTPUT_LOG='no'              # don't log at all
PF_OUTPUT_LOG_LIMIT='3/minute:5'
                                # log 3 events per minute; allow a burst of 5
                                # events
PF_OUTPUT_REJ_LIMIT='1/second:5'
                                # reject 1 connection per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_OUTPUT_UDP_REJ_LIMIT='1/second:5'
```

```
                                # reject 1 udp packet per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_OUTPUT_N='0'                 # number of OUTPUT rules

PF_POSTROUTING_N='1'            # number of POSTROUTING rules
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'
                                # masquerade traffic leaving the subnet

PF_PREROUTING_N='0'             # number of PREROUTING rules
PF_PREROUTING_1='1.2.3.4 dynamic:22 DNAT:@client2'
                                # forward ssh connections coming from 1.2.3.4
                                # to client2

PF_PREROUTING_CT_ACCEPT_DEF='yes'
                                # use default rule set
PF_PREROUTING_CT_N='1'          # number of conntrack PREROUTING rules
PF_PREROUTING_CT_1='tmpl:ftp IP_NET_1 HELPER:ftp'
                                # associate FTP conntrack helper for active FTP
                                # forwarded from within the LAN
PF_PREROUTING_CT_2='tmpl:ftp any dynamic HELPER:ftp'
                                # associate FTP conntrack helper for active FTP
                                # forwarded to the router's external IP

PF_OUTPUT_CT_ACCEPT_DEF='yes'   # use default rule set
PF_OUTPUT_CT_N='0'              # number of conntrack OUTPUT rules
PF_OUTPUT_CT_1='tmpl:ftp HELPER:ftp'
                                # associate FTP conntrack helper for outgoing
                                # active FTP on the router (this rule is added
                                # automatically by the tools package if
                                # OPT_FTP='yes' and FTP_PF_ENABLE_ACTIVE='yes')

PF_USR_CHAIN_N='0'              # number of user-defined rules

#-------------------------------------------------------------------------------
# Domain configuration:
# settings for DNS, DHCP server and HOSTS -> see package DNS_DHCP
#-------------------------------------------------------------------------------
DOMAIN_NAME='lan.fli4l'         # your domain name
DNS_FORWARDERS='194.8.57.8'     # DNS servers of your provider,
                                # e.g. ns.n-ix.net

# optional configuration for the host-entry of the router in /etc/hosts
#HOSTNAME_IP='IP_NET_1_IPADDR'  # IP to bind to HOSTNAME
#HOSTNAME_ALIAS_N='0'           # how many ALIAS names for the router
#HOSTNAME_ALIAS_1='router.lan.fli4l'
                                # first ALIAS name
#HOSTNAME_ALIAS_2='gateway.my.lan'
                                # secound ALIAS name

#-------------------------------------------------------------------------------
# imond configuration:
#-------------------------------------------------------------------------------
START_IMOND='no'                # start imond: yes or no
```

```
IMOND_PORT='5000'               # port (tcp), don't open it to the outside
IMOND_PASS=''                   # imond-password, may be empty
IMOND_ADMIN_PASS=''             # imond-admin-password, may be empty
IMOND_LED=''                    # tty for led: com1 - com4 or empty
IMOND_BEEP='no'                 # beep if connection is going up/down
IMOND_LOG='no'                  # log /var/log/imond.log: yes or no
IMOND_LOGDIR='auto'             # log-directory, e.g. /var/log or auto for
                                # saving in auto-detected savedir
IMOND_ENABLE='yes'              # accept "enable/disable" command
IMOND_DIAL='yes'                # accept "dial/hangup" command
IMOND_ROUTE='yes'               # accept "route" command
IMOND_REBOOT='yes'              # accept "reboot" command

#------------------------------------------------------------------------------
# Generic circuit configuration:
#------------------------------------------------------------------------------
IP_DYN_ADDR='yes'               # use dyn. IP addresses (most providers do)
DIALMODE='auto'                 # standard dialmode: auto, manual, or off

#------------------------------------------------------------------------------
# optional package: syslogd
#------------------------------------------------------------------------------
#OPT_SYSLOGD='no'               # start syslogd: yes or no
#SYSLOGD_RECEIVER='yes'         # receive messages from network
SYSLOGD_DEST_N='1'              # number of destinations
SYSLOGD_DEST_1='*.* /dev/console'
                                # n'th prio & destination of syslog msgs
SYSLOGD_DEST_2='*.* @192.168.6.2'
                                # example: loghost 192.168.6.2
SYSLOGD_DEST_3='kern.info /var/log/dial.log'
                                # example: log infos to file

SYSLOGD_ROTATE='no'             # rotate syslog-files once every day
SYSLOGD_ROTATE_DIR='/data/syslog'
                                # move rotated files to ....
SYSLOGD_ROTATE_MAX='5'          # max number of rotated syslog-files

#------------------------------------------------------------------------------
# Optional package: klogd
#------------------------------------------------------------------------------
#OPT_KLOGD='no'                 # start klogd: yes or no

#------------------------------------------------------------------------------
# Optional package: logip
#------------------------------------------------------------------------------
#OPT_LOGIP='no'                 # logip: yes or no
LOGIP_LOGDIR='auto'             # log-directory, e.g. /boot or auto-detected

#------------------------------------------------------------------------------
# Optional package: y2k correction
#------------------------------------------------------------------------------
#OPT_Y2K='no'                   # y2k correction: yes or no
Y2K_DAYS='0'                    # correct hardware y2k-bug: add x days
```

```
#-------------------------------------------------------------------------
# Optional package: PNP
#-------------------------------------------------------------------------
#OPT_PNP='no'                    # install isapnp tools: yes or no
```

Zu beachten ist, dass diese Datei im DOS-Format gespeichert ist. Das heißt, sie enthält jeweils am Zeilenende ein zusätzliches Carriage-Return (CR). Da die meisten Unix-Editoren damit keine Probleme bekommen wurde dieses Format gewählt, denn umgekehrt hat der Windows-Editor bei fehlendem CR am Zeilenende keine Chance!

Sollte es wider Erwarten unter Unix/Linux doch Probleme mit dem Lieblingseditor geben, kann die Datei vor dem Editieren mit einem Befehl in das Unix-Format konvertiert werden:

```
sh unix/dtou config/base.txt
```

Für die Erstellung des Boot-Mediums ist es völlig unerheblich, ob die Datei CRs am Zeilenende enthält oder nicht. Sie werden beim Schreiben auf das Boot-Medium einschließlich der Kommentare komplett ignoriert.

Jetzt aber zum Inhalt ...

3.2. Allgemeine Einstellungen

HOSTNAME Standardwert: `HOSTNAME='fli4l'`

Als erstes sollte man seinem fli4l-Router einen Namen geben.

PASSWORD Standardwert: `PASSWORD='fli4l'`

Das hier angegebene Passwort wird für das Einloggen in den fli4l-Rechner benötigt – sei es per Tastatur direkt am Router oder per SSH von einem anderen Rechner aus (hierzu wird das sshd-Paket benötigt). Es muss aus mindestens einem und darf aus höchstens 126 Zeichen bestehen.

BOOT_TYPE Standardwert: `BOOT_TYPE='hd'`

`BOOT_TYPE` legt im weitesten Sinne das Bootmedium fest. Diese Variable steuert, welche zusätzlichen Treiber (Kernel-Module) und Start-Skripte mit in das RootFS aufgenommen werden. Zum Verständnis eine kurze Skizze des Bootvorgangs:

- Das BIOS des Rechners lädt/startet den Bootloader auf dem Bootmedium.
- Der Bootloader (i.d.R syslinux) entpackt, lädt und startet den Kernel.
- Der Kernel entpackt das RootFS (= das grundlegende Dateisystem mit darin enthalten Programmen und Skripte), mountet das RootFS und beginnt die Start-Skripte abzuarbeiten.
- Je nach `BOOT_TYPE` werden nun die Kernel-Module für das jeweilige Bootmedium geladen, die Boot-Partition gemountet und das OPT-Archiv (`opt.img`) mit den zusätzlichen Programmen entpackt.
- Im Anschluss beginnt die Konfiguration der einzelnen Dienste des fli4l.

Zur Zeit sind folgende Werte für `BOOT_TYPE` gültig:

ls120 Boot von LS120/240 sowie ZIP Disks.

hd Boot von Festplatte. IDE und SATA Geräte werden direkt erkannt, für SCSI, USB oder besondere Controller wird das Paket HD und/oder USB benötigt. Näheres ist der Dokumentation (Seite 126) zum Paket HD zu entnehmen.

cd Boot von CD-ROM. Es wird lediglich das ISO-Image fli4l.iso der CD erzeugt, welches anschließend mit dem jeweiligen Lieblingsbrennprogramm selbst auf CD gebrannt werden muss. Bezüglich SCSI, USB und spezielle Controller ist das Paket HD bzw. USB nötig.

integrated Bei diesem Typ wird kein Bootmedium zu Grunde gelegt, sondern das OPT-Archiv vollständig ins RootFS integriert. Somit entfällt das Mounten des Bootmediums und der Kernel kann gleich alles entpacken. Dieser `BOOT_TYPE` wird z.B. fürs Booten vom Netzwerk benötigt.
Hinweis: Ein remote Update ist natürlich in diesem Fall nicht möglich.

attached Ähnlich wie **integrated**, jedoch wird das OPT-Archiv als Datei `opt.img` ans RootFS angehängt, mit der Folge, dass es wieder im Verzeichnis `/boot` zu finden ist und gesondert beim Bootvorgang entpackt wird.
Ansonsten gilt das unter `integrated` Gesagte.

netboot Entspricht **integrated**. Es wird jedoch zusätzlich das Skript `mknetboot.sh` gestarted, welches ein Image zum Booten via LAN erzeugt. Weiteres ist bitte dem Wiki https://ssl.nettworks.org/wiki/display/f/fli4l+und+Netzboot zu entnehmen.

pxeboot Es werden zwei Images generiert, kernel und rootfs.img. Das sind die beiden vom PXE-Bootloader nachzuladenden Dateien. Beim Aufruf kann die Lokation des tftp-Verzeichnisses angegeben werden und zusätzlich noch ein Unterverzeichnis innerhalb des tftp-Verzeichnisses (–pxesubdir). Weiteres auch hier im Wiki https://ssl.nettworks.org/wiki/display/f/fli4l+und+Netzboot

Hinweis: wie ein fli4l als passender boot-server (pxe/tftp) zu konfigurieren ist, können sie in der Dokumentation des Pakets dns_dhcp nachlesen.

LIBATA_DMA Mit dieser Variable kann eingestellt werden, ob DMA für libata basierte Geräte aktiviert werden soll. Dies ist z.B. bei einigen unvollständig verdrahteten IDE zu Compact-Flash Adaptern nötig. Um DMA zu aktivieren: 'enabled' Default: 'disabled'

MOUNT_BOOT Standardwert: `MOUNT_BOOT='rw'`

Hier wird eingestellt, wie das Boot-Medium gemountet werden soll. Es gibt drei Möglichkeiten:

rw – Read/Write – Schreiben und Lesen ist möglich

ro – Read-Only – Nur Lesen ist möglich

no – None – Medium wird nach dem Boot wieder abgemeldet und kann dann bei Bedarf entnommen werden.

Bei bestimmten Konfigurationen ist es unbedingt erforderlich, das Medium Read/Write anzumelden, z.B. wenn man den DHCP-Server einsetzen oder die imond-Log-Datei auf dem Medium anlegen möchte.

BOOTMENU_TIME Standardwert: `BOOTMENU_TIME='20'`

Hier wird eingestellt, wie lange der syslinux Bootloader warten soll, bis automatisch mit der Standard-Installation gebootet wird.

Im Paket HD besteht die Möglichkeit, über `OPT_RECOVER` eine Funktion zu aktivieren, mit der eine Notfallinstallation aus einer laufenden Installation erstellt werden kann. Diese kann im Bootmenü über die Wahl der Recover-Version aktiviert werden.

Sollte hier der Wert '0' eingestellt sein, wartet der syslinux Bootloader bis der Anwender die Standard- oder die Recover-Version auswählt und aktiviert!

TIME_INFO Standardwert: `TIME_INFO='MEZ-1MESZ,M3.5.0,M10.5.0/3'`

Uhren ticken in der Unix-Welt und damit auch unter fli4l normalerweise nach der UTC (Universal Time Coordinated), einer weltweit einheitlichen Uhrzeit, die vor der Verwendung in die lokale Zeit umgerechnet wird. `TIME_INFO` liefert fli4l die dafür notwendigen Informationen über die Namen der Zeitzonen, die Differenz zu UTC und Regeln, wann auf Sommerzeit und wieder zurück gewechselt wird. Damit das korrekt funktioniert, muss die Hardware Uhr auf UTC gestellt werden (entspricht der Londoner Winterzeit) oder über das Paket chrony mit einem Timeserver synchronisiert werden (diese liefern UTC aus).

Die Einträge in `TIME_INFO` bedeuten dabei folgendes:

```
TIME_INFO='MEZ-1MESZ,M3.5.0,M10.5.0/3'
```

- *MEZ-1:* Wir befinden uns in der mitteleuropäischen Zeitzone (*MEZ*), die der UTC eine Stunde voraus ist *MEZ-1=UTC*.
- *MESZ:* In dieser Zeitzone gibt es Sommerzeit (Mitteleuropäische Sommerzeit). Da nichts weiter angegeben wird, kommt man zur Sommerzeit, indem man die Zeit eine Stunde vorstellt.
- *M3.5.0,M10.5.0/3:* Am letzten Sonntag im März (um 2 Uhr) wird zur Sommerzeit gewechselt, am letzten Sonntag im Oktober (um 3 Uhr) wieder zurück.

Normalerweise braucht man diesen Wert nie anzufassen, es sei denn man sitzt in einer anderen Zeitzone. Will man die Werte anpassen, sollte man einen Blick auf die Spezifikation der Umgebungsvariable TZ werfen, die unter folgender URL zu finden ist (englisch): http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap08.html

KERNEL_VERSION Legt die Version des zu verwendenden Kerns fest. Entsprechend dieser Variable werden der Kern aus *img/kernel-<kernel version>.<compression extension>* und die Kernel-Module aus *opt/files/lib/modules/<kernel version>* selektiert. Für die Kernel ist auch die Extension -virt möglich, die Unterstützung für virtuelle Maschinen beinhaltet. Diese Version setzt jedoch mindestens einen Pentium-Prozessor mit PAE-Support voraus.

KERNEL_BOOT_OPTION Standardwert: `KERNEL_BOOT_OPTION="`

Der Inhalt dieser Variable wird an die Kommandozeile des Kerns in der syslinux.cfg angehängt. Manche Systeme benötigen für korrekten Reboot 'reboot=bios'. Bei WRAP-Systemen also 'reboot=bios'.

COMP_TYPE_ROOTFS Standardwert: `COMP_TYPE_ROOTFS='xz'`

Der Inhalt dieser Variable legt die Kompressionsmethode für das RootFS-Archiv fest. Mögliche Werte sind 'xz', 'lzma' und 'bzip2'.

COMP_TYPE_OPT Standardwert: `COMP_TYPE_OPT='xz'`

Der Inhalt dieser Variable legt die Kompressionsmethode für das OPT-Archiv fest. Mögliche Werte sind 'xz', 'lzma' und 'bzip2'.

POWERMANAGEMENT Standardwert: `POWERMANAGEMENT='acpi'`

Der Kern unterstützt verschiedene Formen des Powermanagements, das etwas betagte APM und das aktuellere ACPI. Hier kann man einstellen, welche Form er verwenden soll. Mögliche Werte sind 'none' (kein Powermanagement), 'acpi' und die beiden APM-Varianten 'apm' und 'apm_rm'. Letzteres schaltet in einen speziellen Prozessormodus, bevor der Router ausgeschaltet wird.

FLI4L_UUID Standardwert: `FLI4L_UUID="`

Hier wird eine eindeutige UUID eingetragen, mit der der fli4l seine persistenten Daten auf z.B. einem USB-Stick finden kann. Eine UUID kann auf einem beliebigen Linux-System (wie auch dem fli4l) mit dem Befehl `'cat␣/proc/sys/kernel/random/uuid'` erstellt werden. Dies gibt bei jedem Aufruf eine neue UUID aus. Diese muss nun in die Variable eintragen werden. Auf einem persistenten Medium (z.B. auf einer Festplatte (OPT_HD) oder einem USB-Stick (OPT_USB und OPT_HD)) muss dann noch ein Verzeichnis mit demselben Namen angelegt werden. Dort wird dann künftig alles gespeichert, das sich gegenüber der Konfiguration geändert hat, ebenso wie persistente Laufzeitdaten wie z.B. DHCP-Leases. Hierzu muss das entsprechende Paket dies natürlich unterstützen (siehe Dokumentation). Der entsprechende Eintrag für den Speicherpfad ist dort dann in der Regel 'auto'.

Sollte der fli4l bereits vor dem Erstellen der UUID und dem Anlegen des Verzeichnisses einige Daten gespeichert haben, so sind diese unter /boot/persistent zu finden und müssen dann manuell an den neuen Speicherort verschoben werden. Deshalb empfiehlt es sich, die UUID gleich anfangs zu erstellen und nicht erst später zu migrieren.

Zudem ist zu beachten, dass `MOUNT_BOOT='ro'` nicht gewählt werden darf, solange das Verzeichnis sich irgendwo auf der /boot Partition befindet.

Ein empfohlener Ort für das persistente Verzeichnis befindet sich auf der /data Partition (ganz oben) oder einem USB-Stick. Das Dateisystem des USB-Sticks sollte VFAT sein oder bei aktivem OPT_HD alle dort unterstützen schreib-lese-fähigen Dateisysteme.

IP_CONNTRACK_MAX Standardwert: `IP_CONNTRACK_MAX="`

Mit Hilfe dieser Variable kann man die maximale mögliche Anzahl gleichzeitiger Verbindungen manuell einstellen. Normalerweise wird anhand des eingebauten Arbeitsspeichers automatisch ein sinnvoller Wert ermittelt. In Tabelle 3.2 sind die verwendeten Voreinstellungen zusammengefasst dargestellt.

Bei Einsatz von FileSharing-Programmen hinter oder auf dem Router und wenig Arbeitsspeicher ist die maximale Anzahl gleichzeitiger Verbindungen aber sehr schnell erreicht und zusätzliche Verbindungen können nicht mehr aufgebaut werden.
Das äußert sich in Fehlermeldungen wie

Tabelle 3.2.: Automatische Einstellung der maximalen Verbindungsanzahl

Arbeitsspeicher in MiB	gleichzeitige Verbindungen
16	1024
24	1280
32	2048
64	4096
128	8192

```
ip_conntrack: table full, dropping packet
```

oder

```
ip_conntrack: Maximum limit of XXX entries exceeded
```

Mittels `IP_CONNTRACK_MAX` lässt sich nun die maximale Anzahl gleichzeitiger Verbindungen fest auf einen bestimmten Wert einstellen. Jede einzelne mögliche Verbindung kostet 350 Bytes Arbeitsspeicher, der nicht mehr für andere Dinge genutzt werden kann. Setzt man also 10000, so sind gerundet 3,34 MB Arbeitsspeicher für den normalen Gebrauch verloren (Kernel, Ramdisks, Programme).

Bei 32 MiB RAM sollte es kein Problem sein, mal eben 2 oder 3 MiB für die ip_conntrack-Tabelle zu reservieren, bei 16 MiB wird es knapp und bei 12 oder sogar 8 MiB ist absolute Sparwut angesagt.

Die momentan benutze Einstellung lässt sich auf der Konsole mit

```
cat /proc/sys/net/ipv4/ip_conntrack_max
```

anzeigen und mit

```
echo "XXX" > /proc/sys/net/ipv4/ip_conntrack_max
```

zur Laufzeit setzen, wobei XXX für die Anzahl der Einträge steht. Die Einträge in der `IP_CONNTRACK`-Tabelle selbst können auf der Konsole mit

```
cat /proc/net/ip_conntrack
```

angesehen und mit

```
cat /proc/net/ip_conntrack | grep -c use
```

gezählt werden.

LOCALE Standardwert: `LOCALE='de'`

Einige Komponenten sind mittlerweile mehrsprachfähig. Dazu zählen beispielsweise das Konsolen-Menü und die Weboberfläche. Mit dieser Variablen können Sie die bevorzugte Sprache auswählen. Verschiedene Komponenten haben noch eine eigene Einstellung womit diese Grundeinstellung, wenn nötig, überlistet werden kann. Wenn die hier angegebene Sprache bei einer Komponente (noch) nicht verfügbar ist, wird auf Englisch zurückgefallen.

Bei `KEYBOARD_LOCALE='auto'` wird versucht ein zu der `LOCALE`-Einstellung passendes Tastatur-Layout ein zu stellen.

Bisher sind folgende Einstellungen möglich: de, en, es, fr, hu, nl.

3.3. Konsolen-Einstellungen

CONSOLE_BLANK_TIME Standard-Einstellung: `CONSOLE_BLANK_TIME="`

Normalerweise aktiviert der Linux-Kern nach einer gewissen Zeit ohne Eingaben auf dem aktuellen Eingabegerät den Bildschirmschoner. Mit der Variable `CONSOLE_BLANK_TIME` kann man diese Zeit konfigurierereren bzw. den Bildschirmschonermodus ganz deaktivieren (`CONSOLE_BLANK_TIME='0'`).

BEEP Standard-Einstellung: `BEEP='yes'`

Signalton am Ende des Boot- und Shutdownprozesses ausgeben.

Trägt man hier 'yes' ein, ertönt ein Signalton am Ende des Boot- und Shutdownprozesses. Wer extremen Platzmangel auf dem Bootmedium hat oder keinen Signalton möchte, kann hier also 'no' eingetragen lassen.

SER_CONSOLE Standard-Einstellung: `SER_CONSOLE='no'`

fli4l kann gänzlich ohne Tastatur und Grafikkarte eingesetzt werden. Damit man den Router auch ohne Telnet oder SSH bedienen sowie alle Boot-Meldungen des Kernels sehen kann, ist es möglich, die Ein- und Ausgaben auf die serielle Schnittstelle umzuleiten. Dazu muss `SER_CONSOLE`, `SER_CONSOLE_IF` sowie `SER_CONSOLE_RATE` angepasst werden.

Die serielle Konsole kann in drei Modi betrieben werden:

`SER_CONSOLE`	Konsolen-Ein-/Ausgabe
no	Ein- und Ausgabe über tty0
yes	Ein- und Ausgabe über serielle Konsole
primary	Ein- und Ausgabe über serielle Konsole, Ausgabe der Kernelmeldungen zusätzlich über tty0
secondary	Ein- und Ausgabe über tty0, Ausgabe der Kernelmeldungen zusätzlich über serielle Konsole

Wenn der Wert von `SER_CONSOLE` verändert wird, wird diese Änderung nur bei der Erstellung eines neuen Bootmediums oder beim remote-Update der syslinux.cfg wirksam.

Achten Sie beim Ausschalten der Seriellen Konsole darauf, sich einen alternativen Zugang zum Router (ssh oder direkt an der Tastatur) zu erhalten.

Weitere Informationen sind im Anhang unter Serielle Konsole (Seite 365) zu finden.

SER_CONSOLE_IF Standard-Einstellung: `SER_CONSOLE_IF='0'`

Nummer der seriellen Schnittstelle für die Konsolenausgabe.

Hier ist die Nummer der Schnittstelle einzutragen, an die die serielle Konsole angeschlossen wird. 0 entspricht ttyS0 bzw. COM1 in der DOS-Welt.

SER_CONSOLE_RATE Standard-Einstellung: `SER_CONSOLE_RATE='9600'`

Übertragungsrate der seriellen Schnittstelle für die Konsolenausgabe.

Hier trägt man die Baud-Rate ein, mit der die Daten auf der seriellen Schnittstelle übertragen werden. Sinnvolle Werte: 4800, 9600, 19200, 38400, 57600, 115200

3.4. Hilfen zum Einkreisen von Problemen und Fehlern

fli4l loggt die gesamten Ausgaben des Bootvorganges in einer Datei (*/var/tmp/boot.log*). Diese Datei kann man sich am Ende des Bootvorganges auf der Konsole oder über den entsprechenden Menüpunkt im Web-Interface ansehen.

Manchmal ist es jedoch sinnvoll, bei Problemen einen ausführlicheren Ablauf der Start-Sequenz zu generieren, um den Bootvorgang hinterher auf Probleme untersuchen zu können. Dazu dient `DEBUG_STARTUP`. Andere Einstellungen unterstützen Entwickler beim Finden von Fehlern in bestimmten Situationen; auch diese Einstellungen werden in diesem Abschnitt dokumentiert.

DEBUG_STARTUP Standard-Einstellung: `DEBUG_STARTUP='no'`

Steht dieser Wert auf 'yes', wird beim Booten jedes ausgeführte Kommando vor seiner Ausführung auf den Schirm geschrieben. Da für das korrekte Funktionieren Änderungen an der syslinux.cfg vorgenommen werden müssen, gilt das für `SER_CONSOLE` Gesagte auch hier. Wenn man die syslinux.cfg von Hand ergänzen will, ist es nötig, ein `fli4ldebug=yes` einzufügen. `DEBUG_STARTUP` muss dann aber trotzdem auf 'yes' stehen.

DEBUG_MODULES Standard-Einstellung: `DEBUG_MODULES='no'`

Einige Module werden automatisch vom Kern geladen, ohne dass man das vorher erkennen kann. `DEBUG_MODULES='yes'` aktiviert einen Modus, der einem die kompletten Modulladesequenzen zeigt, egal, ob sie von einem Skript oder vom Kern angestoßen werden.

DEBUG_ENABLE_CORE Standard-Einstellung: `DEBUG_ENABLE_CORE='no'`

Wird diese Option aktiviert, verursacht jeder Programmabsturz auf dem Router das Erzeugen einer so genannten "core"-Datei, also eines Speicherabbilds des Prozesses direkt vor dem Absturz. Diese Dateien sind auf dem Router im Verzeichnis `/var/log/dumps` zu finden. Diese Dateien können dann genutzt werden, um den Programmfehler besser zu finden. Genaueres finden Sie hierzu im Abschnitt "Entwanzen von Programmen auf dem fli4l" (Seite 264) in der Dokumentation des SRC-Pakets.

DEBUG_MDEV Standard-Einstellung: `DEBUG_MDEV='no'`

Mit `DEBUG_MDEV='yes'` werden alle Aktionen, die in Zusammenhang mit dem `mdev`-Dämon stehen und somit mit dem Hinzufügen oder Entfernen von Geräteknoten in `/dev` oder dem Laden von Firmware zu tun haben, in der Datei `/dev/mdev.log` protokolliert.

DEBUG_IPTABLES Standard-Einstellung: `DEBUG_IPTABLES='no'`

Mit `DEBUG_IPTABLES='yes'` werden alle `iptables`-Aufrufe inklusive dem Rückgabewert in `/var/log/iptables.log` protokolliert.

DEBUG_IP Standard-Einstellung: `DEBUG_IP='no'`

Diese Variable aktiviert bei `DEBUG_IP='yes'` das Protokollieren aller Aufrufe des Programms `/sbin/ip` in der Datei `/var/log/wrapper.log`.

3.5. Verwendung einer eigenen /etc/inittab

Man kann von Init zusätzliche Programme auf zusätzlichen Konsolen starten lassen oder die Standardkommandos der Init-Konfigurationsdatei verändern. Ein Eintrag sieht wie folgt aus:

```
device:runlevel:action:command
```

Das *device* ist das Gerät, über das das Programm seine Ein-/Ausgaben machen soll. Möglich sind hier die normalen Terminals tty1-tty4 oder serielle Terminals ttyS0-ttySn mit $n <$ Anzahl der vorhandenen seriellen Schnittstellen.

Als *action* kommen in der Regel *askfirst* oder *respawn* in Frage. Bei askfirst wird auf einen Tastendruck gewartet, bevor das Kommando ausgeführt wird, bei respawn wird es automatisch neu gestartet, wenn sich das Programm beendet.

command ist das Programm, das ausgeführt werden soll. Es ist mit vollständiger Pfadangabe zu spezifizieren.

Die Busybox-Dokumentation unter http://www.busybox.net enthält eine genaue Beschreibung des inittab Formats.

Die normale inittab sieht wie folgt aus:

```
::sysinit:/etc/rc
::respawn:cttyhack /usr/local/bin/mini-login
::ctrlaltdel:/sbin/reboot
::shutdown:/etc/rc0
::restart:/sbin/init
```

Diese könnte man z.B. um den Eintrag

```
tty2::askfirst:cttyhack /usr/local/bin/mini-login
```

erweitern, um ein zweites Login auf Terminal zwei zu bekommen. Dazu einfach die Datei opt/etc/inittab nehmen, nach <config verzeichnis>/etc/inittab kopieren und mit einem Editor bearbeiten.

3.6. Länderspezifische Tastaturlayouts

KEYBOARD_LOCALE Standard-Einstellung: `KEYBOARD_LOCALE='auto'`

Wenn man öfter direkt auf dem fli4l Router arbeitet ist ein lokales Tastaturlayout eine willkommene Hilfe. Mit `KEYBOARD_LOCALE='auto'` wird versucht, ein Tastaturlayout passend zu der Einstellung von `LOCALE` zu benutzen. Mit der Einstellung `"` wird kein lokales Tastaturlayout auf dem fli4l–Router installiert; es wird dann das im Kernel anwesende Standardlayout verwendet. Alternativ kann man auch den Namen einer lokalen Tastaturlayoutmap direkt angeben. Wenn z.B. `'de-latin1'` eingestellt wird, prüft der Buildprozess ob in opt/etc eine Datei mit Namen de-latin1.map vorliegt. Wenn ja, wird die entsprechende .map-Datei als Tastaturlayout eingebunden.

OPT_MAKEKBL Standard-Einstellung: `OPT_MAKEKBL='no'`

Wenn man für seine Tastatur eine map Datei erstellen will muss man wie folgt vorgehen:

- `OPT_MAKEKBL` auf 'yes' setzen.
- Auf dem Router 'makekbl.sh' anrufen. Vorzugsweise macht man dies über eine ssh-Verbindung weil das Tastaturlayout sich ändert und das lästig sein kann.
- Die Anweisungen befolgen.
- Ihre neue <locale>.map Datei liegt in der /tmp.
- Die Arbeiten direkt auf dem Router sind jetzt abgeschlossen.
- Nun kopieren Sie die eben erzeugte Tastaturtabelle in Ihr fli4l-Verzeichnis unter opt/etc/<locale>.map. Wenn Sie jetzt `KEYBOARD_LOCALE`='<locale>' setzen wird beim nächsten Buildprozess Ihre neu erzeugtes Tastaturlayout benutzt.
- Vergessen Sie nicht `OPT_MAKEKBL` wieder auf 'no' zu setzen.

3.7. Ethernet-Netzwerkkarten-Treiber

NET_DRV_N Standard-Einstellung: `NET_DRV_N='1'`

Anzahl der benötigten Netzwerkkarten-Treiber.

Wird der Router nur für ISDN verwendet, so gibt es in der Regel nur eine Netzwerkkarte, der Standard-Wert ist also 1.

Bei DSL werden jedoch meistens zwei Netzwerkkarten verwendet.

Hierbei muss man zwei Fälle unterscheiden:

1. Beide Karten sind vom gleichen Typ. Dann muss nur ein Treiber geladen werden, der dann beide Karten anspricht, also `NET_DRV_N='1'`.
2. Es werden zwei verschiedene Karten verwendet, dann muss man hier '2' angeben und den Treiber für die zweite Karte angeben.

NET_DRV_x Standard-Einstellung: `NET_DRV_1='ne2k-pci'`

Hier wird der Treiber für die Netzwerkkarte angegeben. Über die Variable `NET_DRV_1` wird standardmäßig der Treiber für eine NE2000-kompatible Netzwerkkarte geladen.

Weitere verfügbare Treiber für Netzwerkkartenfamilien sind in den Tabellen 3.3 und 3.4 eingetragen.

Die 3COM EtherLinkIII (3c509) muss über das Dostool 3c509cfg.exe (beziehbar von ftp://ftp.ihg.uni-duisburg.de/Hardware/3com/3C5x9n/3C5X9CFG.EXE)

konfiguriert werden. Dabei sollten IRQ und I/O-Port und gegebenenfalls auch der Anschluss (BNC/TP) eingestellt werden.

NET_DRV_x_OPTION Standard-Einstellung: `NET_DRV_x_OPTION="`

Der Eintrag kann in der Regel leer bleiben.

Bei manchen ISA-Karten braucht der Treiber zusätzliche Informationen, um die Karte zu finden, z.B. die I/O-Adresse. Bei NE2000-kompatiblen ISA-Karten und bei der EtherExpress16 ist dies zum Beispiel der Fall.

Hier ist z.B.

```
NET_DRV_x_OPTION='io=0x340'
```

zu setzen (oder der entsprechende numerische Wert).

Ist keine Option nötig, kann die Variable leer gelassen werden.

Sind mehrere Optionen nötig, sind diese durch Leerzeichen zu trennen, z.B.

```
NET_DRV_x_OPTION='irq=9 io=0x340'
```

Werden zwei identische Netzwerkkarten verwendet, z.B. NE2000-ISA-Karten, müssen die verschiedenen I/O-Werte durch Komma getrennt werden, also

```
NET_DRV_x_OPTION='io=0x240,0x300'
```

Die beiden IO-Werte müssen durch Komma ohne Blank getrennt werden!

Dieses funktioniert nicht bei allen Netzwerkkarten-Treibern. Einige muss man auch doppelt laden, also dann `NET_DRV_N='2'`. In diesem Fall müssen aber mit der Option "-o" verschiedene Namen vergeben werden, z.B.

```
NET_DRV_N='2'
NET_DRV_1='3c503'
NET_DRV_1_OPTION='-o 3c503-0 io=0x280'
NET_DRV_2='3c503'
NET_DRV_2_OPTION='-o 3c503-1 io=0x300'
```

Unser Tip: erst die Komma-Methode ausprobieren, danach das mehrfache Laden mit Option "-o" versuchen.

Nachfolgend noch einige Beispiele von Netzwerkkarten:

- 1 x NE2000 ISA

```
NET_DRV_1='ne'
NET_DRV_1_OPTION='io=0x340'
```

- 1 x 3COM EtherLinkIII (3c509)

```
NET_DRV_1='3c509'
NET_DRV_1_OPTION=''
```

  Siehe zu dieser Karte auch:

  http://extern.fli4l.de/fli4l_faqengine/faq.php?display=faq&faqnr=132&catnr=7&prog=1
  http://extern.fli4l.de/fli4l_faqengine/faq.php?display=faq&faqnr=133&catnr=7&prog=1
  http://extern.fli4l.de/fli4l_faqengine/faq.php?display=faq&faqnr=135&catnr=7&prog=1

- 2 x NE2000 ISA

```
NET_DRV_1='ne'
NET_DRV_1_OPTION='io=0x320,0x340'
```

  Oft muss man hier noch die IRQ-Werte mit angeben, also

```
NET_DRV_1_OPTION='io=0x320,0x340 irq=3,5'
```

  Man sollte es hier zunächst ohne Angabe der Interrupts probieren und lediglich dann die Interrupts eintragen, wenn ohne Angabe der Interrupts die Netzwerkkarten nicht gefunden werden.

- 2 x NE2000 PCI

```
NET_DRV_1='ne2k-pci'
NET_DRV_1_OPTION=''
```

- 1 x NE2000 ISA, 1 x NE2000 PCI

```
NET_DRV_1='ne'
NET_DRV_1_OPTION='io=0x340'
NET_DRV_2='ne2k-pci'
NET_DRV_2_OPTION=''
```

- 1 x SMC WD8013, 1 x NE2000 ISA

```
NET_DRV_1='wd'
NET_DRV_1_OPTION='io=0x270'
NET_DRV_2='ne2k'
NET_DRV_2_OPTION='io=0x240'
```

Vollständige Listen der möglichen Treiber finden Sie in der Tabelle der möglichen Kartentreiber und in der Tabelle der möglichen WLAN-Kartentreiber.

Falls Sie ein dummy-Device brauchen, verwenden Sie 'dummy' für `NET_DRV_x` *und* `IP_NET_x_DEV` *(Seite 42)='dummy<Nummer>' als Device-Name.*

Kernel 3.14				Bus	NET_DRV_x	Kartenfamilie
	v	n	vn			
x	x	x	x	isa	3c509	3Com Etherlink III (3c509, 3c509B, 3c529, 3c579) ethernet
x	x	x	x	isa	3c515	3Com 3c515 Corkscrew
x	x	x	x	pcmcia	3c574_cs	3Com 3c574 series PCMCIA ethernet
x	x	x	x	pcmcia	3c589_cs	3Com 3c589 series PCMCIA ethernet
x	x	x	x	pci	3c59x	3Com 3c59x/3c9xx ethernet
x	x	x	x	pci	8139cp	RealTek RTL-8139C+ series 10/100 PCI Ethernet
x	x	x	x	pci	8139too	RealTek RTL-8139 Fast Ethernet
x	x	x	x	pci	acenic	AceNIC/3C985/GA620 Gigabit Ethernet
x	x	x	x	pci	alx	Qualcomm Atheros(R) AR816x/AR817x PCI-E Ethernet Network
x	x	x	x	pci	amd8111e	AMD8111 based 10/100 Ethernet Controller
x	x	x	x	usb	asix	ASIX AX8817X based USB 2.0 Ethernet Devices
x	x	x	x	pci	atl1	Atheros L1 Gigabit Ethernet
x	x	x	x	pci	atl1c	Qualcom Atheros 100/1000M Ethernet Network
x	x	x	x	pci	atl1e	Atheros 1000M Ethernet Network
x	x	x	x	pci	atl2	Atheros Fast Ethernet Network
x	x	x	x	isa	atp	RealTek RTL8002/8012 parallel port Ethernet
x	x	x	x	usb	ax88179_178a	ASIX AX88179/178A based USB 3.0/2.0 Gigabit Ethernet Devices
x	x	x	x	pcmcia	axnet_cs	Asix AX88190 PCMCIA ethernet
x	x	x	x	pci	b44	Broadcom 44xx/47xx 10/100 PCI ethernet
x	x	x	x	pci	be2net	Emulex OneConnect NIC Driver 10.0.600.0u
x	x	x	x	pci	bna	Brocade 10G PCIe Ethernet
x	x	x	x	pci	bnx2	Broadcom NetXtreme II BCM5706/5708/5709/5716
x	x	x	x	pci	bnx2x	Broadcom NetXtreme II BCM57710/ 57711/ 57711E/ 57712/ 57712_MF/ 57800/ 57800_MF/ 57810/ 57810_MF/ 57840/ 57840_MF
x	x	x	x	pci	cassini	Sun Cassini(+) ethernet
x	x	x	x	usb	catc	CATC EL1210A NetMate USB Ethernet
x	x	x	x	usb	cdc_eem	USB CDC EEM
x	x	x	x	usb	cdc_ether	USB CDC Ethernet devices
x	x	x	x	usb	cdc_mbim	USB CDC MBIM host
x	x	x	x	usb	cdc_ncm	USB CDC NCM host
x	x	x	x	usb	cdc_subset	Simple 'CDC Subset' USB networking links
x	x	x	x	usb	cx82310_eth	Conexant CX82310-based ADSL router USB ethernet
x	x	x	x	pci	cxgb	Chelsio 10Gb Ethernet
x	x	x	x	pci	cxgb3	Chelsio T3 Network

Kernel 3.14				Bus	NET_DRV_x	Kartenfamilie
	v	n	vn			
x	x	x	x	pci	cxgb4	Chelsio T4/T5 Network
x	x	x	x	pci	cxgb4vf	Chelsio T4/T5 Virtual Function (VF) Network
x	x	x	x	pci	de2104x	Intel/Digital 21040/1 series PCI Ethernet
x	x	x	x	isa	de4x5	Digital DE425, DE434, DE435, DE450, DE500
x	x	x	x	pci	defxx	DEC FDDIcontroller TC/EISA/PCI (DEFTA/DEFEA/DEFPA) driver v1.10 2006/12/14
x	x	x	x	pci	dl2k	D-Link DL2000-based Gigabit Ethernet Adapter
x	x	x	x	usb	dm9601	Davicom DM96xx USB 10/100 ethernet devices
x	x	x	x	pci	dmfe	Davicom DM910X fast ethernet
x	x	x	x	virtual	dummy	Dummy Network Interface
x	x	x	x	pci	e100	Intel(R) PRO/100 Network
x	x	x	x	pci	e1000	Intel(R) PRO/1000 Network
x	x	x	x	pci	e1000e	Intel(R) PRO/1000 Network
x	x	x	x	pci	enic	Cisco VIC Ethernet NIC
x	x	x	x	pci	epic100	SMC 83c170 EPIC series Ethernet
x	x	x	x	pci	fealnx	Myson MTD-8xx 100/10M Ethernet PCI Adapter
x	x	x	x	pcmcia	fmvj18x_cs	fmvj18x and compatible PCMCIA ethernet
x	x	x	x	pci	forcedeth	Reverse Engineered nForce ethernet
x	x	x	x	usb	gl620a	GL620-USB-A Host-to-Host Link cables
x	x	x	x	pci	hamachi	Packet Engines 'Hamachi' GNIC-II Gigabit Ethernet
x	x	x	x	pci	hp100	HP CASCADE Architecture Driver for 100VG-AnyLan Network Adapters
x	x	x	x	usb	hso	USB High Speed Option
x	x	x	x	usb	huawei_cdc_ncm	USB CDC NCM host driver with encapsulated protocol support
x	x	x	x	pci	i40e	Intel(R) Ethernet Connection XL710 Network
x	x	x	x	pci	i40evf	Intel(R) XL710 X710 Virtual Function Network
x	x	x	x	pci	igb	Intel(R) Gigabit Ethernet Network
x	x	x	x	pci	igbvf	Intel(R) Gigabit Virtual Function Network
x	x	x	x	usb	int51x1	Intellon usb powerline adapter
x	x	x	x	pci	ipg	IC Plus IP1000 Gigabit Ethernet Adapter Linux
x	x	x	x	usb	ipheth	Apple iPhone USB Ethernet
x	x	x	x	pci	ixgb	Intel(R) PRO/10GbE Network
x	x	x	x	pci	ixgbe	Intel(R) 10 Gigabit PCI Express Network
x	x	x	x	pci	ixgbevf	Intel(R) 82599 Virtual Function
x	x	x	x	pci	jme	JMicron JMC2x0 PCI Express Ethernet

Kernel 3.14				Bus	NET_DRV_x	Kartenfamilie
	v	n	vn			
x	x	x	x	usb	kalmia	Samsung Kalmia USB network
x	x	x	x	usb	kaweth	KL5USB101 USB Ethernet
x	x	x	x	pci	ksz884x	KSZ8841/2 PCI network
x	x	x	x	isa	lance	AMD LANCE and PCnet (AT1500, NE2100)
x	x	x	x	usb	lg-vl600	LG-VL600 modem's ethernet link
x	x	x	x	usb	mcs7830	USB to network adapter MCS7830)
x	x	x	x	pci	mlx4_core	Mellanox ConnectX HCA low-level
x	x	x	x	pci	myri10ge	Myricom 10G driver (10GbE)
x	x	x	x	pci	natsemi	National Semiconductor DP8381x series PCI Ethernet
x	x	x	x	isa	ne	NE1000/NE2000 ISA/PnP Ethernet
x	x	x	x	pci	ne2k-pci	PCI NE2000 clone
x	x	x	x	usb	net1080	NetChip 1080 based USB Host-to-Host Links
x	x	x	x	pci	netxen_nic	QLogic/NetXen (1/10) GbE Intelligent Ethernet
x	x	x	x	isa	ni65	AMD Lance Am7990
x	x	x	x	pci	niu	Sun Neptun Ethernet
x	x	x	x	pcmcia	nmclan_cs	New Media PCMCIA ethernet
x	x	x	x	pci	ns83820	National Semiconductor DP83820 10/100/1000
x	x	x	x	pci	pch_gbe	EG20T PCH Gigabit ethernet
x	x	x	x	pci	pcnet32	PCnet32 and PCnetPCI based ethercards
x	x	x	x	pcmcia	pcnet_cs	NE2000 compatible PCMCIA ethernet
x	x	x	x	usb	pegasus	Pegasus/Pegasus II USB Ethernet
x	x	x	x	usb	plusb	Prolific PL-2301/2302/25A1 USB Host to Host Link
x	x	x	x	pci	qla3xxx	QLogic ISP3XXX Network Driver v2.03.00-k5
x	x	x	x	pci	qlcnic	QLogic 1/10 GbE Converged/Intelligent Ethernet
x	x	x	x	pci	qlge	QLogic 10 Gigabit PCI-E Ethernet
x	x	x	x	usb	qmi_wwan	Qualcomm MSM Interface (QMI) WWAN
x	x	x	x	pci	r6040	RDC R6040 NAPI PCI FastEthernet
x	x	x	x	usb	r8152	Realtek RTL8152/RTL8153 Based USB Ethernet Adapters
x	x	x	x	pci	r8169	RealTek RTL-8169 Gigabit Ethernet
x	x	x	x	usb	rndis_host	USB Host side RNDIS
x	x	x	x	usb	rtl8150	rtl8150 based usb-ethernet
x	x	x	x	pci	s2io	Neterion 10GbE Server NIC
x	x	x	x	isa	sb1000	General Instruments SB1000
x	x	x	x	pci	sc92031	Silan SC92031 PCI Fast Ethernet Adapter
x	x	x	x	pci	sfc	Solarflare Communications network
x	x	x	x	pci	sis190	SiS sis190/191 Gigabit Ethernet
x	x	x	x	pci	sis900	SiS 900 PCI Fast Ethernet

Kernel 3.14				Bus	NET_DRV_x	Kartenfamilie
	v	n	vn			
x	x	x	x	pci	skfp	SysKonnect FDDI PCI adapter
x	x	x	x	pci	skge	SysKonnect Gigabit Ethernet
x	x	x	x	pci	sky2	Marvell Yukon 2 Gigabit Ethernet
x	x	x	x	isa	smc-ultra	SMC Ultra/EtherEZ ISA/PnP Ethernet
x	x	x	x	isa	smc9194	SMC's 9000 series of Ethernet cards
x	x	x	x	pcmcia	smc91c92_cs	SMC 91c92 series PCMCIA ethernet
x	x	x	x	usb	smsc75xx	SMSC75XX USB 2.0 Gigabit Ethernet Devices
x	x	x	x	pci	smsc9420	SMSC LAN9420
x	x	x	x	usb	smsc95xx	SMSC95XX USB 2.0 Ethernet Devices
x	x	x	x	usb	sr9700	SR9700 one chip USB 1.1 USB to Ethernet device from http://www.corechip-sz.com/
x	x	x	x	usb	sr9800	SR9800 USB 2.0 USB2NET Dev : http://www.corechip-sz.com
x	x	x	x	pci	starfire	Adaptec Starfire Ethernet
x	x	x	x	pci	stmmac	STMMAC 10/100/1000 Ethernet device
x	x	x	x	pci	sundance	Sundance Alta Ethernet
x	x	x	x	pci	sungem	Sun GEM Gbit ethernet
x	x	x	x	pci	sunhme	Sun HappyMealEthernet(HME) 10/100baseT ethernet
x	x	x	x	pci	tehuti	Tehuti Networks(R) Network
x	x	x	x	pci	tg3	Broadcom Tigon3 ethernet
x	x	x	x	pci	tlan	TI ThunderLAN based ethernet PCI adapters
x	x	x	x	pci	tulip	Digital 21*4* Tulip ethernet
x	x	x	x	pci	typhoon	3Com Typhoon Family (3C990, 3CR990, and variants)
x	x	x	x	pci	uli526x	ULi M5261/M5263 fast ethernet
x	x	x	x	pci	via-rhine	VIA Rhine PCI Fast Ethernet
x	x	x	x	pci	via-velocity	VIA Networking Velocity Family Gigabit Ethernet Adapter
		x	x	virtio	virtio_net	Virtio network
		x	x	pci	vmxnet3	VMware vmxnet3 virtual NIC
x	x	x	x	pci	vxge	Neterion's X3100 Series 10GbE PCIe I/OVirtualized Server Adapter
x	x	x	x	isa	wd	Western Digital wd8003/wd8013 ; SMC Elite, Elite16 ethernet
x	x	x	x	pci	winbond-840	Winbond W89c840 Ethernet
		x	x	xen	xen-netfront	Xen virtual network device frontend
x	x	x	x	pcmcia	xirc2ps_cs	Xircom PCMCIA ethernet
x	x	x	x	pci	xircom_cb	Xircom Cardbus ethernet
x	x	x	x	pci	yellowfin	Packet Engines Yellowfin G-NIC Gigabit Ethernet
x	x	x	x	usb	zaurus	Sharp Zaurus PDA, and compatible products

Tabelle 3.3.: Tabelle der möglichen Netzwerkkartentreiber; Legende: v=virt, n=nonfree, vn=virt-nonfree

Kernel						
3.14				3.16		
	v	n	vn		v	n
x	x	x	x	pci	adm8211	IEEE 802.11b wireless cards based on ADMtek ADM8211
x	x	x	x	isa,pci	airo	Cisco/Aironet 802.11 wireless ethernet cards
x	x	x	x	pcmcia	airo_cs	Cisco/Aironet 802.11 wireless ethernet cards
x	x	x	x	usb	ar5523	Atheros AR5523 based USB dongles
x	x	x	x	usb	at76c50x-usb	Atmel at76x USB Wireless LAN
x	x	x	x	pci	ath10k_pci	Driver support for Atheros QCA988X PCIe devices
x	x	x	x	pci	ath5k	5xxx series of Atheros 802.11 wireless LAN cards
x	x	x	x	usb	ath6kl_usb	Driver support for Atheros AR600x USB devices
x	x	x	x	pci	ath9k	Atheros 802.11n wireless LAN cards
x	x	x	x	usb	ath9k_htc	Atheros driver 802.11n HTC based wireless devices
x	x	x	x	pcmcia	atmel_cs	Atmel at76c50x 802.11 wireless ethernet cards
x	x	x	x	pci	atmel_pci	Atmel at76c50x 802.11 wireless ethernet cards
x	x	x	x	pci,pcmcia	b43	Broadcom B43 wireless
x	x	x	x	pci	b43legacy	Broadcom B43legacy wireless
x	x	x	x	usb	brcmfmac	Broadcom 802.11 wireless LAN fullmac
x	x	x	x	pci	brcmsmac	Broadcom 802.11n wireless LAN
x	x	x	x	usb	carl9170	Atheros AR9170 802.11n USB wireless
x	x	x	x	pcmcia	hostap_cs	Intersil Prism2-based 802.11 wireless LAN cards (PC Card)
x	x	x	x	pci	hostap_pci	Intersil Prism2.5-based 802.11 wireless LAN PCI cards
x	x	x	x	pci	hostap_plx	Intersil Prism2-based 802.11 wireless LAN cards (PLX)
x	x	x	x	pci	ipw2100	Intel(R) PRO/Wireless 2100 Network
x	x	x	x	pci	ipw2200	Intel(R) PRO/Wireless 2200/2915 Network
x	x	x	x	pci	iwl3945	Intel(R) PRO/Wireless 3945ABG/BG Network Connection driver for Linu
x	x	x	x	pci	iwl4965	Intel(R) Wireless WiFi 4965 driver for Linux
x	x	x	x	pci	iwlwifi	Intel(R) Wireless WiFi driver for Linux
x	x	x	x	pcmcia	libertas_cs	Marvell 83xx compact flash WLAN cards
x	x	x	x	usb	libertas_tf_usb	8388 USB WLAN Thinfirm
x	x	x	x	virtual	mac80211_hwsim	Software simulator of 802.11 radio(s) for mac80211
x	x	x	x	pci	mwifiex_pcie	Marvell WiFi-Ex PCI-Express Driver version 1.0
x	x	x	x	usb	mwifiex_usb	Marvell WiFi-Ex USB Driver version1.0
x	x	x	x	pci	mwl8k	Marvell TOPDOG(R) 802.11 Wireless Network
x	x	x	x	pcmcia	orinoco_cs	PCMCIA Lucent Orinoco, Prism II based and similar wireless cards
x	x	x	x	pci	orinoco_nortel	wireless LAN cards using the Nortel PCI bridge
x	x	x	x	pci	orinoco_plx	wireless LAN cards using the PLX9052 PCI bridge
x	x	x	x	pci	orinoco_tmd	wireless LAN cards using the TMD7160 PCI bridge
x	x	x	x	usb	orinoco_usb	Orinoco wireless LAN cards using EZUSB bridge
x	x	x	x	pci	p54pci	Prism54 PCI wireless
x	x	x	x	usb	p54usb	Prism54 USB wireless
x	x	x	x	pcmcia	ray_cs	Raylink/WebGear wireless LAN
x	x	x	x	usb	rndis_wlan	RNDIS based USB Wireless adapters
x	x	x	x	pci	rt2400pci	Ralink RT2400 PCI & PCMCIA Wireless LAN
x	x	x	x	pci	rt2500pci	Ralink RT2500 PCI & PCMCIA Wireless LAN
x	x	x	x	usb	rt2500usb	Ralink RT2500 USB Wireless LAN
x	x	x	x	pci	rt2800pci	Ralink RT2800 PCI & PCMCIA Wireless LAN
x	x	x	x	usb	rt2800usb	Ralink RT2800 USB Wireless LAN
x	x	x	x	pci	rt61pci	Ralink RT61 PCI & PCMCIA Wireless LAN
x	x	x	x	usb	rt73usb	Ralink RT73 USB Wireless LAN
x	x	x	x	pci	rtl8180	RTL8180 / RTL8185 PCI wireless
x	x	x	x	usb	rtl8187	RTL8187/RTL8187B USB wireless
x	x	x	x	pci	rtl8188ee	Realtek 8188E 802.11n PCI wireless

Kernel						
3.14				3.16		
	v	n	vn		v	n
x	x	x	x	pci	rtl8192ce	Realtek 8192C/8188C 802.11n PCI wireless
x	x	x	x	usb	rtl8192cu	Realtek 8192C/8188C 802.11n USB wireless
x	x	x	x	pci	rtl8192de	Realtek 8192DE 802.11n Dual Mac PCI wireless
x	x	x	x	pci	rtl8192se	Realtek 8192S/8191S 802.11n PCI wireless
x	x	x	x	pci	rtl8723ae	Realtek 8723E 802.11n PCI wireless
x	x	x	x	usb	sierra_net	USB-to-WWAN Driver for Sierra Wireless modems
x	x	x	x	pcmcia	spectrum_cs	Symbol Spectrum24 Trilogy cards with firmware downloader
x	x	x	x	usb	usb8xxx	8388 USB WLAN
x	x	x	x	pci	wil6210	60g WiFi WIL6210 card
x	x	x	x	pcmcia	wl3501_cs	Planet wl3501 wireless
x	x	x	x	usb	zd1201	ZyDAS ZD1201 based USB Wireless adapters
x	x	x	x	usb	zd1211rw	USB driver for devices with the ZD1211 chip

Tabelle 3.4.: Tabelle der möglichen WLAN-Kartentreiber; Legende: v=virt, n=nonfree, vn=

3.8. Netzwerke

IP_NET_N Standardwert: `IP_NET_N='1'`

Anzahl der Netzwerke, die an das IP-Protokoll gebunden werden sollen. Im Normalfall also '1'. Falls es keine Netzwerke geben sollte oder sie über einen anderen Weg konfiguriert werden, und daher `IP_NET_N` auf 0 gesetzt wird, wird beim Bauen der Archive eine Warnung ausgegeben. Diese kann man durch setzen von `IGNOREIPNETWARNING`='yes' aussschalten.

IP_NET_x Standardwert: `IP_NET_1='192.168.6.1/24'`

Die IP-Adresse und die Netzmaske in CIDR[1]-Schreibweise des x'ten Devices im fli4l-Router. Wird die IP-Adresse dynamisch durch einen DHCP-Klienten zugewiesen, kann hier auch 'dhcp' als Wert eingetragen werden.

Nachfolgende Tabelle zeigt CIDR und Punkt-Schreibweise (DOT Notation).

[1] Classless Inter-Domain Routing

CIDR	Netzmaske	Anzahl IPs
/8	255.0.0.0	16777216
/16	255.255.0.0	65536
/23	255.255.254.0	512
/24	255.255.255.0	256
/25	255.255.255.128	128
/26	255.255.255.192	64
/27	255.255.255.224	32
/28	255.255.255.240	16
/29	255.255.255.248	8
/30	255.255.255.252	4
/31	255.255.255.254	2
/32	255.255.255.255	1

Anmerkung: Da jeweils eine IP für die Netzwerk- und Broadcast-Adresse reserviert sind, errechnet sich die maximale Anzahl der Hosts im Netzwerk zu: `Anzahl_Hosts = Anzahl_IPs - 2`. Die kleinste mögliche Netzmaske wäre also `/30`, entspricht 4 IPs und somit 2 möglichen Hosts.

IP_NET_x_DEV Standard-Einstellung: `IP_NET_1_DEV='eth0'`

Erforderlich: Device-Name der Netzwerkkarte.

Ab Version 2.1.8 ist die Angabe des verwendeten Device zwingend erforderlich! Die Namen der Devices beginnen in den meisten Fällen mit `'eth'` gefolgt von einer Zahl. Die erste vom System erkannte Netzwerkkarte bekommt den Namen `'eth0'`, die zweite `'eth1'` usw.

Beispiel:

```
IP_NET_1_DEV='eth0'
```

fli4l beherrscht auch IP-Aliasing, also die Zuweisung von mehreren IPs auf eine Netzwerkkarte. Zusätzliche IPs definiert man einfach mit einem weiteren Netzwerk auf dem selben Device. Beim Überprüfen der Konfiguration informiert mkfli4l darüber, dass man ein solches Alias definiert — diese Warnung kann man dann ignorieren.

Beispiel:

```
IP_NET_1='192.168.6.1/24'
IP_NET_1_DEV='eth0'
IP_NET_2='192.168.7.1/24'
IP_NET_2_DEV='eth0'
```

IP_NET_x_MAC Standard-Einstellung: `IP_NET_1_MAC=`"

Optional: MAC-Adresse der Netzwerkkarte.

Hier kann die Hardwareadresse (MAC) der Netzwerkkarte angepasst werden. Dies ist zum Beispiel nützlich, wenn Sie einen DHCP-Provider nutzen wollen, der eine bestimmte MAC-Adresse erwartet. Wird `IP_NET_x_MAC` leer gelassen oder gar nicht angegeben, so wird die voreingestellte MAC-Adresse der Netzwerkkarte verwendet. Die allermeisten Nutzer werden diese Variable nicht benötigen.

Beispiel:

```
IP_NET_1_MAC='01:81:42:C2:C3:10'
```

IP_NET_x_NAME Standard-Einstellung: `IP_NET_x_NAME`=”

Optional: Der IP-Adresse der Netzwerkkarte einen Namen geben.

Bei umgekehrter Namensauflösung der IP-Adresse der Netzwerkkarte erscheint standardmässig ein Name der Form ’fli4l-ethx.<domain>’. In `IP_NET_x_NAME` kann man selbst einen anderen Namen angeben. Dieser Name wird dann bei umgekehrter Namensauflösung angezeigt. Bei einer öffentlichen IP-Adresse kann man so erreichen, dass immer der öffentliche Name aufgelöst wird.

Beispiel:

```
IP_NET_2='80.126.238.229/32'
IP_NET_2_NAME='ajv.xs4all.nl'
```

IP_NET_x_TYPE

IP_NET_x_COMMENT Standard-Einstellung: `IP_NET_x_COMMENT`=”

Optional: Die Angabe dient dazu, einem Device einen ’aussagekräftigen’ Namen zu geben. Dieser kann dann in Paketen wie z.B. rrdtool zur Identifikation des Netzwerks verwendet werden.

3.9. Zusätzliche Routen (optional)

IP_ROUTE_N Standard-Einstellung: `IP_ROUTE_N`=’0’

Anzahl von zusätzlichen Netzwerkrouten. Zusätzliche Netzwerkrouten sind zum Beispiel dann erforderlich, wenn sich im LAN weitere Router befinden, über die andere Netzwerke erreichbar sein sollen.

Im Normalfall ist die Angabe von weiteren Netzwerkrouten nicht erforderlich.

IP_ROUTE_x Die zusätzlichen Routen `IP_ROUTE_1`, `IP_ROUTE_2`, ... haben folgenden Aufbau:

```
network/netmaskbits gateway
```

Dabei ist `network` die Netzwerkadresse, `/netmaskbits` die Netzmaske in CIDR (Seite 42) Schreibweise und `gateway` die Adresse des Rechners, der die Verbindung zu dem Netzwerk herstellt. Der Gateway-Rechner muss natürlich im gleichen Netzwerk, wie der fli4l-Router liegen! Ist z.B. das Netzwerk 192.168.7.0 mit der Netzwerkmaske 255.255.255.0 über das Gateway 192.168.6.99 erreichbar, dann lautet der Eintrag:

```
IP_ROUTE_N='1'
IP_ROUTE_1='192.168.7.0/24 192.168.6.99'
```

Wenn der fli4l-Router nicht als Internet-Router eingesetzt wird sondern nur als reiner Ethernetrouter kann man in einem IP_ROUTE_x Eintrag eine Default-Route angeben. Dazu wird analog zu der network/netmaskbits Schreibweise 0.0.0.0/0 eingetragen, so wie im Beispiel zu sehen ist.

```
IP_ROUTE_N='3'
IP_ROUTE_1='192.168.1.0/24 192.168.6.1'
IP_ROUTE_2='10.73.0.0/16 192.168.6.1'
IP_ROUTE_3='0.0.0.0/0 192.168.6.99'
```

3.10. Der Paketfilter

Der von fli4l verwendete Linux-Kern stellt einen Paketfilter zur Verfügung. Mit Hilfe dieses Paketfilters wird gesteuert, wer mit dem Router bzw. über ihn hinweg kommunizieren darf. Weiterhin können Dinge wie Port-Weiterleitung (ein an den Router gerichtetes Paket wird an einen anderen internen Rechner weitergereicht) und Maskierung (engl. “Masquerading”; Pakete, die von einem Rechner hinter dem Router kommen, werden so verändert, dass sie so aussehen, als kämen sie vom Router selbst) realisiert werden.

Die Struktur des Paketfilters ist in Abbildung 3.1 angedeutet. Pakete kommen über eine Netzwerk-Schnittstelle herein und durchlaufen die `PREROUTING`-Kette (eng. “chain”). Hier werden die an den Router gerichteten Pakete an einen anderen Rechner weitergereicht, indem die Zieladresse und der Zielport manipuliert werden. Ist das Paket an den Router gerichtet, wird es an die `INPUT`-Kette, andernfalls an die `FORWARD`-Kette weitergereicht. Beide Ketten prüfen, ob das Paket zulässig ist. Wird das Paket akzeptiert, wird es an den lokalen Zielprozess zugestellt oder über die `POSTROUTING`-Kette (in der das Maskieren von Paketen stattfindet) an diejenige Netzwerk-Schnittstelle weitergereicht, über die das Paket sein Ziel erreichen kann. Lokal generierte Pakete werden in der `OUTPUT`-Kette gefiltert und schließlich (falls erfolgreich) über die `POSTROUTING`-Kette ebenfalls an die korrekte Netzwerk-Schnittstelle weitergereicht.

Mit der Paketfilterkonfiguration lassen sich die einzelnen Ketten des Paketfilters direkt konfigurieren. Dazu gibt es für jede relevante Kette ein eigenes Array, d.h. eine für die `INPUT`-Kette (`PF_INPUT_%`), eine für die `FORWARD`-Kette (`PF_FORWARD_%`), eine für die `OUTPUT`-Kette (`PF_OUTPUT_%`), eine für die `PREROUTING`-Kette, in der die Portweiterleitung durchgeführt wird (`PF_PREROUTING_%`), und eine für die `POSTROUTING`-Kette, in der das Maskieren der Pakete durchgeführt wird (`PF_POSTROUTING_%`).

Ein Eintrag in einem dieser Arrays besteht im Wesentlichen aus einer Aktion (s.u.), die durch zusätzliche Bedingungen eingeschränkt werden kann. Diese Bedingungen beziehen sich auf

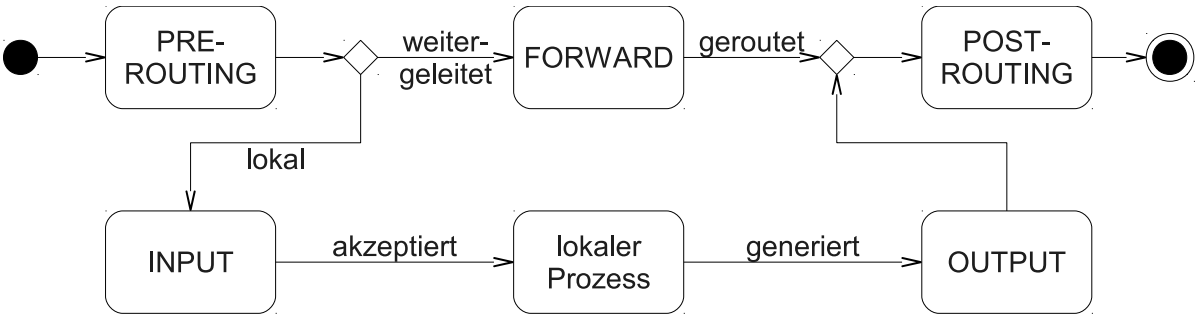


Abbildung 3.1.: Struktur des Paketfilters

Eigenschaften des betrachteten Paketes. Ein Paket enthält Informationen über seine Herkunft (Quelle, welcher Rechner hat das Paket losgeschickt), sein Ziel (an welchen Rechner und welche Anwendung soll das Paket gehen) u.a.m. Bedingungen können sich auf folgende Eigenschaften eines Paketes beziehen:

- die Quelle (Quell-Adresse, Quell-Port oder beides)
- das Ziel (Ziel-Adresse, Ziel-Port oder beides)
- das Protokoll
- die Schnittstelle, über welches das Paket hereinkommt bzw. hinausgeht
- die MAC-Adresse des Rechners, von dem das Paket kommt
- den Zustand des Paketes bzw. der Verbindung, zu der das Paket gehört

Kommt ein Paket herein, werden die Einträge bzw. die daraus generierten Regeln von oben nach unten abgearbeitet und die erste Aktion ausgeführt, bei der alle Bedingungen gelten. Trifft keine der Regeln zu, wird die Standardaktion ausgeführt, die man für (fast) jede Tabelle angeben kann.

Ein Eintrag hat dabei folgendes Format, wobei zu beachten ist, dass alle Einschränkungen optional sind:

```
restriction{0,} [[source] [destination]] action [BIDIRECTIONAL|LOG|NOLOG]
```

An allen Stellen, an denen Netzwerke, IP-Adressen oder Hosts angegeben werden müssen, kann man sich auch auf `IP_NET_%`, `IP_NET_%_IPADDR` oder via `@hostname` auf einen Host aus `HOST_%` beziehen. Ist `OPT_DNS` aktiviert, dann können außerhalb von Aktionen via `@fqdn` auch Hosts über ihren Namen referenziert werden, die *nicht* in `HOST_%` zu finden sind. Das ist insbesondere dann sinnvoll, wenn es sich um externe Hosts handelt, die zudem viele (und wechselnde) IP-Adressen besitzen.

3.10.1. Aktionen des Paketfilters

Aktionen können die folgenden sein:

Aktion	Kette(n)	Bedeutung
ACCEPT	alle	Akzeptiere das Paket.

DROP	`INPUT`
	`FORWARD`
	`OUTPUT`
		Verwirf das Paket (der Absender erkennt das nur daran, dass keine Antwort, aber auch keine Fehlermeldung zurückkommt).

REJECT	`INPUT`
	`FORWARD`
	`OUTPUT`
		Weise das Paket zurück (der Absender erhält eine entsprechende Fehlermeldung).

LOG	alle	Protokolliere das Paket und gehe zur nächsten Regel. Um verschiedene Protokoll-Einträge auseinanderhalten zu können, kann ein Präfix verwendet werden, das via LOG:log-prefix angegeben wird. Dieses Präfix darf maximal 28 Zeichen lang sein und kann aus Buchstaben, Ziffern, dem Bindestrich (-) und dem Unterstrich (_) bestehen.
MASQUERADE	`POSTROUTING`	Maskiere das Paket: Ersetze die Quelladresse des Paketes durch die eigene, der Schnittstelle zugewiesenen Adresse und sorge dafür, dass Antworten für diese Verbindung an den richtigen Rechner weitergeleitet werden.
SNAT	`POSTROUTING`	Ersetze die Quelladresse und den Quellport des Paketes durch die als Parameter für SNAT angegebene Adresse (für alle Pakete, die zu der gerade betrachteten Verbindung gehören).
DNAT	`PREROUTING`	Ersetze die Zieladresse und den Zielport des Paketes durch die als Parameter für DNAT angegebene Adresse (für alle Pakete, die zu der gerade betrachteten Verbindung gehören).
REDIRECT	`PREROUTING` `OUTPUT`	Ersetze den Zielport des Paketes durch den als Parameter für REDIRECT angegebenen Port und stelle das Paket lokal zu (für alle Pakete, die zu der gerade betrachteten Verbindung gehören).
NETMAP	`PREROUTING` `POSTROUTING`	Bilde die Ziel- bzw. Quelladresse des Paketes in den als Parameter für NETMAP angegebenen Bereich ab, die Ports bleiben unverändert (für alle Pakete, die zu der gerade betrachteten Verbindung gehören; in der `PREROUTING`-Kette wird die Zieladresse verändert, in der `POSTROUTING`-Kette die Quelladresse).

Tabelle 3.5.: Aktionen in Paketfilterregeln

Einige dieser Aktionen können durch die Optionen BIDIRECTIONAL, LOG oder NOLOG in ihrem Verhalten modifiziert werden. BIDIRECTIONAL generiert die gleiche Regel noch einmal,

nur mit vertauschter Quell- und Zieladresse (und vertauschtem Quell- und Zielport und/oder vertauschter ein- und ausgehender Netzwerk-Schnittstelle, falls angegeben). LOG/NOLOG aktivieren bzw. deaktivieren das Protokollieren für diese eine Regel.

3.10.2. Einschränkungen in den Regeln

Einschränkungen können durch die in den folgenden Abschnitten aufgeführten Bedingungen vorgenommen werden. Bei den Bedingungen kann man immer `any` angeben, wenn man an irgendeiner Stelle keine Einschränkung vornehmen will, aber trotzdem etwas angeben will/muss. Einschränkungen können in beliebiger Reihenfolge angegeben werden, wenn sie einen vorangestellten Präfix haben. Das gilt für alle Einschränkungen, außer für die Angabe einer Quell- bzw. Zieladresse. Diese müssen immer direkt vor der Aktion stehen, die anderen Einschränkungen müssen vorher erfolgen. Einschränkungen können auch negiert werden, dazu wird einfach ein `!` vorangestellt.

Einschränkungen der Quelle und des Ziels

Jedes Paket enthält eine Quell- und eine Zielangabe, jeweils in Form eines Tupels einer IP-Adresse und eines Ports.[2] Diese Quelle bzw. dieses Ziel kann für eine Einschränkung herangezogen werden. Die Angabe für die Quelle bzw. das Ziel kann folgendermaßen vorgenommen werden:

Ausdruck	Bedeutung
`ip`	eine einfache IP-Adresse
`network`	eine Netzwerkangabe der Form `<ip>/<netmask>`
`port[-port]`	ein Port bzw. ein Port-Bereich
`IP_NET_x_IPADDR`	die IP-Adresse der Schnittstelle `x` des Routers
`IP_NET_x`	das Subnetz `x` des Routers
`IP_ROUTE_x`	das in der Route `x` angegebene Subnetz (Default-Routen können nicht verwendet werden, sie würden `any` entsprechen und werden vorsichtshalber ausgeklammert)
`@name`	einer der via HOST_%_* vergebenen Namen oder Aliase; es wird die zugehörige IP-Adresse an dieser Stelle eingesetzt
`<ip oder netzwerk>:port[-port]`	Host- bzw. Netzwerk-Adresse in einer der obigen Varianten, kombiniert mit einem Port bzw. Port-Bereich

Tabelle 3.6.: Quell- und Zieleinschränkungen in Paketfilterregeln

Das könnte z. B. wie folgt aussehen: `'192.168.6.2 any DROP'`

[2] Ein Port ist nur bei TCP- und UDP-Paketen vorhanden.

Tauchen zwei dieser Angaben auf, wird die erste als Quelle, die zweite als Ziel betrachtet. In diesem Beispiel verwerfen wir also Pakete, die vom Rechner mit der IP-Adresse 192.168.6.2 gesendet wurden, unabhängig davon, an welches Ziel sie gerichtet sind.

Taucht nur eine Angabe auf, wird anhand des Wertes entschieden, ob die Quelle oder das Ziel gemeint ist, wobei die Entscheidung relativ einfach ist:

- Ist eine Port-Angabe enthalten, ist das Ziel gemeint.
- Sonst ist die Quelle gemeint.

Wenn wir also z. B. das obige Beispiel abkürzen wollten, könnten wir einfach `'192.168.6.2 DROP'` schreiben. Es ist kein Port angegeben, die Bedingung gilt also für die Quelle, den Rechner, von dem das Paket gesendet wurde.

Wollen wir die Kommunikation mit dem `ssh`-Dämon erlauben, können wir `'any any:22 ACCEPT'` (Pakete von einem beliebigen Rechner an den `ssh`-Port 22 eines beliebigen Rechners werden akzeptiert) oder kürzer `'22 ACCEPT'` schreiben: Es ist nur ein Port angegeben, also meinen wir das Ziel und damit Pakete, die an den Port 22 gerichtet sind.

Zur Vereinfachung der Regelmenge kann man an die Aktion ein BIDIRECTIONAL dranhängen, um auszudrücken, dass die Regel für beide Kommunikationsrichtungen gilt. Es werden dann Regeln generiert, in denen einfach die Quell- und die Ziel-Adressen und eventuell angegebenene Ports und Netzwerk-Schnittstellen vertauscht sind und der Rest gleich bleibt.

Beispiele:

`127.0.0.1 ACCEPT`	Lokale Kommunikation (Quelle 127.0.0.1) ist erlaubt
`any 192.168.12.1 DROP`	Pakete an die Adresse 192.168.12.1 werden weggeworfen
`any 192.168.12.1 DROP LOG`	Pakete an die Adresse 192.168.12.1 werden weggeworfen und zusätzlich protokolliert
`any 192.168.12.1 DROP NOLOG`	Pakete an die Adresse 192.168.12.1 werden weggeworfen, werden aber nicht protokolliert
`22 ACCEPT`	Pakete an den Port 22 (`ssh`) werden akzeptiert
`IP_NET_1_NET ACCEPT`	Pakete aus dem an der ersten Schnittstelle hängenden Subnetz werden akzeptiert
`IP_NET_1_NET IP_NET_2_NET ACCEPT BIDIRECTIONAL`	Kommunikation zwischen den an der ersten und zweiten Schnittstelle hängenden Subnetzen ist gestattet

Einschränkung der Schnittstelle

Eine Regel kann eingeschränkt werden in Bezug auf die Schnittstelle, über die ein Paket hereinkam bzw. hinausgeht. Das Format sieht wie folgt aus: `if:`*in*`:`*out*

In der `INPUT`-Kette kann man die Schnittstelle für hinausgehende Pakete nicht einschränken (das Paket geht ja nicht mehr hinaus), in der `POSTROUTING`-Kette kann man die Schnittstelle für hereinkommende Pakete nicht einschränken, da die Information darüber nicht mehr vorhanden ist. Lediglich in der `FORWARD`-Kette kann man für beides Bedingungen angeben.

Möglich sind folgende Werte für *in* bzw. *out*:

- `lo` (Loopback-Schnittstelle, lokale Kommunikation auf dem Router)
- `IP_NET_x_DEV`
- `pppoe` (die PPPoE-Schnittstelle; nur bei entsprechend aktiviertem `dsl`- oder `pppoe_server`-Paket)
- `any`

Einschränkungen des Protokolls

Eine Regel kann eingeschränkt werden in Bezug auf das Protokoll, zu dem ein Paket gehört. Das Format sieht wie folgt aus: `prot:`*protocol* bzw. `prot:`*icmp*`:`*icmp-type*. *protocol* kann dabei einen der folgenden Werte annehmen:

- `tcp`
- `udp`
- `gre` (Generic Routing Encapsulation)
- `icmp` (hier kann man zusätzlich noch einen Namen für den zu filternden ICMP-Typ angeben (`echo-reply` oder `echo-request`), etwa `prot:icmp:echo-request`)
- numerischer Wert der Protokoll-ID (z. B. 41 für IPv6)
- `any`

Wenn eine solche Einschränkung nicht vorhanden ist, aber dennoch Portnummern in einer Regel verwendet werden, dann wird die Regel *zweimal* angelegt, nämlich einmal für das `tcp`- und einmal für das `udp`-Protokoll.

Einschränkung der MAC-Adresse

Mittels `mac:`*mac-address* kann eine Einschränkung bezüglich der MAC-Adresse vorgenommen werden.

Einschränkungen in Bezug auf den Zustand eines Paketes

Der von fli4l verwendete Paketfilter sammelt Informationen über den Zustand von Verbindungen. Diese Informationen kann man dann nutzen, um Pakete zu filtern, also z. B. nur Pakete durchzulassen, die zu bereits bestehenden Verbindungen gehören. Die Zustände einer Verbindung können sein:[3]

Zustand	Bedeutung
`INVALID`	Das Paket gehört zu keiner bekannten Verbindung.
`ESTABLISHED`	Das Paket gehört zu einer Verbindung, über die bereits in beide Richtungen Pakete geflossen sind.
`NEW`	Das Paket hat eine neue Verbindung aufgebaut oder gehört zu einer Verbindung, bei der noch nicht Pakete in beide Richtungen geflossen sind.
`RELATED`	Das Paket baut eine neue Verbindung auf, hat aber eine Beziehung zu einer bestehenden Verbindung (z. B. baut `ftp` eine separate Verbindung für den eigentlichen Datentransfer auf).

Tabelle 3.7.: Zustandseinschränkungen in Paketfilterregeln

[3] siehe http://www.sns.ias.edu/~jns/files/iptables_talk/x38.htm für eine genauere Beschreibung der Zustände

Die Zustände werden wie folgt angegeben: `state:` *state(s)*. Will man mehrere Zustände angeben, werden diese mit Komma getrennt. Um z. B. nur Pakete durchzulassen, die direkt oder indirekt zu bestehenden Verbindungen gehören, kann man `state:ESTABLISHED,RELATED` schreiben (dies ist in der `INPUT`- oder `FORWARD`-Kette sinnvoll).

Einschränkung der Häufigkeit einer Aktion

Unter bestimmten Umständen möchte man die Häufigkeit von Aktionen begrenzen, z. B. nur eine ICMP-Echo-Anforderung pro Sekunde zulassen. Das kann man mit der `limit`-Einschränkung erreichen, die wie folgt aussieht: `limit:` *Häufigkeit:Burst*. Die Häufigkeit wird dabei als *n/Zeiteinheit* (second, minute, hour, day) angegeben, wobei zusätzlich noch Ereignisse gehäuft auftreten können (Burst). Die Angabe `limit:3/minute:5` heißt z. B., dass höchstens drei Ereignisse pro Minute erlaubt sind, wobei auch mal fünf Ereignisse dicht aufeinander folgend akzeptiert werden.

3.10.3. Der Einsatz von Schablonen im Paketfilter

Um den Umgang mit dem Paketfilter weiter zu vereinfachen, gibt es die Möglichkeit, häufig vorkommende Regeln zu sogenannten Schablonen (Templates) zusammenzufassen. Damit ist es möglich, eine ganze Reihe von Paketfilterregeln zusammenzufassen und dieser Sammlung von Regeln einen symbolischen Namen zu geben. Anstatt direkt mit Protokollen und Portnummern zu hantieren, verwenden Sie dann Einträge wie `tmpl:ssh`, wenn Sie das `ssh`-Protokoll in einer Regel verwenden wollen. Wie mit Schablonen zu verfahren ist, wird hier am Beispiel von `ssh` gezeigt.

Wollen Sie Ihren fli4l-Router vom Internet aus per `ssh` erreichen, so schreiben Sie in einen Eintrag der Array-Variable `PF_INPUT_%` den entsprechenden Dienstnamen (hier `ssh`) mit vorangestelltem `tmpl:` und der Aktion, die für diesen Dienst gelten soll. Beispiel:

```
PF_INPUT_2='tmpl:ssh ACCEPT'
```

Hierbei steht `tmpl:` dafür, dass eine Regel auf einer Schablone basieren soll. Den Namen des Dienstes geben Sie nach dem ‘:’ an, in unserem Beispiel also `ssh`. Zum Schluss geben Sie an, welche Aktion mit dem Dienst verbunden werden soll. Da Sie den fli4l-Router aus dem Internet erreichen wollen, erlauben wir die Verbindung mit `ACCEPT`. Einschränkungen von IP-Adressen oder Netzen sind nicht angegeben, also ist der `ssh`-Dienst von allen Netzen und über alle Schnittstellen erreichbar. Sie könnten bei Bedarf die gewohnten Schreibweisen vom Paketfilter benutzen, um den Zugriff auf den `ssh`-Dienst weiter einzuschränken.

Für welche Dienste Regeln vorbereitet sind (d.h. Schablonen existieren), kann in der Schablonen-Datei in `opt/etc/fwrules.tmpl/templates` nachgelesen werden. Im Folgenden finden Sie die Aufstellung in Tabellenform (siehe Tabelle 3.8).

Schablone	Protokoll	Port(s)
dhcp	udp	67-68
dns	tcp/udp	53

Schablone	Protokoll	Port(s)
elster	tcp	159.154.8.2:21
elster	tcp	159.154.8.35:21
elster	tcp	193.109.238.26:8000
elster	tcp	193.109.238.27:8000
elster	tcp	193.109.238.58:80
elster	tcp	193.109.238.59:80
elster	tcp	62.157.211.58:8000
elster	tcp	62.157.211.59:8000
elster	tcp	62.157.211.60:8000
elster	tcp	80.146.179.2:80
elster	tcp	80.146.179.3:80
ftp	tcp	21
http	tcp	80
https	tcp	443
hylafax	tcp	4559
imap	tcp	143
imaps	tcp	993
imond	tcp	5000
ipmi	tcp	22
ipmi	tcp	2937
ipmi	tcp	443
ipmi	tcp	5120
ipmi	tcp	5123
ipmi	tcp	5900
ipmi	tcp	5901
ipmi	tcp	80
ipmi	tcp	8889
ipmi	udp	623
irc	tcp	6667
ldap	tcp/udp	389
mail	tcp	110
mail	tcp	143
mail	tcp	25
mail	tcp	465
mail	tcp	993
mail	tcp	995
mysql	tcp	3306
nfs	tcp/udp	111
nfs	tcp/udp	2049
nntp	tcp	119
ntp	udp	123
oracle	tcp	1521
pcanywhere	tcp	5631-5632
ping	icmp:0	
ping	icmp:8	
pop3	tcp	110
pop3s	tcp	995
privoxy	tcp	8118
proxmox	tcp	8006
proxmox	tcp	5900
proxmox	tcp	3128
rdp	tcp	3389
rsync	tcp	873
samba	tcp	139
samba	tcp	445

Schablone	Protokoll	Port(s)
samba	udp	137-138
sip	tcp/udp	5060-5061
smtp	tcp	25
snmp	tcp/udp	161
socks	tcp	1080
squid	tcp	3128
ssh	tcp	22
ssmtp	tcp	465
submission	tcp	587
svn	tcp	3690
syslog	udp	514
teamspeak	tcp	14534
teamspeak	tcp	51234
teamspeak	udp	8767
telmond	tcp	5001
telnet	tcp	23
teredo	udp	3544
tftp	udp	69
time	tcp/udp	37
traceroute	udp	33404-33464
vdr	tcp	6419
vnc	tcp	5900
whois	tcp	43
xbl	tcp/udp	3074
xbl	udp	88
xmppclient	tcp	5222
xmppserver	tcp	5269

Tabelle 3.8.: Im Lieferumfang von fli4l enthaltene Schablonen

Die Syntax für diese Form der Paketfilterregeln lautet also immer

```
tmpl:<Name des Dienstes> <Einschränkungen> <Gewünschte Aktion>
```

wobei als `<Einschränkungen>` alles erlaubt ist, was unter 3.10.2 beschrieben wird. Die möglichen Werte für `<Gewünschte Aktion>` sind in 3.10.1 aufgelistet und beschrieben.

Ein paar weitere Beispiele sollen die Arbeitsweise verdeutlichen. Zuerst wollen wir uns `PF_PREROUTING` ansehen:

```
PF_PREROUTING_N='2'
PF_PREROUTING_1='tmpl:xbl dynamic DNAT:@xbox'
PF_PREROUTING_2='tmpl:https dynamic DNAT:192.168.193.250'
```

Die Regel `PF_PREROUTING_1` versorgt die Xbox mit allem, was für Xbox Live notwendig ist. Im einzelnen werden mit `tmpl:xbl` alle Ports und Protokolle, die für Xbox Live notwendig sind, an den Host `xbox` weitergeleitet. Anstelle der IP-Adresse wird ein Eintrag aus dem `HOST_%_NAME`-Array benutzt. Durch `dynamic` weiß der fli4l, dass die Ports von der Internet-Schnittstelle weitergeleitet werden sollen.

Die zweite Regel leitet das `https`-Protokoll an einen Webserver in der DMZ weiter.

Jetzt sehen wir uns an, wie es mit `PF_INPUT` weitergeht:

```
PF_INPUT_N='3'
PF_INPUT_1='if:IP_NET_1_DEV:any ACCEPT'
PF_INPUT_2='if:pppoe:any prot:tcp 113 ACCEPT'
PF_INPUT_3='if:br0:any tmpl:dns @xbox IP_NET_1_IPADDR ACCEPT'
```

Die erste Regel lässt alle aus dem Netz, das mit `IP_NET_1` definiert ist, auf den Router zugreifen. Die zweite Regel ist für das `oident`-Paket. Dort wird der `ident`-Port geöffnet. Die dritte und letzte Regel erlaubt der Xbox den Zugriff auf den DNS Server auf dem fli4l. Hier ist auch wieder schön zu sehen, wie man einen Host-Alias einsetzt.

In `PF_FORWARD` und `PF_POSTROUTING` steht nichts `tmpl`-spezifisches mehr.

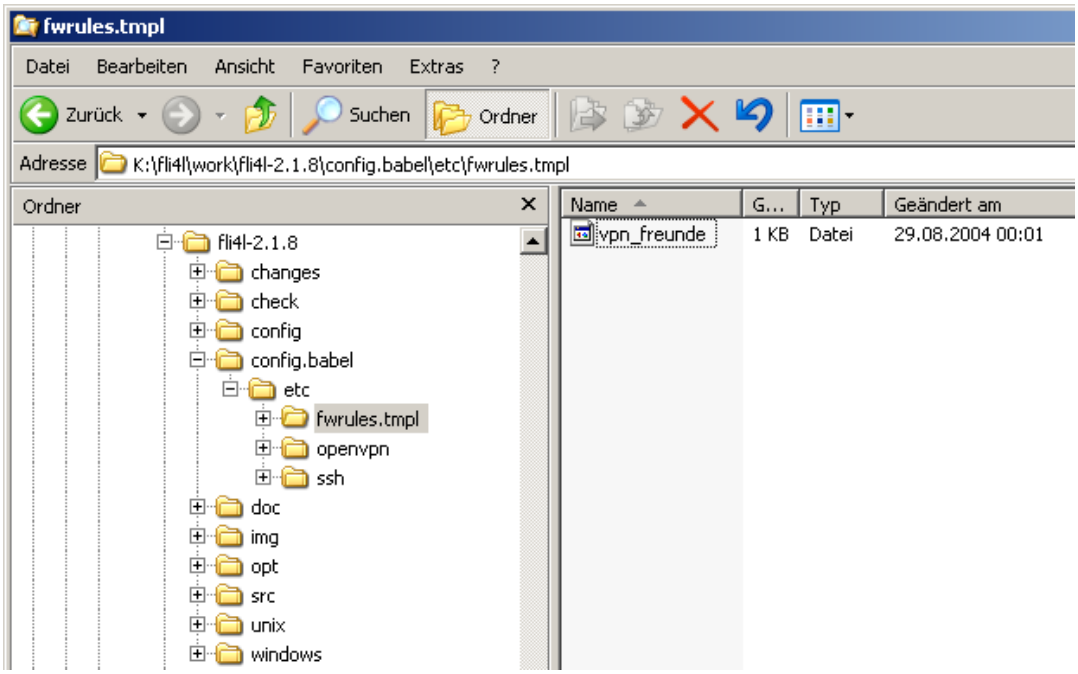


Abbildung 3.2.: Verzeichnisstruktur fli4l

Es ist auch möglich, dass Sie eigene Schablonen anlegen oder dass Pakete ihre eigenen Schablonen mitbringen. Um eine eigene Schablone anzulegen, muss lediglich eine Datei mit den Namen der Schablone erstellt und die entsprechenden Regeln dort aufgenommen werden. Wenn Sie eine private Schablonendatei anlegen wollen, erstellen Sie diese in dem Verzeichnis `etc/fwrules.tmpl` unterhalb Ihres `config`-Verzeichnisses, so wie es die Abbildung 3.2 zeigt. Wenn das Verzeichnis `etc/fwrules.tmpl` unterhalb Ihres `config`-Verzeichnisses noch nicht existiert, legen Sie bitte zuerst beide Verzeichnisse an. Alternativ können Paket-Entwickler oder Benutzer, die Schablonen für mehr als eine Konfiguration anlegen wollen, ihre Regeln direkt in das Verzeichnis `opt/etc/fwrules.tmpl` ablegen. In dieses Verzeichnis kommen dann die neuen Schablonen. Dabei gilt die Regel, dass die Schablonen im `config`-Verzeichnis des Benutzers vorrangig behandelt werden. Zum Schluss wird die Schablonendatei, die zum Lieferumfang von fli4l gehört, ausgewertet. Sie können also Einträge in der fli4l-Schablonendatei dadurch „überschreiben“, indem Sie eine Schablonendatei mit dem Namen der zu überschreibenden Schablone in Ihrem `config`-Verzeichnis anlegen.

Wenn Sie zum Beispiel die Schablone `vpn_freunde` anlegen wollen, legen Sie die Datei

vpn_freunde an. Das Template soll die Dienste ssh, smtp, dns und samba enthalten. Also schreiben Sie in die Datei vpn_freunde Folgendes:

```
prot:tcp 22
prot:tcp 25
53
prot:udp 137-138
prot:tcp 139
prot:tcp 445
```

Wann immer Sie jetzt die Schablone vpn_freunde benutzen, werden daraus Regeln für alle darin aufgeführten Protokolle und Ports erzeugt. PF_FORWARD_x='tmpl:vpn_freunde ACCEPT' etwa erstellt folgende FORWARD-Regeln:

```
prot:tcp 22 ACCEPT
prot:tcp 25 ACCEPT
53 ACCEPT
prot:udp 137-138 ACCEPT
prot:tcp 139 ACCEPT
prot:tcp 445 ACCEPT
```

3.10.4. Die Konfiguration des Paketfilters

Der Paketfilter wird im Wesentlichen durch vier Array-Variablen konfiguriert:

- PF_INPUT_% konfiguriert die INPUT-Kette,
- PF_FORWARD_% konfiguriert die FORWARD-Kette,
- PF_OUTPUT_% konfiguriert die OUTPUT-Kette,
- PF_PREROUTING_% konfiguriert die PREROUTING-Kette und
- PF_POSTROUTING_% konfiguriert die POSTROUTING-Kette.

Für alle folgenden Ketten gilt die in PF_LOG_LEVEL vorgenommene Einstellung der Protokoll-Stufe, deren Inhalt auf einen der folgenden Werte gesetzt werden kann: debug, info, notice, warning, err, crit, alert, emerg.

Die INPUT-Kette

Über die INPUT-Kette wird konfiguriert, wer auf den Router zugreifen darf. Trifft keine der Regeln der INPUT-Kette zu, bestimmt die Standard-Aktion, was mit dem Paket passieren soll, und die Protokoll-Variable bestimmt, ob es bei einer Ablehnung ins System-Protokoll geschrieben werden soll.

Bei den verwendeten Parametern gibt es die folgenden Einschränkungen:

- Es können nur ACCEPT, DROP und REJECT als Aktion angegeben werden.
- Bei einer Schnittstellen-Einschränkung kann man nur die Eingangsschnittstelle einschränken.

PF_INPUT_POLICY Diese Variable beschreibt die Standard-Aktion, die angewandt wird, wenn keine der anderen Regeln zutrifft. Möglich sind:

- ACCEPT (nicht empfohlen)
- REJECT
- DROP (nicht empfohlen)

PF_INPUT_ACCEPT_DEF Steht diese Variable auf 'yes', werden Standard-Regeln generiert, die für ein korrektes Funktionieren des Routers notwendig sind. Standardmäßig sollte man hier 'yes' eintragen.

Möchte man das Verhalten komplett selbst definieren, kann man hier 'no' eintragen, muss dann jedoch alle Regeln selbst definieren. Eine zum Standardverhalten äquivalente Konfiguration würde wie folgt aussehen (die Beschreibung der Liste für benutzerdefinierte Ketten erfolgt hier (Seite 58)):

```
PF_INPUT_ACCEPT_DEF='no'
#
# limit ICMP echo requests - use a separate chain
#
PF_USR_CHAIN_N='1'
PF_USR_CHAIN_1_NAME='usr-in-icmp'
PF_USR_CHAIN_1_RULE_N='2'
PF_USR_CHAIN_1_RULE_1='prot:icmp:echo-request length:0-150 limit:1/second:5 ACCEPT'
PF_USR_CHAIN_1_RULE_2='state:RELATED ACCEPT'

PF_INPUT_N='4'
PF_INPUT_1='prot:icmp usr-in-icmp'
PF_INPUT_2='state:ESTABLISHED,RELATED ACCEPT'
PF_INPUT_3='if:lo:any ACCEPT'
PF_INPUT_4='state:NEW 127.0.0.1 DROP BIDIRECTIONAL'
```

Die erste Regel verzweigt zur ratenlimitierenden "usr-in-icmp"-Kette. Die zweite Regel akzeptiert nur solche Pakete, die zu bestehenden Verbindungen gehören (also Paketen, die entweder den Zustand `ESTABLISHED` oder `RELATED` besitzen), und die dritte erlaubt lokale Kommunikation (`if:lo:any ACCEPT`). Die vierte filtert Pakete heraus, die behaupten, lokale Kommunikation zu sein, aber nicht bereits von der vorherigen Regel akzeptiert wurden.

Arbeitet man mit OpenVPN, muss man die Regeln noch ergänzen, um die von diesen Paketen verwendeteten Ketten einzubinden.

```
PF_INPUT_N='5'
...
PF_INPUT_5='ovpn-chain'
```

PF_INPUT_LOG Definiert, ob abgelehnte Pakete vom Kernel protokolliert werden sollen. Dabei können die Meldungen durch Aktivierung von `OPT_KLOGD` über den syslog-Dämon entsprechend der Konfiguration ausgegeben werden.

PF_INPUT_LOG_LIMIT Definiert, wie häufig Log-Einträge generiert werden. Die Häufigkeit wird analog zur Limit-Einschränkung als *n/Zeiteinheit* mit Bursts beschrieben, also z. B. `3/minute:5`. Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enhält er `none`, wird keine Limitierung durchgeführt.

PF_INPUT_REJ_LIMIT PF_INPUT_UDP_REJ_LIMIT Definiert, wie häufig bei einer Ablehnung eines hereinkommenden Paketes auch ein entsprechendes REJECT-Paket generiert wird. Die Häufigkeit wird analog zur Limit-Einschränkung als *n/Zeiteinheit* mit Bursts beschrieben, also z. B. `3/minute:5`. Ist das Limit überschritten, wird das Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enhält er `none`, wird keine Limitierung durchgeführt.

PF_INPUT_ICMP_ECHO_REQ_LIMIT Definiert, wie häufig auf eine ICMP-Echo-Anfrage reagiert werden soll. Die Häufigkeit wird analog zur Limit-Einschränkung als *n/Zeiteinheit* mit Bursts beschrieben, also z. B. `/minute:5`. Ist das Limit überschritten, wird das Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enhält er `none`, wird keine Limitierung durchgeführt.

PF_INPUT_ICMP_ECHO_REQ_SIZE Definiert, wie groß eine empfangene ICMP-Echo-Anfrage sein darf (in Bytes). In dieser Angabe sind neben den "Nutzdaten" auch die Paket-Header mit zu berücksichtigen. Der Standard-Wert liegt bei 150 Bytes.

PF_INPUT_N PF_INPUT_x PF_INPUT_x_COMMENT Liste der Regeln, die beschreiben, welche Pakete vom Router angenommen bzw. verworfen werden.

Die FORWARD-Kette

Über die FORWARD-Kette wird konfiguriert, welche Pakete vom Router weitergeleitet werden. Trifft keine der Regeln der FORWARD-Kette zu, bestimmt die Standard-Aktion, was mit dem Paket passieren soll,und die Protokoll-Variable bestimmt, ob es bei einer Ablehnung ins System-Protokoll geschrieben werden soll.

Bei den verwendeten Parametern gibt es die Einschränkung, dass nur ACCEPT, DROP und REJECT als Aktion angegeben werden können.

PF_FORWARD_POLICY Diese Variable beschreibt die Standard-Aktion, die angewandt wird, wenn keine der anderen Regeln zutrifft. Möglich sind:

- ACCEPT
- REJECT
- DROP

PF_FORWARD_ACCEPT_DEF Bestimmt, ob der Router Pakete akzeptiert, die zu bestehenden Verbindungen gehören. Steht diese Variable auf 'yes', generiert fli4l automatisch eine Regel, die Pakete mit dem entsprechenden Zustand akzeptiert:

`'state:ESTABLISHED,RELATED ACCEPT'`,

weiterhin eine Regel, die Pakete mit unbekanntem Zustand verwirft:

`'state:INVALID DROP'`.

und schließlich eine Regel, die Pakete mit gefälschten IP-Adressen verwirft:

`'state:NEW 127.0.0.1 DROP BIDIRECTIONAL'`.

Zusätzlich generieren die anderen Subsysteme auch noch Standardregeln – eine Konfiguration ohne Standardregeln mit Portweiterleitung und OpenVPN würde mindestens folgende Regeln enthalten:

```
PF_FORWARD_ACCEPT_DEF='no'
PF_FORWARD_N='5'
PF_FORWARD_1='state:ESTABLISHED,RELATED ACCEPT'
PF_FORWARD_2='state:INVALID DROP'
PF_FORWARD_3='state:NEW 127.0.0.1 DROP BIDIRECTIONAL'
PF_FORWARD_4='pfwaccess-chain'
PF_FORWARD_5='ovpn-chain'
```

PF_FORWARD_LOG Definiert, ob abgelehnte Pakete vom Kernel protokolliert werden sollen. Dabei können die Meldungen durch Aktivierung von `OPT_KLOGD` über den syslog-Dämon entsprechend der Konfiguration ausgegeben werden.

PF_FORWARD_LOG_LIMIT Definiert, wie häufig Log-Einträge generiert werden. Die Häufigkeit wird analog zur Limit-Einschränkung als *n/Zeiteinheit* mit Bursts beschrieben, also z. B. `3/minute:5`. Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enhält er `none`, wird keine Limitierung durchgeführt.

PF_FORWARD_REJ_LIMIT PF_FORWARD_UDP_REJ_LIMIT Definiert, wie häufig bei einer Ablehnung eines hereinkommenden Paketes auch ein entsprechendes REJECT-Paket generiert wird. Die Häufigkeit wird analog zur Limit-Einschränkung als *n/Zeiteinheit* mit Bursts beschrieben, also z. B. `3/minute:5`. Ist das Limit überschritten, wird das Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enhält er `none`, wird keine Limitierung durchgeführt.

PF_FORWARD_N PF_FORWARD_x PF_FORWARD_x_COMMENT Liste der Regeln, die beschreiben, welche Pakete vom Router weitergeleitet bzw. verworfen werden.

Die OUTPUT-Kette

Über die OUTPUT-Kette wird konfiguriert, worauf der Router selbst zugreifen darf. Trifft keine der Regeln der OUTPUT-Kette zu, bestimmt die Standard-Aktion, was mit dem Paket passieren soll, und die Protokoll-Variable bestimmt, ob es bei einer Ablehnung ins System-Protokoll geschrieben werden soll.

Bei den verwendeten Parametern gibt es die folgenden Einschränkungen:

- Es können nur ACCEPT, DROP und REJECT als Aktion angegeben werden.
- Bei einer Schnittstellen-Einschränkung kann man nur die Ausgangsschnittstelle einschränken.

PF_OUTPUT_POLICY Diese Variable beschreibt die Standard-Aktion, die angewandt wird, wenn keine der anderen Regeln zutrifft. Möglich sind:

- ACCEPT
- REJECT
- DROP

PF_OUTPUT_ACCEPT_DEF Steht diese Variable auf 'yes', werden Standard-Regeln generiert, die für ein korrektes Funktionieren des Routers notwendig sind. Standardmäßig sollte man hier 'yes' eintragen.

Möchte man das Verhalten komplett selbst definieren, kann man hier 'no' eintragen, muss dann jedoch alle Regeln selbst definieren. Eine zum Standardverhalten äquivalente Konfiguration würde wie folgt aussehen:

```
PF_OUTPUT_ACCEPT_DEF='no'

PF_OUTPUT_N='1'
PF_OUTPUT_1='state:ESTABLISHED,RELATED ACCEPT'
```

Die erste (und einzige) Regel akzeptiert nur solche Pakete, die zu bestehenden Verbindungen gehören (also Paketen, die entweder den Zustand `ESTABLISHED` oder `RELATED` besitzen).

PF_OUTPUT_LOG Definiert, ob abgelehnte Pakete vom Kernel protokolliert werden sollen. Dabei können die Meldungen durch Aktivierung von `OPT_KLOGD` über den syslog-Dämon entsprechend der Konfiguration ausgegeben werden.

PF_OUTPUT_LOG_LIMIT Definiert, wie häufig Log-Einträge generiert werden. Die Häufigkeit wird analog zur Limit-Einschränkung als *n/Zeiteinheit* mit Bursts beschrieben, also z. B. `3/minute:5`. Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enhält er `none`, wird keine Limitierung durchgeführt.

PF_OUTPUT_REJ_LIMIT PF_OUTPUT_UDP_REJ_LIMIT Definiert, wie häufig bei einer Ablehnung eines hereinkommenden Paketes auch ein entsprechendes REJECT-Paket generiert wird. Die Häufigkeit wird analog zur Limit-Einschränkung als *n/Zeiteinheit* mit Bursts beschrieben, also z. B. `3/minute:5`. Ist das Limit überschritten, wird das Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enhält er `none`, wird keine Limitierung durchgeführt.

PF_OUTPUT_N PF_OUTPUT_x PF_OUTPUT_x_COMMENT Liste der Regeln, die beschreiben, welche Pakete vom Router versandt bzw. verworfen werden.

Benutzerdefinierte Listen

Aus verschiedenen Gründen besteht manchmal der Bedarf, eigene Ketten anzulegen und dort die Pakete genauer zu filtern. Diese Ketten kann man mittels `PF_USR_CHAIN_%` definieren und mit Regeln füllen. Die Namen der Ketten müssen dabei mit *usr-* beginnen und können nach ihrer Definition überall in der `INPUT`- oder `FORWARD`-Kette statt einer Aktion eingesetzt werden. Als Beispiel soll hier die bereits vorher verwendete ICMP-Filterkette dienen:

```
PF_USR_CHAIN_N='1'
#
# create usr-in-icmp
#
PF_USR_CHAIN_1_NAME='usr-in-icmp'
#
# add rule to usr-in-icmp
#
PF_USR_CHAIN_1_RULE_N='2'
PF_USR_CHAIN_1_RULE_1='prot:icmp:echo-request length:0-150 limit:1/second:5 ACCEPT'
PF_USR_CHAIN_1_RULE_2='state:RELATED ACCEPT'
#
# use chain in PF_INPUT
#
PF_INPUT_2='prot:icmp usr-in-icmp'
```

PF_USR_CHAIN_N Definiert die Anzahl der benutzerdefinierten Ketten.

PF_USR_CHAIN_x_NAME Definiert den Namen der benutzerdefinierten Kette. Dieser muss mit *usr-* beginnen.

PF_USR_CHAIN_x_RULE_N

PF_USR_CHAIN_x_RULE_x

PF_USR_CHAIN_x_RULE_x_COMMENT Hier werden die Regeln definiert, die in die benutzerdefinierte Kette eingefügt werden sollen. Es können alle Regeln verwendet werden, die auch in einer `FORWARD`-Kette verwendet werden könnten. Sollte keine Regel der benutzerdefinierten Kette zutreffen, wird zur Ausgangskette zurückgekehrt und mit der Regel nach der Verzweigung fortgefahren.

Die NAT-Ketten (Network Address Translation)

Pakete können vor und nach Routing-Entscheidungen noch manipuliert werden. Sie können zum Beispiel eine neue Zieladresse erhalten, um an einen anderen Rechner weitergeleitet zu werden (Portweiterleitung) oder eine andere Quelladresse erhalten, um das hinter dem Router liegende Netzwerk zu maskieren. Maskieren nutzt man beispielsweise, um ein privates Netz über eine öffentliche IP ins Netz zu bringen oder in einem DMZ-Setup die Struktur des lokalen Netzes vor den Rechnern in der DMZ zu verbergen.

Die Konfiguration erfolgt über zwei Ketten, die `PREROUTING`- und die `POSTROUTING`-Kette. Über die `POSTROUTING`-Kette wird konfiguriert, welche Pakete vom Router maskiert werden. Trifft keine der Regeln der `POSTROUTING`-Kette zu, werden die Pakete unmaskiert weitergeleitet.

Beim Maskieren gibt es zwei Varianten: eine für Netzwerk-Schnittstellen, die bei der Einwahl erst eine IP-Adresse zugewiesen bekommen (MASQUERADE) und eine für Netzwerk-Schnittstellen mit statischer IP-Adresse (SNAT). SNAT erwartet dabei zusätzlich die IP-Adresse, die im Paket als Quelle eingetragen werden soll. Diese kann als

- IP-Adresse (Beispiel: SNAT:1.2.3.4),
- IP-Bereich (Beispiel: SNAT:1.2.3.4-1.2.3.10)
- oder als symbolische Referenz (Beispiel: SNAT:IP_NET_1_IPADDR)

angegeben werden.

Sowohl bei SNAT als auch bei MASQUERADE kann schließlich ein Port bzw. Portbereich angegeben werden, auf den der Quellport abgebildet werden soll. Normalerweise ist das nicht nötig, da der Kern die Ports allein auswählen kann. Es gibt aber Anwendungen, die verlangen, dass der Quellport unverändert bleibt (und somit ein 1:1-NAT erfordern) oder die ein PAT (Port Address Translation) oder NAPT (Network Address and Port Translation) verbieten. Der Portbereich wird einfach hinten angehängt, z. B. so: SNAT:IP_NET_1_IPADDR:4000-8000.

Bei der POSTROUTING-Kette können nur ACCEPT, SNAT, NETMAP und MASQUERADE als Aktionen verwendet werden.

PF_POSTROUTING_N PF_POSTROUTING_x PF_POSTROUTING_x_COMMENT

Eine Liste der Regeln, die beschreiben, welche Pakete vom Router maskiert werden (bzw. unmaskiert weitergeleitet werden). Will man Pakete vom Maskieren ausklammern, kann man eine ACCEPT-Regel für die auszuklammernden Pakete der MASQUERADE-Regel voranstellen.

Über die PREROUTING-Kette wird konfiguriert, welche Pakete an einen anderen Rechner weitergeleitet werden sollen. Trifft keine der Regeln der PREROUTING-Kette zu, werden die Pakete unverändert weiterbehandelt. Die Aktion DNAT erwartet dabei die IP-Adresse, die im Paket als Ziel eingetragen werden soll. Diese kann als

- IP-Adresse (Beispiel: DNAT:1.2.3.4),
- IP-Bereich (Beispiel: DNAT:1.2.3.4-1.2.3.10)
- oder als Hostname (Beispiel: DNAT:@client1)

angegeben werden.

Schließlich kann noch ein Port bzw. Portbereich angegeben werden, auf den der Zielport abgebildet werden soll. Das ist aber nur nötig, wenn der Port geändert werden soll. Der Port bzw. Portbereich wird einfach hinten angehängt, z. B. so: DNAT:@server:21.

REDIRECT verhält sich wie DNAT, nur dass die Ziel-IP-Adresse immer auf 127.0.0.1 gesetzt wird und damit das Paket lokal zugestellt wird. Dies wird z. B. für transparente Proxys benötigt, siehe OPT_TRANSPROXY (Seite 206).

Will man eine Portweiterleitung auf Schnittstellen mit dynamischen Adressen machen, weiß man zum Zeitpunkt der Konfiguration noch nicht, an welche IP die Pakete gerichtet sein werden. Daher kann man in der PREROUTING-Kette dynamic als Platzhalter für die später zugewiesene IP verwenden, etwa wie folgt:

```
'dynamic:80  DNAT:1.2.3.4'           # leite http-Pakete an die
                                     # IP-Adresse 1.2.3.4 weiter
'prot:gre any dynamic DNAT:1.2.3.4'  # leite gre-Pakete (Teil des PPTP-
                                     # Protokolls) an die IP-Adresse
                                     # 1.2.3.4 weiter
```

Bei der PREROUTING-Kette können nur ACCEPT, DNAT, NETMAP und REDIRECT als Aktionen verwendet werden.

Für weitere Beispiele zur Portweiterleitung siehe den nächsten Abschnitt.

PF_PREROUTING_N PF_PREROUTING_x PF_PREROUTING_x_COMMENT

Eine Liste der Regeln, die beschreiben, welche Pakete vom Router an ein anderes Ziel weitergeleitet werden sollen.

3.10.5. Beispiele

Im Folgenden sind einige Beispiele für die Paketfilter-Konfiguration angegeben.

Die fli4l-Standardkonfiguration

Die fli4l-Standardkonfiguration der Distribution sieht für die `INPUT`-Kette wie folgt aus:

```
PF_INPUT_POLICY='REJECT'
PF_INPUT_ACCEPT_DEF='yes'
PF_INPUT_LOG='no'
PF_INPUT_N='1'
PF_INPUT_1='IP_NET_1 ACCEPT'
```

Damit erreichen wir, dass

- Rechner im lokalen Netzwerk auf den Router zugreifen dürfen (`PF_INPUT_1='IP_NET_1 ACCEPT'`),
- lokale Kommunikation auf dem Router erlaubt ist (`PF_INPUT_ACCEPT_DEF='yes'`),
- Pakete, die zu vom Router aufgebauten Verbindungen gehören, akzeptiert werden (`PF_INPUT_ACCEPT_DEF='yes'`),
- alles andere abgelehnt wird (`PF_INPUT_POLICY='REJECT'`),
- aber nicht ins System-Protokoll geschrieben wird (`PF_INPUT_LOG='no'`).

Für die `FORWARD`-Kette sieht das so ähnlich aus: Nur Pakete unseres lokalen Netzes und Pakete, die zu Verbindungen gehören, die von Rechnern im lokalen Netz aufgebaut wurden, sollen weitergeleitet werden. Des Weiteren werden NetBIOS- und CIFS-Pakete verworfen.

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='2'
PF_FORWARD_1='tmpl:samba DROP'
PF_FORWARD_2='IP_NET_1 ACCEPT'
```

Was man hier gut sieht, ist die Abhängigkeit von der Reihenfolge der Regeln: *Zuerst* werden NetBIOS-Pakete verworfen, und *danach* werden die Pakete des lokalen Netzes akzeptiert.

Nun kann das lokale Netz mit dem Router kommunizieren, seine Pakete werden weitergeleitet, es fehlt nur noch das Maskieren, welches für den Zugriff eines privaten Netzwerkes auf das Internet notwendig ist:

```
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'
```

Trusted Nets

Wollen wir lokal mehrere Subnetze haben, die frei und unmaskiert miteinander kommunizieren können, müssen wir dafür sorgen, dass Pakete zwischen diesen Subnetzen nicht verworfen und auch nicht maskiert werden. Dazu fügen wir einfach eine Regel hinzu oder modifizieren die vorhandene.

Angenommen, wir haben einen DSL-Zugang über PPPoE, und die beiden Subnetze sind `IP_NET_1` (192.168.6.0/24) und `IP_NET_2` (192.168.7.0/24). Dann würde die Konfiguration wie folgt aussehen:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='4'
PF_FORWARD_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_FORWARD_2='tmpl:samba DROP'
PF_FORWARD_3='IP_NET_1 ACCEPT'
PF_FORWARD_4='IP_NET_2 ACCEPT'

PF_POSTROUTING_N='3'
PF_POSTROUTING_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_POSTROUTING_2='IP_NET_1 MASQUERADE'
PF_POSTROUTING_3='IP_NET_2 MASQUERADE'
```

Regel eins sorgt jetzt dafür, dass Pakete zwichen den beiden Subnetzen ohne weitere Prüfung weitergeleitet werden. Die Regeln drei und vier sorgen dafür, dass beide Subnetze auch ins Internet kommen. Die erste Regel der `POSTROUTING`-Kette sorgt dafür, dass die Kommunikation zwischen den Subnetzen unmaskiert erfolgt.

Alternativ könnten wir auch sagen, dass nur Pakete, die über die `pppoe`-Schnittstelle hinausgehen, maskiert werden sollen:

```
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'
```

Genauso hätte man die Filterung der Ports auch auf die `pppoe`-Schnittstelle beschränken und die beiden Subnetze zu einem zusammenfassen können, das würde dann wie folgt aussehen:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='2'
PF_FORWARD_1='if:any:pppoe tmpl:samba DROP'
PF_FORWARD_2='192.168.6.0/23 ACCEPT'

PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'
```

Pakete, die über die `pppoe`-Schnittstelle hinausgehen und die an die `udp`-Ports 137-138 oder an die `tcp`-Ports 139 und 445 adressiert sind, werden verworfen (Regel 1), alle anderen Pakete, die aus dem Subnetz 192.168.6.0/23 kommen, werden weitergeleitet (Regel 2).

Route Network

Fügen wir dem Ganzen noch ein Netzwerk 10.0.0.0/24 hinzu (z. B. ein Dial-In-Netzwerk), mit dem wir unmaskiert kommunizieren wollen, wobei Pakete an die `udp`-Ports 137-138 sowie an die `tcp`-Ports 139 und 445 verworfen werden sollen, dann würde das wie folgt aussehen:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='4'
PF_FORWARD_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_FORWARD_2='tmpl:samba DROP'
PF_FORWARD_3='192.168.6.0/23 ACCEPT'
PF_FORWARD_4='10.0.0.0/24 ACCEPT'

PF_POSTROUTING_N='2'
PF_POSTROUTING_1='10.0.0.0/24 ACCEPT BIDIRECTIONAL'
PF_POSTROUTING_2='192.168.6.0/23 MASQUERADE'
```

- Regel 1 erlaubt die ungehinderte Kommunikation zwischen den Subnetzen `IP_NET_1` und `IP_NET_2`.
- Regel 2 verwirft Pakete an die Samba-Ports.
- Die Regeln 3 und 4 erlaubt die Weiterleitung von Paketen, die aus den Subnetzen 192.168.6.0/24, 192.168.7.0/24 und 10.0.0.0/24 kommen; die Rückrichtung wird von der Einstellung `PF_FORWARD_ACCEPT_DEF='yes'` abgedeckt.
- Regel 1 der `POSTROUTING`-Kette sorgt dafür, dass Pakete in das bzw. aus dem 10.0.0.0/24-Subnetz nicht maskiert werden.

Alternativ ginge auch:

```
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'
```

Diese Regel besagt, dass nur Pakete, die über die `pppoe`-Schnittstelle hinausgehen, maskiert werden.

Blacklists, Whitelists

Blacklists (ein Rechner in dieser Liste darf etwas nicht) und Whitelists (ein Rechner in dieser Liste darf etwas) werden prinzipiell ähnlich umgesetzt. Es werden Regeln geschrieben, die am Anfang sehr speziell sind und nach hinten immer allgemeiner werden. Bei einer Blacklist stehen am Anfang Regeln, die etwas verbieten und am Ende Regeln, die allen bisher nicht erwähnten etwas erlauben. Bei einer Whitelist ist es genau umgekehrt.

Beispiel 1: Alle Rechner im Subnetz 192.168.6.0/24 außer Rechner 12 dürfen ins Internet, solange sie nicht mit den CIFS Ports 137-138 (`udp`), 139 und 445 (`tcp`) kommunizieren wollen:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
```

```
PF_FORWARD_N='3'
PF_FORWARD_1='192.168.6.12 DROP'
PF_FORWARD_2='tmpl:samba DROP'
PF_FORWARD_3='192.168.6.0/23 ACCEPT'

PF_POSTROUTING_N='1'
PF_POSTROUTING_2='192.168.6.0/24 MASQUERADE'
```

Beispiel 2: Nur Rechner 12 darf ins Internet (aber nicht an die o. g. Ports ...), alle anderen dürfen nur lokal mit einem anderen Subnetz kommunizieren:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='3'
PF_FORWARD_1='192.168.6.0/24 192.168.7.0/24 ACCEPT BIDIRECTIONAL'
PF_FORWARD_2='tmpl:samba DROP'
PF_FORWARD_3='192.168.6.12 ACCEPT'

PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'
```

3.10.6. Standardkonfigurationen

Einfacher maskierender Router mit einem Netz dahinter

```
#
# Zugriff auf den Router
#
PF_INPUT_POLICY='REJECT'
PF_INPUT_ACCEPT_DEF='yes'
PF_INPUT_LOG='no'
PF_INPUT_N='1'
PF_INPUT_1='IP_NET_1 ACCEPT'   # alle Hosts im lokalen Netz dürfen
                               # auf den Router zugreifen

#
# Zugriff auf das ``Internet''
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

PF_FORWARD_N='2'
PF_FORWARD_1='tmpl:samba DROP' # Samba-Pakete, die das Netz
                               # verlassen wollen, werden verworfen
PF_FORWARD_2='IP_NET_1 ACCEPT' # alle anderen Pakete dürfen das
                               # lokale Netz verlassen

#
# Maskieren des lokalen Netzes
#
PF_POSTROUTING_N='1'
```

```
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'  # maskiere Pakete, die das Subnetz
                                        # verlassen
```

Einfacher maskierender Router mit zwei Netzen dahinter

```
#
# Zugriff auf den Router
#
PF_INPUT_POLICY='REJECT'
PF_INPUT_ACCEPT_DEF='yes'
PF_INPUT_LOG='no'
PF_INPUT_N='2'
PF_INPUT_1='IP_NET_1 ACCEPT'   # alle Hosts im lokalen Netz dürfen
                               # auf den Router zugreifen
PF_INPUT_2='IP_NET_2 ACCEPT'   # alle Hosts im lokalen Netz dürfen
                               # auf den Router zugreifen

#
# Zugriff auf das ``Internet''
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

#
# Freie Kommunikation zwischen den Netzen
#
PF_FORWARD_N='4'
PF_FORWARD_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_FORWARD_2='tmpl:samba DROP' # Samba-Pakete, die das Netz
                               # verlassen wollen, werden verworfen
PF_FORWARD_3='IP_NET_1 ACCEPT' # alle anderen Pakete dürfen das
                               # lokale Netz verlassen
PF_FORWARD_4='IP_NET_2 ACCEPT' # alle anderen Pakete dürfen das
                               # lokale Netz verlassen

#
# Maskieren der lokalen Netze, unmaskierte Kommunikation zwischen den
# Netzen
#
PF_POSTROUTING_N='3'
PF_POSTROUTING_1'IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_POSTROUTING_2='IP_NET_1 MASQUERADE'  # maskiere Pakete, die das Subnetz
                                        # verlassen
PF_POSTROUTING_3='IP_NET_2 MASQUERADE'  # maskiere Pakete, die das Subnetz
                                        # verlassen
```

Maskierender DSL-Router mit zwei Netzen dahinter und SSH/HTTP-Zugriff aus dem Internet

```
#
# Zugriff auf den Router
#
```

```
PF_INPUT_POLICY='REJECT'
PF_INPUT_ACCEPT_DEF='yes'
PF_INPUT_LOG='no'
PF_INPUT_N='4'
PF_INPUT_1='IP_NET_1 ACCEPT'   # alle Hosts im lokalen Netz dürfen
                               # auf den Router zugreifen
PF_INPUT_2='IP_NET_2 ACCEPT'   # alle Hosts im lokalen Netz dürfen
                               # auf den Router zugreifen
PF_INPUT_3='tmpl:ssh ACCEPT'   # gestatte Zugriff auf SSH-Dienst
                               # von überall her
PF_INPUT_4='tmpl:http 1.2.3.4/24 ACCEPT'  # gestatte Rechner aus
                               # einem bestimmten Subnetz Zugriff
                               # auf HTTP-Dienst


#
# Zugriff auf das ``Internet''
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

#
# Keine Kommunikation zwischen den Netzen, beide Netze dürfen ins
# Internet, Samba-Pakete werden verworfen
#
PF_FORWARD_N='2'
PF_FORWARD_1='tmpl:samba if:any:pppoe DROP' # Samba-Pakete, die das Netz
                               # verlassen wollen, werden verworfen
PF_FORWARD_2='if:any:pppoe ACCEPT' # alle anderen Pakete dürfen das
                               # lokale Netz verlassen

#
# Maskieren der lokalen Netze, unmaskierte Kommunikation zwischen den
# Netzen
#
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'  # maskiere Pakete, die das Subnetz
                                            # verlassen
```

Portweiterleitung

Portweiterleitungen lassen sich mit den `PREROUTING`-Regeln wie folgt umsetzen (`TARGET` bezeichnet die ursprüngliche Zieladresse (optional) und den ursprünglichen Zielport, `NEW_TARGET` bezeichnet die neue Zieladresse und den neuen Zielport (optional), `PROTOCOL` bezeichnet das jeweilige Protokoll):

```
TARGET='<port>'
NEW_TARGET='<ip>'
PROTOCOL='<proto>'
PF_PREROUTING_x='prot:<proto> dynamic:<port> DNAT:<ip>'
```

```
TARGET='<port1>-<port2>'
NEW_TARGET='<ip>'
PROTOCOL='<proto>'
PF_PREROUTING_x='prot:<proto> dynamic:<port1>-<port2> DNAT:<ip>'

TARGET='<ip>:<port-a>'
NEW_TARGET='<ip>:<port-b>'
PROTOCOL='<proto>'
PF_PREROUTING_x='prot:<proto> any <ip>:<port-a> DNAT:<ip>:<port-b>'
```

Transparenter Proxy

Will man bestimmte Zugriffe auf das Internet nur über einen lokalen Proxy zulassen, kann man das mit Hilfe der `PREROUTING`- und `POSTROUTING`-Ketten erzwingen, ohne dass der Client davon etwas merkt. Prinzipiell sind dazu drei Schritte notwendig:

1. Anfragen an den HTTP-Port, die nicht vom Proxy kommen, an den Proxy umleiten (`PREROUTING`).
2. Die umgeleiteten Pakete so verändern, dass der Proxy denkt, sie kommen vom Router, so dass er sie wieder dorthin zurückschickt (`POSTROUTING`).
3. Die Pakete durch die FORWARD-Kette durchlassen, sofern ein Eintrag à la

   ```
   PF_FORWARD_x='IP_NET_1 ACCEPT'
   ```

   nicht existiert (`FORWARD`).

Beispiel 1: Angenommen, wir haben nur ein Netz `IP_NET_1`, in dem auf einem Rechner namens `proxy` ein Squid-Proxy läuft, und wollen den gesamten `http`-Datenverkehr über ihn leiten. Squid lauscht auf Port 3128. Der Einfachheit halber beziehen wir uns via `@proxy` auf den eingetragenen Host aus `HOST_1_NAME='proxy'` (vgl. Domainkonfiguration (Seite 71)).

Das Ganze würde wie folgt aussehen:

```
...
  PF_PREROUTING_x='@proxy ACCEPT'
      # Pakete vom Proxy sollen nicht umgeleitet werden

  PF_PREROUTING_x='prot:tcp IP_NET_1 80 DNAT:@proxy:3128'
      # HTTP-Pakete aus IP_NET_1 mit einem beliebigen Ziel werden
      # umgeleitet nach @proxy, Port 3128

  PF_POSTROUTING_x='any @proxy:3128 SNAT:IP_NET_1_IPADDR'
      # alle Pakete an den Proxy-Port 3128 so umschreiben, als wären sie
      # vom fli4l (IP_NET_1_IPADDR)

  PF_FORWARD_x='prot:tcp @proxy 80 ACCEPT'
      # HTTP-Pakete vom Proxy durch die FORWARD-Kette durchlassen (wenn nötig)
...
```

Gibt es mehrere Netze oder potentielle Konflikte mit anderen Portweiterleitungen (die ja auch nichts anderes sind als DNAT-Regeln), muss man die Regeln vielleicht noch etwas enger formulieren.

Beispiel 2: Unser Proxy namens `proxy` steht in `IP_NET_1`, lauscht auf Port 3128 und soll nur für Clients aus `IP_NET_1` wirksam werden. `IP_NET_1` ist über `IP_NET_1_DEV` erreichbar. Pakete aus weiteren Netzen sollen nicht berücksichtigt werden.

```
...
  PF_PREROUTING_x='if:IP_NET_1_DEV:any !@proxy 80 DNAT:@proxy:3128'
      # Anfragen an den HTTP-Port, die nicht vom Proxy, aber über eine
      # interne Schnittstelle (IP_NET_1_DEV) kommen, an den Proxy-Port umleiten.
      # An dieser Stelle ist es wichtig, mit if:IP_NET_1_DEV:any zu
      # überprüfen, ob die Pakete von innen kommen, da sonst auch Pakete von
      # außen umgeleitet würden (Sicherheitslücke!).

  PF_POSTROUTING_x='prot:tcp IP_NET_1 @proxy:3128 SNAT:IP_NET_1_IPADDR'
      # HTTP-Pakete die aus IP_NET_1 stammen und für den Proxy-Port 3128
      # gedacht sind, so umschreiben, als wären sie vom fli4l (IP_NET_1_IPADDR)

  PF_FORWARD_x='prot:tcp @proxy 80 ACCEPT'
      # HTTP-Pakete vom Proxy durch die FORWARD-Kette durchlassen (wenn nötig)
...
```

Beispiel 3: Um sich das Leben etwas zu erleichtern und die Regeln kürzer zu gestalten, kann man auch Templates einsetzen (vgl. Templates im Paketfilter (Seite 50)). Zweckmäßig ist an dieser Stelle das `tmpl:http`, das in `prot:tcp any any:80` übersetzt wird. So wird z. B. aus `tmpl:http IP_NET_1 DNAT:@proxy:3128` dann `prot:tcp IP_NET_1 80 DNAT:@proxy:3128`.

Sowohl `IP_NET_1` als auch `IP_NET_2` sollen transparent über den Proxy umgeleitet werden. Damit ließe sich vereinfacht auch schreiben:

```
...
  PF_PREROUTING_x='tmpl:http @proxy   ACCEPT'
      # HTTP-Pakete vom Proxy sollen nicht umgeleitet werden

  PF_PREROUTING_x='tmpl:http IP_NET_1 DNAT:@proxy:3128'
      # HTTP-Pakete aus IP_NET_1 sollen umgeleitet werden

  PF_PREROUTING_x='tmpl:http IP_NET_2 DNAT:@proxy:3128'
      # HTTP-Pakete aus IP_NET_2 sollen umgeleitet werden

  PF_POSTROUTING_x='IP_NET_1 @proxy:3128 SNAT:IP_NET_1_IPADDR'
  PF_POSTROUTING_x='IP_NET_2 @proxy:3128 SNAT:IP_NET_2_IPADDR'

  PF_FORWARD_x='tmpl:http @proxy ACCEPT'
...
```

Und so ließe sich das endlos fortsetzen ...

3.10.7. DMZ – Demilitarisierte Zone

fli4l gestattet auch den Aufbau einer DMZ. Hier sei erstmal auf das Wiki verwiesen. https://ssl.nettworks.org/wiki

3.10.8. Conntrack-Helfer

Die Verwendung von IP-Masquerading hat zwar den Vorteil, dass mehrere Rechner im LAN über eine einzige offizielle IP-Adresse geroutet werden kann, es gibt aber auch Nachteile, die man in Kauf nehmen muss.

Ein großes Problem ist zum Beispiel, dass kein Rechner von außen von sich aus eine Verbindung zu einem Rechner aufnehmen kann. Das ist zwar aus Sicherheitsgründen eigentlich durchaus erwünscht, aber bestimmte Protokolle funktionieren nicht mehr, weil sie einen Verbindungsaufbau von außen einfach erfordern.

Ein klassisches Beispiel ist FTP. Neben dem Kommunikationskanal, auf dem Befehle und Antworten ausgetauscht werden, wird ein weiterer Kanal (in Form eines IP-Ports) verwendet, um die eigentlichen Nutzdaten zu versenden. fli4l verwendet dafür bestimmte Conntrack-Helfer, um solche zusätzlichen Ports, die verwendet werden, ad hoc dann freizuschalten und an den internen Rechner weiterzuleiten, wenn sie benötigt werden. Dabei "horcht" der Conntrack-Helfer in den Datenstrom, um zu erkennen, wann ein zusätzlicher Port benötigt wird.

Typische Anwendungen für Conntrack-Helfer sind Chat-Protokolle und Spiele im Internet.

Ein solcher Conntrack-Helfer wird über Regeln in zwei speziellen Arrays aktiviert. Das Array `PF_PREROUTING_CT_%` enthält Helfer-Zuordnungen zu Paketen, die von außen kommen, das Array `PF_OUTPUT_CT_%` enthält Helfer-Zuordnungen zu Paketen, die auf dem Router generiert werden. Einige Beispiele aus der Praxis sollen dies verdeutlichen.

Beispiel 1: Soll aktives FTP aus dem LAN erlaubt werden, ist das aus der Sicht des Routers eine Verbindung von außerhalb, somit muss ein Eintrag in `PF_PREROUTING_CT_%` vorgenommen werden:

```
PF_PREROUTING_CT_N='1'
PF_PREROUTING_CT_1='tmpl:ftp IP_NET_1 HELPER:ftp'
```

Damit wird für alle TCP-Verbindungen aus dem lokalen Netz (`IP_NET_1`) zu irgendeiner anderen Adresse an Port 21 (dies ist der `ftp`-Port) das `ftp`-Hilfsmodul geladen. Dieses Modul erlaubt dann im Laufe der Verbindung, dass der FTP-Server zurück zum Client eine Datenverbindung aufbauen kann, indem temporär ein "Loch" in der Firewall aufgemacht wird.

Beispiel 2: Soll passives FTP für einen FTP-Server im LAN ermöglicht werden (dabei wird die Datenverbindung von außen nach innen aufgebaut, so dass auch hier kurzfristig ein Loch in der Firewall geöffnet werden muss), ist dies ebenfalls aus der Sicht des Router eine Verbindung von außerhalb des Routers. Hier sieht die Regel folgendermaßen aus:

```
PF_PREROUTING_CT_N='1'
PF_PREROUTING_CT_1='tmpl:ftp any dynamic HELPER:ftp'
```

Mit dieser Regel wird ausgedrückt, dass alle FTP-Verbindungen, die an die dynamische Adresse des Routers gesandt werden, mit dem FTP-Conntrack-Helfer assoziiert werden. Hier wurde `dynamic` verwendet, da angenommen wird, dass der Router für die Einwahl ins Internet verantwortlich ist und somit eine externe IP-Adresse besitzt. Falls der Router eine Einwahl via DSL durchführt, kann man die Regel auch so schreiben:

```
PF_PREROUTING_CT_N='1'
PF_PREROUTING_CT_1='tmpl:ftp if:pppoe:any HELPER:ftp'
```

Mit dieser Regel wird ausgedrückt, dass alle FTP-Verbindungen, die von der DSL-Schnittstelle (pppoe) kommen, mit dem FTP-Conntrack-Helfer assoziiert werden.

Falls der Router sich nicht einwählt, sondern z. B. hinter einem anderen Router (Fritz!Box, Kabelmodem etc.) hängt, so kann die folgende Regel verwendet werden:

```
PF_PREROUTING_CT_N='1'
PF_PREROUTING_CT_1='tmpl:ftp if:IP_NET_2_DEV:any HELPER:ftp'
```

Dabei wird im Beispiel angenommen, dass die Verbindung zum anderen Router über die Schnittstelle durchgeführt wird, die dem zweiten Subnetz zugeordnet ist (IP_NET_2_DEV).

Zu beachten ist, dass natürlich *zusätzlich* eine entsprechende Konfiguration der FORWARD-Kette nötig ist, um die FTP-Pakete auch tatsächlich weiterzuleiten. Eine typische Regel wäre etwa

```
PF_PREROUTING_1='tmpl:ftp any dynamic DNAT:@ftpserver'
```

wobei angenommen wird, dass der Host, auf dem das FTP-Serverprogramm läuft, den Namen ftpserver hat.

Beispiel 3: Schließlich muss auch, wenn man vom fli4l direkt aktives FTP benutzen möchte (etwa mit Hilfe des ftp-Programms aus dem tools-Paket), die Firewall dafür vorbereitet werden, diesmal in der OUTPUT-Kette, die mit Hilfe des Arrays PF_OUTPUT_CT_% konfiguriert wird:

```
PF_OUTPUT_CT_N='1'
PF_OUTPUT_CT_1='tmpl:ftp HELPER:ftp'
```

Diese Regel ist jedoch unnötig, falls FTP_PF_ENABLE_ACTIVE='yes' benutzt wird – siehe hierzu die Dokumentation des ftp-OPTs im tools-Paket.

Es folgt eine Übersicht über die existierenden Conntrack-Helfer:

Helfer	**Erläuterung**
ftp	File Transfer Protocol
h323	H.323 (Voice over IP)
irc	Internet Relay Chat
pptp	PPTP Masquerading (Mit diesem Modul lässt sich mehr als ein PPTP-Client gleichzeitig hinter einem fli4l-Router betreiben.)
sip	Session Initiation Protocol
sane	SANE Network Procotol
snmp	Simple Network Management Protocol
tftp	Trivial File Transfer Protocol

Tabelle 3.9.: Verfügbare Conntrack-Helfer im Paketfilter

Es folgt eine Übersicht der zu konfigurierenden Variablen:

PF_PREROUTING_CT_ACCEPT_DEF Steht diese Variable auf 'yes', werden Standard-Regeln generiert, die für ein korrektes Funktionieren des Routers notwendig sind. Standardmäßig sollte man hier 'yes' eintragen.

PF_PREROUTING_CT_N PF_PREROUTING_CT_x PF_PREROUTING_CT_x_COMMENT Liste der Regeln, die beschreiben, welche eingehenden Pakete vom Router mit Conntrack-Helfern verbunden werden.

PF_OUTPUT_CT_ACCEPT_DEF Steht diese Variable auf 'yes', werden Standard-Regeln generiert, die für ein korrektes Funktionieren des Routers notwendig sind. Standardmäßig sollte man hier 'yes' eintragen.

PF_OUTPUT_CT_N PF_OUTPUT_CT_x PF_OUTPUT_CT_x_COMMENT Liste der Regeln, die beschreiben, welche auf dem Router generierten Pakete vom Router mit Conntrack-Helfern verbunden werden.

3.11. Domain-Konfiguration

Windows-PCs im LAN haben eine unangenehme Eigenschaft: Sobald ein Nameserver benötigt wird und man diesen deshalb im Windows einstellt, fragen diese Windows-Rechner den angegebenen Nameserver in regelmäßigen Abständen ab – auch wenn man gar nicht daran arbeitet! Würde man also auf dem Windows-PC einen DNS-Server im Internet angeben, wird's teuer...

Der Trick ist nun folgender: Wenn im LAN nicht bereits ein DNS-Server vorhanden ist, kann man den DNS-Server im fli4l-Router verwenden.

Es wird DNSMASQ als DNS-Server eingesetzt.

Wenn man jedoch mit der DNS-Konfiguration beginnt, sollte man sich zunächst Gedanken über den Domain-Namen und die Namen der PCs im Netz machen. Der verwendete Domain-Name wird nicht im Internet sichtbar. Deshalb kann man sich hier prinzipiell beliebige Domain-Namen ausdenken.

Außerdem sollte man jedem Windows-Rechner einen Namen verpassen. Diese Namen müssen dem fli4l-Router bekannt sein.

DOMAIN_NAME Standard-Einstellung: `DOMAIN_NAME='lan.fli4l'`

Hier kann sich jeder austoben, da die lokal verwendete Domain nicht im Internet sichtbar wird. Sie sollten lediglich vermeiden, einen Namen zu benutzen, den es im Internet geben könnte (z.B. irgendwas.de), da Sie sonst nicht auf diese Domain werden zugreifen können.

DNS_FORWARDERS Standard-Einstellung: `DNS_FORWARDERS="`

Hier ist die DNS-Server-Adresse des Internet-Providers anzugeben, wenn fli4l als Router in das Internet verwendet wird. Der fli4l-Router gibt dann sämtliche DNS-Anfragen, die er nicht selbst beantworten kann, an diese Adresse weiter.

Möchte man mehrere DNS-Forwarder angeben, trennt man die IP-Adressen durch Leerzeichen.

Sind mehrere DNS-Server konfiguriert werden diese in der angegebenen Reihenfolge für DNS-Anfragen genutzt, somit wird der zweite angegebene Server nur genutzt, wenn der erste keine Antwort liefert usw.

Es ist auch möglich, optional Port-Nummern an die IP-Adressen durch Doppelpunkt getrennt anzugeben. Allerdings muss dann OPT_DNS='yes' (Seite 99) sein (Paket dns_dhcp (Seite 98)) und es darf nirgends die Option *_USEPEERDNS benutzt werden.

Achtung: Auch wenn

- PPPOE_USEPEERDNS (Seite 111),
- ISDN_CIRC_x_USEPEERDNS (Seite 161) oder
- DHCP_CLIENT_x_USEPEERDNS (Seite 97)

gesetzt (='yes') ist, ist hier die Eintragung eines Servers nötig, da sonst direkt nach dem Start keine Namensauflösung möglich ist.

Ausnahme: fli4l als Router in einem lokalen Netz *ohne* Anschluss an das Internet oder (Firmen-)Netze mit weiteren DNS-Servern. In diesem Fall ist 127.0.0.1 anzugeben, um das Weiterleiten zu unterbinden.

HOSTNAME_IP (optional)

Hiermit kann optional festgelegt werden, an welches Netz IP_NET_x der HOSTNAME gebunden wird.

HOSTNAME_ALIAS_N (optional)

Anzahl der zusätzlichen Alias-Hostnamen für den Router.

HOSTNAME_ALIAS_x (optional)

Zusätzlicher Alias-Name für den Router.

3.12. imond-Konfiguration

START_IMOND Standard-Einstellung: START_IMOND='no'

Mit START_IMOND kann man einstellen, ob der imond-Server aktiviert werden soll. imond übernimmt dabei das Monitoring/Controlling und Least-Cost-Routing des fli4l-Routers. Der Beschreibung von imond (Seite 284) ist deshalb ein extra Kapitel gewidmet (s.u.).

Wichtig: Die LC-Routing-Features von fli4l können nur mit imond genutzt werden. Ein zeitabhängiges Umschalten von Verbindungen ist ohne imond nicht möglich!

Für ISDN- und DSL-Routing ist imond ab Version 1.5 zwingend erforderlich. In diesem Fall ist START_IMOND='yes' einzustellen.

Wird fli4l lediglich als Router zwischen 2 Netzwerken eingesetzt, sollte START_IMOND='no' eingestellt werden.

IMOND_PORT TCP/IP-Port, auf dem imond auf Verbindungen horcht. Der Standard-Wert '5000' sollte nur in Ausnahmefällen geändert werden.

IMOND_PASS Standard-Einstellung: IMOND_PASS="

Hier kann ein spezielles User-Password für imond gesetzt werden. Meldet sich ein Client auf Port 5000 an, erwartet imond (und damit auch seine Clients) die Eingabe dieses Passworts, bevor er irgendeinen Befehl korrekt beantwortet. Ausnahme: Befehle "quit", "help" und "pass". Ist IMOND_PASS leer, wird kein Password benötigt.

Ob der Client im User-Modus bestimmte Steuerbefehle, wie Dial, Hangup, Reboot, Umschalten der Default-Route bereits ausführen kann oder dafür die Eingabe des Admin-Passwords zwingend notwendig ist, wird über die Variablen

- IMOND_ENABLE (Seite 74),
- IMOND_DIAL (Seite 74),
- IMOND_ROUTE (Seite 74) und
- IMOND_REBOOT (Seite 74)

eingestellt, siehe unten.

IMOND_ADMIN_PASS Standard-Einstellung: `IMOND_ADMIN_PASS="`

Mit Hilfe der Admin-Passwords erhält der Client alle Rechte und kann so sämtliche Steuerfunktionen des imond-Servers nutzen – und zwar unabhängig von den Variablen `IMOND_ENABLE`, `IMOND_DIAL` usw. Lässt man `IMOND_ADMIN_PASS` leer, so reicht die Eingabe des User-Passwords, um sämtliche Rechte zu erhalten!

IMOND_LED imond kann den Online/Offline-Status nun über eine LED anzeigen. Diese wird folgendermaßen an einen COM-Port angeschlossen:

Verbindung 25-polig:

```
20 DTR  -------- 1kOhm ----- >| ---------- 7 GND
```

Verbindung 9-polig:

```
 4 DTR  -------- 1kOhm ----- >| ---------- 5 GND
```

Ist eine ISDN- oder DSL-Verbindung aufgebaut, leuchtet die LED. Ansonsten ist sie ausgeschaltet. Sollte es genau umgekehrt sein, ist die Leuchtdiode umzupolen. Sollte die LED zu schwach leuchten, kann der Vorwiderstand bis auf 470 Ohm reduziert werden.

Es ist auch möglich, zwei verschiedenfarbige LEDs anzuschließen. Dann ist die zweite LED ebenso über einen Vorwiderstand zwischen DTR und GND anzuschließen, jedoch genau umgekehrt. Dann leuchtet je nach Zustand die eine oder die andere LED. Oder man verwendet direkt eine DUO-LED (zweifarbig, drei Anschlussbeinchen).

Im Moment verhält sich der RTS-Anschluss der seriellen Schnittstelle genauso wie DTR. Hier könnte also noch eine weitere LED angechlossen werden, die den Online/Offline-Zustand anzeigt. Das könnte sich jedoch in einer zukünftigen fli4l-Version ändern.

Als Wert von `IMOND_LED` muss ein COM-Port angegeben werden, also 'com1', 'com2', 'com3' oder 'com4'. Ist keine LED angeschlossen, sollte die Variable leer gelassen werden.

IMOND_BEEP Mit `IMOND_BEEP`='yes' gibt imond einen Zweiklang-Ton über den PC-Lautsprecher aus, wenn der Zustand von Offline nach Online wechselt und umgekehrt. Im ersten Fall wird zuerst ein tiefer, dann ein hoher Ton ausgegeben. Beim Wechsel in den Offline-Status zurück wird zuerst der höhere, dann der tiefere Ton ausgegeben.

IMOND_LOG Standard-Einstellung: `IMOND_LOG='no'`

Wird `IMOND_LOG`='yes' benutzt, werden in der Datei `/var/log/imond.log` die Verbindungen protokolliert. Diese Datei kann z.B. für Statistikzwecke per scp auf einen Rechner im LAN kopiert werden. Für den scp-Zugriff ist aber dann noch das Paket sshd zu installieren und so zu konfigurieren, dass es auch scp zur Verfügung stellt.

Das Format der Logdateieinträge ist in Tabelle 3.10 beschrieben.

Tabelle 3.10.: Format der Imond-Logdatei

Eintrag	Bedeutung
Circuit	der Name des Circuits, für den der Eintrag erzeugt wurde
Startzeit	Datum und Uhrzeit der Einwahl dieses Circuits
Stopzeit	Datum und Uhrzeit des Auflegens dieses Circuits
Online-Zeit	die Zeit, die dieser Circuit online war
Abgerechnete Zeit	die Zeit, die der Provider abrechnen wird (hängt vom Takt ab)
Kosten	die Kosten, die der Provider für die Zeit in Rechnung stellt
Bandbreite	die genutzte Bandbreite getrennt nach in und out (in zuerst), dargestellt als zwei vorzeichenlose Integerzahlen, für die gilt: Bandbreite = *4GiB* $*$ *<erste Zahl>* + *<zweite Zahl>*
Device	das Gerät, über das kommuniziert wurde
Abrechnungstakt	der Takt, der vom Provider zur Abrechnung herangezogen wird (Daten der Circuit-Konfiguration)
Taktgebühren	die Gebühren, die pro Takt fällig werden (Daten der Circuit-Konfiguration)

Die Kosten werden in Euro ausgegeben. Wichtig ist dabei die korrekte Definition der entsprechenden Circuit-Variablen `ISDN_CIRC_x_TIMES` (Seite 168).

IMOND_LOGDIR Ist das Protokollieren eingeschaltet, kann über `IMOND_LOGDIR` ein alternatives Verzeichnis statt /var/log angegeben werden, z.B. '/boot'. Dann wird die Log-Datei imond.log auf dem Bootmedium angelegt. Dazu muss dieses aber auch Read/Write „gemounted“ sein. Default ist 'auto' was den Speicherort automatisch bestimmt. Je nach weiterer Konfiguration liegt das dann unter /boot/persistent/base oder an einem anderen durch `FLI4L_UUID` bestimmten Pfad. Ist /boot nicht Read/Write und `FLI4L_UUID` nicht gesetzt, befindet sich das File unter /var/run.

IMOND_ENABLE IMOND_DIAL IMOND_ROUTE IMOND_REBOOT Durch diese Variablen werden bestimmte Kommandos, die von imonc-Clients zum imond-Server gesendet werden, bereits im User-Modus freigeschaltet.

Hiermit kann man einstellen, ob der imond-Server die ISDN-Schnittstelle ein- bzw. ausschalten, wählen/einhängen, eine neue Default-Route setzen und/oder den Rechner booten darf.

Standard-Einstellungen:

```
IMOND_ENABLE='yes'
IMOND_DIAL='yes'
IMOND_ROUTE='yes'
IMOND_REBOOT='yes'
```

Alle weiteren Features der Client-/Server-Schnittstelle von imond sind in einem eigenen Kapitel (Seite 284) beschrieben.

3.13. Allgemeine Circuit-Konfiguration

IP_DYN_ADDR Wird eine Verbindung mit dynamischer Adressvergabe verwendet, ist `IP_DYN_ADDR` auf 'yes' zu stellen, ansonsten auf 'no'. Die meisten Internet-Provider verwenden eine dynamische Adressvergabe.

Standard-Einstellung: `IP_DYN_ADDR`='yes'

DIALMODE Der Standard-Wählmodus von fli4l ist 'auto', d.h. es wird automatisch gewählt, wenn eine IP-Adresse außerhalb des eigenen Netzes angesprochen wird. Es ist aber auch möglich, als Dialmode 'manual' oder 'off' anzugeben. In diesem Fall kann man eine Wählverbindung nur über den Imonc-Client oder das Web-Interface auslösen.

Standard-Einstellung: `DIALMODE`='auto'

4. Pakete

Neben der Basisinstallation (BASE) gibt es weitere Pakete. Diese enthalten ein oder mehrere "OPTs"[1], die bei Bedarf zu BASE hinzuinstalliert werden können. Einige dieser Pakete sind Bestandteil des Basis-Paketes, andere kommen separat. Eine Übersicht über die vom fli4l-Team bereitgestellten Pakete finden Sie auf der Download-Seite (http://www.fli4l.de/download/stabile-version/), die von anderen Autoren bereitgestellten Pakete sind in der OPT-Datenbank (http://extern.fli4l.de/fli4l_opt-db3/) zu finden. Im folgenden werden die vom fli4l-Team bereitgestellten Pakete beschrieben.

4.1. Werkzeuge im Basispaket

Im Basispaket befinden sich die folgenden OPTs:

Name	Beschreibung
OPT_SYSLOGD	Werkzeug zum Protokollieren von Systemmeldungen (Seite 76)
OPT_KLOGD	Werkzeug zum Protokollieren von Kernelmeldungen (Seite 78)
OPT_LOGIP	Werkzeug zum Protokollieren von WAN-IP-Adressen (Seite 78)
OPT_Y2K	Datumskorrektur bei nicht Y2K-festen Rechnern (Seite 78)
OPT_PNP	Installation der isapnp-Werkzeuge (Seite 79)

4.1.1. OPT_SYSLOGD – Protokollieren von Systemmeldungen

Viele Programme verwenden die Syslog-Schnittstelle, um Meldungen auszugeben. Damit diese auch auf der Konsole sichtbar werden, muss in diesem Falle der Daemon syslogd gestartet werden.

Sind Debug-Meldungen gewünscht, stellt man OPT_SYSLOGD auf 'yes', ansonsten auf 'no'.
Siehe auch ISDN_CIRC_x_DEBUG (Seite 167) und PPPOE_DEBUG (Seite 112).
Standard-Einstellung: OPT_SYSLOGD='no'

SYSLOGD_RECEIVER Mit SYSLOGD_RECEIVER kann man festlegen, ob fli4l Syslog-Nachrichten vom Netzwerk empfangen soll oder nicht.

SYSLOGD_DEST_N SYSLOGD_DEST_x Mit SYSLOGD_DEST_x gibt man Ziele an, wohin die System-Meldungen, die von syslogd entgegengenommen werden, ausgegeben werden. Im Normalfall ist dies die Konsole von fli4l, also

```
SYSLOGD_DEST_1='*.* /dev/console'
```

Möchte man eine Datei als Ziel verwenden, ist z.B. einzutragen:

[1] Abk. für "OPTionales Modul"

```
SYSLOGD_DEST_1='*.* /var/log/messages'
```

Ist ein sog. Loghost im Netz vorhanden, können die Meldungen auch auf diesen Rechner umgeleitet werden – unter Angabe der IP-Adresse.
Beispiel:

```
SYSLOGD_DEST_1='*.* @192.168.4.1'
```

Das @-Zeichen ist dann der IP-Adresse voranzustellen.

Wenn die Systemmeldungen an mehrere Ziele "ausgeliefert" werden sollen, ist es nötig, die Variable `SYSLOGD_DEST_N` (Anzahl der Ziele) entsprechend zu erhöhen und die Variablen `SYSLOG_DEST_1`, `SYSLOG_DEST_2` usw. zu füllen.

Der Parameter '*.*' bedeutet, dass sämtliche Meldungen protokolliert werden. Man kann jedoch die Meldungen für bestimmte Ziele über sog. "Prioritäten" einschränken. In diesem Fall ersetzt man das Sternchen (*) hinter dem Punkt (.) durch eines der folgenden Schlüsselwörter:

- debug
- info
- notice
- warning (veraltet: warn)
- err (veraltet: error)
- crit
- alert
- emerg (veraltet: panic)

Die Reihenfolge in der Liste spiegelt dabei das "Gewicht" der Meldungen wider. Die Schlüsselwörter "error", "warn" und "panic" sind veraltet und sollten nicht mehr verwendet werden.

Vor dem Punkt kann eine sog. "Facility" statt des Sternchens (*) eingetragen werden. Eine Erklärung würde aber hier zu weit gehen. Der geneigte Leser kann hierfür eine Suchmaschine seiner Wahl bemühen. Eine Übesicht über mögliche Facilities finden sich auf den Manpages der syslog.conf:

http://linux.die.net/man/5/syslog.conf

Normalerweise ist das Sternchen aber vollkommen ausreichend. Beispiel:

```
SYSLOGD_DEST_1='*.warning @192.168.4.1'
```

Nicht nur Unix-/Linux-Rechner können als Loghost dienen, sondern auch Windows-Rechner. Auf http://www.fli4l.de/sonstiges/links/ findet man Verweise auf entsprechende Software. Die Verwendung eines Loghosts wird dringend empfohlen, wenn eine detaillierte Protokollierung gewünscht ist. Auch hilft die Protokollierung bei der Fehlersuche. Auch imonc "versteht" als Windows-Client das Syslog-Protokoll und kann die Meldungen in einem Fenster ausgeben.

Leider lassen sich die Boot-Meldungen von fli4l nicht über syslogd umlenken. Jedoch kann man fli4l auch so konfigurieren, dass die Konsole ein serielles Terminal (bzw. Terminalemulation) ist. Wie das geht, steht in dem Abschnitt Konsolen-Einstellungen (Seite 30).

SYSLOGD_ROTATE Mit SYSLOGD_ROTATE kann man festlegen, ob fli4l Syslog-Nachrichten einmal täglich rotiert. Es werden dabei die Meldungen der letzten x Tage archiviert.

SYSLOGD_ROTATE_DIR Mit der optionalen Variable SYSLOGD_ROTATE_DIR kann man festlegen, dass die rotierten Syslog-Dateien nicht in /var/log/ sondern in das angegebene Verzeichnis rotiert werden.

SYSLOGD_ROTATE_MAX Mit der optionalen Variablen SYSLOGD_ROTATE_MAX legt man die Anzahl der archivierten/rotierten Syslog-Dateien fest.

SYSLOGD_ROTATE_AT_SHUTDOWN Mit der optionalen Variable SYSLOGD_ROTATE_AT_SHUTDOWN kann man den das Rotieren beim Herunterfahren des Routers deaktivieren. Dies sollte man jedoch nur einstellen, wenn die syslog-Dateien bereits auf ein nichtflüchtiges Ziel geschrieben werden.

4.1.2. OPT_KLOGD – Protokollieren von Kernelmeldungen

Viele der auftretenden Fehler – z.B. fehlgeschlagene Einwahl – werden vom Linux-Kernel direkt auf die Konsole geschrieben. Mit OPT_KLOGD='yes' werden diese Meldungen an den syslogd umgelenkt, welcher diese dann entweder in Dateien protokollieren oder an einen Loghost weiterleiten kann, s.o. Dann hat man auf der fli4l-Konsole (fast) Ruhe.
Empfehlung: Wenn Sie OPT_SYSLOGD='yes' benutzen, sollte man auch OPT_KLOGD='yes' setzen.
Standard-Einstellung: OPT_KLOGD='no'

4.1.3. OPT_LOGIP – Protokollieren von WAN-IP-Adressen

Mit LOGIP ist es möglich, die WAN-IP in einer Log-Datei festzuhalten. Mit Angabe von OPT_LOGIP='yes' wird die Funktion aktiviert.
Standard-Einstellung: OPT_LOGIP='no'

LOGIP_LOGDIR Verzeichnis der Log-Datei festlegen

Mit LOGIP_LOGDIR wird das Verzeichnis festgelegt, in welchem die Log-Datei angelegt wird oder 'auto' für autodetect.

Standard-Einstellung: LOGIP_LOGDIR='auto'

4.1.4. OPT_Y2K – Datumskorrektur bei nicht Y2K-festen Rechnern

Meist werden fli4l-Router aus alten Hardware-Teilen zusammengesetzt. Dabei kann das BIOS des Mainboards nicht Y2K-fest sein. Das kann dazu führen, dass bei einer Einstellung des Datums auf den 27.05.2000 im BIOS beim nächsten Booten der 27.05.2094 im BIOS zu finden ist! Linux zeigt dann übrigens den 27.05.1994 an.

Eigentlich kann das eingestellte Datum für den fli4l-Router egal sein und sollte deshalb keine Rolle spielen. Wird fli4l jedoch als Least-Cost-Router eingesetzt, kann dies sehr wohl eine Rolle spielen.

Grund: Der 27.05.1994 war ein Freitag, der 27.05.2000 jedoch ein Samstag. Und am Wochenende gibt's günstigere Tarife bzw. günstigere Provider ...

Eine erste Lösung lautet: Das BIOS-Datum wird vom 27.05.2000 auf den 28.05.1994 gestellt. Das war ebenso ein Samstag. Damit ist das Problem aber noch nicht gänzlich gelöst, denn fli4l

verwendet nicht nur den Wochentag und die momentane Uhrzeit für das LC-Routing, sondern berücksichtigt auch die Feiertage.

Y2K_DAYS – N Tage auf Systemdatum addieren

Da das gesetzte Datum genau 2191 Tage zum tatsächlichen Datum differiert, werden bei Angabe von

```
Y2K_DAYS='2191'
```

auf das BIOS-Datum 2191 Tage addiert und dieses dann als Linux-Datum gesetzt. Das BIOS-Datum bleibt davon jedoch unberührt, sonst würde beim nächsten Booten das Datum wieder das Jahr 2094 (bzw. 1994) aufweisen.

Es gibt noch eine Alternative:

Mit dem Zugriff auf einen Time-Server kann sich fli4l die aktuelle Datum/Uhrzeit aus dem Internet holen. Dafür steht das Paket CHRONY (Seite 94) zur Verfügung. Beide Einstellungen lassen sich kombinieren. Das ist sinnvoll, um mit Y2K_DAYS zunächst das Datum schon einmal zu korrigieren und anschließend über den Time-Server die genaue Uhrzeit einzustellen.

Wer mit Y2K keine Probleme hat: Variable OPT_Y2K='no' setzen und einfach nicht mehr darüber nachdenken ...

4.1.5. OPT_PNP – Installation von isapnp tools

Teilweise müssen ISAPnP-Karten über das Werkzeug "isapnp" konfiguriert werden. Dies betrifft insb. ISDN-Karten mit ISDN_TYPE 7, 12, 19, 24, 27, 28, 30 und 106 – aber nur, wenn es sich auch wirklich um ISAPnP-Karten handelt.

Zur Konfiguration ist die Erstellung einer Konfigurations-Datei etc/isapnp.conf notwendig.

Hier eine Kurzanleitung zur Erstellung:

- In <config>/base.txt die Variable OPT_PNP='yes' und MOUNT_BOOT='rw' setzen
- fli4l booten – die ISAPnP-Karte wird wahrscheinlich nicht erkannt
- Auf fli4l-Konsole eingeben:

  ```
  pnpdump -c >/boot/isapnp.conf
  umount /boot
  ```

  Damit ist die Konfiguration auf dem Boot-Medium gespeichert.

Weiter auf PC (Unix/Linux/Windows):

- Die Datei isapnp.conf vom Boot-Medium nach <config>/etc/isapnp.conf kopieren
- isapnp.conf mit Editor öffnen, bearbeiten und abspeichern
  Die vorgegebenen Werte kann man hier beibehalten oder auch durch andere ersetzen, die man aus den möglichen Werten wählt. Relevant sind dabei die folgenden Zeilen im folgenden Beispiel:

```
      #     Start dependent functions: priority acceptable
      #       Logical device decodes 16 bit IO address lines
      #             Minimum IO base address 0x0160
      #             Maximum IO base address 0x0360
      #             IO base alignment 8 bytes
      #             Number of IO addresses required: 8
1)       (IO 0 (SIZE 8) (BASE 0x0160))
      #       IRQ 3, 4, 5, 7, 10, 11, 12 or 15.
      #             High true, edge sensitive interrupt (by default)
2)       (INT 0 (IRQ 10 (MODE +E
```

1) – Hier kann als „BASE“ eine Adresse zwischen die angegebene Minimum und Maximum eingegeben werden, wobei man das „base alignment“ in betracht ziehen muss. Bei mehr als einer ISA-Karte im System muss immer darauf geachtet werden, dass es hier keine Überschneidungen gibt, achte dabei auch auf die benötigte Anzahl Adressen (number of addresses required).

2) – Hier kann aus der angezeigten Liste ein IRQ eingesetzt werden. Dabei ist 2(9), 3, 4, 5 und 7 eher eine schlechte Wahl, da sich diese normalerweise mit den Seriellen und Parallelen Schnittstellen bzw der Cascadierung ins Gehege kommen.
ISA Karten können IRQs nicht teilen, deshalb darf ein für diese Karte verwendeter nicht anderweitig belegt sein.

- Die entsprechenden Daten (IRQ/IO) in die <config>/isdn.txt übernehmen
- Es ist nötig in der <config>/base.txt die `OPT_PNP` Einstellung auf ‘yes’ zu belassen, anderenfalls werden die erforderliche Dateien nicht mit auf das Boot-Medium kopiert. Die Einstellung `MOUNT_BOOT` kann man beliebig ändern.
- Neues Boot-Medium erzeugen

Die automatisch generierte Datei ist im Unix-Format gespeichert und enthält keine CRs. Startet man unter Windows den Notepad-Editor, zeigt dieser alle Zeilen in einer einzigen Zeile an. Der DOS-Editor “edit” kann jedoch mit Unix-Dateien umgehen. Er speichert sie dann als DOS-Datei (mit CRs) ab.

Abhilfe:

- DOS-Box starten
- In das Verzeichnis <config>/etc wechseln
- Eingeben: edit isapnp.conf
- Datei bearbeiten und abspeichern

Anschließend kann die Datei auch wieder mit Notepad bearbeitet werden.

Man kann auch unter Windows einfach den Wordpad-Editor verwenden.

Die zusätzlich generierten CRs werden beim Booten von fli4l wieder herausgefiltert. Sie stören also nicht.

Zunächst sollte man versuchen, ohne `OPT_PNP` auszukommen. Wird die Karte nicht erkannt, sollte wie oben beschrieben vorgegangen werden.

Bei einem Update auf eine neuere fli4l-Version kann die früher erstellte Datei isapnp.conf weiter verwendet werden.

Standard-Einstellung: `OPT_PNP='no'`

4.2. Advanced Networking

Das Advanced Networking Paket beinhaltet die Möglichkeit den fli4l–Router um VLAN, Bonding und Bridging Funktionen zu erweitern. Zusätzlich gibt es die Möglichkeit EBTables (http://ebtables.sourceforge.net/) Unterstützung zu aktivieren. Damit wird es möglich einen transparenten Paketfilter aufzubauen.

Generell gilt bei allen Paketen aus dem advanced_networking Paket:

Dieses Paket ist nur für Anwender gedacht, die sich sehr gut im Bereich Netzwerke auskennen. Insbesondere sind fundierte Routingkenntnisse notwendig.

Gerade beim aktivieren der EBTables Unterstützung können sehr ungewöhnliche Probleme auftreten, wenn man sich nicht 100% mit den verschiedenen Wirkungsweisen von Layer 2 und 3 auskennt. Einige Paketfilterregeln arbeiten mit aktivierter EBTables Unterstützung vollkommen anders als gewohnt.

4.2.1. Broadcast Relay - Weiterleitung von IP Broadcasts

Mithilfe eines Broadcast Relays können IP Broadcasts über Interface-Grenzen hinweg weitergeleitet werden. Dies ist notwendig für Applikationen, die Geräte im Netzwerk mittels Broadcast ermitteln (z.B. QNAP Finder), da Broadcasts normalerweise von Routern nicht über Netzgrenzen hinweg weitergegeben werden. Durch Einrichtung eines Broadcast Relays kann dieses Problem umgangen werden.

Innerhalb eines Broadcast Relays werden Broadcasts immer an alle angeschlossenen Interfaces weitergeleitet. Das bedeutet, dass eine Einrichtung eines weiteren Broadcast Relays mit vertauschten Interfaces nicht notwenig ist. Außerdem sind mehrere Broadcast Relays, die das gleiche Interface beinhalten, nicht erlaubt.

OPT_BCRELAY Weiterleiten von Broadcasts

Default: `OPT_BCRELAY='no'`

Mit `'yes'` wird das Broadcast Relay Paket aktiviert. Die Einstellung `'no'` deaktiviert das Broadcast Relay Paket komplett.

BCRELAY_N Default: `BCRELAY_N='0'`

Die Anzahl der zu konfigurierenden Broadcast Relays.

BCRELAY_x_IF_N Default: `BCRELAY_x_IF_N='1'`

Anzahl der Interfaces, die diesem Broadcast Relay zugeordnet sind.

BCRELAY_x_IF_x Default: `BCRELAY_x_IF_x=""`

Name des Interfaces, das diesem Broadcast Relay zugeordnet werden soll.

Zur Verdeutlichung folgt ein Beispiel, bei dem der Rechner mit der Applikation (z.B. QNAP Finder) im internen Netzwerk (angeschlossen an `eth0`) hängt und das NAS sich in einem anderen Netzwerk (angeschlossen an `eth1`) befindet.

```
OPT_BCRELAY='yes'
BCRELAY_N='1'
BCRELAY_1_IF_N='2'
BCRELAY_1_IF_1='eth0'
BCRELAY_1_IF_2='eth1'
```

4.2.2. Bonding - mehrere Netzwerkkarten zusammenfassen zu einem Link

Unter Bonding versteht man das Zusammenfassen von mindestens zwei Netzwerkkarten, die auch unterschiedlichen Typs (also 3Com und Intel) und Geschwindigkeit (10 Mbit/s oder 100 Mbit/s) sein können, zu einer gemeinsamen Verbindung. Dabei können entweder entsprechende Linux Rechner direkt verbunden werden, oder eine Verbindung zu einem Switch aufgebaut werden. So kann z.B. ohne grossen Aufwand eine 200 Mbit/s Full-Duplex Verbindung vom fli4l-Router zu einem Switch geschaltet werden. Jeder der sich für Bonding interessiert sollte vorher die Dokumentation dazu im Kernelverzeichnis (bonding.txt) gelesen haben. Die Namen der Bondingeinstellungen entsprechen weitestgehend den dort verwendeten Namen. Unter dem Linuxkernel 2.6.x findet sich im Verzeichnis der Kernelsourcen unter Documentation/networking die Datei bonding.txt.

OPT_BONDING_DEV Default: `OPT_BONDING_DEV='no'`

Mit `'yes'` wird das Bonding Paket aktiviert. Die Einstellung `'no'` deaktiviert das Bonding Paket komplett.

BONDING_DEV_N Default: `BONDING_DEV_N='0'`

Die Anzahl der zu konfigurierenden Bondinggeräte.

BONDING_DEV_x_DEVNAME Default: `BONDING_DEV_x_DEVNAME="`

Der Name des Bondinggerätes das erstellt werden soll. Der Name muß dabei mit 'bond' beginnen und es muß eine Zahl ohne führende '0' folgen. Die Namen der Bondinggeräte müssen nicht mit '0' beginnen und müssen nicht aufeinanderfolgend sein. Mögliche Werte wären z.B. 'bond0', 'bond8' oder 'bond99'.

BONDING_DEV_x_MODE Default: `BONDING_DEV_x_MODE="`

Gibt eines der Bonding-Methoden an. Der Standardwert ist Round-Robin 'balance-rr'. Mögliche Werte sind hier aufgelistet:

balance-rr Round-Robin-Methode: Übermittle der Reihe nach über alle Slaves von ersten bis zum letzten. Diese Methode bietet sowohl Load- Balancing als auch Fehlertoleranz.

active-backup Aktives Backup: Nur ein Slave im Bond ist aktiv. Die anderen Slaves werden nur dann aktiviert, wenn der aktive Slave ausfällt. Die MAC-Adresse des Bonds ist nur auf einem Port (Netzwerkadapter) sichtbar, um den Switch nicht zu verwirren. Dieser Modus bietet Fehlertoleranz.

balance-xor XOR-Methode: Übermittle basierend auf der Formel [(Quell-MAC-Adresse XOR Ziel-MAC-Adresse) modulo Anzahl der Slaves]. Dadurch wird immer der selbe Slave für die selben Ziel-MAC-Adresse benutzt. Diese Methode bietet sowohl Load-Balancing als auch Fehlertoleranz.

broadcast Broadcast-Methode: Übermittelt alles auf allen Slave-Devices. Dieser Modus bietet Fehlertoleranz.

802.3ad Dynamische IEEE 802.3ad Verbindungsaggregation. Erstellt Aggregationsgruppen, die die selben Geschwindigkeits und Duplex- Einstellungen teilen. Übermittelt auf allen Slaves im aktiven Aggregator.

Voraussetzungen:

- Unterstützt für ethtool im Basistreiber,um Geschwindigkeit und Duplex-Status für jede Device abzufragen.
- Ein Switch, der dynamische IEEE 802.3ad Verbindungsaggregation unterstützt.

balance-tlb Adaptives Load-Balancing für ausgehende Daten: Kanal-Bonding, dass keine speziellen Features im Switch benötigt. Der ausgehende Netzwerktraffic wird entsprechend der momentanen Last (relativ zur Geschwindigkeit berechnet) auf jeden Slave verteilt. Eingehender Netzwerktraffic wird vom aktuellen Slave empfangen. Wenn der empfangende Slave ausfällt, übernimmt ein anderer Slave die MAC-Adresse des ausgefallenen Empfangsslaves.

Voraussetzungen:

Unterstützt für ethtool im Basistreiber,um Geschwindigkeit und Duplex-Status für jede Device abzufragen.

balance-alb Adaptives Load-Balancing: schliesst sowohl balance-tlb, als auch Eingehendes Load-Balancing (rlb) für IPV4 Traffic ein und benötigt keine speziellen Voraussetzungen beim Switch. Load- Balancing für eingehenden Traffic wird über ARP-Absprache erreicht. Der Bonding-Treiber fängt ARP-Antworten vom Server auf ihrem Weg nach aussen hin ab und überschreibt die Quell- Hardware-Adresse mit der eindeutigen HW-Adresse eines Slaves im Bond, so dass unterschiedliche Clients unterschiedliche HW-Adressen für den Server verwenden.

Eingehender Traffic von Verbindungen, die vom Server erstellt wurden wird auch verteilt. Wenn der Server ARP-Anfragen sendet, kopiert und speichert der Bonding-Treiber die Client-IP aus dem ARP. Wenn die ARP-Antwort des Client ankommt, wird seine HW-Adresse ermittelt und der Bonding-Treiber erstellt eine ARP-Antwort an diesen Client und ordnet ihn so zu einem Client im Bond zu. Ein Problematischer Effekt von ARP-Absprachen für die Lastverteilung ist, dass jedes Mal wenn eine ARP-Anfrage übermittelt wird, sie die HW-Adresse des Bonds benutzt. Also lernen die Clients die HW-Adresse des Bonds und der eingehende Traffic auf dem aktuellen Slave bricht zusammen. Diesem Umstand wird begegnet, indem Updates (ARP-Antworten) zu allen Clients mit ihrer jeweiligen HW-Adresse gesandt wird, sodass der Traffic wieder aufgeteilt ist. Eingehender Traffic wird auch dann neu aufgeteilt, wenn ein neuer Slave zum Bond hinzugefügt wird oder ein inaktiver Slave reaktiviert wird. Die Empfangslast wird der Reihe nach (Round-Robin) in der Gruppe der Slave mit der grössten Geschwindigkeit im Bond verteilt.

Wenn eine Verbindung wiederhergestellt wird oder ein neuer Slave zum Bond hinzukommt wird der eingehende Traffic neu auf alle aktiven Slaves im Bond verteilt, indem ARP-Antworten mit den ausgewählten MAC-Adressen zu jedem Client gesandt werden. Der Parameter updelay muss auf einen Wert grösser oder gleich der

Weiterleitungsverzögerung (forwarding delay) des Switchs eingestellt sein, sodass ARP-Antworten an die Clients nicht vom Switch geblockt werden.

Voraussetzungen:

- Unterstützt für ethtool im Basistreiber,um Geschwindigkeit und Duplex-Status für jede Device abzufragen.
- Unterstützung im Basistreiber, die HW-Adresse auch dann setzen zu können, wenn das Device offen ist. Das ist notwendig, damit immer ein Slave im Team die HW-Adresse des Bonds tragen kann, (der curr_active_slave) obwohl jeder Slave im Bond eine eigene, eindeutige HW-Adresse hat. Wenn der curr_active_slave ausfällt, wird seine HW-Adresse mit dem neuen curr_active_slave ausgetauscht.

BONDING_DEV_x_DEV_N Default: `BONDING_DEV_x_DEV_N='0'`

Gibt an, aus wievielen Geräten dieses Bondinggeräte besteht. Wenn z.B. ein Bondinggerät aus 'eth0' und 'eth1' gebildet werden soll muß hier eine '2' (für die beiden eth–Geräte) eingetragen werden.

BONDING_DEV_x_DEV_x Default: `BONDING_DEV_x_DEV_x="`

Der Name eines Gerätes, welches zu diesem Bondinggerät gehören soll. Ein möglicher Wert wäre z.B. 'eth0'. Bitte beachten Sie, dass ein Gerät, welches Sie für ein Bondinggerät benutzen, exklusiv dafür benutzt werden muß. Insbesondere ist es nicht möglich das Gerät für ein DSL-Modem, eine Bridge, ein VLAN oder in der base.txt zu benutzen.

BONDING_DEV_x_MAC Default: `BONDING_DEV_x_MAC="`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Ein Bondinggerät benutzt standardmäßig die MAC Adresse des ersten Gerätes, welches für das Bonding benutzt wird. Wenn Sie dies nicht wollen können Sie auch eine MAC Adresse angegeben, die das Bondinggerät benutzen soll.

BONDING_DEV_x_MIIMON Default: `BONDING_DEV_x_MIIMON='100'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Gibt an in welchen Zeitabständen (in Millisekunden) die einzelnen Verbindungen eines Bondinggerätes auf Ihren Linkstatus geprüft werden. Es wird also der Linkstatus jedes einzelnen Gerätes dieses Bondinggerätes alle x Millisekunden geprüft. Mit '0' wird die MIIMON Überwachung deaktiviert.

BONDING_DEV_x_USE_CARRIER Default: `BONDING_DEV_x_USE_CARRIER='yes'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Wenn eine Überwachung des Linkstatus per MIIMON aktiviert wird kann man hier auswählen, ob die Überwachung des Linkstatus durch die netif_carrier_ok() Funktion (bei der Einstellung 'yes') erfolgen soll, oder durch direkte Aufrufe von MII oder ETHTOOL ioctl() Systemaufrufen (mit der Einstellung 'no'). Die netif_carrier_ok() Methode ist effizienter, aber nicht alle Treiber unterstützen diese Methode.

BONDING_DEV_x_UPDELAY Default: `BONDING_DEV_x_UPDELAY='0'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Der Wert dieser Einstellung multipliziert mit der Einstellung von BONDING_DEV_x_MIIMON gibt an nach welcher Zeit eine Verbindung des Bondinggerätes aktiviert wird wenn der entsprechende Link (z.B. ein eth–Gerät) aufgebaut wurde. Damit wird eine Verbindung des Bondinggerätes solange aktiviert, bis der Linkstatus auf 'nicht verbunden' schaltet.

BONDING_DEV_x_DOWNDELAY Default: BONDING_DEV_x_DOWNDELAY='0'

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Der Wert dieser Einstellung multipliziert mit der Einstellung von BONDING_DEV_x_MIIMON gibt an nach welcher Zeit eine Verbindung des Bondinggerätes deaktiviert wird wenn der entsprechende Link (z.B. ein eth–Gerät) ausfällt. Damit wird also eine Verbindung des Bondinggerätes zeitweise deaktiviert, solange bis der Linkstatus wieder auf 'aktiv' schaltet.

BONDING_DEV_x_LACP_RATE Default: BONDING_DEV_x_LACP_RATE='slow'

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Wenn bei BONDING_DEV_x_MODE=" der Wert '802.3ad' eingestellt wird, kann man hier angeben wie oft die Linkinformationen mit dem Verbindungspartner (also einem Switch oder einem anderen Linuxrechner) ausgetauscht werden. 'slow' tauscht alle 30 Sekunden die Linkinformationen aus, bei 'fast' werden die Linkinformationen jede Sekunde ausgetauscht.

BONDING_DEV_x_PRIMARY Default: BONDING_DEV_x_PRIMARY="

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Wenn als Mode 'active-backup' eingestellt bestimmt man hiermit, welches Gerät primär als Ausgabegerät benutzt werden soll. Das ist vor allem sinnvoll, wenn die unterschiedlichen Geräte eine unterschiedliche Geschwindigkeit haben. Ein String (eth0, eth2, etc) der als Primäres Devices verwendet werden soll. Wenn ein Wert eingegeben wird, und das Device ist online, wird es als erstes Ausgabemedium benutzt. Nur wenn das Device offline ist, wird ein anderes Devices benutzt. Andernfalls, sobald ein Ausfall erkannt wird, wird ein neues Standardausgabemedium bestimmt. Dies ist dann praktisch, wenn ein Slave Vorrang gegenüber einem anderen haben soll - wenn bspw. ein Slave 1000 Mbit/s schnell ist und ein anderer 100 Mbit/s. Wenn der 1000 Mbit/s-Slave ausfällt und später wieder hergestellt wurde, kann es von Vorteil sein, dass der schnellere Slave wieder aktiv gesetzt werden kann, ohne beim 100 Mbit/s-Slave künstlich einen Ausfall herbei- zuführen.

BONDING_DEV_x_ARP_INTERVAL Default: BONDING_DEV_x_ARP_INTERVAL='0'

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Gibt die Frequenz in Millisekunden an nach dem die unter BONDING_DEV_x_ARP_IP_TARGET_x angegebenen IP–Adressen (bzw. deren ARP Antwort) geprüft werden. Wenn ARP-Überwachung im Load-Balancing-Mode (mode 0 or 2) genutzt werden soll, sollte der Switch so eingestellt werden, dass er alle Pakete gleich auf alle Verbindungen verteilt - wie etwa Round-Robin. Wenn der Switch so eingestellt ist, dass er die Pakete nach der XOR-Methode verteilt, werden alle Antworten der ARP-Ziele auf der selben Verbindung ankommen und das könnte bei den anderen Team- Mitgliedern zum Ausfall führen. ARP-Überwachung sollte nicht zusammen mit miimon verwandt werden. Wird als Wert 0 übergeben, ist ARP-Überwachung deaktiviert.

BONDING_DEV_x_ARP_IP_TARGET_N Default: `BONDING_DEV_x_ARP_IP_TARGET_N="`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Die Anzahl der IP–Adressen die für die ARP Prüfung benutzt werden sollen. Es können maximal 16 IP–Adressen überprüft werden.

BONDING_DEV_x_ARP_IP_TARGET_x Default: `BONDING_DEV_x_ARP_IP_TARGET_x="`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Hier wird jeweils eine IP-Adressen angegeben, wenn `BONDING_DEV_x_ARP_INTERVAL` > 0 ist. Diese werden als Ziele der ARP-Anfragen verwandt, die verschickt werden, um die Qualität der Verbindung zu den Zielen festzustellen. Geben sie diese Werte im Format ddd.ddd.ddd.ddd an. Damit ARP-Überwachung funktioniert, muss zumindest eine IP-Adresse angegeben werden.

4.2.3. VLAN - 802.1Q Unterstüzung

Die Unterstützung für VLAN nach 802.1Q ist nur in Verbindung mit entsprechenden Switches sinnvoll. Port-Based VLAN Switche sind *nicht* dafür geeignet. Eine allgemeine Einführung in das Thema VLAN findet sich unter http://www.hoteltravelmovie.de/chrisi/downloads/VLAN_802frame.pdf in deutscher Sprache. Gerade für den Einstieg in das Thema VLAN ist diese Seite geeignet. Auf der Seite http://de.wikipedia.org/wiki/VLAN finden sich auch noch ein wenig Informationen wo man etwas nachlesen kann.

Bitte beachten Sie, dass nicht jede Netzwerkkarte mit VLANs umgehen kann. Einige Netzwerkkarten können überhaupt nicht mit VLANs umgehen, andere benötigen eine angepaßte MTU und einige wenige Karten arbeiten vollkommen problemlos. Der Autor des advanced_networking Paketes benutzt Intelnetzwerkkarten mit dem 'e100' Treiber ohne jedes Problem, eine MTU Anpassung ist nicht notwendig. Der 3COM '3c59x' Treiber benötigt eine MTU Anpassung, die MTU muß auf 1496 eingestellt werden, sonst arbeitet die Karte nicht korrekt. Der 'starfire' Treiber arbeitet nicht korrekt wenn ein VLAN Gerät in einer Bridge aufgenommen wird. In diesem Fall können keine Pakete mehr empfangen werden. Wer also mit VLANs arbeiten will sollte sicherstellen, dass der jeweilige Linux Netzwerkkartentreiber VLANs auch korrekt unterstützt.

OPT_VLAN_DEV Default: `OPT_VLAN_DEV='no'`

Mit `'yes'` wird das VLAN Paket aktiviert, mit `'no'` wird es deaktiviert.

VLAN_DEV_N Default: `VLAN_DEV_N="`

Anzahl der zu konfigurierenden VLAN Geräte.

VLAN_DEV_x_DEV Default: `VLAN_DEV_x_DEV="`

Der Name des Gerätes, das an den VLAN fähigen Switch angeschlossen ist. Das kann z.B. `'eth0'`, `'br1'` oder `'eth2'` sein.

VLAN_DEV_x_VID Default: `VLAN_DEV_x_VID="`

Die VLAN ID, für welches das entsprechende VLAN Gerät erstellt werden soll. Der Name des VLAN Gerätes wird aus dem Prefix 'ethX' und der angehängten VLAN ID (ohne führende '0') erstellt. Wird hier z.B. '42' eingetragen existiert später auf dem fli4l–Router das VLAN Gerät `'eth0.42'`.

Die VLAN Geräte auf dem fli4l–Router heissen immer '<device>.<vid>'. Also wenn ich ein eth–Gerät habe was an einem VLAN–fähigen Switch angeschlossen ist und ich auf dem fli4l–Router die VLANs 10, 11 und 23 nutzen will konfiguriere ich 3 VLAN Geräte mit dem eth–Gerät als `VLAN_DEV_x_DEV='ethX'` und der jeweiligen VLAN ID unter `VLAN_DEV_x_VID="`. Aber wie immer sagt ein Beispiel mehr als tausend Worte, daher hier das passende Beispiel:

```
OPT_VLAN_DEV='yes'
VLAN_DEV_N='3'
VLAN_DEV_1_DEV='eth0'
VLAN_DEV_1_VID='10' # Ergibt device: eth0.10
VLAN_DEV_2_DEV='eth0'
VLAN_DEV_2_VID='11' # Ergibt device: eth0.11
VLAN_DEV_3_DEV='eth0'
VLAN_DEV_3_VID='23' # Ergibt device: eth0.23
```

Bitte immer daran denken die MTU aller beteiligten Geräte zu prüfen. Durch den VLAN Header werden die Frame 4 Bytes länger. Wenn es notwendig ist muss bei den entsprechenden Geräten die MTU auf 1496 geändert werden.

4.2.4. Device MTU - Anpassen der MTU

Unter seltenen Umständen kann es notwendig sein, die MTU eines Gerätes anzupassen. Z.B. einige nicht 100% VLAN kompatible Netzwerkkarten benötigen eine Anpassung der MTU. Bitte denken Sie daran, dass nur wenige Netzwerkkarten in der Lage sind Ethernetframes die größer als 1500 Bytes sind zu verarbeiten!

DEV_MTU_N Default: `DEV_MTU_N="`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Gibt die Anzahl der Geräte an deren MTU geändert werden soll.

DEV_MTU_x Default: `DEV_MTU_x="`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Der Gerätename dessen MTU geändert werden soll gefolgt von der einzustellenden MTU. Beide Angaben werden durch ein Leerzeichen getrennt. Um z.B. für `'eth0'` eine MTU von `'1496'` einzustellen geben Sie folgendes ein:

```
DEV_MTU_N='1'
DEV_MTU_1='eth0 1496'
```

4.2.5. BRIDGE - Ethernet Bridging für fli4l

Hierbei handelt sich um eine vollwertige Ethernet-Bridge, die bei Bedarf nach dem Spanning Tree Protokoll arbeiten kann. Für den Anwender scheint der Rechner an den konfigurierten Ports danach wie ein Layer 3 Switch zu arbeiten.

Weiterführende Informationen zum Thema Bridging finden Sie hier:

Die Homepage des Linux Bridge Projektes: http://bridge.sourceforge.net/.

Die ausführliche und verbindliche Beschreibung des Briding Standards: http://standards.ieee.org/getieee802/download/802.1D-2004.pdf. Vor allem die Informationen ab Seite 153

sind interessant. Bitte beachten Sie, dass der Linux Bridging Code noch nach dem Standard von 1998 arbeitet. Dort gibt es z.B. nur 16 Bit Werte für die Pathcost.

Hier kann man sich die unterschiedlichen Zeitwerte für das Spanning Tree Protocoll berchnen lassen: http://www.dista.de/netstpclc.htm

Wie STP arbeitet kann man auf dieser Seite anhand einiger netter Beispiele sehen: http://web.archive.org/web/20060114052801/http://www.zyxel.com/support/supportnote/ves1012/app/stp.htm

OPT_BRIDGE_DEV Default: `OPT_BRIDGE_DEV='no'`

Mit `'yes'` wird das Bridge Paket aktiviert, mit `'no'` wird es deaktiviert.

BRIDGE_DEV_BOOTDELAY Default: `BRIDGE_DEV_BOOTDELAY='yes'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Da eine Bridge mindestens 2 × `BRIDGE_DEV_x_FORWARD_DELAY` in Sekunden an Zeit benötigt, um aktiv zu werden, ist diese Zeitspanne abzuwarten, wenn die Devices beim Start von fli4l sofort benötigt werden, um z.B. Syslogmeldungen zu verschicken oder sich per DSL einzuwählen. Wird der Eintrag auf `'yes'` gelassen wird automatisch 2 × `BRIDGE_DEV_x_FORWARD_DELAY` gewartet. Werden die Bridges nicht direkt beim Start benötigt, sollten Sie hier den Wert `'no'` eintragen um den Startvorgang des fli4l-Routers zu beschleunigen.

BRIDGE_DEV_N Default: `BRIDGE_DEV_N='1'`

Die Anzahl der voneinander unabhängigen Bridges. Jede Bridge ist von den anderen vollkommen isoliert zu betrachten. Das gilt insbesondere für die Einstellung von `BRIDGE_DEV_x_STP`. Es wird pro Bridge ein virtuelles Device mit Namen `'br<nummer>'` angelegt.

BRIDGE_DEV_x_NAME Default: `BRIDGE_DEV_x_NAME="`

Der symbolische Name der Bridge. Dieser Name kann von anderen Paketen benutzt werden um die Bridge unabhängig von dem Gerätenamen zu benutzen.

BRIDGE_DEV_x_DEVNAME Default: `BRIDGE_DEV_x_DEVNAME="`

Jedes Bridgegerät braucht einen Namen in der Form von `'br<nummer>'`. Dabei darf <nummer> eine Zahl zwischen '0' und '99' ohne führende '0' sein. Mögliche Einträge sind also `'br0'`, `'br9'` oder `'br42'`. Die Namen können beliebig gewählt werden, die erste Bridge kann also `'br3'` heissen und die zweite `'br0'`.

BRIDGE_DEV_x_DEV_N Default: `BRIDGE_DEV_x_DEV_N='0'`

Wie viele Netzwerkgeräte gehören der Bridge an? Die Anzahl der Devices, die fest an die Bridge gebunden werden sollen. Kann auch `'0'` sein, wenn die Bridge nur als Platzhalter für eine IP-Adresse sein soll, die dann von einem an die Bridge gebundenen VPN-Tunnel übernommen werden soll.

BRIDGE_DEV_x_DEV_x_DEV Gibt an, welches Device an die Bridge gehängt werden kann. hier kann ein eth-Device (z.B. `'eth0'`), ein Bonding-Device (z.B. `'bond0'`) oder auch ein Vlan-Device eingetragen werden (z.B. `'vlan11'`). Ein hier eingebundenes Device darf nicht mehr an anderer Stelle verwendet werden und auch keine IP erhalten.

```
BRIDGE_DEV_1_DEV_N='3'
BRIDGE_DEV_1_DEV_1_DEV='eth0.11' #VLAN 11 auf eth0
BRIDGE_DEV_1_DEV_2_DEV='eth2'
BRIDGE_DEV_1_DEV_3_DEV='bond0'
```

BRIDGE_DEV_x_AGING Default: `BRIDGE_DEV_x_AGING='300'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Gibt an nach welcher Zeit alte Einträge in der MAC Tabelle der Bridge gelöscht werden. Wenn die hier angegebene Zeit in Sekunden keine Daten von dem Rechner mit der Netzwerkkarte empfangen oder verschickt worden sind wird diese entsprechende MAC Adresse aus der Bridge MAC Tabelle entfernt.

BRIDGE_DEV_x_GARBAGE_COLLECTION_INTERVAL Default: `BRIDGE_DEV_x_GARBAGE_COLLECTION_IN`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Gibt an nach wieviel Sekunden eine „Müllsammlung“ gemacht wird. Dabei werden alle dynamischen Einträge der Bridge auf ihre Gültigkeit geprüft und veraltete oder nicht mehr gültige Einträge entfernt. Insbesondere bedeutet dies, dass alte nicht mehr gültige Verbindungen entfernt werden.

BRIDGE_DEV_x_STP Default: `BRIDGE_DEV_x_STP='no'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Das Spanning Tree Protokoll erlaubt es, mehrere Verbindungen zu anderen Switches zu unterhalten. Dadurch wird eine Redundanz erziehlt, die die Funktionsfähigkeit des Netzwerkes im Falle eines Ausfalls einer Leitung gewährleistet. Ohne den Einsatz von STP sind redundante Leitungen zwischen Switches nicht möglich, dass Netzwerk würde nicht funktionieren. Das STP versucht immer die schnellstmögliche Verbindung zwischen zwei Switches zu benutzen, so dass auch der Einsatz von zwei unterschiedlich schnellen Leitungen sinnvoll ist. So könnte man z.B. eine 1 Gbit/s Verbindung als Hauptverbindung benutzen und eine zweite 100 Mbit/s Verbindung als Sicherheit.

Einen guten Artikel mit einigen Hintergrundinformationen finden Sie auf dieser Seite: http://de.wikipedia.org/wiki/Spanning_Tree_Protocol.

BRIDGE_DEV_x_PRIORITY Default: `BRIDGE_DEV_x_PRIORITY="`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Nur gültig wenn `BRIDGE_DEV_x_STP='yes'` gesetzt wird!

Welche Priorität hat diese Bridge? Die Bridge mit der geringsten Priorität in der aktuellen Landschaft gewinnt die Wahl zur Hauptbridge. Jede Bridge sollte eine unterschiedliche Priorität haben. Bitte beachten Sie, dass die Bridge mit der geringsten Priorität auch über die größte Bandbreite verfügen sollte, da diese alle 2 Sekunden (oder die unter `BRIDGE_DEV_x_HELLO`) Steuerpakete verschickt und auch der gesamte restliche Datenverkehr über sie abgewickelt wird.

Gültige Werte sind `'0'` bis `'61440'` in Schritten von 4096.

BRIDGE_DEV_x_FORWARD_DELAY Default: `BRIDGE_DEV_x_FORWARD_DELAY='15'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Nur gültig wenn `BRIDGE_DEV_x_STP='yes'` gesetzt wird!

Wenn ein Bridgeanschluß deaktiviert war und erneut aktiviert werden soll, oder auch wenn der Anschluß gerade neu zur Bridge hinzugekommen ist, dauert es die angegebene Zeitspanne (in Sekunden) × 2 bis der Anschluß Daten weiterleiten kann. Dieser Parameter ist maßgebend für die Dauer, die die Bridge benötigt um einer tote Verbindung zu erkennen. Die Zeitspanne wird nach folgender Formel in Sekunden berechnet:

`BRIDGE_DEV_x_MAX_MESSAGE_AGE`+(2 × `BRIDGE_DEV_x_FORWARD_DELAY`)

Daraus ergibt sich mit den Standardwerten: $20 + (2 \times 15) = 50$ Sekunden. Die Zeit bis eine tote Verbindung erkennt wird kann minimiert werden, wenn `BRIDGE_DEV_x_HELLO` auf 1 Sekunde und die `BRIDGE_DEV_x_FORWARD_DELAY` auf 4 Sekunden gestellt wird. Zusätzlich muss `BRIDGE_DEV_x_MAX_MESSAGE_AGE` auf 4 Sekunden eingestellt werden. Daraus ergibt sich folgender Wert: $4 + (2 \times 4) = 12$ Sekunden. Schneller geht es nicht.

BRIDGE_DEV_x_HELLO Default: `BRIDGE_DEV_x_HELLO='2'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Nur gültig wenn `BRIDGE_DEV_x_STP='yes'` gesetzt wird!

Die mit `BRIDGE_DEV_x_HELLO` angegeben Zeit ist der Zeitabstand in Sekunden, in dem die sogenannten Hello Nachrichten von der Hauptbridge verschickt werden. Diese Nachrichten sind für die automatische Konfiguration von STP notwendig.

BRIDGE_DEV_x_MAX_MESSAGE_AGE Default: `BRIDGE_DEV_x_MAX_MESSAGE_AGE='20'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Nur gültig wenn `BRIDGE_DEV_x_STP='yes'` gesetzt wird!

Die maximale Gültigkeitsdauer der letzten Hello Nachricht. Wenn innerhalb dieser Zeit (in Sekunden) keine neue Hello Nachricht empfangen wird, wird eine neue Wahl der Hauptbridge ausgelöst. Deshalb darf dieser Wert **nie** kleiner als 2 × `BRIDGE_DEV_x_HELLO` sein.

BRIDGE_DEV_x_DEV_x_PORT_PRIORITY Default: `BRIDGE_DEV_x_DEV_x_PORT_PRIORITY='128'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Nur gültig wenn `BRIDGE_DEV_x_STP='yes'` gesetzt wird!

Ist nur relevant, wenn mehrere Verbindungen mit der selben `BRIDGE_DEV_x_DEV_x_PATHCOST` zum selben Ziel führen. Ist dies der Fall wird die Verbindung mit der geringsten Priorität ausgewählt.

Gültige Werte sind '0' bis '240' in Schritten von '16'.

BRIDGE_DEV_x_DEV_x_PATHCOST Default: `BRIDGE_DEV_x_DEV_x_PATHCOST='100'`

Diese Einstellung ist optional und kann auch komplett weggelassen werden.

Nur gültig wenn `BRIDGE_DEV_x_STP='yes'` gesetzt wird!

Bestimmt indirekt die Bandbreite für diese Verbindung. Je geringer der Wert, desto höher ist die Bandbreite und damit wird die Verbindung höher priorisiert.

Die vorgeschlagene Berechnungsgrundlage ist 1000000 / kbit/s, was zu den in Tabelle 4.1 aufgelisteten Werten führt. Bitte beachten Sie, dass bei der Berechnung die tatsächlich nutzbare Bandbreite in die Formel eingesetzt werden muss. Dadurch ergeben sich vor allem für WLAN deutliche niedrigere Werte als man erwarten würde.

Hinweis: Der aktuelle IEEE Standard von 2004 benutzt für die Bandbreitenberechnung 32 Bitzahlen, die von Linux noch nicht unterstützt werden.

Bandbreite	Einstellung von `BRIDGE_DEV_x_DEV_x_PATHCOST`
64 kbit/s	15625
128 kbit/s	7812
256 kbit/s	3906
10 Mbit/s	100
11 Mbit/s	190
54 Mbit/s	33
100 Mbit/s	10
1 Gbit/s	1

Tabelle 4.1.: Werte für BRIDGE_DEV_x_DEV_x_PATHCOST in Abhängigkeit von der Bandbreite

4.2.6. Anmerkungen

Eine Bridge leitet jede Art von Ethernetdaten weiter - somit läßt sich z.B. auch ein normales DSL-Modem über WLAN ansprechen als hätte es eine WLAN Schnittstelle. Es wird kein Paket, welches die Bridge passiert auf irgendwelche unerwünschten Aktivitäten hin untersucht (das heisst der fli4l Paketfilter wird nicht aktiv!), wodurch der Einsatz z.B. als WLAN Access Point nur unter sorgfältiger Abwägung der Sicherheitsrisiken zu empfehlen ist. Es gibt allerdings auch die Möglichkeit die EBTables Unterstützung zu aktivieren.

4.2.7. EBTables - EBTables für fli4l

Ab der Version 2.1.9 hat fli4l auch rudimentäre EBTables Unterstützung. Mit `OPT_EBTABLES='yes'` wird die EBTables Unterstützung aktiviert. Damit werden alle EBTables Kernelmodule geladen und das ebtables Programm auf dem fli4l–Router zur Verfügung gestellt. Im Gegensatz zur wesentlich vereinfachten Konfiguration des netfilter durch die verschiedenen Filterlisten von fli4l ist es notwendig selbstständig ein ebtables Skript zu schreiben. Das bedeutet, sie müssen das komplette ebtables Skript selbst schreiben.

Für Hintergrundinformationen zu der EBTables Unterstützung lesen sie bitte die Dokumentation auf der EBTables Homepage unter http://ebtables.sourceforge.net.

Es gibt die Möglichkeit ebtables Kommandos vor und nach dem Einrichten des netfilters (die `PF_INPUT_x`, `PF_FORWARD_x` usw) auf dem fli4l–Router abzusetzen. Legen Sie dazu je nach Bedarf im Verzeichnis config/ebtables die Dateien ebtables.pre und ebtables.post an. Die Datei ebtables.pre wird vor der Konfiguration des netfilters ausgeführt, die ebtables.post Datei danach. Bitte bedenken sie, dass ein Fehler in den ebtables Skripten unter Umständen den Startvorgang des fli4l–Routers unterbricht!

Vor Einsatz der EBTables Unterstützung sollten sie unbedingt die komplette EBTables Dokumentation lesen. Durch den Einsatz von EBTables ändert sich das

Verhalten des fli4l–Router unter Umständen! So funktioniert z.B. das mac: Filtern in der PF_FORWARD nicht mehr wie gewohnt.

Sehr interessant ist folgende Seite, die einen kleinen Einblick in die Funktionsweise der EBTables Unterstützung bringt: http://ebtables.sourceforge.net/br_fw_ia/br_fw_ia.html.

4.2.8. ETHTOOL - Einstellungen für Ethernet-Netzwerkadapter

Mit `OPT_ETHTOOL='yes'` wird das ethtool-Programm mit auf den fli4l kopiert und stellt es so zur Nutzung durch andere Pakete zu Verfügung. Mit Hilfe dieses Programms können diverse Einstellungen von Ethernet-Netzwerkkarten und -treibern angezeigt und verändert werden.

ETHTOOL_DEV_N Hier kann die Anzahl der Einstellungen angegeben werden, die beim Booten gesetzt werden.

Default: `ETHTOOL_DEV_N='0'`

ETHTOOL_DEV_x `ETHTOOL_DEV_x` gibt an, für welches Netzwerkgerät die Einstellung gelten soll.

Beispiel: `ETHTOOL_DEV_1='eth0'`

ETHTOOL_DEV_x_OPTION_N `ETHTOOL_DEV_x_OPTION_N` gibt die Anzahl der Einstellungen für das Gerät an.

ETHTOOL_DEV_x_OPTION_x_NAME

ETHTOOL_DEV_x_OPTION_x_VALUE Die Variable `ETHTOOL_DEV_x_OPTION_x_NAME` gibt den Namen und `ETHTOOL_DEV_x_OPTION_x_VALUE` den Wert für die zu ändernde Einstellung an.

Hier ist eine Liste der Optionen und möglichen Werte, die zur Zeit aktiviert sind:

- speed 10|100|1000|2500|10000 jeweils erweiterbar mit HD oder FD (default FD = Full-Duplex)
- autoneg on|off
- advertise %x
- wol p|u|m|b|a|g|s|d

Beispiel:

```
OPT_ETHTOOL='yes'
ETHTOOL_DEV_N='2'
ETHTOOL_DEV_1='eth0'
ETHTOOL_DEV_1_OPTION_N='1'
ETHTOOL_DEV_1_OPTION_1_NAME='wol'
ETHTOOL_DEV_1_OPTION_1_VALUE='g'
ETHTOOL_DEV_2='eth1'
ETHTOOL_DEV_2_OPTION_N='2'
ETHTOOL_DEV_2_OPTION_1_NAME='wol'
ETHTOOL_DEV_2_OPTION_1_VALUE='g'
ETHTOOL_DEV_2_OPTION_2_NAME='speed'
ETHTOOL_DEV_2_OPTION_2_VALUE='100hd'
```

Für nähere Informationen ist die Dokumentation von ethtool zu Rate zu ziehen: http://linux.die.net/man/8/ethtool

4.2.9. Beispiel

Für das Verständnis ist ein einfaches Beispiel sicher hilfreich. Wir gehen in unserem Beispiel von 2 Gebäudeteile aus, die mit 2 x 100 Mbit/s Verbindungen verbunden sind. Es sollen darüber 4 getrennte Netze von einem Gebäude in das andere geroutet werden.

Um dies zu verwirklichen, bietet sich eine Kombinatinon aus Bonding (die 2 vorhandenen 100 Mbit/s Verbindungen in deren Übertragungsleistung zusammenfassen), VLAN (um mehrere getrennte Netze auf einer zusammengefassten Leitung transportieren zu können) und Bridging (um die Netze in den Gebäuden in das Bond/VLAN Gebilde einhängen zu können. (erfolgreich getestet mit 2x Intel e100 Karten und 1x Adaptec 4-Port Karte ANA6944.) Die beiden e100 haben in diesem Beispiel die Gerätenamen `'eth0'` und `'eth1'` und werden für die Gebäudeverbindung verwendet. Aktuell sind uns keine anderen Kartentypen außer die Intel e100 bekannt, die reibungslos mit VLAN zusammenarbeiten. Gigabit-Karten sollten aber prinzipiell auch funktionieren. Die 4 Anschlüsse der Multiportkarte dienen für die jeweiligen Netzwerke und haben die Gerätenamen `'eth2'` bis `'eth5'`.

Zuerst werden die beiden 100 Mbit/s Strecken gebondet:

```
OPT_BONDING_DEV='yes'
BONDING_DEV_N='1'
BONDING_DEV_1_DEVNAME='bond0'
BONDING_DEV_1_MODE='balance-rr'
BONDING_DEV_1_DEV_N='2'
BONDING_DEV_1_DEV_1='eth0'
BONDING_DEV_1_DEV_2='eth1'
```

Dadurch wir das Gerät `'bond0'` erzeugt. Auf diese Bondingverbindung werden jetzt die VLANs aufgebaut, wir verwenden im Beispiel die VLAN-IDs 11, 22, 33 und 44:

```
OPT_VLAN_DEV='yes'
VLAN_DEV_N='4'
VLAN_DEV_1_DEV='bond0'
VLAN_DEV_1_VID='11'
VLAN_DEV_2_DEV='bond0'
VLAN_DEV_2_VID='22'
VLAN_DEV_3_DEV='bond0'
VLAN_DEV_3_VID='33'
VLAN_DEV_4_DEV='bond0'
VLAN_DEV_4_VID='44'
```

Und über diese VLAN Verbindungen wird nun eine Bridge in die einzelnen Netzwerksegmente gelegt. Ein Routing ist somit nicht notwendig.

```
OPT_BRIDGE_DEV='yes'
BRIDGE_DEV_N='4'
BRIDGE_DEV_1_NAME='_VLAN11_'
BRIDGE_DEV_1_DEVNAME='br11'
BRIDGE_DEV_1_DEV_N='2'
BRIDGE_DEV_1_DEV_1='bond0.11'
```

```
BRIDGE_DEV_1_DEV_2='eth2'
BRIDGE_DEV_2_NAME='_VLAN22_'
BRIDGE_DEV_2_DEVNAME='br22'
BRIDGE_DEV_2_DEV_N='2'
BRIDGE_DEV_2_DEV_1='bond0.22'
BRIDGE_DEV_2_DEV_2='eth3'
BRIDGE_DEV_3_NAME='_VLAN33_'
BRIDGE_DEV_3_DEVNAME='br33'
BRIDGE_DEV_3_DEV_N='2'
BRIDGE_DEV_3_DEV_1='bond0.33'
BRIDGE_DEV_3_DEV_2='eth4'
BRIDGE_DEV_4_NAME='_VLAN44_'
BRIDGE_DEV_4_DEVNAME='br44'
BRIDGE_DEV_4_DEV_N='2'
BRIDGE_DEV_4_DEV_1='bond0.44'
BRIDGE_DEV_4_DEV_2='eth5'
```

Damit sind jetzt alle 4 Netze vollkommen transparent miteinander verbunden und teilen sich die 200 Mbit/s Verbindung. Selbst bei Ausfall einer 100 Mbit/s Verbindung funktioniert die Verbindung noch. Bei Bedarf kann auch noch die EBTablesunterstützung aktiviert werden um z.B. bestimmte Paketfilter zu aktivieren.

Diese Konfiguration wird auf zwei fli4l-Routern eingerichtet. Ich denke dieses Beispiel zeigt eindrucksvoll welche Möglichkeiten das advanced_networking Paket ermöglicht.

4.3. CHRONY - Network Time Protocol Server/Client

OPT_CHRONY erweitert fli4l um das Network Time Protocol (Seite 96) (NTP). Dies ist nicht mit dem *normalen* Time Protokoll zu verwechseln, welches das alte OPT_TIME bereitstellt. Die Protokolle sind nicht kompatibel und somit werden gegebenenfalls neue Client-Programme, die NTP beherrschen, benötigt. Falls man nicht auf das einfache Time Protokoll verzichten kann, so läßt sich dieses Protokoll zusätzlich aktivieren.
OPT_CHRONY arbeitet sowohl im Server, als auch im Client Modus. In der Funktion des Client gleicht es die Zeit des fli4l mit Zeitreferenzen (Time Server) im Internet ab. In der Grundeinstellung nutzt OPT_CHRONY bis zu drei verschiedene Time Server aus dem Fundus von pool.ntp.org (Seite 96). Es ist jedoch auch möglich, über die Konfigurationsdatei eine andere Auswahl an Time Servern zu treffen. So ist es beispielsweise sinvoll Server aus Europa zu wählen. Möglich ist das, indem man als Server de.pool.ntp.org angibt, wenn der Router bzw. der Provider in Deutschland ist. Weitere Informationen dazu auf der Webseite von pool.ntp.org (Seite 96).

In der Funktion des Server dient OPT_CHRONY als Zeitreferenz für das lokale Netzwerk (LAN). NTP arbeitet auf Port 123.

Chrony zeichnet sich dadurch aus, dass es keine fortlaufende Verbindung zum Internet benötigt. Sobald die Verbindung getrennt wird (offline), erhält chrony hiervon Kenntnis und stellt den Zeitabgleich mit den Internet Time Servern ein. Somit löst chrony keinen neuen Verbindungsaufbau aus. Weiterhin verhindert chrony nicht die automatische Verbindungsunterbrechung, falls die `HUP_TIMEOUT`, also die Zeit, in der keine Daten mit dem Internet ausgetauscht werden, erreicht wurde.

Damit der Zeitabgleich reibungslos funktioniert, sollte folgendes beachtet werden:

- Chrony erwartet, dass die BIOS-Uhr in der Zeitzone UTC läuft. Falls nicht, muß dies in der Konfigurationsdatei geändert werden!
  UTC = Deutsche Zeit minus 1 (Winter) bzw. 2 (Sommer) Stunde(n)
- Seit der Version 2.1.12 setzt Chrony die Uhrzeit mit der ersten Verbindung zum Internet korrekt, auch wenn der Zeitunterschied sehr groß sein sollte (beispielsweise bei defekter Mainboardbatterie).
- Sollte das BIOS Jahreszahlen nach 1999 nicht richtig darstellen können (Year 2000 Bug) bzw. die Implementation der BIOS Uhr fehlerhaft sein, so muß `OPT_Y2K='yes'` (Seite 78) aktiviert werden!

Es können nur Time Server im Internet über die Default Route (`0.0.0.0/0`) erreicht werden, da nur die Default Route Chrony in den online Zustand versetzt. Ist der Router als LAN-Router konfiguriert, also keine DSL oder ISDN Circuits definiert, ist Chrony permanent im online Zustand.

Disclaimer: *Der Autor gibt weder eine Garantie auf die Funktionsfähigkeit des OPT_-CHRONY, noch haftet er für Schäden, z.B. Datenverlust, die durch den Einsatz von OPT_-CHRONY entstehen.*

4.3.1. Konfiguration des OPT_CHRONY

Die Konfiguration erfolgt, wie bei allen fli4l Opts, durch Anpassung der Datei `Pfad/fli4l-3.10.4/<config>/chrony.txt` an die eigenen Anforderungen. Jedoch sind fast alle Variablen des OPT_CHRONY optional. Optional heißt, die Variablen können, müssen aber nicht in der Konfigurationsdatei auftauchen. Somit ist die chrony Konfigurationsdatei im Auslieferungszustand fast leer und die optionalen Variablen sind sinnvoll vorbelegt. Möchte man dennoch eine anderen Konfiguration, müssen die Variablen von Hand eingefügt werden. Im weiteren erfolgt nun die Beschreibung der einzelnen Variablen:

OPT_CHRONY Default: `OPT_CHRONY='no'`

Die Einstellung `'no'` deaktiviert das OPT_CHRONY vollständig. Es werden keine Änderungen an dem fli4l Bootmedium bzw. dem Archiv `opt.img` vorgenommen. Weiter überschreibt das OPT_CHRONY grundsätzlich keine anderen Teile der fli4l Installation, mit einer Ausnahme. Es wird die Filterdatei ausgetauscht, die dafür sorgt, das Anfragen von außen nicht als Traffic gewertet werden (fli4l legt sicher nach erreichen der Hangup Time auf). Die neue Filterdatei legt fest, dass der chrony-Traffic ebenfalls nicht mitgezählt wird, somit legt der Router weiterhin sicher auf.
Um OPT_CHRONY zu aktivieren, ist die Variable `OPT_CHRONY` auf `'yes'` zu setzen.

CHRONY_TIMESERVICE Default: `CHRONY_TIMESERVICE='no'`

Mit `CHRONY_TIMESERVICE` kann ein weiteres Protokoll zur Zeitübermittlung aktiviert werden. Dieses ist nur dann nötig, wenn die lokalen Rechner nicht mit NTP arbeiten können. Das zusätzliche Protokoll ist RFC 868 konform und arbeitet auf Port 37. Wenn immer möglich, sollte NTP vorgezogen werden.

Einen herzlichen Dank an Christoph Schulz, der das Programm `srv868` beigesteuert hat.

CHRONY_TIMESERVER_N Default: `CHRONY_TIMESERVER_N='3'`

`CHRONY_TIMESERVER_N` legt die Anzahl der als Referenz benutzten Time Server fest. Der Anzahl entsprechend sind `CHRONY_TIMESERVER_x` Variablen anzulegen. Der Index `x` muß fortlaufend bis zur Gesamtanzahl heraufgezählt werden.
In der Grundeinstellung nutzt chrony drei Internet Time Server aus dem Fundus von pool.ntp.org (Seite 96).

CHRONY_TIMESERVER_x Default: `CHRONY_TIMESERVER_x='pool.ntp.org'`

Mit `CHRONY_TIMESERVER_x` kann eine eigene Liste von Internet Time Servern angelegt werden. Die Time Server können sowohl durch ihre IP als auch über ihren DNS Namen spezifiziert werden.

CHRONY_LOG Default: `CHRONY_LOG='/var/run'`

`CHRONY_LOG` bezeichnet das Verzeichnis, indem chrony Information über die BIOS Uhr und die Zeitkorrektur ablegt. Sollte in der Regel nicht verändert werden.

CHRONY_BIOS_TIME Default: `CHRONY_BIOS_TIME='utc'`

Damit chrony die Zeit der BIOS Uhr (RTC = real time clock) richtig auswerten kann, wird mittels `CHRONY_BIOS_TIME` übermittelt, ob die Uhr auf lokaler `'local'` oder universaler Zeit `'utc'` (UTC - Universal Coordinated Time) läuft.

4.3.2. Support

Support wird nur im Rahmen der fli4l Newsgroups (Seite 96) geleistet.

4.3.3. Literatur

Homepage von chrony: http://chrony.tuxfamily.org/
NTP: The Network Time Protocol: http://www.ntp.org/
pool.ntp.org: public ntp time server for everyone: http://www.pool.ntp.org/de/
RFC 1305 - Network Time Protocol (Version 3) Specification, Implementation:
http://www.faqs.org/rfcs/rfc1305.html
fli4l Newsgroups und ihre Spielregeln: http://www.fli4l.de/hilfe/newsgruppen/

4.4. DHCP_CLIENT - Dynamic Host Configuration Protocol

Mit Hilfe dieses Paketes kann der Router IP-Adressen für seine Interfaces dynamisch beziehen. Das Paket und seine Parameter werden im Folgenden erklärt.

4.4.1. OPT_DHCP_CLIENT

Ein DHCP-Client kann verwendet werden, um eine IP-Adresse für ein oder mehrere Interface(s) des Routers zu beziehen - dies erfolgt meist vom Service Provider. Diese Möglichkeit der Anbindung kommt derzeit hauptsächlich bei Kabelmodem-Betreibern und in der Schweiz, in den Niederlanden und in Frankreich vor. Manchmal wird diese Konfiguration auch benötigt, wenn der Router hinter einem anderen Router eingebunden wird, der die Adressen per DHCP verteilt (z.B. FritzBox).

Beim Start des Routers wird für die angegebenen Interfaces eine IP-Adresse bezogen. Anschließend wird diese dem Interface zugewiesen und bei Bedarf die Default- Route auf dieses Interface gelegt.

OPT_DHCP_CLIENT Muss auf 'yes' gesetzt werden, wenn einer der DHCP-Clienten verwendet werden soll.

Standard-Einstellung: `OPT_DHCP_CLIENT`='no'

DHCP_CLIENT_TYPE Das Paket kommt momentan mit zwei verschiedenen Implementierungen eines DHCP-CLienten, dem dhclient und dem dhcpcd. Hier kann man auswählen, welchen man verwenden möchte.

Standard-Einstellung: `DHCP_CLIENT_TYPE`='dhcpcd'

DHCP_CLIENT_N Hier wird die Anzahl zu konfigurierender Interfaces angegeben.

DHCP_CLIENT_x_IF Hier wird das zu konfigurierende Interface als Referenz auf `IP_NET_x_DEV` angegeben, z.B. `DHCP_CLIENT_1_IF='IP_NET_1_DEV'`. Der dhcp-client entnimmt der entsprechenden Variable das passende Device. In der base.txt sollte als Platzhalter anstelle einer IP-Adresse mit Netzmaske '*dhcp*' eingetragen werden.

DHCP_CLIENT_x_ROUTE Hier kann angegeben werden, ob und wie über das Interface eine Route konfiguriert werden soll. Die Variable kann auf folgende Werte gesetzt werden:

none Es wird keine Route über das Interface gelegt.

default Es wird eine default-Route über das Interface gelegt.

imond Der Imond managed die default-Route für dieses Interface.

Standard-Einstellung: `DHCP_CLIENT_x_ROUTE`='default'

DHCP_CLIENT_x_USEPEERDNS Wird dieser Parameter auf 'yes' gesetzt und über das Device eine default-Route gelegt, so wird der vom ISP über- gebene DNS als Forwarder für den DNS auf dem Router verwendet - dieser muss dafür natürlich aktiviert werden - siehe base.txt.

Standard-Einstellung: `DHCP_CLIENT_x_USEPEERDNS`='no'

DHCP_CLIENT_x_HOSTNAME Manche Provider verlangen eine Übermittlung eines Hostnamens. Dieser Name ist vom Provider zu erfahren und hier anzugeben. Es muß nicht identisch mit dem Hostnamen des Routers sein.

Standard-Einstellung: `DHCP_CLIENT_x_HOSTNAME`=''

DHCP_CLIENT_x_STARTDELAY Mit dieser Variable kann optional der Start des DHCP-Clients verzögert werden.

In machen Installation (z.B. fli4l als dhcp-Client hinter einem Kabelmodem, Fritzbox, ...) ist es notwendig, zu warten bis der genutzte DHCP-Server ebenfalls neu gestartet ist wenn es z.b. einen Stromausfall gab.

Standard-Einstellung: `DHCP_CLIENT_x_STARTDELAY`='0'

DHCP_CLIENT_x_WAIT Normalerweise wird der DHCP-Client im Hintergrund gestartet. Das bedeutet, dass der Boot-Prozess nicht durch das Ermitteln der IPv4-Adresse verzögert wird. Gelegentlich ist es jedoch notwendig, dass die Adresse konfiguriert ist, bevor der Boot-Vorgang voranschreitet, etwa wenn ein installiertes Paket zwingend eine konfigurierte Adresse benötigt (dies ist z. B. bei OPT_IGMP der Fall). In diesem Fall kann man `DHCP_CLIENT_x_WAIT='yes'` verwenden, um das Warten auf die Adresse zu erzwingen.

Standard-Einstellung: `DHCP_CLIENT_x_WAIT`='no'

DHCP_CLIENT_DEBUG Liefert weitere Informationen, wenn eine Adresse angefordert wird.

Standard-Einstellung: weglassen oder `DHCP_CLIENT_DEBUG`='no'

4.5. DNS_DHCP - DNS- und DHCP-Server sowie DHCP-Relay und Slave DNS Server

4.5.1. Hostnamen

Hosts

OPT_HOSTS Mit der optionalen Variable `OPT_HOSTS` kann die Konfiguration von Hostname deaktiviert werden!

HOST_N HOST_x_{attribute} Es sollten alle Rechner im LAN beschrieben werden - mit IP-Adresse, Namen, Aliasnamen und evtl. Mac-Adressen für die dhcp-Konfiguration . Dazu setzt man zunächst die Anzahl der Rechner mit der Variablen `HOST_N`.

Hinweis: Seit Version 3.4.0 wird der Eintrag für den Router aus den Angaben in der `<config>/base.txt` generiert. Sollen zusätzliche Aliasnamen aufgenommen werden, siehe auch `HOSTNAME_ALIAS_N` (Seite 72).

Anschließend werden mit den Attributen die Eigenschaften des Hostes definiert. Dabei sind einige Attribute Pflicht, wie z.B. IP-Adresse und Name, die anderen optional, d.h. man kann, aber man muß sie nicht spezifizieren.

NAME – Name des n-ten Hostes

IP4 – IP-Adresse (ipv4) des n-ten Hostes

IP6 – IP-Adresse (ipv6) des n-ten Hostes (optional). Wenn man “auto” verwendet, dann wird die Adresse automatisch berechnet – entweder wird die Adresse `::ffff:<IPv4>` verwendet, oder (bei aktiviertem OPT_IPV6) es wird eine “ordentliche” IPv6-Adresse gesetzt, bestehend aus einem IPv6-Präfix (mit /64er-Netzmaske) und der MAC-Adresse des jeweiligen Hosts. Damit das funktioniert, muss die MAC-Adresse via `HOST_x_MAC` gesetzt (siehe unten) und das ipv6-Paket entsprechend konfiguriert werden.

DOMAIN – DNS-Domain des n-ten Hostes (optional)

ALIAS_N – Anzahl der Alias-Namen des n-ten Hostes

ALIAS_m – m-ter Alias-Name für den n-ten Host

MAC – Mac Adresse des n-ten Hostes

DHCPTYP – Vergabe der IP-Adresse per DHCP abhängig von MAC oder NAME (optional)

In der Beispiel-Datei sind 4 Rechner konfiguriert - nämlich die PCs “client1”, “client2”, “client3” und “client4”.

```
HOST_1_NAME='client1'                # 1st host: ip and name
HOST_1_IP4='192.168.6.1'
```

Aliasnamen müssen mit kompletter Domain angegeben werden.

Die MAC-Adresse ist optional und ist nur dann relevant, wenn fli4l zusätzlich als DHCP-Server eingesetzt wird. Dies wird in der Beschreibung zum optionalen Programmpaket “OPT_DHCP” erklärt, siehe unten. Ohne Einsatz als DHCP-Server sind lediglich die IP-Adresse, der Name des Rechners und eventuell Aliasnamen einzusetzen. Die MAC-Adresse ist eine 48-Bit-Adresse und besteht aus 6 Hex-Werten, welche durch einen Doppelpunkt voneinander getrennt werden, z.B.

```
HOST_2_MAC='de:ad:af:fe:07:19'
```

Hinweis: Wird fli4l um das IPv6-Paket ergänzt, brauchen keine IPv6-Adressen hinterlegt zu werden, wenn gleichzeitig die MAC-Adressen der Hosts vorliegen, weil das IPv6-Paket dann die IPv6-Adressen automatisch berechnet (modifiziertes EUI-64). Natürlich kann man aber den Automatismus unterbinden und feste IPv6-Adressen vorgeben, wenn man dies wünscht.

Extra Hosts

HOST_EXTRA_N HOST_EXTRA_x_NAME HOST_EXTRA_x_IP4 HOST_EXTRA_x_IP6
Mit diesen Variablen können weitere Hosts hinzugefügt werden die nicht der lokalen Domain angehören wie z.b. Hosts die sich auf der anderen Seite eines VPNs befinden.

4.5.2. DNS-Server

OPT_DNS Um den DNS-Server zu aktivieren ist die Variable OPT_DNS mit ‘yes‘ zu belegen.

Werden im LAN keine Windows-Rechner verwendet oder ist bereits ein DNS-Server vorhanden, kann man OPT_DNS auf ‘no’ setzen und den Rest in diesem Abschnitt übergehen.

Im Zweifel immer (Standard-Einstellung): OPT_DNS=’yes’

Allgemeine DNS-Optionen

DNS_BIND_INTERFACES Mit der Einstellung ‘yes’ horcht dnsmasq *nicht* auf allen IP-Adressen und bindet sich nur an die IP-Adressen die unter DNS_LISTEN konfiguriert sind. Mit der Einstellung ‘no’ horcht der dnsmasq auf allen Interfaces und IP-Addressen und verwirft Anfragen an IP-Adressen auf denen dnsmasq eigentlich nicht reagieren soll. Das macht Probleme wenn man auf unterschiedlichen IP-Adressen unterschiedliche DNS Server nutzen möchte. Unterschiedliche DNS Server auf dem fli4l machen z.B. dann Sinn, wenn man einen Slave DNS Server direkt auf dem fli4l Router betreiben will. Will man

also nicht nur exklusiv den dnsmasq auf dem fli4l einsetzen muss die Einstellung 'yes' gewählt werden und die für den dnsmasq zu nutzenden IP-Adressen per DNS_LISTEN konfiguriert werden.

DNS_LISTEN_N DNS_LISTEN_x Wenn Sie `OPT_DNS`='yes' gewählt haben, können Sie mit Hilfe von `DNS_LISTEN_N` die Anzahl, und mit `DNS_LISTEN_1` bis `DNS_LISTEN_N` lokale IPs angeben, auf denen `dnsmasq` DNS-Anfragen annehmen darf. Sollten Sie bei `DNS_LISTEN_N` eine 0 eingetragen haben, beantwortet `dnsmasq` DNS-Anfragen auf allen lokalen IPs. An dieser Stelle dürfen nur IPs von existierenden Schnittstellen (ethernet, wlan ...) verwendet werden, es kommt sonst zu Warnmeldungen beim Start des Routers. Alternativ ist nun möglich hier auch ALIAS-Namen zu verwende, z. B. `IP_NET_1_IPADDR`

Für alle hier angegebenen Adressen werden bei `PF_INPUT_ACCEPT_DEF='yes'` und/oder `PF6_INPUT_ACCEPT_DEF='yes'` entsprechende ACCEPT-Regeln in der INPUT-Kette der Firewall erzeugt. Im Falle `DNS_LISTEN='0'` werden ebenfalls Regeln erzeugt, die den DNS-Zugriff auf *allen* konfigurierten Schnittstellen erlauben.

Im Zweifelsfalle können die Standardeinstellungen übernommen werden.

DNS_VERBOSE Logging von DNS-Queries: 'yes' oder 'no'

Für ausführlichere Ausgaben des DNS, muß `DNS_VERBOSE` auf yes gesetzt werden. In diesem Fall werden DNS-Anfragen an den Nameserver protokolliert - und zwar über die syslog-Schnittstelle. Damit die Ausgaben auch sichtbar werden, ist dann auch die Variable `OPT_SYSLOGD='yes'` (Seite 76) zu setzen, s.u.

DNS_MX_SERVER Mit dieser Variable gibt man hier den Hostnamen für den MX-Record (Mail-Exchanger) für die in `DOMAIN_NAME` definierte Domain an. Ein MTA (Mail-Transport-Agent, wie z.B. sendmail) auf einem internen Server fragt per DNS nach einem Mail-Exchanger für die Zieldomain der zuzustellenden Mail. Der DNS-Server liefert hiermit dem MTA den entsprechenden Host, der für Mails der Domain `DOMAIN_NAME` zuständig ist.

Dies ist keine automatische Konfiguration für Mail-Clients, wie z.B. Outlook! Also bitte nicht gmx.de hier eintragen und dann wundern, warum Outlook nicht funktioniert.

DNS_FORBIDDEN_N DNS_FORBIDDEN_x Hier können Sie Domains angeben, bei denen DNS-Queries vom DNS-Server prinzipiell als "nicht vorhanden" beantwortet werden sollen.

Beispiel:

```
DNS_FORBIDDEN_N='1'
DNS_FORBIDDEN_1='foo.bar'
```

In diesem Fall wird zum Beispiel eine Anfrage nach www.foo.bar mit einem Fehler beantwortet.

Man kann damit auch ganze Top-Level-Domains verbieten:

```
DNS_FORBIDDEN_1='de'
```

Dann ist die Namensauflösung für sämtliche Rechner in der DE-Topleveldomain abgeschaltet.

DNS_REDIRECT_N DNS_REDIRECT_x DNS_REDIRECT_x_IP Hier können Domains angegeben werden, bei welchen DNS-Queries vom DNS-Server auf eine spezielle IP umgeleitet werden.

Beispiel:

```
DNS_REDIRECT_N='1'
DNS_REDIRECT_1='yourdom.dyndns.org'
DNS_REDIRECT_1_IP='192.168.6.200'
```

In diesem Fall wird zum Beispiel eine Anfrage nach yourdom.dyndns.org mit der IP 192.168.6.200 beantwortet. Somit kann man externe Domains auf andere IPs umleiten.

DNS_BOGUS_PRIV Setzt man diese Variable auf ‘yes‘, werden reverse-lookups für IP-Adressen nach RFC1918 (Private Address Bereiche) nicht vom dnsmasq an andere DNS-Server weitergeleitet, sondern vom dnsmasq beantwortet.

DNS_FILTERWIN2K Setzt man diese Variable auf ’yes’, werden DNS-Anfragen vom Typ SOA, SRV und ANY geblockt. Dienste, die diese Anfragen verwenden, werden dann nicht mehr ohne weitere Konfiguration funktionieren.
Dazu zählen zum Beispiel:

- XMPP (Jabber)
- SIP
- LDAP
- Kerberos
- Teamspeak3 (seit Client-Version 3.0.8)
- Minecraft (seit Vollversion 1.3.1)
- Ermittlung des Domänencontrollers (Win2k)

Siehe hierzu auch:

- Generelle Erklärung der DNS Record Arten:
  http://en.wikipedia.org/wiki/List_of_DNS_record_types
- Manpage von dnsmasq:
  http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html
- SRV-Record im Speziellen:
  http://de.wikipedia.org/wiki/SRV_Resource_Record

Durch Setzen von ’no’ können durch die zusätzlichen weitergeleiteten DNS-Anfragen ungewollte Einwahlverbindungen aufgebaut oder bestehende nicht abgebaut werden. Insbesondere bei ISDN- und UMTS-Verbindungen können dadurch Mehrkosten entstehen. Sie müssen selbst abwägen, was für Sie wichtiger ist.

DNS_FORWARD_LOCAL setzt man diese Variable auf 'yes' kann der fli4l-Router in einer Domäne mit DOMAIN_NAME='example.local' konfiguriert werden, die wiederrum per DNS_ZONE_DELEGATION_x_DOMAIN='example.local' von einem anderen Nameserver aufgelöst wird.

DNS_LOCAL_HOST_CACHE_TTL Gibt die TTL (Time to live, in Sekunden) für Einträge aus den /etc/hosts Dateien und den per DHCP vergebenen IP-Adressen an. Der Standardwert für den fli4l-Router beträgt 60 Sekunden. Standardmäßig setzt der dnsmasq die TTL für lokale Einträge auf 0 und deaktiviert damit faktisch das nachfolgende Caching der DNS Einträge. Die Idee dahinter ist das ablaufende DHCP Leases usw. zeitnah weitergegeben werden können. Fragt allerdings z.B. ein lokaler IMAP Proxy die DNS Einträge dadurch mehrfach pro Sekunde ab ist das eine deutliche Belastung für das Netzwerk. Ein Kompromiss ist daher ein relativ kurzer TTL von 60 Sekunden. Es kann ja auch ohne die kurze TTL von 60 Sekunden jederzeit zu einem simplen abschalten eines Hosts kommen, so dass die abfragende Software sowieso mit nicht antwortetenden Hosts klarkommen muss.

DNS_SUPPORT_IPV6 (optional)

setzt man diese optionale Variable auf 'yes' wird die Unterstützung für IPV6- Adressen des DNS-Servers aktiviert.

DNS-Zonenkonfiguration

Der dnsmasq kann auch eine DNS-Domäne eigenständig verwalten, d.h. er ist "authoritativ" für diese Domäne. Dazu muss man zweierlei tun: Zum einen muss angegeben werden, welcher externe (!) DNS-Namensdienst auf den eigenen fli4l verweist und über welche Netzwerk-Schnittstelle dies passiert. Die Angabe der externen Referenz ist erforderlich, denn die Domäne, welche der fli4l verwaltet, ist ja immer eine Unterdomäne einer anderen Domäne.[2] Die Angabe der Schnittstelle ist wichtig, weil sich der dnsmasq dort "nach außen" anders verhält als auf den anderen Schnittstellen "nach innen": Nach außen beantwortet der dnsmasq niemals Anfragen für Namen außerhalb der konfigurierten eigenen Domäne. Nach innen funktioniert der dnsmasq natürlich auch als DNS-Relay ins Internet, damit die Auflösung von nicht-lokalen Namen funktioniert.

Zum anderen muss konfiguriert werden, welche Netze nach außen via Namensauflösung erreichbar sind. Hierbei sollten natürlich nur Netze mit öffentlichen IP-Adressen angegeben werden, denn über private Adressen können Hosts von außen ohnehin nicht erreicht werden.

Im Folgenden wird die Konfiguration an einem Beispiel beschrieben. Dieses Beispiel setzt das IPv6-Paket sowie ein öffentlich geroutetes IPv6-Präfix voraus; letzteres kann z.B. von einem 6in4-Tunnel-Provider wie SixXS oder Hurricane Electric bereitgestellt werden.

DNS_AUTHORITATIVE Die Einstellung `DNS_AUTHORITATIVE='yes'` aktiviert den authoritativen Modus des dnsmasq. Dies reicht jedoch nicht aus, da weitere Angaben gemacht werden müssen (s.u.).

Standard-Einstellung: `DNS_AUTHORITATIVE='no'`

Beispiel: `DNS_AUTHORITATIVE='yes'`

[2] Wir gehen hier mal davon aus, dass niemand einen fli4l als DNS-Rootserver verwendet...

DNS_AUTHORITATIVE_NS Mit dieser Variable wird der DNS-Name konfiguriert, über den auf den fli4l von außen mit Hilfe eines DNS-NS-Records verwiesen wird. Das kann auch ein DNS-Name sein, der zu einem Dynamic DNS-Dienst gehört.

Beispiel: `DNS_AUTHORITATIVE_NS='fli4l.noip.me'`

DNS_AUTHORITATIVE_IPADDR Mit dieser Variable wird konfiguriert, an welcher Adresse bzw. Schnittstelle der dnsmasq DNS-Anfragen für die eigene Domäne authoritativ beantwortet. Symbolische Namen wie `IP_NET_2_IPADDR` sind erlaubt. Der dnsmasq kann nur an *einer* Adresse/Schnittstelle authoritativ antworten.

Momentan können nur fest zugewiesene Adressen angegeben werden. Adressen, die sich erst durch eine Einwahl ergeben (z.B. mit Hilfe einer PPP-Verbindung), können nicht verwendet werden. Dies wird in einer späteren fli4l-Version korrigiert werden.

Wichtig: *Zu beachten ist, dass dies niemals eine Adresse/Schnittstelle sein darf, an der das eigene LAN hängt, weil sonst keine nicht-lokalen Namen mehr im LAN aufgelöst werden können!*

Beispiel: `DNS_AUTHORITATIVE_IPADDR='IP_NET_2_IPADDR'`

DNS_ZONE_NETWORK_N DNS_ZONE_NETWORK_x Hier werden die Netzadressen angegeben, für die der dnsmasq authoritativ die Namen auflösen soll. Dabei funktioniert sowohl die Vorwärts- (Name zu Adresse) als auch die Rückwärtsauflösung (Adresse zu Name).

Ein komplettes Beispiel:

```
DNS_AUTHORITATIVE='yes'
DNS_AUTHORITATIVE_NS='fli4l.noip.me'
DNS_AUTHORITATIVE_IPADDR='IP_NET_2_IPADDR' # Uplink hängt an eth1
DNS_ZONE_NETWORK_N='1'
DNS_ZONE_NETWORK_1='2001:db8:11:22::/64'   # lokales IPv6-LAN
```

Dabei wird angenommen, dass “2001:db8:11::/48” ein zu dem fli4l öffentlich geroutetes IPv6-Präfix ist und dass für das LAN das Subnetz 22 gewählt wurde.

DNS Zone Delegation

DNS_ZONE_DELEGATION_N DNS_ZONE_DELEGATION_x Es gibt besondere Situationen, wo die Angabe eines oder mehrerer DNS Server sinnvoll ist, z.B. bei Einsatz von fli4l im Intranet ohne Internetanschluss oder einem Mix von diesen (Intranet mit eigenem DNS Server und zusätzlich Internetanschluss).

Stellen wir uns folgendes Szenario vor:

- Circuit 1: Einwahl in das Internet
- Circuit 2: Einwahl in das Firmen-Netz 192.168.1.0 (firma.de)

Dann wird man `ISDN_CIRC_1_ROUTE` auf ‘0.0.0.0’ und `ISDN_CIRC_2_ROUTE` auf ‘192.168.1.0’ setzen. Bei Zugriff auf Rechner mit IP-Adresse 192.168.1.x wird fli4l dann den Circuit 2, sonst den Circuit 1 benutzen. Wenn das Firmennetz aber nicht öffentlich ist, wird in

diesem vermutlich ein eigener DNS Server betrieben. Nehmen wir an, die Adresse dieses DNS Servers wäre 192.168.1.12 und der Domainname wäre “firma.de”.

In diesem Fall gibt man an:

```
DNS_ZONE_DELEGATION_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_1_IP='192.168.1.12'
DNS_ZONE_DELEGATION_1_DOMAIN_N='1'
DNS_ZONE_DELEGATION_1_DOMAIN_1='firma.de'
```

Dann werden bei DNS Anfragen an die Domain firma.de der firmeninterne DNS Server benutzt. Alle anderen DNS Anfragen gehen wie üblich an die DNS Server im Internet.

Ein anderer Fall:

- Circuit 1: Internet
- Circuit 2: Firmen-Netz 192.168.1.0 *mit* Internetanschluss

Hier hat man also die Möglichkeit, auf 2 Wegen in das Internet zu gelangen. Möchte man geschäftliches und privates trennen, bietet sich dann folgendes an:

```
ISDN_CIRC_1_ROUTE='0.0.0.0'
ISDN_CIRC_2_ROUTE='0.0.0.0'
```

Man legt also auf beide Circuits eine Defaultroute und schaltet dann die Route mit dem imond-Client um - je nach Wunsch. Auch in diesem Fall sollte man `DNS_ZONE_DELEGATION_N` und `DNS_ZONE_DELEGATION_x_DOMAIN_x` wie oben beschrieben einstellen.

Möchte man auch die Reverse-DNS-Auflösung für ein so erreichbares Netz nutzen, z.B. wird ein Reverselookup von einigen Mailserver gemacht, gibt man in der optionalen Variable `DNS_ZONE_DELEGATION_x_NETWORK_x`, das/die Netz(werke) an, für die der Reverselookup aktiviert werden soll. Das folgende Beispiel verdeutlicht das:

```
DNS_ZONE_DELEGATION_N='2'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_1_IP='192.168.1.12'
DNS_ZONE_DELEGATION_1_DOMAIN_N='1'
DNS_ZONE_DELEGATION_1_DOMAIN_1='firma.de'
DNS_ZONE_DELEGATION_1_NETWORK_N='1'
DNS_ZONE_DELEGATION_1_NETWORK_1='192.168.1.0/24'
DNS_ZONE_DELEGATION_2_UPSTREAM_SERVER_N='1'
DNS_ZONE_DELEGATION_2_UPSTREAM_SERVER_1_IP='192.168.2.12'
DNS_ZONE_DELEGATION_2_DOMAIN_N='1'
DNS_ZONE_DELEGATION_2_DOMAIN_1='bspfirma.de'
DNS_ZONE_DELEGATION_2_NETWORK_N='2'
DNS_ZONE_DELEGATION_2_NETWORK_1='192.168.2.0/24'
DNS_ZONE_DELEGATION_2_NETWORK_2='192.168.3.0/24'
```

Mit der Konfigurationsoption `DNS_ZONE_DELEGATION_x_UPTREAM_SERVER_x_QUERYSOURCEIP` kann man die IP-Adresse für die ausgehenden DNS Anfragen an den oder die Upstream

DNS Server setzen. Das ist z.B. dann sinnvoll wenn man den Upstream DNS Server über ein VPN erreicht und nicht möchte, dass die lokale VPN Adresse vom fli4l-Router als Quell IP-Adresse beim Upstream DNS Server auftaucht. Ein anderer Anwendungsfall ist eine vom Upstream DNS Server aus gesehen nicht routebare IP-Adresse (die durch ein VPN Interface evtl. auftritt). Auch in diesem Fall ist es notwendig die vom dnsmasq benutzte ausgehende IP-Adresse fest auf eine vom fli4l-Router benutzte und vom Upstream DNS Server aus erreichbar IP-Adresse zu setzen.

```
DNS_ZONE_DELEGATION_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_1_IP='192.168.1.12'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_1_QUERYSOURCEIP='192.168.0.254'
DNS_ZONE_DELEGATION_1_DOMAIN_N='1'
DNS_ZONE_DELEGATION_1_DOMAIN_1='firma.de'
DNS_ZONE_DELEGATION_1_NETWORK_N='1'
DNS_ZONE_DELEGATION_1_NETWORK_1='192.168.1.0/24'
```

DNS_REBINDOK_N DNS_REBINDOK_x_DOMAIN Der Nameserver *dnsmasq* lehnt normalerweise Antworten anderer Nameserver ab, die IP-Adressen aus privaten Netzwerken enthalten. Er verhindert dadurch eine bestimmte Klasse von Angriffen auf das Netzwerk. Hat man allerdings eine Domain in einem Netzwerk mit privaten IP-Adressen und einen extra Nameserver, der für dieses Netz zuständig ist, liefert der genau die Antworten, die vom *dnsmasq* abgelehnt werden würden. Diese Domains kann man in `DNS_REBINDOK_x` auflisten, die entsprechenden Antworten auf Anfragen zu der Domain werden dann akzeptiert. Ein weiteres Beispiel für Nameserver, die private IP-Adressen als Antwort liefern, sind sogenannte "Real-Time Blacklist Server". Ein Beispiel basierend auf diesen könnte wie folgt aussehen:

```
DNS_REBINDOK_N='8'
DNS_REBINDOK_1_DOMAIN='rfc-ignorant.org'
DNS_REBINDOK_2_DOMAIN='spamhaus.org'
DNS_REBINDOK_3_DOMAIN='ix.dnsbl.manitu.net'
DNS_REBINDOK_4_DOMAIN='multi.surbl.org'
DNS_REBINDOK_5_DOMAIN='list.dnswl.org'
DNS_REBINDOK_6_DOMAIN='bb.barracudacentral.org'
DNS_REBINDOK_7_DOMAIN='dnsbl.sorbs.net'
DNS_REBINDOK_8_DOMAIN='nospam.login-solutions.de'
```

4.5.3. DHCP-Server

OPT_DHCP Mit `OPT_DHCP` kann man einstellen, ob der DHCP-Server aktiviert wird.

DHCP_TYPE (optional)

Mit dieser Variable legt man fest, ob man die interne DHCP-Funktion des dnsmasq benutzt, oder ob man auf den externen ISC-DHCPD zurückgreifen will. Im Falle des ISC-DHCPD entfällt der Support für DDNS.

DHCP_VERBOSE aktiviert zusätzliche DHCP-Ausgaben im log.

DHCP_LS_TIME_DYN legt die standard Lease-Time für dynamisch vergebene IP-Adressen fest.

DHCP_MAX_LS_TIME_DYN legt die maximale Lease-Time für dynamisch vergebene IP-Adressen fest.

DHCP_LS_TIME_FIX Standard Lease-Time für statisch zugeordnete IP-Adressen.

DHCP_MAX_LS_TIME_FIX legt die maximale Lease-Time für statisch zugeordnete IP-Adressen fest.

DHCP_LEASES_DIR legt das Verzeichnis für die Leases-Datei fest. Möglich ist die Angabe eines absoluten Pfades oder des Wertes *auto*. Bei Angabe von *auto* wird die lease-Datei im Unterverzeichnis dhcp des persistent-Verzeichnisses (siehe Base-Dokumentation) abgelegt.

DHCP_LEASES_VOLATILE Befindet sich das Verzeichnis für die *Leases* in der Ram-Disk (da der Router z.B. von CD oder einem anderen nicht schreibbaren Medium bootet), gibt der Router beim Booten eine Warnung wegen einer fehlenden *Lease*-Datei aus. Diese Warnung entfällt, wenn man `DHCP_LEASES_VOLATILE` auf *yes* setzt.

DHCP_WINSSERVER_1 legt die Adresse des ersten WINS-Server fest. Bei installiertem und aktiviertem WINS-Server wird die Adresse des WINS-Server des SAMBA-Paketes übernommen.

DHCP_WINSSERVER_2 legt die Adresse des zweiten WINS-Server fest. Bei installiertem und aktiviertem WINS-Server wird die Adresse von WINS-Server des SAMBA-Paketes übernommen.

Lokale DHCP-Range

DHCP_RANGE_N Anzahl der DHCP-Ranges

DHCP_RANGE_x_NET Referenz zu einem in `IP_NET_x` definiertem Netz

DHCP_RANGE_x_START legt die erste zu vergebende IP-Adresse fest.

DHCP_RANGE_x_END legt die letzte zu vergebende IP-Adresse fest. Die beiden Variablen `DHCP_RANGE_x_START` und `DHCP_RANGE_x_END` kann man auch leer lassen, es wird dann keine DHCP-Range angelegt und nur die weiteren Variablen genutzt, um einem Host der per MAC-Zuordnung seine DHCP-IP bezieht, die Werte der Variablen zu übergeben.

DHCP_RANGE_x_DNS_SERVER1 legt die Adresse des DNS-Server für DHCP-Hosts des Netzes fest. Diese Variable ist optional. Wird hier nichts eingetragen, oder die Variable einfach weggelassen, wird die IP-Adresse, des zugeordneten Netzes verwendet. Es ist auch möglich, diese Variable auf 'none' zu setzen. Dann wird kein DNS-Server übertragen.

DHCP_RANGE_x_DNS_SERVER2 legt die IP-Adresse des zweiten DNS-Servers fest. Es gelten die gleiche Option wie in der vorherigen Variable

DHCP_RANGE_x_DNS_DOMAIN legt eine spezielle DNS-Domain für DHCP-Hosts dieser Range fest. Diese Variable ist optional. Wird hier nichts eingetragen, oder die Variable einfach weggelassen, wird der Default DNS-Domain `DOMAIN_NAME` verwendet.

DHCP_RANGE_x_NTP_SERVER legt die Adresse des NTP-Servers für DHCP-Hosts dieser Range fest. Diese Variable ist optional. Wird hier nichts eingetragen, oder die Variable einfach weggelassen, wird die IP-Adresse des in `DHCP_RANGE_x_NET` referenzierten Netzes verwendet, wenn ein Zeitserverpaket auf dem Router aktiviert ist. Es ist auch möglich, diese Variable auf 'none' zu setzen. Dann wird kein NTP-Server übertragen.

DHCP_RANGE_x_GATEWAY legt die Adresse des Gateways für diese Range fest. Diese Variable ist optional. Wird hier nichts eingetragen, oder die Variable einfach weggelassen, wird die IP-Adresse des in `DHCP_RANGE_x_NET` referenzierten Netzes verwendet. Es ist auch möglich, diese Variable auf 'none' zu setzen. Dann wird kein Gatway übertragen.

DHCP_RANGE_x_MTU legt die MTU für Clients in diesem Range fest. Diese Variable ist optional.

DHCP_RANGE_x_OPTION_N gestattet die Angabe Nutzer-definierter Optionen für diesen Bereich. Die verfügbaren Optionen kann man dem Manual des dnsmasq entnehmen (`http://thekelleys.org.uk/dnsmasq/docs/dnsmasq.conf.example`). Sie werden ungeprüft übernommen, können also bei Fehlern zu Problemen mit dem DNS/DHCP-Server führen. Diese Variable ist optional.

Extra DHCP-Range

DHCP_EXTRA_RANGE_N legt die Anzahl von DHCP-Bereichen fest, die an nicht lokale Netze vergeben werden. Hierzu ist am Gateway zum entsprechenden Netz ein DHCP-Relay zu installieren.

DHCP_EXTRA_RANGE_x_START erste zu vergebende IP-Adresse.

DHCP_EXTRA_RANGE_x_END letzte zu vergebende IP-Adresse.

DHCP_EXTRA_RANGE_x_NETMASK Netzwerkmaske für diesen Bereich.

DHCP_EXTRA_RANGE_x_DNS_SERVER Adresse des DNS-Servers für diesen Bereich.

DHCP_EXTRA_RANGE_x_NTP_SERVER Adresse des NTP-Servers für diesen Bereich.

DHCP_EXTRA_RANGE_x_GATEWAY Adresse des Default-Gateway für diesen Bereich.

DHCP_EXTRA_RANGE_x_MTU legt die MTU für Clients in dieser Range fest. Diese Variable ist optional.

DHCP_EXTRA_RANGE_x_DEVICE Netzwerkinterface über den dieser Bereich erreicht wird.

Nicht zugelassene DHCP-Clients

DHCP_DENY_MAC_N Anzahl der MAC-Adressen von Host, dennen der Zugriff auf DHCP-Adressen verweigert wird.

DHCP_DENY_MAC_x MAC-Adresse des Hosts, dem der Zugriff auf DHCP-Adressen verweigert wird.

Unterstützung fürs Booten vom Netz

Der dnsmasq unterstützt Clients, die via Bootp/PXE übers Netz booten. Die dafür nötigen Informationen werden vom dnsmasq bereitgestellt und pro Subnetz und Host konfiguriert. Die dafür nötigen Variablen sind in den DHCP_RANGE_%- und HOST_%-Abschnitten untergebracht und beschreiben das zu bootende File (*_PXE_FILENAME), den Server, der dieses File bereitstellt (*_PXE_SERVERNAME und *_PXE_SERVERIP) und evtl. notwendige Optionen (*_PXE_OPTIONS). Weiterhin kann man den internen tftp-Server aktivieren, so dass das Booten komplett von dnsmasq unterstützt wird.

HOST_x_PXE_FILENAME DHCP_RANGE_x_PXE_FILENAME Hier wird das zu bootende Image angegeben. Im Falle von PXE wird hier der zu ladende pxe-Bootloader, wie z.B. pxegrub, pxelinux oder ein anderer passender Bootloader angegeben.

HOST_x_PXE_SERVERNAME HOST_x_PXE_SERVERIP DHCP_RANGE_x_PXE_SERVERNAME Name und IP des tftp-Servers, werden diese Variablen leer gelassen, wird der Router selbst als tftp-Server übermittelt.

DHCP_RANGE_x_PXE_OPTIONS HOST_x_PXE_OPTIONS Einige Bootloader benötigen spezielle Optionen zum Booten. So erfragt zum Beispiel pxegrub über die Option 150 den Namen der Menu-Datei. Diese Optionen können hier angegeben werden und werden dann ins Konfigfile übernommen. Im Falle von pxegrub könnte das z.B. wie folgt aussehen:

```
HOST_x_PXE_OPTIONS='150,"(nd)/grub-menu.lst"'
```

Sind mehrere Optionen nötig, werden sie einfach mit Leerzeichen voneinander getrennt angegeben.

4.5.4. DHCP-Relay

Das DHCP-Relay wird dann verwendet, wenn ein anderer DHCP-Server die Verwaltung der Ranges übernimmt, der nicht direkt von den Clients erreicht werden kann.

OPT_DHCPRELAY Dieser Wert ist auf 'yes' zu setzen, damit der Router als DHCP-Relay arbeitet. Es darf nicht gleichzeitig ein DHCP-Server aktiv sein.

Standard-Einstellung: `OPT_DHCPRELAY`='no'

DHCPRELAY_SERVER An dieser Stelle wird der richtige DHCP-Server eingetragen, an den die Anfragen weitergereicht werden sollen.

DHCPRELAY_IF_N DHCPRELAY_IF_x Mit `DHCPRELAY_IF_N` gibt man die Anzahl der Netzwerkkarten an, auf denen der Relay-Server lauschen soll. In `DHCPRELAY_IF_x` werden dann die entsprechenden Netzwerkkarten angegeben.

Das Interface, über das die Antworten des DHCP-Servers wieder reinkommen, muß in der Liste mit aufgeführt werden.Zusätzlich muss sichergestellt werden, dass die Routen auf dem Rechner, auf dem der DHCP-Server läuft, korrekt gesetzt sind. Die Antwort des DHCP-Servers geht an die IP des Interfaces, an dem der DHCP-Client hängt. Nehmen wir folgendes Scenario an:

- Relay mit zwei Interfaces
- Interfaces zum Client: eth0, 192.168.6.1
- Interfaces zum DHCP-Server: eth1, 192.168.7.1
- DHCP-Server: 192.168.7.2

Dann muss es auf dem DHCP-Server eine Route geben, über den die Antworten an die 192.168.6.1 ihr Ziel erreichen. Ist der Router, auf dem das Relay läuft, der default gateway für den DHCP-Server, ist bereits alles ok. Ist dem nicht so, wird eine extra Route benötigt. Ist der DHCP-Server ein fli4l-Router, würde folgender Konfig-Eintrag dieses Ziel erreichen: IP_ROUTE_x='192.168.6.0/24 192.168.7.1'

Im Betrieb kann es zu Warnungen kommen, dass bestimmte Pakete ignoriert werden. Diese Warnungen kann man ignorieren, sie stören nicht den normalen Betrieb.

Beispiel:

```
OPT_DHCPRELAY='yes'
DHCPRELAY_SERVER='192.168.7.2'
DHCPRELAY_IF_N='2'
DHCPRELAY_IF_1='eth0'
DHCPRELAY_IF_2='eth1'
```

4.5.5. TFTP-Server

Der TFTP-Server wird dann verwendet, wenn der fli4l per TFTP Dateien ausliefern soll. Dies kann zum Beispiel dazu dienen, das ein Client per Netboot startet.

OPT_TFTP Aktiviert den internen TFTP-Server des dnsmasq. Standard-Wert ist 'no'.

TFTP_PATH Spezifiziert das Verzeichnis, in dem die Dateien liegen, die der tftp-Server an die Klienten ausliefern soll. Die entsprechenden Dateien sind mit Hilfe eines geeigneten Programms (z.B. scp) im entsprechenden Pfad abzulegen.

4.5.6. YADIFA - Slave DNS Server

OPT_YADIFA Aktiviert den YADIFA Slave DNS Server. Standard-Wert ist 'no'.

OPT_YADIFA_USE_DNSMASQ_ZONE_DELEGATION Wenn diese Einstellung aktiviert wird erzeugt das yadifa Startscript automatisch für alle Slavezonen entsprechende Zone Delegation Einträge für den dnsmasq. Damit sind die Slavezonen auch direkt über

den dnsmasq abfragbar und man benötigt im Prinzip keine YADIFA_LISTEN_x Einträge mehr. Die Anfragen werden dann vom dnsmasq beantwortet und einen nur auf localhost:35353 horchenden yadifa weitergeleitet.

YADIFA_LISTEN_N Wenn Sie `OPT_YADIFA`='yes' gewählt haben, können Sie mit Hilfe von `YADIFA_LISTEN_N` die Anzahl, und mit `YADIFA_LISTEN_1` bis `YADIFA_LISTEN_N` lokale IPs angeben, auf denen YADIFA DNS-Anfragen annehmen darf. Eine Portnummer ist optional möglich, mit der Angabe 192.168.1.1:5353 würde der YADIFA Slave DNS Server auf DNS Anfragen auf Port 5353 horchen. Achten Sie darauf, dass der dnsmasq in diesem Fall nicht auf allen Schnittstellen horchen darf (siehe `DNS_BIND_INTERFACES`). An dieser Stelle dürfen nur IPs von existierenden Schnittstellen (ethernet, wlan ...) verwendet werden, es kommt sonst zu Warnmeldungen beim Start des Routers. Alternativ ist nun möglich hier auch ALIAS-Namen zu verwende, z. B. `IP_NET_1_IPADDR`

YADIFA_ALLOW_QUERY_N

YADIFA_ALLOW_QUERY_x Gibt IP-Adressen und Netze an denen der Zugriff auf YADIFA erlaubt ist. YADIFA nutzt die Angaben um den fli4l Paketfilter entsprechend zu konfigurieren und die Konfigurationsdateien von YADIFA zu erstellen. Mit dem Prefix ! wird der IP-Adresse oder dem Netz der Zugriff auf YADIFA verweigert.

Der fli4l Paketfilter wird für YADIFA so konfiguriert, dass alle erlaubten Netze aus dieser Einstellung und der für die einzelnen Zonen zusammen in eine ipset Liste (yadifa-allow-query) aufgenommen werden. Eine Unterscheidung nach Zonen ist beim Paketfilter leider nicht möglich. Zusätzlich werden alle IP-Adressen und Netze aus dieser globalen Einstellung, denen der Zugriff verweigert wird, in diese Liste aufgenommen. Es ist daher nicht möglich den Zugriff später für einzelne Zonen wieder auszuweiten.

YADIFA_SLAVE_ZONE_N Gibt die Anzahl der Slave DNS Zonen an die YADIFA verwalten soll.

YADIFA_SLAVE_ZONE_x Der Name der Slave DNS Zone.

OPT_YADIFA_SLAVE_ZONE_USE_DNSMASQ_ZONE_DELEGATION Aktiviert ('yes') oder deaktiviert ('no') die dnsmasq Zone Delegation nur für die Slavezone.

YADIFA_SLAVE_ZONE_x_MASTER Die IP-Adresse mit einer optionalen Portnummer des DNS Master Server.

YADIFA_SLAVE_ZONE_x_ALLOW_QUERY_N

YADIFA_SLAVE_ZONE_x_ALLOW_QUERY_x Gibt IP-Adressen und Netze an denen der Zugriff auf diese YADIFA DNS Zone erlaubt ist. Damit kann der Zugriff auf bestimmte DNS Zonen weiter eingeschränkt werden. YADIFA nutzt die Angaben um die Konfigurationsdateien von YADIFA zu erstellen.

Mit dem Prefix ! wird die IP-Adresse oder das Netz der Zugriff auf YADIFA verweigert.

4.6. DSL - DSL über PPPoE, Fritz!DSL und PPTP

fli4l unterstützt DSL in drei verschiedenen Varianten:

- PPPoE (externe, über Ethernet angeschlossene DSL-Modem, über die pppoe gefahren wird)
- PPTP (externe, über Ethernet angeschlossene Modem, über die pptp gefahren wird)
- Fritz!DSL (DSL über DSL-Adapter von AVM)

Man kann immer nur eine Variante auswählen, gleichzeitiger Betrieb ist leider noch nicht möglich.

Die Konfiguration dieser Varianten ähnelt sich, daher werden die allgemeinen Parameter vorab beschrieben und anschließend wird auf die Spezial-Optionen der einzelnen Varianten eingegangen. Der DSL-Zugang wird vom imond als Circuit[3] verwaltet, daher muß bei Aktivierung einer der DSL-Varianten auch der imond aktiviert werden (siehe START_IMOND (Seite 72)).

4.6.1. Allgemeine Konfigurationsvariablen

Die Pakete verwenden alle die gleichen Konfigurationsvariablen, sie unterscheiden sich nur durch den vorangestellten Paketnamen. Z.B. wird in allen Paketen der Nutzername verlangt, die Variable heißt lediglich je nach Paket PPPOE_USER, PPTP_USER oder FRITZDSL_USER. Im folgenden werden die Variablen ohne ihre Präfixe beschrieben, der fehlende Präfix wird durch einen Stern repräsentiert und in konkreten Beispielen wird von PPPOE ausgegangen (sie sind aber auch mit jedem anderen Präfix gültig).

***_NAME** Hier sollte ein Name für den Circuit vergeben werden - max. 15 Stellen lang. Dieser wird im imon-Client imonc angezeigt. Leerstellen (Blanks) sind nicht erlaubt.

Beispiel: PPPOE_NAME='DSL'

***_USEPEERDNS** Hiermit wird festgelegt, ob die vom Internet-Provider bei der Einwahl übergebenen Nameserver für die Dauer der Onlineverbindung in die Konfigurationsdatei des lokalen Nameservers eingetragen werden sollen.

Sinnvoll ist die Nutzung dieser Option also nur bei Circuits für Internet-Provider. Inzwischen unterstützen fast alle Provider diese Art der Übergabe.

Nachdem die Nameserver-IP-Adressen übertragen wurden, werden die unter DNS_FORWARDERS eingetragenen Nameserver durch die vom Provider vergebenen IP-Adressen als Forwarder verwendet. Danach wird der lokale Nameserver veranlasst, seine Konfiguration neu einzulesen. Dabei gehen bis dahin aufgelöste Namen nicht aus dem Nameserver-Cache verloren.

Diese Option bietet den Vorteil, immer mit den am nächsten liegenden Nameservern arbeiten zu können, sofern der Provider die korrekten IP-Adressen übermittelt - dadurch geht die Namensauflösung schneller.

Im Falle eines Ausfalls eines DNS-Servers beim Provider werden in der Regel die übergebenen DNS-Server-Adressen sehr schnell vom Provider korrigiert.

[3] Im Augenblick ist leider nur genau ein DSL-Circuit möglich – will man mehrere Circuits verwenden, muß man sich mehrere Floppies bauen

Trotz allem ist vor jeder ersten Einwahl die Angabe eines gültigen Nameservers in DNS_FORWARDERS zwingend erforderlich, da sonst die erste Anfrage nicht korrekt aufgelöst werden kann. Außerdem wird beim Beenden der Verbindung die originale Konfiguration des lokalen Nameservers wieder hergestellt.

Standard-Einstellung: *_USEPEERDNS='yes'

***_DEBUG** Soll pppd zusätzliche Debug-Informationen ausgeben, muss man *_DEBUG auf 'yes' setzen. In diesem Fall schreibt pppd zusätzlichen Informationen über die syslog-Schnittstelle.

WICHTIG: Damit diese auch über syslogd ausgegeben werden, muss die Variable OPT_SYSLOGD (s.o.) ebenso auf 'yes' gesetzt sein.

***_USER, *_PASS** Hier sind Benutzerkennung und Passwort für den jeweils benutzten Provider anzugeben. *_USER enthält die Benutzerkennung, *_PASS das Passwort.

WICHTIG: Für einen T-Online-Zugang ist folgendes zu beachten:

Der Username AAAAAAAAAAAATTTTTT#MMMM setzt sich aus der zwölfstelligen Anschlußkennung, der T-Online-Nummer und der Mitbenutzernummer zusammen. Hinter der T-Online-Nummer muß ein '#' angegeben werden, wenn die Länge der T-Online-Nummer kürzer als 12 Zeichen ist.

Sollte dies in Einzelfällen nicht zum Erfolg führen (offenbar abhängig von der Vermittlungsstelle), muß zusätzlich zwischen der Anschlußkennung und der T-Online-Nummer ein weiteres '#'-Zeichen eingefügt werden.

Ansonsten (T-Online-Nr ist 12stellig) sind keine '#'-Zeichen anzugeben.

Die Benutzerkennung muß bei T-Online mit '@t-online.de' abgeschlossen werden!

Beispiel:

```
PPPOE_USER='11111111111222222#0001@t-online.de'
```

Infos zu der Benutzerkennung bei anderen Providern finden sich in der FAQ:

- http://extern.fli4l.de/fli4l_faqengine/faq.php?list=category&catnr=3&prog=1

***_HUP_TIMEOUT** Hier kann die Zeit in Sekunden angegeben werden, nach welcher die Verbindung beendet werden soll, wenn nichts mehr über die DSL-Leitung läuft. Dabei steht ein Timeout von '0' oder 'never' für kein Timeout - bei 'never' wählt sich der Router zusätzlich nach einem Zwangsauflegen sofort wieder neu ein. Eine Veränderung des Dialmodes ist dann nicht mehr möglich - er muß auf 'auto' stehen und bleiben. 'never' ist momentan nur für PPPOE und FRITZDSL verfügbar.

***_CHARGEINT** Charge-Interval: Hier ist der Zeittakt in Sekunden anzugeben. Dieser wird dann für die Kosten-Berechnung verwendet.

Die meisten Provider rechnen minutengenau ab. In diesem Fall ist der Wert '60' richtig. Bei Providern mit sekundengenauer Abrechnung setzt man *_CHARGEINT besser auf '1'.

Leider wird bei DSL der Zeittakt nicht voll ausgenutzt, so wie es bei ISDN der Fall ist. Hier wird immer nach der Zeit, die in *_HUP_TIMEOUT angegeben ist, eingehängt.

Hier ist deshalb *_CHARGEINT lediglich für die Berechnung von Gebühren maßgeblich.

***_TIMES** Die hier angegebenen Zeiten bestimmen, wann dieser Circuit aktiviert werden soll und wann er wieviel kostet. Dadurch wird es möglich, zu verschiedenen Zeiten verschiedene Circuits mit Default-Routen zu verwenden (Least-Cost-Routing). Dabei kontrolliert der Daemon imond die Routen-Zuweisung.

Aufbau der Variablen:

```
PPPOE_TIMES='times-1-info [times-2-info] ...'
```

Jedes Feld times-?-info besteht aus 4 Unterfeldern - durch Doppelpunkt (':') getrennt.

1. Feld: W1-W2

   Wochentag-Zeitraum, z.B. Mo-Fr oder Sa-Su usw. Sowohl die deutsche als auch die englische Schreibweise ist erlaubt. Soll ein einzelner Wochentag eingetragen werden, ist zu W1-W1 schreiben, also z.B. Su-Su.

2. Feld: hh-hh

   Stunden-Bereich, z.B. 09-18 oder auch 18-09. 18-09 ist gleichbedeutend mit 18-24 plus 00-09. 00-24 meint den ganzen Tag.

3. Feld: Charge

   Hier werden in Euro-Werten die Kosten pro Minute angegeben, z.B. 0.032 für 3.2 Cent pro Minute. Diese werden unter Berücksichtigung der Taktzeit umgerechnet für die tatsächlich anfallenden Kosten, welche dann im imon-Client angezeigt werden.

   Feld: LC-Default-Route

   Der Inhalt kann Y oder N sein. Dabei bedeutet:

   Y: Der angegebene Zeitbereich wird beim LC-Routing als Default-Route verwendet.

   N: Der angegebene Zeitbereich dient nur zum Berechnen von Kosten, er wird beim automatischen LC-Routing jedoch nicht weiter verwendet.

Beispiel (als eine lange Zeile zu lesen):

```
PPPOE_TIMES='Mo-Fr:09-18:0.049:N
             Mo-Fr:18-09:0.044:Y
             Sa-Su:00-24:0.039:Y'
```

Wichtig: *Die bei *_TIMES angegebenen Zeiten muessen die ganze Woche abdecken. Ist das nicht der Fall, kann keine gültige Konfiguration erzeugt werden.*

Wenn die Zeitbereiche aller LC-Default-Route-Circuits ("Y") zusammengenommen nicht die komplette Woche beinhalten, gibt's zu diesen Lückenzeiten keine Default-Route. Damit ist dann Surfen im Internet zu diesen Zeiten ausgeschlossen!

Noch ein ganz einfaches Beispiel:

```
PPPOE_TIMES='Mo-Su:00-24:0.0:Y'
```

für diejenigen, die eine Flatrate nutzen.

Und noch eine letzte Bemerkung zum LC-Routing: *Feiertage werden wie Sonntage behandelt.*

***_FILTER** fli4l legt automatisch auf, wenn während der über hangup timeout angegebenen Zeit keine Daten über das pppoe-Interface gehen. Leider wertet das Interface auch Datentransfers mit, die von außen kommen, z.B. durch Verbindungsversuche eines P2P-Clients wie eDonkey. Da man heutzutage eigentlich pemanent von anderen kontaktiert wird, kann es passieren, dass fli4l die DSL-Verbindung nie beendet.

Hier hilft die Option `*_FILTER`. Setzt man es auf yes, wird nur noch Verkehr gewertet, der von der eigenen Maschine generiert wird und externer Traffic wird komplett ignoriert. Da von draußen reinkommender Traffic in der Regel dazu führt, dass der Router oder dahinter liegende Rechner reagieren, indem sie z.B. Verbindungswünsche ablehnen, werden zusätzlich noch einige rausgehende Pakete ignoriert. Wie das genau funktioniert, kann man hier nachlesen:

- http://www.fli4l.de/hilfe/howtos/basteleien/hangup-problem-loesen/ und
- http://web.archive.org/web/20061107225118/http://www.linux-bayreuth.de/dcforum/DCForumID2/46.html.

Eine genauere Beschreibung des Ausdrucks und seiner Einbindung ist im Anhang zu finden - das ist aber nur interessant, wenn man Änderungen vornehmen will.

***_FILTER_EXPR** Hier steht der zu nutzende Filter, wenn `*_FILTER` auf 'yes' gesetzt ist.

***_MTU *_MRU** Mit diesen optionalen Variablen können die sog. **MTU** (maximum transmission unit) und die **MRU** (maximum receive unit) eingestellt werden. Optional bedeutet, die Variable muß nicht in der Konfigurationsdatei stehen, sie ist bei Bedarf durch den Benutzer einzufügen!
Normal beträgt die MTU und die MRU jeweils 1492. Diese Einstellung sollte nur in Sonderfällen geändert werden! Diese Variablen gibt es nicht fuer OPT_PPTP.

***_NF_MSS** Bei manchen Providern treten Effekte folgender Art auf:

- Webbrowser bekommt eine Verbindung, macht aber danach nichts mehr,
- kleine Mail kann verschickt werden, große Mail nicht,
- ssh funktioniert, scp hängt nach dem initialen Verbindungsaufbau.

Um diese Probleme zu umgehen, manipuliert fli4l standardmässig die MTU. In einige Faellen reicht das allerdings nicht, daher gestattet fli4l explizit das Setzen der MSS (message segment size) auf einen vom Provider vorgegebenen Wert. Falls der Provider nichts vorgibt, ist 1412 ein guter Startwert zum probieren; falls er die MTU vorgibt, tragen sind hier 40 Byte weniger einzutragen ($mss = mtu - 40$). Diese Variable ist optional, was bedeutet, die Variable muß nicht in der Konfigurationsdatei stehen, sie ist bei Bedarf durch den Benutzer einzufügen! Diese Variable gibt es nicht fuer OPT_PPTP.

4.6.2. OPT_PPPOE - DSL über PPPoE

Für die Kommunikation über einen DSL-Anschluss ist in der Regel das PPPoE-Paket notwendig, weil die Provider keinen richtigen Router, sondern lediglich ein DSL-Modem zur Verfügung

stellen. Zwischen dem fli4l-Router und dem Modem wird das Protokoll PPP benutzt, jedoch hier speziell über das Ethernet.

Dabei können eine oder zwei Ethernet-Karten im fli4l-Router zum Einsatz kommen:

- Nur eine Karte mit IP für das LAN und PPP zum DSL-Modem
- Zwei Karten: eine für IP im LAN, die andere für PPP zum DSL-MODEM

Die bessere Wahl ist die Alternative mit den zwei Ethernet-Karten. Dann sind beide Protokolle - IP und PPPoE - sauber voneinander getrennt.

Aber die Methode mit einer Ethernet-Karte funktioniert ebenso. In diesem Fall wird das T-DSL-Modem einfach mit an den Netzwerk-Hub angeschlossen. Es muss dann aber eventuell mit leichten Einbußen bei der maximalen Übertragungsgeschwindigkeit gerechnet werden.

Bei Kommunikationsproblemen zwischen DSL-Modem und Router kann man versuchen die Geschwindigkeit der Netzwerkkarte zu verringern und eventuell deren Half-Duplex-Betrieb zu aktivieren. Alle neueren PCI- aber nur einige ISA-Netzwerkkarten lassen sich auf verschiedene Modi konfigurieren. Dazu kann man entweder das ethtool aus dem Paket advanced_networking verwenden oder sich ein DOS-Bootmedium erstellen und das eigene Konfigurationstool der Karte dort mit abspeichern. Dann bootet man den fli4l-Router mit diesem Medium, startet das Konfigurationsprogramm und stellt die Karte fest auf einen langsameren Modus ein. Das Konfigurationsprogramm für die Karte liegt beim Kauf auf dem Treibermedium bei oder kann von der Webseite des Kartenherstellers heruntergeladen werden. Eventuell findet man es auch bei einer Recherche im Wiki:

- https://ssl.nettworks.org/wiki/display/f/Netzwerkkarten

Wenn man zwei Karten benutzt, sollte man die erste Karte für das LAN und die zweite für die Verbindung zum DSL-Modem verwenden.

Dabei muss nur die erste Karte mit einer IP-Adresse belegt werden.

Das heißt:

```
IP_NET_N='1'                # Nur *eine* Karte mit IP-Adresse!
IP_NET_1xxx='...'              # Die üblichen Parameter
```

Bei `PPPOE_ETH` gibt man 'eth1' für die zweite Ethernetkarte an und definiert *keine* `IP_NET_2-xxx`-Variablen.

OPT_PPPOE Aktiviert die Unterstützung für PPPoE. Standard-Einstellung: `OPT_PPPOE='no'`.

PPPOE_ETH Name des Ethernet-Interfaces

'eth0'	1. Ethernet-Karte
'eth1'	2. Ethernet-Karte
...	...

Standard-Einstellung: `PPPOE_ETH='eth1'`

PPPOE_TYPE *PPPOE* steht für die Übertragung von PPP-Paketen über Ethernet-Leitungen. D.h., die zu übertragenden Daten werden im ersten Schritt vom ppp-Daemon in ppp-Pakete und dann in einem zweiten Schritt für die Übertragung übers Ethernet nochmals in pppoe-Pakete verpackt, um dann ans DSL-Modem geschickt zu werden. Das

Wert	Beschreibung
async	Die Pakete werden durch den pppoe-Daemon erzeugt; die Kommunikation zwischen *pppd* und *pppoed* erfolgt asynchron.
sync	Die Pakete werden durch den pppoe-Daemon erzeugt; die Kommunikation zwischen *pppd* und *pppoed* erfolgt syncron. Das führt zu einer effizienteren Kommunikation und damit zu einer geringeren Prozessorlast.
in_kernel	Die ppp-Pakete werden direkt an den Linux-Kern gereicht, der daraus pppoe-Pakete macht. Dadurch entfällt die Kommunikation mit einem zweiten Daemon und damit eine Menge Kopieraufwand, was wiederum zu geringerer Prozessorlast führt.

Tabelle 4.2.: Arten der pppoe-Paketerzeugung

zweite Verpacken kann durch den pppoe-Daemon oder durch den Kern erfolgen. Mittels `PPPOE_TYPE` wird die Art und Weise der pppoe-Paketerzeugung definiert.

Jemand hat mal einen Vergleich der verschiedenen Varianten gemacht und kam auf einem Fujitsu Siemens PCD-H, P75 zu den in Tabelle 4.3 dargestellten Ergebnissen[4].

fli4l	NIC	Bandbreite (down stream)	CPU-Auslastung
2.0.8	rtl8029 + rtl8139	310 kB/s	100%
2.0.8	2x 3Com Etherlink III	305 kB/s	100%
2.0.8	SMC + 3Com Etherlink III	300 kB/s	100%
2.1.7	SMC + 3Com Etherlink III	375 kB/s	40%

Tabelle 4.3.: Bandbreite und CPU-Auslastung bei pppoe

PPPOE_HUP_TIMEOUT Verwendet man als PPPoE-Typ in_kernel und als Dialmode auto, kann man als Timeout 'never' angeben. Der Router legt dann nicht mehr auf und wählt sich nach einer Zwangstrennung des Providers automatisch wieder ein. Nachträgliche Änderungen des Dialmodes ist nicht mehr möglich.

4.6.3. OPT_FRITZDSL - DSL per Fritz!Card DSL

Hier wird die Internetverbindung per Fritz!Card DSL aktiviert. Fuer die Internetverbindung wird die Fritz!Card DSL von AVM benutzt. Da die Treiber fuer diese Karten nicht der GPL unterliegen ist es nicht moeglich diese mit dem DSL Paket zu liefern. Es ist daher unbedingt nötig diese Treiber vorher von `http://www.fli4l.de/download/stabile-version/avm-treiber/` herunter zu laden und einfach in das fli4l Verzeichnis zu entpacken.

Die Circuit-Unterstützung für die Fritz!Card DSL wurde mit freundlicher Unterstützung von Stefan Uterhardt (E-Mail: `zer0@onlinehome.de`) realisiert.

[4]Die Zahlen wurden einem Posting in spline.fli4l entnommen und nicht weiter geprüft. Der Artikel trug die Message ID <caf9fkala1@bla.spline.inf.fu-berlin.de>.

OPT_FRITZDSL Aktiviert die Unterstützung für Fritz!DSL. Standard-Einstellung: `OPT_FRITZDSL='no'`.

FRITZDSL_TYPE Es gibt verschiedene Fritz!-Karten, ueber die eine DSL-Anbindung erfolgen kann. Die verwendete Karte wird ueber `FRITZDSL_TYPE` eingestellt, wobei die in Tabelle 4.4 aufgefuehrten Typen zur Verfuegung stehen.

Kartentyp	Verwendung
fcdsl	Fritz!Card DSL
fcdsl2	Fritz!Card DSLv2
fcdslsl	Fritz!Card DSL SL
fcdslusb	Fritz!Card DSL USB
fcdslslusb	Fritz!Card DSL SL USB
fcdslusb2	Fritz!Card DSL USBv2

Tabelle 4.4.: Fritz-Karten

Standard-Einstellung:

```
FRITZDSL_TYPE='fcdsl'
```

FRITZDSL_PROVIDER Mit dieser Option wird der Typ der Gegenstelle eingestellt. Moegliche Optionen sind:
U-R2, ECI, Siemens, Netcologne, oldArcor, Switzerland, Belgium, Austria1, Austria2, Austria3, Austria4

In Deutschland handelt es sich fast immer um UR-2. Siemens und ECI kommen nur bei sehr alten Anschluessen zum Einsatz.
Fuer Schweiz und Belgien sollten die Optionen selbsterklaerend sein und in Oesterreich heisst es ausprobieren.
Sollte jemand fuer Oesterreich eine bessere Beschriftung der Optionen haben moege er diese bitte mitteilen.

Standard-Einstellung:

```
FRITZDSL_PROVIDER='U-R2'
```

4.6.4. OPT_PPTP - DSL über PPTP in Österreich/den Niederlanden

In Österreich (und anderen europäischen Ländern) wird statt PPPoE das PPTP-Protokoll verwendet. Auch hier wird eine separate Ethernet-Karte an ein PPTP-Modem angeschlossen.

Ab Version 2.0 ist der Zugang über PPTP als Circuit realisiert - mit freundlicher Unterstützung von Rudolf Hämmerle (E-Mail: rudolf.haemmerle@aon.at).

Bei PPTP werden zwei Karten verwendet. Hierbei sollte man die erste Karte für den Anschluß des LANs verwenden und die zweite für die Verbindung zum DSL-Modem.

Nur die erste Karte darf mit IP-Adressen belegt werden.

Das heißt:

```
IP_NET_N='1'                    # Nur *eine* Karte mit IP-Adresse!
IP_NET_1xxx='...'               # Die üblichen Parameter
```

Bei PPTP_ETH gibt man 'eth1' für die zweite Ethernetkarte an und definiert *keine* IP_NET_-2-xxx-Variablen.

OPT_PPTP Aktiviert die Unterstützung für PPTP. Standard-Einstellung: OPT_PPTP='no'.

PPTP_ETH Name des Ethernet-Interfaces

'eth0'	1. Ethernet-Karte
'eth1'	2. Ethernet-Karte
...	...

Standard-Einstellung: PPTP_ETH='eth1'

PPTP_MODEM_TYPE Es gibt verschiedene PPTP-Modemtypen, über die eine pptp-Anbindung erfolgen kann. Das verwendete Modem wird über PPTP_MODEM_TYPE eingestellt, wobei die in Tabelle 4.5 aufgeführten Typen zur Verfügung stehen.

Modemtyp	Verwendung
telekom	Östereich (Telekom Austria)
xdsl	Östereich (Inode xDSL@home)
mxstream	die Niederlande, Dänemark

Tabelle 4.5.: PPTP-Modemtypen

Standard-Einstellung:

```
PPTP_MODEM_TYPE='telekom'
```

Inode xDSL@home

Implementiert wurde die Unterstützung von Inode xDSL@home in Anlehnung an das im Supportbereich von Inode beschriebene[5].

Es gibt momentan evtl. noch Probleme mit dem Erneuern der Lease für das Interface (die IP für das Interface wird über dhcp zugeteilt und muß regelmäßig neu angefordert werden) und das Auflegen und wieder Einwählen per imonc funktioniert noch nicht so richtig. Hier ist Hilfe in Form von Patches oder zur Verfügungstellung als Testkaninchen willkommen.

Bei xdsl gibt es zwei weitere Optionen für pptp:

PPTP_CLIENT_REORDER_TO Der pptp-client, der für xdsl genutzt wird, muß unter Umständen Pakete zwichenpuffern und umordnen. Normalerweise wartet er 0,3s auf ein ausstehendes Paket. Mit dieser Variabel kann man das Timeout zwischen 0.00 (gar nicht puffern) und 10.00 variieren. Die Zeiten müssen immer mit zwei Nachkommstellen angegeben werden.

PPTP_CLIENT_LOGLEVEL Hier kann angegeben werden, wieviel Debug-Ausgaben der pptp-client produziert. Möglich sind 0 (wenig), 1 (default) und 2 (viel).

[5] Siehe http://www6.inode.at/support/internetzugang/xdsl_home/konfiguration_ethernet_linux.html

4.6.5. OPT_POESTATUS - PPPoE-Status-Monitor auf fli4l-Console

Auch diesen PPPoE-Status-Monitor für DSL-Verbindungen hat Thorsten Pohlmann entwickelt.

Bei `OPT_POESTATUS`='yes' kann auf der 3. fli4l-Console jederzeit der DSL-Status eingesehen werden. Auf die 3. Console kann mit der Tastenkombination ALT-F3 gewechselt werden, zurück auf die 1. Console mit ALT-F1.

4.7. DYNDNS - Dynamische Updates für Domain Name Services

Dieses Paket ist dafür gedacht, automatisch bei jeder Einwahl einen dynamischen Hostname zu aktualisieren. Folgende Dienste werden unterstützt:

Anbieter	FreeDNS (afraid.org)
DYNDNS_x_PROVIDER	AFRAID
Homepage	http://freedns.afraid.org

Wichtig: *Als Passwort ist hier der letzte Teil (hinter dem Fragezeichen) der URL anzugeben, die man auf der Homepage von Afraid.org abrufen kann (Einloggen ⇒„Dynamic DNS" ⇒Die URL, die sich hinter dem Link „Direct URL" versteckt). Alle anderen Angaben werden ignoriert.*

Anbieter	Companity
DYNDNS_x_PROVIDER	COMPANITY
Homepage	http://www.staticip.de/

Anbieter	DHS International
DYNDNS_x_PROVIDER	DHS
Homepage	http://www.dhs.org/

Anbieter	DNS2Go
DYNDNS_x_PROVIDER	DNS2GO
Homepage	http://dns2go.com/

Anbieter	DNS-O-Matic
DYNDNS_x_PROVIDER	DNSOMATIC
Homepage	http://www.dnsomatic.com

Anbieter	DtDNS
DYNDNS_x_PROVIDER	DTDNS
Homepage	http://www.dtdns.com/

Anbieter	DynAccess
DYNDNS_x_PROVIDER	DYNACCESS
Homepage	http://dynaccess.de/

Wichtig: *DynAccess bietet im Rahmen der fli4l-DynAccess-Kooperation für die Subdomains *.dyn-fli4l.de, *.dyn-fli4l.info und *.dyn-eisfair.de Sondertarife an. Informationen hierzu gibt es auf der Internet-Seite* http://www.dyn-fli4l.de/ *bzw.* http://www.dyn-eisfair.de/.

Anbieter	DynDNS.org
DYNDNS_x_PROVIDER	DYNDNS
Homepage	http://dyn.com/

Anbieter	DynDNS.org (custom)
DYNDNS_x_PROVIDER	DYNDNSC
Homepage	http://dyn.com/standard-dns/

Anbieter	DynDNS DK
DYNDNS_x_PROVIDER	DYNDNSDK
Homepage	http://dyndns.dk/

Anbieter	dyndns:free
DYNDNS_x_PROVIDER	DYNDNSFREE
Homepage	http://dyndnsfree.de/

Anbieter	eisfair.net
DYNDNS_x_PROVIDER	DYNEISFAIR
Homepage	http://www.intersales.de/it-infrastruktur/dyneisfair.html

Wichtig: *Mit der Benutzung dieses Dienstes unterstützt man die Arbeit der fli4l- und eisfair-Entwickler.*

Anbieter	DyNS
DYNDNS_x_PROVIDER	DYNSCX
Homepage	http://www.dyns.cx/

Anbieter	GnuDIP Dynamic DNS
DYNDNS_x_PROVIDER	GNUDIP
Homepage	http://gnudip2.sourceforge.net/

Anbieter	Provider Hurricane Electric
DYNDNS_x_PROVIDER	HE
Homepage	https://dns.he.net/

Anbieter	IN-Berlin e.V.
DYNDNS_x_PROVIDER	INBERLIN
Homepage	http://www.in-berlin.de

Anbieter	KONTENT
DYNDNS_x_PROVIDER	KONTENT
Homepage	http://www.kontent.de/

Anbieter	Nerdcamp.net
DYNDNS_x_PROVIDER	NERDCAMP
Homepage	http://nerdcamp.net/dynamic/dns.cgi

Anbieter	No-IP.com
DYNDNS_x_PROVIDER	NOIP
Homepage	http://www.no-ip.com/

Anbieter	noxaDynDNS
DYNDNS_x_PROVIDER	NOXA
Homepage	http://www.noxa.de/

Anbieter	OVH.DE
DYNDNS_x_PROVIDER	OVHDE
Homepage	http://www.ovh.de/

Anbieter	PHPDYN
DYNDNS_x_PROVIDER	PHPDYN
Homepage	http://www.webnmail.de/phpdyn/

Wichtig: *diese Lösung muß man selber hosten*

Anbieter	Regfish.com
DYNDNS_x_PROVIDER	REGFISH
Homepage	http://www.regfish.de/

Anbieter	SelfHost.de
DYNDNS_x_PROVIDER	SELFHOST
Homepage	http://selfhost.de/cgi-bin/selfhost

Anbieter	Securepoint Dynamic DNS Service
DYNDNS_x_PROVIDER	SPDNS
Homepage	http://www.spdns.de/

Anbieter	Strato
DYNDNS_x_PROVIDER	STRATO
Homepage	http://www.strato.de/

Anbieter	T-Link.de
DYNDNS_x_PROVIDER	TLINK
Homepage	http://www.t-link.de/

Anbieter	twodns.de
DYNDNS_x_PROVIDER	TWODNS
Homepage	http://www.twodns.de/

Anbieter	ZoneEdit.com
DYNDNS_x_PROVIDER	ZONEEDIT
Homepage	http://zoneedit.com/

Wir versuchen diese Daten aktuell zu halten. Trotzdem übernehmen wir keine Haftung für die Richtigkeit dieser Daten. Wer einen Fehler oder eine Änderung entdeckt sollte eine Mail an das fli4l-Team (E-Mail: team@fli4l.de) schicken.

Diese Liste ist komplett, andere Provider werden ohne Änderung nicht unterstützt. Wie man das Paket um eigene Anbieter erweitern kann, steht im Anhang.

Der dynamische Hostname wird automatisch bei jeder Einwahl ins Internet aktualisiert. Das Paket beinhaltet eine Sperre, die das mehrmalige aktualisieren der gleichen IP verhindert, da dies bei einigen DynDNS-Anbietern nicht gerne gesehen wird und im Extremfall zur Sperrung des Accounts führen kann.

Hinweis: Es kann einige Minuten dauern, bis die Änderung des dynamischen Hostnamens wirksam wird.

Bevor man mit der Einrichtung dieses Paketes beginnen kann, muss man sich bei einem der oben genannten Anbietern einen Account holen. Falls man das schon hat, kann man sofort loslegen. Hat man noch keinen Account, so kann man sich an obiger Tabelle orientieren, um einen Hostname zu finden, der den Ansprüchen genügt und den persönlichen Geschmack trifft.

Für die nun folgende Konfiguration benötigt man folgende Daten:

- Name des Anbieters
- Benutzername
- Passwort
- Den DynDNS-Hostnamen

Die benötigten Angaben können je nach Anbieter variieren, es wird versucht eine möglichst konsistene Konfiguration zu bieten. Manchmal ist z.B. der Hostname gleich dem Benutzernamen, in so einem Fall werden wir versuchen, immer das Host-Feld zu benutzen und den Benutzernamen einfach ignorieren. Jetzt aber los:

OPT_DYNDNS Steht dieser Parameter auf `'yes'`, wird `OPT_DYNDNS` aktiviert.

DYNDNS_SAVE_OUTPUT Wird dieser Parameter auf `'yes'` gestellt, wird das Ergebnis der DynDNS-Anfrage(n) in einer Datei gespeichert und kann über den Webserver[6] abgefragt werden.

DYNDNS_N Hat man bei mehreren DynDNS-Anbietern einen Account und will deswegen bei jeder Einwahl mehrere Namen updaten, so ist dieser Wert entsprechend anzupassen.

DYNDNS_x_PROVIDER Hier wird der Name des zu benutzenden Providers angegeben (siehe Tabelle weiter oben und Hinweis in der Config-Datei).

DYNDNS_x_USER Benutzername bei dem DynDNS-Anbieter. Häufig ist dies eine E-Mail-Adresse, ein selbstgewählter Name oder gleich dem DynDNS-Hostname.

DYNDNS_x_PASSWORD Hier ist das Passwort des DynDNS-Accounts anzugeben. Aufpassen, dass niemand anderes beim Editieren der Config-Datei zusieht!

DYNDNS_x_HOSTNAME Hier ist der *komplette* DynDNS-Hostname des Accounts einzutragen. Beispielsweise könnte hier folgendes stehen:

- `cool.nerdcamp.net`
- `user.dyndns.org`
- `fli4luser.fli4l.net`

DYNDNS_x_UPDATEHOST Hier wird für den Provider PHPDYN angegeben, auf welchem Host der Updater installiert ist. Dies ist nötig, da dies kein herkömmlicher Provider ist sondern nur ein Script, welches einen PowerDNS Server mit MySQL aktualisiert und welches unter der GPL steht.

DYNDNS_x_CIRCUIT Hier kann angegeben werden, bei welchen Circuits dieser Hostname aktualisiert wird. Die einzelnen Circuits werden mit Leerzeichen voneinander getrennt. Es kann z.B. erwünscht sein, den Hostnamen nur bei der DSL-Einwahl zu benutzen. Hier ein paar Beispiele:

```
DYNDNS_1_CIRCUIT='1 2 3'          # Nur ISDN: Circuits 1 bis 3
oder
DYNDNS_1_CIRCUIT='pppoe'          # Nur DSL: pppoe-Circuit
oder
DYNDNS_1_CIRCUIT='dhcp'           # Update bei DHCP-Providern
                                  # (opt_dhcp wird benötigt)
oder
DYNDNS_1_CIRCUIT='pppoe 1'        # DSL und ISDN

oder
DYNDNS_1_CIRCUIT='static'         # fli4l hinter z.B. LTE Router
```

DYNDNS_x_RENEW Manche Provider erwarten, dass alle n Tage ein Update ausgeführt wird, auch wenn sich die IP nicht verändert hat. Dieses Intervall kann man hier angeben. Gibt man keinen Wert an, wird nach 29 Tagen ein Update durchgeführt.

[6] OPT_HTTPD im Paket HTTPD (Seite 131) auf http://www.fli4l.de/download/stabile-version/

Zu beachten ist hierbei, dass ein Update nur bei einer Einwahl angestoßen wird - also bei einer Einwahl über DSL oder ISDN oder einer Erneuerung einer Lease bei einem via DHCP konfigurierten Interface, wie man es bei einem Kabelmodem findet. Findet über längere Zeit keine Einwahl statt, muß man das Update auf andere Weise anstoßen.

DYNDNS_x_EXT_IP Mit dieser Variable wird die Methode, mit der die externe IP Adresse ermittelt wird, konfiguriert. Im Moment gibt es die Möglichkeit mit `'no'` überhaupt keinen externen Dienst nach der IP Adresse zu befragen sondern direkt die externe IP Adresse anhand des WAN Interfaces zu bestimmen. Das funktioniert in der Regel aber nur bei WAN Verbindungen, die direkt auf dem fli4l terminieren, wie z.B. DSL via PPPoE. Mit der Einstellung `'dyndns'` wird die beim Update verwendete IP Adresse über den externen Dienst von checkip.dyndns.org ermittelt. Wird die Einstellung `'stun'` benutzt wird die Liste der STUN Server der Reihe nach abgefragt bis eine erfolgreiche Antwort geliefert wird. Die Nutzung eines externen Dienstes zur Ermittlung der IP Adresse ist notwendig, wenn der Router selbst nicht derjenige ist, der die externe IP erhält. Dabei ist zu beachten, dass der Router in diesem Falle momentan eine Änderung der externen IP nicht mitbekommt, den dyndns-Namenseintrag also nicht zeitnah aktualisieren kann.

DYNDNS_x_LOGIN Manche Provider erfordern, dass sich der Benutzer regelmäßig auf ihrer Internet-Seite unter seinem Benutzerkonto anmeldet, damit der Dienst nicht deaktiviert wird. Wenn diese Variable auf `'yes'` gesetzt ist, erledigt das der fli4l für Sie. Allerdings funktioniert das nur, wenn das dyndns-Paket für den jeweiligen Provider vorbereitet wurde. Momentan ist eine solche regelmäßige Anmeldung nur für den Provider "DYNDNS" erforderlich und möglich. Bedenken Sie bitte auch, dass die Nutzung dieser Funktion `OPT_EASYCRON='yes'` im Paket easycron erfordert.

DYNDNS_LOGINTIME Nutzen Sie einen Provider, bei dem sich der fli4l regelmäßig anmelden soll, um eine Deaktivierung des Dienstes zu verhindern (s.o.), dann können Sie mit dieser Variable einstellen, wann diese Anmeldung stattfinden soll. Benötigt wird eine Zeitangabe im Cron-Format; zu Details lesen Sie sich bitte die Dokumentation des easycron-Pakets durch. Die Standardbelegung lautet `0 8 * * *`, was einer täglichen Anmeldung um acht Uhr morgens entspricht.

DYNDNS_ALLOW_SSL Ist diese Variable auf `'yes'` gesetzt, wird das Update wenn möglich über SSL (verschlüsselte Verbindung) durchgeführt.

DYNDNS_LOOKUP_NAMES Ein Update der IP sollte eigentlich nur erfolgen, wenn sich die IP geändert hat. Viele fli4l-Router haben jedoch keinen permanenten Speicher, auf der die Information über die registierte IP gesichert werden kann, daher steht diese Information direkt nach dem Booten dort nicht zur Verfügung. Um trotzdem unnötige Updates zu vermeiden, kann fli4l in dieser Situation (und nur in dieser Situation) beim Namensdienst nach der aktuell registrierten IP fragen. Die ermittelte IP wird dann zwischengespeichert und für jedes weitere Update genutzt.

Zu beachten ist dabei, dass nach einem Reboot das Update-Intervall neu beginnt, wenn fli4l den Namensdienst zur Ermittlung der IP nutzt.

DYNDNS_DEBUG_PROVIDER Ist diese Variable auf `'yes'` gesetzt, wird ein trace des Update-Vorgangs aufgezeichnet, so dass man im Nachhinein bei einem Problem prüfen kann, was schief gegangen ist. Default: `DYNDNS_DEBUG_PROVIDER='no'`

4.8. EASYCRON - Befehle zeitgesteuert ausführen

Dieses Paket wurde von Stefan Manske E-Mail: fli4l@stephan.manske-net.de zusammengestellt und vom fli4l-Team an 2.1 angepaßt.

4.8.1. Konfiguration

Mit OPT_EASYCRON kann man über das entsprechende config-file gesteuert zu bestimmten Zeiten Befehle ausführen lassen.

Dabei werden folgende Einträge benutzt:

OPT_EASYCRON mit OPT_EASYCRON='yes' wird das Paket aktiviert

Standard-Einstellung: OPT_EASYCRON='no'

EASYCRON_MAIL Da immer wieder Probleme auftraten, dass der crond unerwünschte Mails verschickt, kann man dies generell mit diesem Flag verhindern.

Standard-Einstellung: EASYCRON_MAIL='no'

EASYCRON_N Die Anzahl der verschiedenen Befehle, die von cron gestartet werden sollen.

EASYCRON_x_CUSTOM Wer sich mit den Einstellungen in der crontab auskennt, kann hier für jeden Eintrag eigene Einstellungen wie MAILTO, PATH, ... einstellen. Mehrere Einträge müssen durch \\ getrennt werden. Hier sollte man sich aber wirklich mit cron auskennen.

Standard-Einstellung: EASYCRON_CUSTOM=''

EASYCRON_x_COMMAND In EASYCRON_x_COMMAND wird der gewünschte Befehl eigetragen, wie z.B.

```
EASYCRON_1_COMMAND='echo 1 '>' /dev/console'
```

EASYCRON_x_TIME In EASYCRON_x_TIME wird die Ausführungszeit gemäß der üblichen cron-Syntax eingetragen.

4.8.2. Beispiele

- Der Computer wünscht uns "Ein gutes neues Jahr"

```
EASYCRON_1_COMMAND = 'echo Ein gutes neues Jahr! > /dev/console'
EASYCRON_1_TIME    = '0 0 31 12 *'
```

- xxx wird von Montag bis Freitag jeweils von 7-20 Uhr zu jeder vollen Stunde ausgeführt.

```
EASYCRON_1_COMMAND = 'xxx'
EASYCRON_1_TIME    = '0 7-20 0 * 1-5'
```

- Der Router beendet jede Nacht um 03:40 die Internet-Verbindung die per DSL aufgebaut ist baut sie nach 5sec Wartezeit wieder auf. Die folgenden Devicenamen sind möglich: pppoe, ippp[1-9], ppp[1-9].

```
EASYCRON_1_COMMAND = 'fli4lctrl hangup pppoe; sleep 5; fli4lctrl dial pppoe'
EASYCRON_1_TIME    = '40 3 * * *'
```

Weitere Informationen zur cron-Syntax finden Sie unter

- http://www.pro-linux.de/artikel/2/146/der-batchdaemon-cron.html
- http://de.linwiki.org/wiki/Linuxfibel_-_System-Administration_-_Zeit_und_Steuerung#Die_Datei_crontab
- http://web.archive.org/web/20021229004331/http://www.linux-magazin.de/Artikel/ausgabe/1998/08/Cron/cron.html
- http://web.archive.org/web/20070810063838/http://www.newbie-net.de/anleitung_cron.html

4.8.3. Voraussetzungen

- fli4l in einer Version > 2.1.0
- für ältere Versionen bitte die entspechenden opt_easycron-Versionen aus der OPT-Datenbank verwenden

4.8.4. Installation

`OPT_EASYCRON` wird einfach wie jedes andere OPT im aktuelle fli4l-Verzeichnis entpackt.

4.9. HD - Unterstützung von Festplatten, Flash-Karten, USB-Sticks usw.

4.9.1. OPT_HDINSTALL - Installation auf Festplatte/CompactFlash

fli4l unterstützt eine Vielzahl an Bootmedien (CD, HD, Netzwerk, Compact-Flash,...). Die Diskette zählt aus Platzgründen ab Version 4.0 nicht mehr dazu.

Im Folgenden werden die notwendigen Schritte zur Installation auf einer Festplatte erklärt.

Der übliche Weg ist die Installation mit einem Bootmedium, es kann aber auch über Netzwerk-Boot installiert werden. Das `OPT_HDINSTALL` bereitet die Festplatte vor. Ist beim Erstellen des Bootmediums sowohl dort als auch beim Ziel der Installation der gewählte `BOOT_TYPE='hd'` werden die Installationsdateien direkt übertragen. Sollte ein direktes Kopieren nicht möglich sein werden diese später über scp oder über ein Remote-Update per Imonc übertragen.
Eine Einführung in die verschiedenen Festplatten- Installationsvarianten A oder B befindet sich am Anfang der fli4l-Dokumentation (Seite 15). Bitte unbedingt vorher lesen!

HD-Installation in sechs einfachen Schritten

1. lauffähiges fli4l-Bootmedium mit dem Paket base sowie `OPT_HDINSTALL` erstellen. Zusätzlich muss dieses Bootmedium ein Remote-Update ermöglichen. Es muss also entweder `OPT_SSHD` aktiv sein oder `START_IMOND` auf `'yes'` stehen. Wenn zum Ansprechen des Datenträgers Treiber erforderlich sind, die in der Standardinstallation nicht enthalten sind, müssen diese zusätzlich über `OPT_HDDRV` aktiviert werden.

2. den Router mit diesem Bootmedium booten.
3. am Router einloggen und den Befehl “hdinstall.sh” ausführen.
4. wenn die Aufforderung dazu erscheint die Dateien syslinux.cfg, kernel, rootfs.img, opt.img und rc.cfg mittels scp oder Imonc auf den Router nach /boot kopieren. Es wird empfohlen, dazu mit zwei fli4l-Verzeichnissen zu arbeiten, eines für das Setup und ein zweites für die spätere HD-Version. Bei der HD-Version stellen Sie die Variable `BOOT_TYPE='hd'` ein und beim Bootmedium dessen Typ entsprechend.

   Beim Remote-Update müssen natürlich die Dateien der HD-Version auf den Router übertragen werden!

5. Bootmedium entfernen, Router herunterfahren und neu starten (unter Verwendung von halt/reboot/poweroff). Der Router bootet jetzt von der Festplatte
6. bei Problemen den folgenden Abschnitt gut durchlesen.

HD-Installation ausführlich erklärt (inklusive Beispielen)

Zuerst muss ein Router-Bootmedium erstellt werden, bei dem in der Datei config/hd.txt das `OPT_HDINSTALL` mit den Installationsscripten und eventuell das `OPT_HDDRV` (falls zusätzliche Treiber benötigt werden) richtig konfiguriert wurden. Bitte dazu auch den Abschnitt zu `OPT_HDDRV` gründlich durchlesen!

Die Variable `BOOT_TYPE` in der base.txt wird entsprechend dem gewählten Setup-Medium eingestellt, es soll ja schließlich ein Setup durchgeführt werden. Die Variable `MOUNT_BOOT` in der base.txt muss auf `'rw'` eingestellt werden, damit später ggf. neue Archive (*.img) über das Netzwerk aufgespielt werden können.

Anschließend wird der Router von diesem Setup-Bootmedium gebootet. Durch Eingabe von “hdinstall.sh” an der fli4l-Console wird dann das Installationsprogramm gestartet. Nach Beantwortung von ein paar Fragen wird auf die Festplatte installiert. Eventuell erscheint am Ende noch die Aufforderung, dass man die für den Router benötigten Dateien per Remote-Update aufspielen soll.

Dieses Remote-Update keinesfalls vergessen, der Router bootet sonst nicht von der Festplatte. Zum Neustarten des Routers nach dem Remote-Update unbedingt reboot/halt/poweroff verwenden, andernfalls können die beim Remote-Update vorgenommenen Änderungen verloren gehen.

Das Installationsscript kann sowohl direkt am Router als auch über ssh von einem anderen PC aus gestartet werden. Im jedem Fall muss man sich vorher durch Eingabe des Passwortes am Router anmelden. Als ssh-Client für Windows-Rechner kann z.B. die Freeware Putty verwendet werden.

Konfiguration des Setup-Bootmediums

Bereits hier muss die Netzwerkkonfiguration richtig eingestellt sein damit man später noch Dateien über das Netzwerk aufspielen kann. Es wird empfohlen, DNS_DHCP zu diesem Zeitpunkt noch nicht zu aktivieren, da dies regelmäßig zu Problemen führt (der DHCP-Server hat vielleicht noch eine lease für den zu installierenden Router). Für ein Remote-Update mittels

`BOOT_TYPE`	entsprechend dem Bootmedium für die Installation einstellen
`MOUNT_BOOT='rw'`	notwendig, um später neue Archive (*.img) über Netzwerk auf die Platte kopieren zu können
`OPT_HDINSTALL='yes'`	notwendig um das Setup-Skript und die Tools zum Formatieren der Partitionen auf dem Bootmedium zu haben
(`OPT_HDDRV='yes'`)	nur dann notwendig, wenn ohne spezielle Treiber nicht auf die Festplatte zugegriffen werden kann
`OPT_SSHD='yes'`	nach dem Vorbereiten der Festplatte werden eventuell noch Dateien per remote Update übertragen. Dazu benötigt man entweder den sshd, imond (`IMOND='yes'`) oder ein anderes Paket, das einen Filetransfer erlaubt.

Tabelle 4.6.: Beispiel für die Konfiguration des Setup-Mediums

scp (befindet sich im Paket SSHD) bitte `OPT_SSHD='yes'` einstellen. Alternativ dazu kann man die Dateien per IMOND übertragen, dafür wird zusätzlich allerdings eine gültige DSL oder ISDN-Konfiguration benötigt. Alle nicht unbedingt nötigen Pakete bitte weglassen, also kein DNS_DHCP, SAMBA_LPD, LCD, Portforwarding usw.

Falls die Installation mit der Fehlermeldung

```
*** ERROR: can't create new partition table, see docu ***
```

abbricht, können mehrere Fehlerquellen in Frage kommen:

- die Festplatte ist in Benutzung, evtl. durch einen abgebrochenen Installationsversuch. Einfach neu booten und noch einmal versuchen.
- es werden zusätzliche Treiber benötigt, siehe `OPT_HDDRV`
- es gibt Hardwareprobleme, mehr dazu bitte im Anhang nachlesen.

Im letzten Schritt kann man nun die endgültige Fassung der Konfigurationsdateien erstellen und alle gewünschten Pakete hinzufügen.

Beispiele für eine fertige Installation nach Typ A und Typ B:

Ein Beispiel für jede Konfiguration finden Sie in Tabelle 4.7.

`BOOT_TYPE='hd'`	notwendig, da sie ja jetzt von Festplatte starten
`MOUNT_BOOT='rw|ro|no'`	nach Wahl. Um später neue fli4l-Archive über Netzwerk auf die Platte kopieren zu können ist 'rw' nötig.
`OPT_HDINSTALL='no'`	nach der erfolgreichen Installation ist dieses Paket nicht mehr notwendig.
`OPT_MOUNT`	nur aktivieren, falls eine Datenpartition erstellt wurde
(`OPT_HDDRV='yes'`)	nur notwendig, wenn ohne zusätzliche Treiber nicht auf die Festplatte zugegriffen werden kann.

Tabelle 4.7.: Beispiel für eine Installation nach Typ A oder B

Das Erstellen einer Swap-Partition wird nur angeboten, falls weniger als 32MB RAM im Router stecken und die Installation NICHT auf ein Flash-Medium durchgeführt wird!

4.9.2. OPT_MOUNT - Automatisches Einhängen von Dateisystemen

OPT_MOUNT hängt eine bei der Installation erstellte Datenpartition nach /data ein, eine Prüfung der Partition auf Fehler wird bei Bedarf automatisch durchgeführt. Ein evtl. vorhandenes CD-ROM wird nach /cdrom eingehängt, falls eine CD eingelegt ist. Für die swap-Partition wird das OPT_MOUNT nicht mehr benötigt!

OPT_MOUNT liest die Konfigurationsdatei hd.cfg auf der Boot-Partition und hängt die dort angegebenen Partitionen ein. Wenn das `OPT_MOUNT` mit einem Remote-Update auf einen bereits installierten Router übertragen wurde, muss diese Konfigurationsdatei ggf. geändert werden.

Auch bei einem Boot von CD-ROM kann das OPT_MOUNT nicht genutzt werden. Die CD kann in diesem Fall mit `MOUNT_BOOT='ro'` eingehängt werden.

Die Datei hd.cfg auf der DOS-Partition hat für einen Router nach Typ B mit Swap und Datenpartition den folgenden Inhalt (Beispiel):

```
hd_boot='sda1'
hd_opt='sda2'
hd_swap='sda3'
hd_data='sda4'
hd_boot_uuid='4A32-0C15'
hd_opt_uuid='c1e2bfa4-3841-4d25-ae0d-f8e40a84534d'
hd_swap_uuid='5f75874c-a82a-6294-c695-d301c3902844'
hd_data_uuid='278a5d12-651b-41ad-a8e7-97ccbc00e38f'
```

Nicht existierende Partitionen werden einfach weggelassen, bei einem Router Typ A ohne weitere Partitionen sieht das also so aus:

```
hd_boot='sda1'
hd_boot_uuid='4863-65EF'
```

4.9.3. OPT_EXTMOUNT - Manuelles Einhängen von Dateisystemen

OPT_EXTMOUNT hängt Datenpartitionen an jedem beliebigen Mountpoint im Dateisystem ein. Damit ist es möglich von Hand erstellte Dateisysteme einzuhängen und beispielsweise für einen Rsync-Server zur Verfügung zu stellen.

EXTMOUNT_N Die Anzahl der Datenpartitionen die extra eingehängt werden sollen.

EXTMOUNT_x_VOLUMEID Device, Label oder UUID des Volumens, das eingehängt werden soll. Mit dem Befehl 'blkid' kann man sich Device, Label und UUID aller verfügbaren Volumen anzeigen lassen.

EXTMOUNT_x_FILESYSTEM Das verwendete Dateisystem der Partition. fli4l unterstützt zur Zeit die Dateisysteme isofs, fat, vfat, ext2, ext3 und ext4.
(Der Standardwert `EXTMOUNT_x_FILESYSTEM='auto'` versucht das verwendete Dateisystem automatisch festzustellen.)

EXTMOUNT_x_MOUNTPOINT Der Pfad (Mountpoint) im Dateisystem in dem das Device eingehängt wird. Der Pfad muss vorher nicht existieren, er wird automatisch erzeugt.

EXTMOUNT_x_OPTIONS Wenn spezielle Optionen an den 'mount' Aufruf übergeben werden sollen können diese hier angegeben werden.

Beispiel:

```
EXTMOUNT_1_VOLUMEID='sda2'        # device
EXTMOUNT_1_FILESYSTEM='ext3'      # filesystem
EXTMOUNT_1_MOUNTPOINT='/mnt/data' # mountpoint for device
EXTMOUNT_1_OPTIONS=''             # extra mount options passed via mount -o
```

4.9.4. OPT_HDSLEEP – automatisches Abschalten für Festplatten einstellen

Eine Festplatte kann sich automatisch abschalten, wenn eine bestimmte Zeit ohne Aktivität verstreicht. Damit benötigt die Platte kaum noch Strom und macht keine Geräusche mehr. Wenn ein Zugriff auf die Festplatte erfolgt, läuft sie automatisch wieder an.

Nicht alle Festplatten vertragen häufiges Wiederanlaufen. Daher sollte man die Zeit nicht zu klein wählen. Ältere IDE-Platten bieten diese Funktion erst gar nicht an. Bei Flash-Medien ist diese Einstellung nicht sinnvoll und auch nicht notwendig.

HDSLEEP_TIMEOUT Diese Variable legt fest, nach welcher Zeit ohne Zugriff die Festplatte in den Power-Down-Modus gehen soll. Dann schaltet sie sich automatisch nach der Wartezeit aus und beim nächsten Zugriff wieder ein. Hierbei sind Wartezeiten in Minutenabständen von einer bis 20 Minuten sowie in Abständen von 30 Minuten von einer Halben bis zu fünf Stunden möglich. Eine Wartezeit von 21 oder 25 Minuten z.B. wird also auf 30 Minuten aufgerundet. Manche Festplatten ignorieren zu hohe Werte und stoppen dann schon nach einigen Minuten. Bitte unbedingt die korrekte Funktion durchtesten, da dies sehr von der jeweiligen Hardware abhängig ist!

```
HDSLEEP_TIMEOUT='2'               # wait 2 minutes until power down
```

4.9.5. OPT_RECOVER – Notfalloption

Diese Variable legt fest, ob Funktionen zur Erstellung einer Notfalloption verfügbar sind. Wenn die Option aktiviert ist wird der Befehl "mkrecover.sh" mit auf den Router übertragen. Mit diesem kann an der Kommandokonsole durch einfachen Aufruf die Notfallinstallation aktiviert werden. Beim installierten Paket "HTTPD" kann die Übertragung der aktuell laufenden Installation in eine Notfallinstallation im Menü Recover durchgeführt werden.

Um die Notfallinstallation zu nutzen, ist beim nächsten Reboot im Bootmenü die Auswahl Recover auszuwählen.

```
OPT_RECOVER='yes'
```

4.9.6. OPT_HDDRV - Treiber für Festplattencontroller

Mit OPT_HDDRV='yes' können eventuell benötigte zusätzliche Treiber aktiviert und installiert werden. Für IDE und SATA ist es in der Regel nicht nötig einen speziellen Treiber zu laden, da diese bereits vom Paket Base geladen werden.

HDDRV_N Die Anzahl der Treiber, die geladen werden sollen, wird hier eingestellt.

HDDRV_x Mit HDDRV_1 usw. werden die entsprechenden Treiber für die verwendeten Host-Adapter ausgewählt. Eine Liste der unterstützten Hostadapter ist in der initialen Konfigurationsdatei enthalten.

HDDRV_x_OPTION Mit HDDRV_x_OPTION können Optionen übergeben werden, die einige Treiber zum laden benötigen. Dies kann z.B. eine IO-Adresse sein. Bei den meisten Treibern kann diese Variable einfach leer gelassen werden.

Im Anhang (Seite 379) finden Sie eine Übersicht der Fehler, die bei Festplatten und CompactFlash am häufigsten auftreten.

Beispiel 1: Zugriff auf SCSI-Festplatte an einem Adaptec 2940

```
OPT_HDDRV='yes'             # install Drivers for Harddisk: yes or no
HDDRV_N='1'                 # number of HD drivers
HDDRV_1='aic7xxx'           # various aic7xxx based Adaptec SCSI
HDDRV_1_OPTION=''           # no need for options yet
```

Beispiel 2: Beschleunigter IDE-Zugriff beim PC-Engines ALIX

```
OPT_HDDRV='yes'             # install Drivers for Harddisk: yes or no
HDDRV_N='1'                 # number of HD drivers
HDDRV_1='pata_amd'          # AMD PCI IDE/ATA driver (e.g. ALIX)
HDDRV_1_OPTION=''           # no need for options yet
```

4.10. HTTPD - Status-Webserver

4.10.1. OPT_HTTPD - Mini-Webserver als Statusmonitor

Wer aus irgendeinem Grund keine Möglichkeit hat, den IMONC zu benutzen, weil er z.B. einen Mac benutzt, kann den Webserver benutzen, um den Status des fli4l-Routers abzurufen oder zu ändern. Mit OPT_HTTPD='yes' kann man den Statusmonitor verwenden.

Um den Status abzurufen, muss man in seinen Browser eine der folgenden Adressen eingeben:

```
http://fli4l/
http://fli4l.domain.lan/
http://192.168.6.1/
```

Hat der fli4l-Router einen abweichenden Namen, muss dieser statt "fli4l" verwendet werden. Dies gilt auch für den Domain-Namen und die obige IP-Adresse. Hat man den Webserver auf einen anderen Port gelegt (per HTTPD_PORT), muss man diesen mit angeben:

```
http://fli4l:81/
```

Es wird seit der Version 2.1.12 eine Login-Seite angezeigt, die nicht passwortgeschützt ist. Die passwortgeschützten Seiten befinden sich im Unterverzeichnis admin, also beispielsweise:

```
http://fli4l.domain.lan/admin/
```

Der Webserver lässt sich über folgende Variablen anpassen:

HTTPD_GUI_LANG Hiermit wird die Sprache eingestellt, in der das Webinterface dargestellt werden soll. Wird hier 'auto' eingetragen, wird die Spracheinstellung der Variablen `LOCALE` (in der base.txt) verwendet.

HTTPD_GUI_SKIN Diese Variable beeinflusst das Aussehen der Oberfläche. Das Paket *httpd_skins* stellt mit den Werten *fixed* und *notfixed* zwei alternative Darstellungen zur Verfügung. Standardwert dieser Variable ist *default*.

HTTPD_LISTENIP Der Webserver bindet sich normalerweise an eine sogenannte Wildcard-Adresse, so dass er auf einem beliebigen Interface angesprochen werden kann. Soll er sich nur an eine IP-Adresse binden, so kann es mit diesem Parameter eingestellt werden. Dazu die IP-Adressse wie folgt eintragen: `IP_NET_x_IPADDR`. Normalerweise bleibt dieser Parameter leer, damit die Standardeinstellung (ansprechbar auf einer beliebigen IP) greift.

Dieser Parameter dient lediglich dazu, den httpd nur an eine IP zu binden, so dass sich andere Instanzen an die anderen IPs des Routers binden können. Er kann nicht ohne weiteres dazu genutzt werden, den Zugang einzelner Subnetze zum Webinterface des Routers zu sperren. Dazu benötigt man weiterhin die Hilfe des Paketfilters.

Es ist auch möglich, hier durch Leerzeichen getrennt mehrere IP anzugeben.

HTTPD_PORT Soll der Webserver auf einem anderen Port laufen als 80, so ist dieser Wert anzupassen. Das ist normalerweise nicht zu empfehlen, da dann z.B. mit http://fli4l:81/ auf den Webserver zugegriffen werden muß.

HTTPD_PORTFW Setzt man diese Variable auf 'yes', kann man über das Webinterface Änderungen an der Portweiterleitung vornehmen. Es können Regeln gelöscht und hinzugefügt werden, Änderungen werden sofort wirksam. Änderungen an den Regeln gelten nur für die Laufzeit des Routers. Wird der Router neu gestartet, sind die Änderungen weg.

Diese Variable hat einen Defaultwert von 'yes'.

HTTPD_ARPING Der Webserver stellt den Online-Zustand der mit `HOST_x` aufgelisteten Hosts dar. Dazu verwendet er den *"Arp-Cache"*, ein Zwischenspeicher der die Adressen der lokalen Hosts zwischenspeichert. Hat ein Rechner lange nicht mehr mit dem Router kommuniziert, verschwindet seine Adresse aus dem *"Arp-Cache"* und der Host scheint aus zu sein. Will man den *"Arp-Cache"* aktuell halten (also das rausfallen eigentlich nicht benötigter Einträge verhindern), kann man `HTTPD_ARPING` auf *'yes'* setzen.

HTTPD_ARPING_IGNORE_N Legt die Anzahl der zu Ignorierenden Einträge fest

HTTPD_ARPING_IGNORE_x IP-Adresse oder Name des Hosts der nicht bei den ARPING-Test geprüft werden soll. Dies kann z.B. sinnvoll sein, bei Hosts die durch die regelmäßigen Netzwerpakete Ihren Akku schneller verbrauchen (Handys im WLAN).

4.10.2. Nutzerverwaltung

Der Webserver bietet eine ausgefeilte Benutzerverwaltung:

HTTPD_USER_N Hiermit wird die Anzahl der Benutzer eingestellt. Wird diese Variable auf 0 gesetzt, wird die Benutzerverwaltung komplett deaktiviert und jeder hat die Möglichkeit, auf den Webserver zuzugreifen.

HTTPD_USER_x_USERNAME HTTPD_USER_x_PASSWORD HTTPD_USER_x_RIGHTS Hier werden Benutzername und Passwort der einzelnen Benutzer eingetragen. Desweiteren wird für jeden Benutzer angegeben, auf welche Funktionen des Webservers er zugreifen darf. Diese Funktion wird mit der Variable `HTTPD_RIGHTS_x` geregelt. Im einfachsten Fall steht dort nur 'all', was bedeutet, dass der entsprechende Benutzer alles darf. Ansonsten hat die Variable den folgenden Aufbau:

```
'bereich1:recht1,recht2,... bereich2:...'
```

Statt für einen Bereich die einzelnen Rechte anzugeben, darf auch hier das Wort "all" eingesetzt werden, was wiederum heißt, das dieser Benutzer in diesem Bereich alle Rechte hat. Dabei gibt es folgende Bereiche und Rechte:

Bereich "status" Alles, was im Menü Status zu sehen ist.

- **view** Der Benutzer darf alle Menüpunkte aufrufen.
- **dial** Der Benutzer darf wählen und auflegen.
- **boot** Der Benutzer darf den Router herunterfahren & neu starten.
- **link** Der Benutzer darf Kanalbündlung an- und abschalten.
- **circuit** Der Benutzer darf den Circuit wechseln.
- **dialmode** Der Benutzer daf den Dialmode (Auto, Manual, Off) ändern.
- **conntrack** Der Benutzer darf die aktuell über den Router laufenden Verbindungen ansehen.
- **dyndns** Der Benutzer darf Log-Meldungen des `DYNDNS` (Seite 119) Paketes sehen.

Bereich "logs" Alles, was mit Logdateien zu tun hat (Verbindungen, Anrufe, Syslog)

- **view** Der Benutzer darf die Logdateien betrachten.
- **reset** Der Benutzer darf die Logdateien löschen.

Bereich "support" Alles, was nützlich ist, wenn man beispielsweise in der Newsgroup Hilfestellung sucht.

- **view** Der Benutzer darf die Links zur Doku, fli4l-Webseite, usw. abrufen
- **systeminfo** Der Benutzer darf detaillierte Informationen zur Konfiguration und zum aktuellen Status des Routers (z. B.: Firewall) abfragen.

Hier noch einige Beispiele:

HTTPD_USER_1_RIGHTS='all' Diese Angabe erlaubt einem Benutzer alles!

HTTPD_USER_2_RIGHTS='status:view logs:view support:all' Dieser Benutzer darf sich zwar alles ansehen, aber nichts ändern.

HTTPD_USER_3_RIGHTS='status:view,dial,link' Dieser Benutzer darf sich den Status der Internetverbindung ansehen, wählen und Kanalbündelung ein- und ausschalten.

HTTPD_USER_4_RIGHTS='status:all' Dieser Benutzer darf alles mit der Internetverbindung machen und neu starten (natürlich auch herunterfahren). Er darf aber nicht die Logdateien sehen oder löschen, die Timetable darf er auch nicht sehen...

4.10.3. OPT_OAC - Online Access Control

OPT_OAC (optionale Variable)

Aktiviert das Modul 'Online Access Control'. Hierüber kann der Internet-Zugang jedes im Paket dns_dhcp (Seite 98) konfigurierten Rechners selektiv deaktiviert werden.

Es gibt auch ein Kommandozeilen-Tool, welches die Steuerung über andere Pakete wie z.B. EasyCron möglich macht:

/usr/local/bin/oac.sh

Die Optionen werden beim Aufruf angezeigt.

OAC_WANDEVICE (optionale Variable)

Schränkt die Online Zugangssperre auf Verbindungen über dieses Netzwerkdevice ein. z.B. 'pppoe'

OAC_INPUT (optionale Variable)

Bietet Schutz vor der Umgehung via Proxy.

OAC_INPUT='default' sperrt die konfigurierten Ports von: Privoxy, Squid, Tor, SS5, Transproxy.

OAC_INPUT='tcp:8080 tcp:3128' sperrt TCP Port 8080 und 3128. Dies ist eine Space-separierte Liste an zu sperrenden Ports mit dazugehörendem Protokoll (udp, tcp). Fehlt das Protokoll, werden udp und tcp Ports gesperrt.

Weglassen dieser Variable oder Inhalt 'no' deaktiviert die Funktion.

OAC_ALL_INVISIBLE (optionale Variable)

Schaltet die Gesamtübersicht aus, wenn mindestens eine sichtbare Gruppe existiert. Existiert nicht mindestens eine sichtbare Gruppe, so hat diese Variable keine Wirkung.

OAC_LIMITS (optionale Variable)

Gibt eine durch Leerzeichen getrennte Liste der Zeitlimits an, die zur Auswahl stehen. Die Limits werden in Minuten angegeben. Damit kann eine zeitlich limitierte Sperre oder Freigabe erreicht werden.

Default: '30 60 90 120 180 360 540'

OAC_MODE (optionale Variable)

Mögliche Werte: `'DROP'` oder `'REJECT'` (default)

OAC_GROUP_N (optionale Variable)

Anzahl der Clientgruppen. Dient der Übersichtlichkeit, erlaubt aber auch über das Web-Interface eine gesamte Gruppe gesammelt freizugeben oder zu sperren.

OAC_GROUP_x_NAME (optionale Variable)

Name der Gruppe - Dieser Name wird im Web-Interface angezeigt und ist auch über die das Kommandozeilenscript 'oac.sh' nutzbar.

OAC_GROUP_x_BOOTBLOCK (optionale Variable)

Wenn hier 'yes' steht, werden alle Clients der Gruppe beim Bootvorgang bereits gesperrt. Hilfreich, wenn Rechner idr. gesperrt sein sollen und nur ausnahmsweise nicht.

OAC_GROUP_x_INVISIBLE (optionale Variable)

Markiert die Gruppe als unsichtbar. Sinnvoll, wenn einige Rechner vorgesperrt werden sollen aber diese nicht als eigene Gruppe im Web-If sichtbar sein sollen. Das Kommandozeilentool oac.sh kann diese trotzdem ansprechen, wenn man das braucht, z.B. von easycron aus.

OAC_GROUP_x_CLIENT_N (optionale Variable)

Anzahl der Clients in der Gruppe.

OAC_GROUP_x_CLIENT_x (optionale Variable)

Name des Clients - Siehe Paket dns_dhcp (Seite 98).

OAC_BLOCK_UNKNOWN_IF_x (optionale Variable)

Liste der in base.txt definierten Interfaces, auf welchen nur in dns_dhcp.txt definierte Hosts ins Internet dürfen. Nicht-definierte Hosts sind hier damit generell gesperrt.

4.11. HWSUPP - Unterstützung von Hardware

4.11.1. Beschreibung

Das Paket stellt die Unterstützung für die Nutzung spezieller Hardwarekomponenten bereit.
Unterstützte Hardwarekomponenten/-elemente:

- Temeratursensoren
- LEDs
- Spannungssensoren
- Lüfterdrehzahlen
- Taster
- Watchdog
- VPN-Karten

Unterstützung gibt es für die folgende Systeme/Mainboards/VPN-Karten:

- Standard PC-Hardware
  - LEDs einer PC-Tastatur
- ACPI-PC-Hardware
- Embedded Systeme
  - AEWIN SCB6971
  - Fujitsu Siemens Futro S200
  - PC Engines ALIX
  - PC Engines APU
  - PC Engines WRAP
  - Soekris net4801
  - Soekris net5501
- Mainboards
  - Commell LE-575
  - GigaByte GA-M521-S3
  - LEX CV860A
  - SuperMicro PDSME
  - SuperMicro X7SLA
  - Tyan S5112
  - WinNet PC640
  - WinNet PC680
- VPN Karten (PCI, miniPCI and miniPCIe)
  - vpn1401 vpn1411

4.11.2. Konfiguration des Paketes HWSUPP

Die Konfiguration erfolgt, wie bei allen fli4l Paketen, durch Anpassung der Datei `Pfad/fli4l-3.10.4/<config>/hwsupp.txt` an die eigenen Anforderungen.

OPT_HWSUPP Die Einstellung `'no'` deaktiviert das OPT_HWSUPP vollständig. Es werden keine Änderungen am fli4l Archiv `rootfs.img` bzw. dem Archiv `opt.img` vorgenommen. Weiterhin überschreibt das OPT_HWSUPP grundsätzlich keine anderen Teile der fli4l Installation.
Um OPT_HWSUPP zu aktivieren, ist die Variable `OPT_HWSUPP` auf `'yes'` zu setzen.

HWSUPP_TYPE In dieser Konfigurationsvariable wird die zu unterstützende Hardware festgelegt. Folgende Werte stehen zur Verfügung:

- sim
- generic-pc

- generic-acpi
- aewin-scb6971
- commell-le575
- fsc-futro-s200
- gigabyte-ga-m52l-s3
- lex-cv860a
- pcengines-alix
- pcengines-apu
- pcengines-wrap
- soekris-net4801
- soekris-net5501
- supermicro-pdsme
- supermicro-x7sla
- tyan-s5112
- winnet-pc640
- winnet-pc680

HWSUPP_WATCHDOG Die Einstellung `'yes'` aktiviert den Watchdog-Daemon falls die gewählte Hardware einen Watchdog besitzt. Durch den Watchdog wird ein hängendes System automatisch neu gestartet werden.

HWSUPP_CPUFREQ Die Einstellung `'yes'` aktiviert die Anpassung der Prozessor-Taktfrequenz.

HWSUPP_CPUFREQ_GOVERNOR Auswahl des CPU-Frequenz-Reglers. Die Auswahl des Reglers steuert das Verhalten der Anpassung der Prozessor-Taktfrequenz. Zur Auswahl stehen:

- performance

  Der Prozessor läuft immer mit der maximalen Taktfrequenz.

- ondemand

  Die CPU-Frequenz wird an die Rechenleistung angepasst. Dabei kann die CPU-Frequenz u.U. sprunghaft angehoben oder abgesenkt werden.

- conservative

  Die CPU-Frequenz wird an die Rechenleistung angepasst. Die CPU-Frequenz wird schrittweise angehoben bzw. abgesenkt.

- powersave

  Der Prozessor läuft immer mit der minimmalen Taktfrequenz.

- userspace

  Der Prozessortakt kann manuell oder von einem Anwenderskript über die sysfs-Variable `/devices/system/cpu/cpu`<*n*>`/cpufreq/scaling_setspeed` gesetzt werden.

HWSUPP_LED_N Definiert die Anzahl der LEDs. Hier sollte die Anzahl der LEDs die die verwendete Hardware bereitstellt stehen.

HWSUPP_LED_x Definiert die Information, die durch das LED angezeigt werden soll. Folgende Informationen sind möglich:

- ready - Der fli4l-Router ist betriebsbereit[7]
- online - der fli4l-Router ist mit den Internet verbunden
- trigger - Anzeige wird durch einen LED-Trigger gesteuert
- user - Anzeige wird durch ein Benutzerscript gesteuert

Die Liste der möglichen Information kann durch andere Pakete erweitert werden. So ist bei geladenem WLAN-Paket z.B. die Anzeige

- wlan - das WLAN ist aktiviert

möglich.

Im Anhang B.10 finden sich Hinweise für Paket-Entwickler wie eine solche Erweiterung anzulegen ist.

HWSUPP_LED_x_DEVICE Gibt das LED-Device an.

Hier wird entweder ein LED-Device eingetragen (zu finden unter `/sys/class/leds/` im Dateisystem des Routers) oder eine GPIO[8]-Nummer.

Eine Liste gültiger Namen der LED-Devices für den jeweiligen `HSUPP_TYPE` findet sich im Anhang B.7.1.

Die GPIO-Nummer musss im Format `gpio::x` eingegeben werden. Wenn man ein GPIO eingeträgt, so wird das dazugehörige LED-Device automatisch angelegt. Durch Voranstellen von `/` wird die Funktionsweise des GPIO invertiert.

Beispiele:

```
HWSUPP_LED_1_DEVICE='apu::1'
HWSUPP_LED_2_DEVICE='gpio::237'
HWSUPP_LED_3_DEVICE='/gpio::245'
```

HWSUPP_LED_PARAM Definiert Parameter für die ausgewählte LED Anzeige.

Je nach Auswahl in `HWSUPP_LED_x` hat `HWSUPP_LED_x_PARAM` eine unterschiedliche Bedeutung.

Ist `HWSUPP_LED_x='trigger'`, so ist der Name des LED-Triggers, der die Ansteuerung der LED kontrolliert, in `HWSUPP_LED_x_PARAM` einzutragen.

Die verfügbaren Trigger können mit dem Shell-Kommando `cat /sys/class/leds/*/trigger` angezeigt werden.

[7] Ist `HWSUPP_LED_x='ready'`, so wird der Bootfortschritt durch eine Blink-folge angezeigt (siehe Anhang B.9).

[8] Ein GPIO (General Purpose Input/Output) ist ein Kontaktstift an einem integrierten Schaltkreis, dessen Verhalten durch Programmierung bestimmbar ist, unabhängig, ob als Ein- oder Ausgabekontakt.

Neben den Triggern die von z.B. netfilter oder Hardwaretreibern wie ath9k erzeugt werden, können weitere Trigger-Module über `HWSUPP_DRIVER_x` geladen werden.

Beispiele:

```
HWSUPP_LED_1='trigger'
HWSUPP_LED_1_PARAM='heartbeat'
HWSUPP_LED_2='trigger'
HWSUPP_LED_2_PARAM='netfilter-ssh'
```

Ist `HWSUPP_LED_x='user'`, so ist in `HWSUPP_LED_PARAM` ein gültiger Skriptname inclusive Pfad einzutragen.

Beispiel:

```
HWSUPP_LED_1='user'
HWSUPP_LED_1_PARAM='/usr/local/bin/myledscript'
```

Ist `HWSUPP_LED_x='wlan'`, so definiert `HWSUPP_LED_PARAM` ein oder mehrere WLAN Devices, deren Zustand angezeigt wird. Mehrere WLAN Devices sind durch Leerzeichen zu trennen.

Wird der Zustand mehrerer WLAN Devices mit einer LED Angezeigt, so hat die LED folgende Bedeutung:

- aus - alle WLAN Devices sind inaktiv
- blinkt - ein Teil der WLAN Devices ist aktiv
- an - alle WLAN Devices sind aktiv

Beispiel:

```
HWSUPP_LED_1='wlan'
HWSUPP_LED_1_PARAM='wlan0 wlan1'
```

HWSUPP_BOOT_LED Definiert eine LED die während des Bootvorgangs den Fortschritt durch eine Blinkfolge angezeigt.

Wenn für eine LED `HWSUPP_LED_x='ready'` gesetzt ist so hat diese Einstellung Vorrang und `HWSUPP_BOOT_LED` wird ignoriert.

HWSUPP_BUTTON_N Definiert die Anzahl der BUTTONs. Hier sollte die Anzahl der Taster die die verwendete Hardware bereitstellt stehen.

HWSUPP_BUTTON_x Definiert die Aktion die durch drücken des Tasters durchgeführt werden soll. Folgende Aktionen sind möglich:

- reset - Startet den fli4l-Router neu
- online - Baut die Internetverbindung auf bzw. beendet diese

- user - Ein User-Script wird ausgeführt

Die Liste der möglichen Aktionen kann durch andere Pakete erweitert werden. So ist bei geladenem WLAN-Paket z.B. die Aktion

- wlan - WLAN aktivieren bzw. deaktivieren

möglich.

HWSUPP_BUTTON_x_DEVICE Gibt das Button-Device an. Hier ist eine GPIO-Nummer einzutragen.

Die GPIO-Nummer musss im Format `gpio::x` eingegeben werden. Durch Voranstellen von `/` wird die Funktionsweise des GPIO invertiert.

Eine Liste der vordefinierten GPIO's für den jeweiligen `HSUPP_TYPE` findet sich im Anhang B.7.2.

Beispiele:

```
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_2_DEVICE='/gpio::237'
```

HWSUPP_BUTTON_x_PARAM Definiert Parameter für die ausgewählte Aktion.

Je nach Wert in `HWSUPP_BUTTON_x` hat `HWSUPP_BUTTON_x_PARAM` eine unterschiedliche Funktion.

Ist `HWSUPP_BUTTON_x='user'`, so definiert `HWSUPP_BUTTON_x_PARAM` ein Script das beim Drücken des Tasters ausgeführt werden soll.

Beispiel:

```
HWSUPP_BUTTON_1='user'
HWSUPP_BUTTON_2_WLAN='/usr/local/bin/myscript'
```

Ist `HWSUPP_BUTTON_x_ACTION='wlan'`, so sind in `HWSUPP_BUTTON_x_PARAM` das oder die WLAN Devices einzutragen, die durch Drücken des Tasters aktiviert bzw. deaktiviert werden. Mehrere WLAN Devices sind durch Leerzeichen zu trennen.

Beispiel:

```
HWSUPP_BUTTON_2='wlan'
HWSUPP_BUTTON_2_WLAN='wlan0 wlan1'
```

4.11.3. Experten-Einstellungen

Die folgenden Einstellungen sollten nur gemacht werden, wenn man genau weiss

- welche Hardware man hat und welche zusätzlichen Treiber man dafür benötigt

- an welchen Adressen welche I²C-Geräte[9]liegen.

Nach dem Aktivieren der Experteneinstellungen erhält man eine Warnung beim mkfli4l Bau.

HWSUPP_DRIVER_N Anzahl der zusätzlich zu ladenden Treiber. Die Treiber in `HWSUPP_DRIVER_x` werden in der angegebenen Reihenfolge geladen.

HWSUPP_DRIVER_x Name des Treibers (ohne Dateiendung `.ko`).

Beispiel:

```
HWSUPP_DRIVER_N='2'
HWSUPP_DRIVER_1='i2c-piix4'     # I2C Bus Treiber
HWSUPP_DRIVER_2='gpio-pcf857x'  # I2C GPIO Expander
```

HWSUPP_I2C_N Anzahl der zu ladenden I²C-Geräte.

I²C unterstützt keine PnP-Mechanismen. Daher sind für jedes zu ladende I²C-Gerät die Busnummer, die Geräteadresse und der Gerätetyp anzugeben.

HWSUPP_I2C_x_BUS I²C-Busnummer an der das zu ladende Gerät angeschlossen ist.

Die Bussnummer ist im Format `i2c-x` anzugeben.

HWSUPP_I2C_x_ADDRESS I²C-Busadresse des Geräts.

Die Adresse ist als Hexadezimalzahl im Bereich von `0x03` bis `0x77` anzugeben.

HWSUPP_I2C_x_DEVICE Der Typ des I²C-Geräts der vom einem zuvor geladenen Treiber erkannt wird.

Beispiel:

```
HWSUPP_I2C_N='1'
HWSUPP_I2C_1_BUS='i2c-1'
HWSUPP_I2C_1_ADDRESS='0x38'
HWSUPP_I2C_1_DEVICE='pcf8574a' # Unterstützt von gpio-pcf857x Treiber
```

4.11.4. Unterstützung von VPN-Karten

OPT_VPN_CARD Die Einstellung `'no'` deaktiviert das OPT_VPN_CARD vollständig. Es werden keine Änderungen am fli4l Archiv `rootfs.img` bzw. dem Archiv `opt.img` vorgenommen. Weiterhin überschreibt das OPT_VPN_CARD grundsätzlich keine anderen Teile der fli4l Installation.
Um OPT_VPN_CARD zu aktivieren, ist die Variable `OPT_VPN_CARD` auf `'yes'` zu setzen.

VPN_CARD_TYPE In dieser Konfigurationsvariable wird der zu unterstützende VPN Beschleuniger festgelegt. Folgende Werte stehen zur Verfügung:

- hifn7751 - Soekris vpn1401 und vpn1411
- hifnhipp

[9]Ein I²C-Bus oder SMBus ist ein serieller Bus der im PC z.B. zum Auslesen von Temperatur-Sensoren verwendet wird. Vielfach ist der I²C-Bus oder SMBus auf einer Stiftleiste verfügbar und kann für eigene Hardwareerweiterungen genutzt werden.

4.12. IPv6 - Internet Protokoll Version 6

4.12.1. Einleitung

Dieses Paket ermöglicht es, den fli4l-Router in vielerlei Hinsicht IPv6-tauglich zu machen. Dazu gehören Angaben über die IPv6-Adresse des Routers, die verwalteten IPv6-(Sub-)Netze, vordefinierte IPv6-Routen und Firewall-Regeln bzgl. IPv6-Paketen. Des Weiteren lassen sich IPv6-Dienste wie DHCPv6 konfigurieren. Schließlich ist es möglich, Tunnel zu IPv6-Anbietern automatisch aufbauen zu lassen. Dies funktioniert momentan aber leider nur mit so genannten 6in4-Tunneln, wie sie etwa der Anbieter "SixXS" unterstützt. Andere Technologien (AYIYA, 6to4, Teredo) werden zur Zeit nicht unterstützt.

IPv6 ist der Nachfolger des Internet-Protokolls IPv4. Hauptsächlich wurde es entwickelt, um die relativ kleine Menge von eindeutigen Internet-Adressen zu vergrößern: Während IPv4 ungefähr 2^{32} Adressen unterstützt,[10] sind es bei IPv6 bereits 2^{128} Adressen. Dadurch kann mit IPv6 jedem kommunizierenden Host eine eindeutige Adresse zugeordnet werden, und man ist nicht mehr auf Techniken wie NAT, PAT, Masquerading etc. angewiesen.

Neben diesem Aspekt spielten bei der Entwicklung des IPv6-Protokolls auch Themen wie Selbstkonfiguration und Sicherheit eine Rolle. Dies wird in späteren Abschnitten aufgegriffen.

Das größte Problem bei IPv6 ist dessen Verbreitung: Momentan wird IPv6 – verglichen mit IPv4 – nur sehr spärlich verwendet. Das liegt daran, dass IPv6 und IPv4 technisch nicht miteinander kompatibel sind und somit alle Software- und Hardware-Komponenten, die an der Paket-Weiterleitung im Internet beteiligt sind, für IPv6 nachgerüstet werden müssen. Auch bestimmte Dienste wie DNS (Domain Name System) müssen für IPv6 entsprechend aufgebohrt werden.

Hier tut sich also ein Teufelskreis auf: Die geringe Verbreitung von IPv6 bei Dienstanbietern im Internet führt zu Desinteresse seitens der Router-Hersteller, ihre Geräte mit IPv6-Funktionalität auszustatten, was wiederum dazu führt, dass Dienstanbieter die Umstellung auf IPv6 scheuen, weil sie fürchten, dass sich der Aufwand nicht lohnt. Erst langsam wendet sich das Blatt zugunsten von IPv6, nicht zuletzt unter dem immer stärkeren Druck der knappen Adressvorräte.[11]

4.12.2. Adressformat

Eine IPv6-Adresse besteht aus acht Zwei-Byte-Werten, die hexadezimal notiert werden:

Beispiel 1: `2001:db8:900:551:0:0:0:2`

Beispiel 2: `0:0:0:0:0:0:0:1` (IPv6-Loopback-Adresse)

Um die Adressen etwas übersichtlicher zu gestalten, werden aufeinander folgende Nullen zusammengelegt, indem sie entfernt werden und lediglich zwei unmittelbar aufeinander folgende Doppelpunkte verbleiben. Die obigen Adressen können also auch so geschrieben werden:

Beispiel 1 (kompakt): `2001:db8:900:551::2`

Beispiel 2 (kompakt): `::1`

Eine solche Kürzung ist aber nur höchstens einmal erlaubt, um Mehrdeutigkeiten zu vermeiden. Die Adresse `2001:0:0:1:2:0:0:3` kann also entweder zu `2001::1:2:0:0:3` oder zu `2001:0:0:1:2::3` verkürzt werden, nicht aber zu `2001::1:2::3`, da jetzt unklar wäre, wie die vier Nullen jeweils auf die zusammengezogenen Bereiche verteilt werden sollen.

[10] nur ungefähr, weil einige Adressen speziellen Zwecken dienen, etwa für Broad- und Multicasting

[11] Inzwischen sind die letzten IPv4-Adressblöcke von der IANA vergeben worden.

Eine weitere Mehrdeutigkeit existiert, wenn eine IPv6-Adresse mit einem Port (TCP oder UDP) kombiniert werden soll: In diesem Fall darf man den Port nicht unmittelbar mit Doppelpunkt und Wert anschließen, weil der Doppelpunkt bereits innerhalb der Adresse verwendet wird und somit in manchen Fällen unklar wäre, ob die Port-Angabe nicht vielleicht doch eine Adress-Komponente darstellt. Deshalb muss in solchen Fällen die IPv6-Adresse in eckigen Klammern angegeben werden. Dies ist auch die Syntax, wie sie in URLs gefordert wird (etwa wenn im Web-Browser eine numerische IPv6-Adresse verwendet werden soll).

Beispiel 3: `[2001:db8:900:551::2]:1234`

Ohne die Verwendung von Klammern entsteht die Adresse `2001:db8:900:551::2:1234`, die unverkürzt der Adresse `2001:db8:900:551:0:0:2:1234` entspricht und somit keine Port-Angabe besitzt.

4.12.3. Konfiguration

Allgemeine Einstellungen

Die allgemeinen Einstellungen beinhalten zum einen die Aktivierung der IPv6-Unterstützung und zum anderen die optionale Vergabe einer IPv6-Adresse an den Router.

OPT_IPV6 Aktiviert die IPv6 Unterstützung.

Standard-Einstellung: `OPT_IPV6='no'`

HOSTNAME_IP6 (optional) Diese Variable stellt explizit die IPv6-Adresse des Routers ein. Falls die Variable nicht gesetzt wird, wird die IPv6-Adresse auf die Adresse des ersten konfigurierten IPv6-Subnetzes gesetzt (`IPV6_NET_x`, siehe unten).

Beispiel: `HOSTNAME_IP6='IPV6_NET_1_IPADDR'`

Subnetz-Konfiguration

In diesem Abschnitt wird die Konfiguration von einem oder mehreren IPv6-Subnetzen vorgestellt. Ein IPv6-Subnetz ist ein IPv6-Adressraum, der über ein so genanntes Präfix spezifiziert wird und an eine bestimmte Netzwerk-Schnittstelle gebunden ist. Weitere Einstellungen betreffen das Veröffentlichen des Präfixes und des DNS-Dienstes innerhalb des Subnetzes sowie einen optionalen Router-Namen innerhalb dieses Subnetzes.

IPV6_NET_N Diese Variable enthält die Anzahl der verwendeten IPv6-Subnetze. Mindestens ein IPv6-Subnetz sollte definiert werden, um IPv6 im lokalen Netz nutzen zu können.

Standard-Einstellung: `IPV6_NET_N='0'`

IPV6_NET_x Diese Variable enthält für ein bestimmtes IPv6-Subnetz die IPv6-Adresse des Routers sowie die Größe der Netzmaske in CIDR-Notation. Soll das Subnetz öffentlich geroutet werden, so stammt es in der Regel vom Internet- bzw. Tunnel-Anbieter.

Wichtig: *Soll in dem Subnetz die zustandslose Selbstkonfiguration (siehe den Abschnitt zu `IPV6_NET_x_ADVERTISE` weiter unten) aktiviert werden, dann muss die Länge des Subnetz-Präfixes 64 Bit betragen!*

Wichtig: *Ist das Subnetz an einen Tunnel angeschlossen (siehe `IPV6_NET_x_TUNNEL` weiter unten), dann darf hier nur der Teil der Router-Adresse angegeben werden, der* nicht

zum dem Tunnel zugeordneten Subnetz-Präfix (zu finden in `IPV6_TUNNEL_x_PREFIX`) gehört, da jenes Präfix und diese Adresse miteinander kombiniert werden! In früheren Versionen des IPv6-Pakets gab es die Variable `IPV6_TUNNEL_x_PREFIX` noch nicht, und Subnetz-Präfix sowie Router-Adresse wurden zusammen in `IPV6_NET_x` festgelegt. Das funktioniert aber nicht, wenn das vom Tunnelanbieter zugeordnete Subnetz-Präfix erst dynamisch beim Tunnelaufbau mitgeteilt wird. Außerdem bleibt dabei die Länge des Subnetz-Präfixes (hier: /48) im Dunkeln, so dass bestimmte vordefinierte Routen nicht ordentlich gesetzt werden können, was beim Routen zu bestimmten Zieladressen dann zu seltsamen Effekten führt. Deswegen wurde die Konfiguration entsprechend aufgetrennt.

Beispiele:

```
IPV6_NET_1='2001:db8:1743:42::1/64'       # ohne Tunnel: komplette Adresse
IPV6_NET_1_TUNNEL=''

IPV6_NET_2='0:0:0:42::1/64'               # mit Tunnel: partielle Adresse
IPV6_NET_2_TUNNEL='1'
IPV6_TUNNEL_1_PREFIX='2001:db8:1743::/48' # s. Abschnitt "Tunnel-Konfiguration"
```

IPV6_NET_x_DEV Diese Variable enthält für ein bestimmtes IPv6-Subnetz den Namen der Netzwerk-Schnittstelle, an welche das IPv6-Netz gebunden wird. Dies kollidiert *nicht* mit den in der Basis-Konfiguration (`base.txt`) vergebenen Netzwerk-Schnittstellen, da einer Netzwerk-Schnittstelle sowohl IPv4- als auch IPv6-Adressen zugeordnet werden dürfen.

Beispiel: `IPV6_NET_1_DEV='eth0'`

IPV6_NET_x_TUNNEL Diese Variable enthält für ein bestimmtes IPv6-Subnetz den Index des zugehörigen Tunnels. Das Präfix des angegebenen Tunnels wird mit der Router-Adresse kombiniert, um die vollständige IPv6-Adresse des Routers zu erhalten. Wenn die Variable leer oder nicht definiert ist, dann gehört zu dem Subnetz kein Tunnel, und in `IPV6_NET_x` muss die vollständige IPv6-Adresse des Routers inklusive Netzmaske angegeben werden (siehe oben).

Ein Tunnel kann mehreren Subnetzen zugeordnet werden, da ein Tunnel-Subnetzpräfix normalerweise groß genug ist, dass er in mehrere Subnetze aufgeteilt werden kann (/56 oder größer). Andersherum ist es natürlich nicht möglich, einem Subnetz mehrere Tunnel-Subnetzpräfixe zuzuordnen, da die Adresse des Subnetzes in diesem Fall mehrdeutig wäre.

Beispiel: `IPV6_NET_1_TUNNEL='1'`

IPV6_NET_x_ADVERTISE Diese Variable legt fest, ob das eingestellte Subnetz-Präfix im LAN mittels "Router Advertisements" verbreitet wird. Dies wird für die so genannte "stateless autoconfiguration" (zustandslose Selbstkonfiguration) verwendet und ist nicht mit DHCPv6 zu verwechseln. Mögliche Werte sind "yes" und "no".

Es ist empfehlenswert, diese Einstellung zu aktivieren, es sei denn, alle Adressen im Netz werden statisch vergeben oder ein anderer Router ist bereits dafür zuständig, das Subnetz-Präfix anzukündigen.

Wichtig: *Die automatische Verteilung des Subnetzes funktioniert nur, wenn das Subnetz ein /64-Netz ist, d.h. wenn die Länge des Subnetz-Präfixes 64 Bit beträgt! Der Grund*

hierfür ist, dass die anderen Hosts im Netzwerk ihre IPv6-Adresse aus dem Präfix und ihrer Host-MAC-Adresse berechnen und dies nicht funktioniert, wenn der Host-Anteil nicht 64 Bit beträgt. Wenn die Selbstkonfiguration fehlschlägt, sollte also geprüft werden, ob das Subnetz-Präfix nicht vielleicht falsch angegeben worden ist (z.B. als /48).

Standard-Einstellung: `IPV6_NET_1_ADVERTISE='yes'`

IPV6_NET_x_ADVERTISE_DNS Diese Variable legt fest, ob auch der lokale DNS-Dienst im IPv6-Subnetz mittels "Router Advertisements" angekündigt werden soll. Dies funktioniert aber nur, wenn die IPv6-Funktionalität des DNS-Dienstes mit Hilfe der Variable `DNS_SUPPORT_IPV6`='yes' aktiviert ist. Mögliche Werte sind "yes" und "no".

Standard-Einstellung: `IPV6_NET_1_ADVERTISE_DNS='no'`

IPV6_NET_x_DHCP Diese Variable aktiviert den DHCPv6-Dienst für dieses IPv6-Subnetz. Mögliche Werte sind "yes" und "no". DHCPv6 wird dabei nur verwendet, um Hosts in diesem Subnetz Informationen zum Domänen-Namen und zur Adresse des zu verwendenden DNS-Servers zukommen zu lassen. Die Zuordnung von IPv6-Adressen ist über DHCPv6 mit fli4l momentan nicht möglich.

Die Adresse des DNS-Servers wird nur dann via DHCPv6 veröffentlicht, wenn die IPv6-Unterstützung des DNS-Dienstes via `DNS_SUPPORT_IPV6` im dns_dhcp-Paket eingeschaltet ist.

Wichtig: *Die Variablen `IPV6_NET_x_ADVERTISE_DNS` und `IPV6_NET_x_DHCP` schließen sich nicht gegenseitig aus, sondern können beide aktiviert sein. In diesem Fall kann die Adresse des DNS-Servers auf zweierlei Weise von Hosts im lokalen Netzwerk in Erfahrung gebracht werden.*

Pro Netzwerkschnittstelle darf höchstens ein gebundenes IPv6-Subnetz für DHCPv6 konfiguriert werden!

Standard-Einstellung: `IPV6_NET_1_DHCP='no'`

IPV6_NET_x_NAME (optional) Diese Variable legt einen Interface-spezifischen Hostnamen für den Router in diesem IPv6-Subnetz fest.

Beispiel: `IPV6_NET_1_NAME='fli4l-subnet1'`

Tunnel-Konfiguration

Dieser Abschnitt stellt die Konfiguration von 6in4-IPv6-Tunneln vor. Ein solcher Tunnel bietet sich an, wenn der eigene Internet-Anbieter kein IPv6 von Haus aus unterstützt. In diesem Fall wird mit einem bestimmten Internet-Host eines Tunnel-Brokers, dem so genannten PoP (Point of Presence), via IPv4 eine bidirektionale Verbindung aufgebaut, über die dann alle IPv6-Pakete verpackt geroutet werden (deswegen 6 "in" 4, weil die IPv6-Pakete innerhalb von IPv4-Paketen gekapselt werden).[12] Damit das funktioniert, muss zum einen der Tunnel aufgebaut und zum anderen der Router so konfiguriert werden, dass die IPv6-Pakete, die ins Internet sollen, auch über den Tunnel geroutet werden. Der erste Teil wird in diesem Abschnitt konfiguriert, der zweite Teil wird im nächsten Abschnitt beschrieben.

[12] Es handelt sich um das IPv4-Protokoll 41, "IPv6 encapsulation".

IPV6_TUNNEL_N Diese Variable enthält die Anzahl der aufzubauenden 6in4-Tunnel.

Beispiel: `IPV6_TUNNEL_N='1'`

IPV6_TUNNEL_x_TYPE Diese Variable bestimmt den Typ des Tunnels. Momentan werden die Werte "raw" für "rohe" Tunnel, "static" für statische Tunnel, "sixxs" für dynamische Heartbeat-Tunnel des Anbieters SixXS und "he" für Tunnel des Anbieters Hurricane Electric unterstützt. Mehr zu Heartbeat-Tunneln steht im nächsten Absatz.

Beispiel: `IPV6_TUNNEL_1_TYPE='sixxs'`

IPV6_TUNNEL_x_DEFAULT Diese Variable legt fest, ob IPv6-Pakete, die nicht an das lokale bzw. die lokalen Netze adressiert sind, über diesen Tunnel geroutet werden sollen. Es kann nur einen solchen Tunnel geben (weil nur eine Default-Route existieren kann). Mögliche Werte sind "yes" und "no".

Wichtig: *Genau ein Tunnel sollte ein Default-Gateway für nausgehende IPv6-Daten sein, da andernfalls eine Kommunikation mit IPv6-Hosts im Internet nicht möglich ist! Die ausschließliche Verwendung von Nicht-Default-Tunneln ist nur sinnvoll, wenn ausgehende IPv6-Daten über eine separat konfigurierte Default-Route geschickt werden, die nicht mit einem Tunnel zusammenhängt. Siehe hierzu auch die Einleitung zum Unterabschnitt "Routen-Konfiguration" sowie die Beschreibung der Variable `IPV6_ROUTE_x` weiter unten.*

Standard-Konfiguration: `IPV6_TUNNEL_1_DEFAULT='no'`

IPV6_TUNNEL_x_PREFIX Diese Variable enthält den IPv6-Subnetzpräfix des Tunnels in CIDR-Notation, d.h. es wird sowohl eine IPv6-Adresse als auch die Länge des Präfixes angegeben. Diese Angabe wird in der Regel vom Tunnelanbieter vorgegeben. Bei Tunnelanbietern, die den Präfix beim Tunnelaufbau jedes Mal neu vergeben, ist diese Angabe unnötig. (Momentan werden solche Anbieter aber noch nicht unterstützt.) Diese Variable muss auch bei rohen ("raw") Tunneln leer bleiben.

Wichtig: *Diese Variable* darf *leer bleiben, wenn dem Tunnel noch kein Subnetz-Präfix zugewiesen worden ist. Allerdings kann dieser Tunnel dann nicht einem IPv6-Subnetz (`IPV6_NET_x`) zugeordnet werden, weil die IPv6-Adressen im Subnetz nicht berechnet werden können. Sinnvoll ist eine solche Konfiguration also nur übergangsweise, etwa wenn der Tunnel einige Zeit aktiv sein muss, bevor der Tunnelanbieter einem ein Subnetz-Präfix zuweist (diese Vorgehensweise ist z.B. beim Tunnelanbieter SixXS üblich).*

Beispiele:

```
IPV6_TUNNEL_1_PREFIX='2001:db8:1743::/48'      # /48-Subnetz
IPV6_TUNNEL_2_PREFIX='2001:db8:1743:5e00::/56' # /56-Subnetz
```

IPV6_TUNNEL_x_LOCALV4 Diese Variable enthält die lokale IPv4-Adresse des Tunnels oder den Wert 'dynamic', wenn die dynamisch zugewiesene IPv4-Adresse des aktiven WAN-Circuits verwendet werden soll. Letzteres ist nur sinnvoll, wenn es sich um einen Heartbeat-Tunnel handelt (siehe `IPV6_TUNNEL_x_TYPE` weiter unten).

Beispiele:

```
IPV6_TUNNEL_1_LOCALV4='172.16.0.2'
IPV6_TUNNEL_2_LOCALV4='dynamic'
```

IPV6_TUNNEL_x_REMOTEV4 Diese Variable enthält die entfernte IPv4-Adresse des Tunnels. Diese Angabe wird in der Regel vom Tunnel-Anbieter vorgegeben.

Beispiel (entspricht dem PoP deham01 von Easynet):

```
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
```

Wichtig: *Wenn* `PF_INPUT_ACCEPT_DEF` *auf "no" steht, d.h. wenn die IPv4-Firewall manuell konfiguriert wird, dann wird eine Regel benötigt, die alle IPv6-in-IPv4-Pakete (IP-Protokoll 41) vom Tunnelendpunkt akzeptiert. Für den o.g. Tunnelendpunkt sähe die entsprechende Regel wie folgt aus:*

```
PF_INPUT_x='prot:41 212.224.0.188 ACCEPT'
```

IPV6_TUNNEL_x_LOCALV6 Diese Variable legt die lokale IPv6-Adresse des Tunnels inklusive verwendeter Netzmaske in CIDR-Notation fest. Diese Angabe wird vom Tunnelanbieter vorgegeben. Bei Tunnelanbietern, welche die Tunnelendpunkte beim Tunnelaufbau jedes Mal neu vergeben, ist diese Angabe unnötig. (Momentan werden solche Anbieter aber noch nicht unterstützt.)

Beispiel: `IPV6_TUNNEL_1_LOCALV6='2001:db8:1743::2/112'`

IPV6_TUNNEL_x_REMOTEV6 Diese Variable legt die entfernte IPv6-Adresse des Tunnels fest. Diese Angabe wird vom Tunnelanbieter vorgegeben. Eine Netzmaske wird nicht benötigt, da sie der Variable `IPV6_TUNNEL_x_LOCALV6` entnommen wird. Bei Tunnelanbietern, welche die Tunnelendpunkte beim Tunnelaufbau jedes Mal neu vergeben, ist diese Angabe unnötig. (Momentan werden solche Anbieter aber noch nicht unterstützt.)

Beispiel: `IPV6_TUNNEL_1_REMOTEV6='2001:db8:1743::1'`

IPV6_TUNNEL_x_DEV (optional) Diese Variable enthält den Namen der zu erstellenden Tunnel-Netzwerkschnittstelle. Verschiedene Tunnel müssen unterschiedlich benannt werden, damit alles funktioniert. Falls die Variable nicht definiert ist, wird ein Tunnelname automatisch generiert ("v6tun" + Tunnelindex).

Beispiel: `IPV6_TUNNEL_1_DEV='6in4'`

IPV6_TUNNEL_x_MTU (optional) Diese Variable enthält die Größe der MTU (Maximum Transfer Unit) in Bytes, d.h. des größten Pakets, das noch getunnelt werden kann. Diese Angabe wird in der Regel vom Tunnelanbieter vorgegeben. Die Standard-Einstellung, falls nichts angegeben wird, lautet "1280" und sollte mit allen Tunneln funktionieren.

Standard-Konfiguration: `IPV6_TUNNEL_1_MTU='1280'`

Einige Tunnelanbieter verlangen, dass über den Tunnel permanent ein Lebenszeichen vom Router an den Anbieter gesandt wird, um zu verhindern, dass ein Host einen Tunnel in Anspruch nimmt, obwohl er ihn nicht nutzt. Dazu wird ein so genanntes Heartbeat-Protokoll

(dt. “Herzschlag”) verwendet. Zusätzlich verlangen Anbieter in der Regel eine erfolgreiche Anmeldung mit Benutzernamen und Passwort, um Missbrauch zu vermeiden. Soll ein solcher Heartbeat-Tunnel genutzt werden (wie ihn etwa SixXS anbietet), dann müssen entsprechende Angaben gemacht werden, die im Folgenden beschrieben werden.

IPV6_TUNNEL_x_USERID Diese Variable enthält den Namen des Benutzers, der beim Tunnel-Login erforderlich ist.

Beispiel: `IPV6_TUNNEL_1_USERID='ABCDE-SIXXS'`

IPV6_TUNNEL_x_PASSWORD Diese Variable enthält das Passwort für den oben angegebenen Benutzernamen. Es darf keine Leerzeichen enthalten.

Beispiel: `IPV6_TUNNEL_1_PASSWORD='passwort'`

IPV6_TUNNEL_x_TUNNELID Diese Variable enthält den Identifikator des Tunnels. Der Name eines SixXS-Tunnels beginnt immer mit einem großen ‘T’.

Beispiel: `IPV6_TUNNEL_1_TUNNELID='T1234'`

IPV6_TUNNEL_x_TIMEOUT (optional) Diese Variable enthält die Zeitspanne in Sekunden, die beim Tunnelaufbau maximal gewartet wird. Der Standard-Wert ist abhängig vom eingestellten Tunnelanbieter.

Beispiel: `IPV6_TUNNEL_1_TIMEOUT='30'`

Routen-Konfiguration

Routen sind Wege für IPv6-Pakete. Damit der Router weiß, welches eingehende Paket er wohin schicken soll, greift er auf eine Routing-Tabelle zurück, in der genau diese Informationen zu finden sind. Im Falle von IPv6 ist es wichtig zu wissen, wohin IPv6-Pakete geschickt werden, die nicht ins lokale Netz sollen. Eine Default-Route, die alle Pakete an das andere Ende eines IPv6-Tunnels schickt, wird automatisch konfiguriert, wenn `IPV6_TUNNEL_x_DEFAULT` für den entsprechenden Tunnel gesetzt ist. Andere Routen, die etwa benachbarte IPv6-Subnetze miteinander verbinden, können hier konfiguriert werden.

IPV6_ROUTE_N Diese Variable legt die Anzahl der zu spezifizierenden IPv6-Routen fest. In der Regel werden keine zusätzlichen IPv6-Routen benötigt.

Standard-Einstellung: `IPV6_ROUTE_N='0'`

IPV6_ROUTE_x Diese Variable enthält die Route in der Form ‘Zielnetz Gateway’, wobei das Zielnetz in der CIDR-Notation erwartet wird. Für die Default-Route muss als Zielnetz `::/0` verwendet werden. Es ist aber nicht nötig, Default-Routen, die über einen Tunnel gehen, hier zu konfigurieren (siehe Einleitung zu diesem Abschnitt).

Beispiel: `IPV6_ROUTE_1='2001:db8:1743:44::/64␣2001:db8:1743:44::1'`

IPv6-Firewall

Wie für IPv4 wird auch für IPv6-Netzwerke eine Firewall benötigt, damit nicht jeder von außen jeden Rechner im lokalen Netz erreichen kann. Dies ist um so wichtiger, als dass jeder Rechner im Normalfall eine weltweit eindeutige IPv6-Adresse erhält, die dem Rechner permanent zugeordnet werden kann, da sie auf der MAC-Adresse der verwendeten Netzwerkkarte

aufbaut.[13] Deshalb verbietet die Firewall erst einmal jegliche Zugriffe von außen und kann dann durch entsprechende Einträge in diesem Abschnitt Stück für Stück – je nach Bedarf – geöffnet werden.

Die Konfiguration der IPv6-Firewall entspricht im Großen und Ganzen der Konfiguration der IPv4-Firewall. Auf Besonderheiten und Unterschiede wird gesondert eingegangen.

PF6_LOG_LEVEL Für alle folgenden Ketten gilt die in `PF6_LOG_LEVEL` vorgenommene Einstellung der Protokoll-Stufe, deren Inhalt auf einen der folgenden Werte gesetzt werden kann: debug, info, notice, warning, err, crit, alert, emerg.

PF6_INPUT_POLICY Diese Variable legt die Standard-Strategie für auf dem Router eingehende Pakete fest (INPUT-Kette). Mögliche Werte sind "REJECT" (Standard, weist alle Pakete ab), "DROP" (verwirft klammheimlich alle Pakete) und "ACCEPT" (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_POLICY`.

Standard-Einstellung: `PF6_INPUT_POLICY='REJECT'`

PF6_INPUT_ACCEPT_DEF Diese Variable aktiviert die voreingestellten Regeln für die INPUT-Kette der IPv6-Firewall. Mögliche Werte sind "yes" und "no".

Die voreingestellten Regeln öffnen die Firewall für eingehende ICMPv6-Pings (ein Ping pro Sekunde als Limit) sowie für NDP-Pakete (Neighbour Discovery Procotol), das zur zustandslosen Selbstkonfiguration von IPv6-Netzen benötigt wird. Verbindungen von localhost sowie Antwortpakete zu lokal initiierten Verbindungen werden ebenfalls erlaubt. Schließlich wird die IPv4-Firewall dahingehend angepasst, dass für jeden Tunnel gekapselte IPv6-in-IPv4-Pakete vom Tunnelendpunkt akzeptiert werden.

Standard-Einstellung: `PF6_INPUT_ACCEPT_DEF='yes'`

PF6_INPUT_LOG Diese Variable aktiviert das Logging aller zurückgewiesenen eingehenden Pakete. Mögliche Werte sind "yes" und "no". Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_LOG`.

Standard-Einstellung: `PF6_INPUT_LOG='no'`

PF6_INPUT_LOG_LIMIT Diese Variable konfiguriert das Log-Limit der INPUT-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_LOG_LIMIT`.

Standard-Einstellung: `PF6_INPUT_LOG_LIMIT='3/minute:5'`

PF6_INPUT_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von eingehenden TCP-Paketen ein. Überschreitet ein solches Paket dieses Limit, wird das Paket klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_REJ_LIMIT`.

Standard-Einstellung: `PF6_INPUT_REJ_LIMIT='1/second:5'`

[13] Eine Ausnahme existiert, wenn auf den LAN-Hosts die so genannten "Privacy Extensions" aktiviert werden, weil dann ein Teil der IPv6-Adresse zufällig generiert wird. Diese Adressen sind jedoch per Definition nicht nach außen hin bekannt und somit für die Firewall-Konfiguration nur bedingt bis gar nicht relevant.

PF6_INPUT_UDP_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von eingehenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_UDP_REJ_LIMIT`.

Standard-Einstellung: `PF6_INPUT_UDP_REJ_LIMIT='1/second:5'`

PF6_INPUT_ICMP_ECHO_REQ_LIMIT Definiert, wie häufig auf eine ICMPv6-Echo-Anfrage reagiert werden soll. Die Häufigkeit wird analog zur Limit-Einschränkung als 'n/Zeiteinheit' mit Bursts beschrieben, also z.B. '3/minute:5'. Ist das Limit überschritten, wird das Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert '1/second:5' verwendet; enhält er 'none', wird keine Limitierung durchgeführt.

Standard-Einstellung: `PF6_INPUT_ICMP_ECHO_REQ_LIMIT='1/second:5'`

PF6_INPUT_ICMP_ECHO_REQ_SIZE Definiert, wie groß eine empfangene ICMPv6-Echo-Anfrage sein darf (in Bytes). In dieser Angabe sind neben den "Nutzdaten" auch die Paket-Header mit zu berücksichtigen. Der Standard-Wert liegt bei 150 Bytes.

Standard-Einstellung: `PF6_INPUT_ICMP_ECHO_REQ_SIZE='150'`

PF6_INPUT_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln für eingehende Pakete (INPUT-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste erlaubt allen lokalen Hosts Zugriff auf den Router über so genannte Link-Level-Adressen, und die zweite erlaubt die Kommunikation von Hosts aus dem ersten definierten IPv6-Subnetz mit dem Router.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die zweite Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_INPUT_N='2'`

PF6_INPUT_x Diese Variable spezifiziert eine Regel für die INPUT-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

```
PF6_INPUT_1='[fe80::0/10] ACCEPT'
PF6_INPUT_2='IPV6_NET_1 ACCEPT'
PF6_INPUT_3='tmpl:samba DROP NOLOG'
```

PF6_INPUT_x_COMMENT Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen INPUT-Regel.

Beispiel: `PF6_INPUT_3_COMMENT='no␣samba␣traffic␣allowed'`

PF6_FORWARD_POLICY Diese Variable legt die Standard-Strategie für von dem Router weiterzuleitenden Pakete fest (FORWARD-Kette). Mögliche Werte sind "REJECT" (Standard, weist alle Pakete ab), "DROP" (verwirft klammheimlich alle Pakete) und "ACCEPT" (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_POLICY`.

Standard-Einstellung: `PF6_FORWARD_POLICY='REJECT'`

PF6_FORWARD_ACCEPT_DEF Diese Variable aktiviert die voreingestellten Regeln für die FORWARD-Kette der IPv6-Firewall. Mögliche Werte sind "yes" und "no".

Die voreingestellten Regeln öffnen die Firewall für ausgehende ICMPv6-Pings (ein Ping pro Sekunde als Limit). Antwortpakete zu bereits erlaubten Verbindungen werden ebenfalls erlaubt.

Standard-Einstellung: `PF6_FORWARD_ACCEPT_DEF='yes'`

PF6_FORWARD_LOG Diese Variable aktiviert das Logging aller zurückgewiesenen weterzuleitenden Pakete. Mögliche Werte sind "yes" und "no". Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_LOG`.

Standard-Einstellung: `PF6_FORWARD_LOG='no'`

PF6_FORWARD_LOG_LIMIT Diese Variable konfiguriert das Log-Limit der FORWARD-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_LOG_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_LOG_LIMIT='3/minute:5'`

PF6_FORWARD_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von weiterzuleitenden TCP-Paketen ein. Überschreitet ein solches TCP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_REJ_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_REJ_LIMIT='1/second:5'`

PF6_FORWARD_UDP_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von weiterzuleitenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_UDP_REJ_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_UDP_REJ_LIMIT='1/second:5'`

PF6_FORWARD_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln für weiterzuleitende Pakete (FORWARD-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste verhindert die Weiterleitung aller lokalen Samba-Pakete in nicht-lokale Netze, und die zweite erlaubt letzteres für alle anderen lokalen Pakete aus dem ersten definierten IPv6-Subnetz.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die letzte Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_FORWARD_N='2'`

PF6_FORWARD_x Diese Variable spezifiziert eine Regel für die FORWARD-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

```
PF6_FORWARD_1='tmpl:samba DROP'
PF6_FORWARD_2='IPV6_NET_1 ACCEPT'
```

PF6_FORWARD_x_COMMENT Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen FORWARD-Regel.

Beispiel: `PF6_FORWARD_1_COMMENT='no␣samba␣traffic␣allowed'`

PF6_OUTPUT_POLICY Diese Variable legt die Standard-Strategie für vom Router ausgehende Pakete fest (OUTPUT-Kette). Mögliche Werte sind “REJECT” (Standard, weist alle Pakete ab), “DROP” (verwirft klammheimlich alle Pakete) und “ACCEPT” (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_POLICY`.

Standard-Einstellung: `PF6_OUTPUT_POLICY='REJECT'`

PF6_OUTPUT_ACCEPT_DEF Diese Variable aktiviert die voreingestellten Regeln für die OUTPUT-Kette der IPv6-Firewall. Mögliche Werte sind “yes” und “no”. Momentan existieren keine voreingestellten Regeln.

Standard-Einstellung: `PF6_OUTPUT_ACCEPT_DEF='yes'`

PF6_OUTPUT_LOG Diese Variable aktiviert das Logging aller zurückgewiesenen ausgehenden Pakete. Mögliche Werte sind “yes” und “no”. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_LOG`.

Standard-Einstellung: `PF6_OUTPUT_LOG='no'`

PF6_OUTPUT_LOG_LIMIT Diese Variable konfiguriert das Log-Limit der OUTPUT-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_LOG_LIMIT`.

Standard-Einstellung: `PF6_OUTPUT_LOG_LIMIT='3/minute:5'`

PF6_OUTPUT_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von ausgehenden TCP-Paketen ein. Überschreitet ein solches Paket dieses Limit, wird das Paket klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_REJ_LIMIT`.

Standard-Einstellung: `PF6_OUTPUT_REJ_LIMIT='1/second:5'`

PF6_OUTPUT_UDP_REJ_LIMIT Diese Variable stellt das Limit für das Zurückweisen von ausgehenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_UDP_REJ_LIMIT`.

Standard-Einstellung: `PF6_OUTPUT_UDP_REJ_LIMIT='1/second:5'`

PF6_OUTPUT_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln für eingehende Pakete (OUTPUT-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste erlaubt allen lokalen Hosts Zugriff auf den Router über so genannte Link-Level-Adressen, und die zweite erlaubt die Kommunikation von Hosts aus dem ersten definierten IPv6-Subnetz mit dem Router.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die zweite Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_OUTPUT_N='1'`

PF6_OUTPUT_x Diese Variable spezifiziert eine Regel für die OUTPUT-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

```
PF6_OUTPUT_1='tmpl:ftp IPV6_NET_1 ACCEPT HELPER:ftp'
```

PF6_OUTPUT_x_COMMENT Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen OUTPUT-Regel.

Beispiel: `PF6_OUTPUT_3_COMMENT='no␣samba␣traffic␣allowed'`

PF6_USR_CHAIN_N Diese Variable enthält die Anzahl der vom Benutzer definierten IPv6-Firewall-Tabellen. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_N`.

Standard-Einstellung: `PF6_USR_CHAIN_N='0'`

PF6_USR_CHAIN_x_NAME Diese Variable enthält den Namen der entsprechenden benutzerdefinierten IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_NAME`.

Beispiel: `PF6_USR_CHAIN_1_NAME='usr-myvpn'`

PF6_USR_CHAIN_x_RULE_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln in der zugehörigen benutzerdefinierten IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_RULE_N`.

Beispiel: `PF6_USR_CHAIN_1_RULE_N='0'`

PF6_USR_CHAIN_x_RULE_x Diese Variable spezifiziert eine Regel für die benutzerdefinierte IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_RULE_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

PF6_USR_CHAIN_x_RULE_x_COMMENT Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen Regel.

Beispiel: `PF6_USR_CHAIN_1_RULE_1_COMMENT='some␣user-defined␣rule'`

PF6_POSTROUTING_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln fürs Maskieren (POSTROUTING-Kette). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_POSTROUTING_N`.

Beispiel: `PF6_POSTROUTING_N='2'`

PF6_POSTROUTING_x PF6_POSTROUTING_x_COMMENT
Eine Liste der Regeln, die beschreiben, welche IPv6-Pakete vom Router maskiert werden (bzw. unmaskiert weitergeleitet werden). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_POSTROUTING_x`.

PF6_PREROUTING_N Diese Variable enthält die Anzahl der IPv6-Firewallregeln fürs Weiterleiten an ein anderes Ziel (PREROUTING-Kette). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_PREROUTING_N`.

Beispiel: `PF6_PREROUTING_N='2'`

PF6_PREROUTING_x PF6_PREROUTING_x_COMMENT
Eine Liste der Regeln, die beschreiben, welche IPv6-Pakete vom Router an ein anderes Ziel weitergeleitet werden sollen. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_PREROUTING_x`.

4.12.4. Web-GUI

Das Paket installiert einen Menüeintrag “Paketfilter (IPv6)” im Mini-HTTPD, unter dem die Einträge des für IPv6 konfigurierten Paketfilters eingesehen werden können.

4.13. ISDN - Kommunikation über aktive und passive ISDN-Karten

fli4l ist vornehmlich zum Einsatz als ISDN- und/oder DSL-Router gedacht. Mit der Einstellung `OPT_ISDN='yes'` wird das ISDN-Paket aktiviert. Voraussetzung ist eine ISDN-Karte, die von fli4l unterstützt wird.

Soll kein ISDN verwendet werden, kann mit der Einstellung `OPT_ISDN='no'` die ISDN-Installation abgeschaltet werden. Dann werden alle in diesem Kapitel folgenden ISDN-Variablen ignoriert.

Standard-Einstellung: `OPT_ISDN='no'`

4.13.1. Herstellen einer ISDN-Verbindung

Das Einwählverhalten von fli4l wird von drei verschiedenen Variablen bestimmt, `DIALMODE`, `ISDN_CIRC_X_ROUTE_X`, `ISDN_CIRC_X_TIMES`. Es wird von `DIALMODE` (Seite 75) (in <config>/base.txt) bestimmt, ob bei Eintreffen eines Paketes auf einem aktiven Circuit automatisch eine Verbindung aufgebaut werden soll oder nicht. `DIALMODE` kann folgende Werte annehmen:

auto Trifft ein Paket auf einem ISDN-Circuit (bzw. dem daraus abgeleiteten ISDN-Interface ippp*) ein, wird automatisch eine Verbindung aufgebaut. Ob und wann ein Paket auf einem ISDN-Circuit eintrifft, wird von `ISDN_CIRC_X_ROUTE_X` und `ISDN_CIRC_X_TIMES` bestimmt.

manual Im manuellen Modus muß der Verbindungsaufbau über imond/imonc angestoßen werden. Wie das geht, steht im Abschnitt über imonc/imond.

off Es werden keine ISDN-Verbindungen hergestellt.

Auf welchem der konfigurierten Circuits Pakete eintreffen und damit eine Einwahl auslösen können, wird über `ISDN_CIRC_X_ROUTE_X` definiert. Standardmäßig ist es auf '0.0.0.0/0', die sogenannte 'default route' gesetzt. Das heißt, dass alle Pakete, die das lokale Netz verlassen, über diesen Circuit gehen, wenn er aktiv ist. Ob und wann er aktiv ist, wird dabei von `ISDN_CIRC_X_TIMES` bestimmt, da fli4l über die definierten Circuits ein *least cost routing* durchführt (siehe Abschnitt Least-Cost-Routing - Funktionsweise (Seite 284) in der Dokumentation des Grundpaketes). Möchte man nicht alle Pakete über diesen Circuit leiten, sondern nur Pakete in ein bestimmtes Netz (z.B. Firmennetz), kann man hier ein oder mehrere Netze angeben. Dann richtet fli4l eine Route über das dem Circuit zugeordnete ISDN-Interface ein, die permanent aktiv ist. Wird nun ein Paket in dieses Netz geschickt, erfolgt automatisch der Verbindungsaufbau.

Wie schon erwähnt, beschreibt `ISDN_CIRC_X_TIMES` neben den Verbindungskosten eines Circuits auch, ob und wann ein Circuit mit einer 'default route' aktiv ist und damit einen Verbindungsaufbau auslösen kann. Das 'wann' spezifiert man über die Zeit, die ersten beiden Elemente einer time-info (z.B. Mo-Fr:09-18), das 'ob' durch den vierten Parameter lc-default-route (y/n). fli4l (bzw. imond) sorgt dann dafür, dass die Pakete, die das lokale Netz verlassen, immer über den zu diesem Zeitpunkt aktiven Circuit gehen und damit ein Herstellen der Verbindung zum Internet-Provider auslösen.

Zusammenfassend kann man also für die Standardanwendungsfälle folgendes sagen:

- Will man einfach nur ins Internet, stellt man `DIALMODE` auf auto, definiert 1-n Circuits, die als erste Route '0.0.0.0/0' haben und deren Zeiten (Zeiten mit lc-default-route = y) die gesamte Woche abdecken.

```
ISDN_CIRC_%_ROUTE_N='1'
ISDN_CIRC_%_ROUTE_1='0.0.0.0/0'
ISDN_CIRC_%_TIMES='Mo-Su:00-24:0.0148:Y'
```

- Will man nebenbei noch über eine spezielle Nummer ins Firmennetz, definiert man sich einen Circuit (oder mehrere Circuits) mit Route ungleich '0.0.0.0/0' und hat damit einen permanent aktiven Zugang zum Firmennetz.

```
ISDN_CIRC_%_ROUTE_N='1'
ISDN_CIRC_%_ROUTE_1='network/netmaskbits'
ISDN_CIRC_%_TIMES='Mo-Su:00-24:0.0148:Y'
```

4.13.2. ISDN-Karte

ISDN_TYPE ISDN_IO ISDN_IO0 ISDN_IO1 ISDN_MEM ISDN_IRQ ISDN_IP ISDN_PORT

Hier sind die technischen Daten für die ISDN-Karte anzugeben.

Die im Beispiel aufgeführten Werte funktionieren für eine TELES 16.3, wenn die Karte auf IO-Adresse 0xd80 eingestellt ist (über Dip-Switches). Bei einer anderen Einstellung der Karte muss der Wert geändert werden.

Häufig gemachter Fehler (Beispiel):

```
ISDN_IO='240' -- richtig wäre: ISDN_IO='0x240'
```

Bei IRQ 12 muss man einen eventuell vorhandenen PS/2-Maus-Anschluss im BIOS abschalten. Sonst besser einen anderen IRQ wählen! „Gute“ sind meist 5, 10 und 11.

`ISDN_TYPE` entspricht prinzipiell den Typen-Nummern für den HiSax-Treiber. Ausnahme: nicht-HiSax-Karten wie z.B. AVM-B1. Für diese wurde der Nummernkreis für die Typen erweitert (siehe unten). Die Liste der möglichen HiSax-Typen basiert auf `linux-2.x.y/Documentation/isdn/README.HiSax`.

Typ	Karte	Benötigte Parameter
Dummy Type-Nummer:		
0	no driver (dummy)	none
Typen-Nummern für HiSax-Treiber:		
1	Teles 16.0	irq, mem, io
2	Teles 8.0	irq, mem
3	Teles 16.3 (non PnP)	irq, io
4	Creatix/Teles PnP	irq, io0 (ISAC), io1 (HSCX)
5	AVM A1 (Fritz)	irq, io
5	AVM (Fritz!Card Classic)	irq, io
6	ELSA PCC/PCF cards	io or nothing for autodetect (the iobase is required only if you have more than one ELSA card in your PC)
7	ELSA Quickstep 1000	irq, io (from isapnp setup)
8	Teles 16.3 PCMCIA	irq, io
9	ITK ix1-micro Rev.2	irq, io (from isapnp setup?)
10	ELSA PCMCIA	irq, io (set with card manager)
11	Eicon.Diehl Diva ISA PnP	irq, io

Typ	Karte	Benötigte Parameter
11	Eicon.Diehl Diva PCI	no parameter
12	ASUS COM ISDNLink	irq, io (from isapnp setup)
13	HFC-2BS0 based cards	irq, io
14	Teles 16.3c PnP	irq, io
15	Sedlbauer Speed Card	irq, io
15	Sedlbauer PC/104	irq, io
15	Sedlbauer Speed PCI	no parameter
16	USR Sportster internal	irq, io
17	MIC card	irq, io
18	ELSA Quickstep 1000PCI	no parameter
19	Compaq ISDN S0 ISA card	irq, io0, io1, io (from isapnp setup io=IO2)
20	NETjet PCI card	no parameter
21	Teles PCI	no parameter
22	Sedlbauer Speed Star (PCMCIA)	irq, io (set with card manager)
23	reserved (AMD 7930)	n.a.
24	Dr. Neuhaus Niccy PnP	irq, io0, io1 (from isapnp setup)
24	Dr. Neuhaus Niccy PCI	no parameter
25	Teles S0Box	irq, io (of the used lpt port)
26	AVM A1 PCMCIA (Fritz!)	irq, io (set with card manager)
27	AVM PnP (Fritz!PnP)	irq, io (from isapnp setup)
27	AVM PCI (Fritz!PCI)	no parameter
28	Sedlbauer Speed Fax+	irq, io (from isapnp setup)
29	Siemens I-Surf 1.0	irq, io, memory (from isapnp setup)
30	ACER P10	irq, io (from isapnp setup)
31	HST Saphir	irq, io
32	Telekom A4T	none
33	Scitel Quadro	subcontroller (4*S0, subctrl 1...4)
34	Gazel ISDN cards (ISA)	irq,io
34	Gazel ISDN cards (PCI)	none
35	HFC 2BDS0 PCI	none
36	W6692 based PCI cards	none
37	2BDS0 S+, SP	irq,io
38	NETspider U PCI	none
39	2BDS0 SP/PCMCIA [14]	irq,io (set with card manager)
40	not used (hotplug)	n.a.
41	Formula-n enter:now PCI	none
81	ST5481 USB ISDN adapters	none
82	HFC USB based ISDN adapters	none
83	HFC-4S/8S based ISDN cards	none
84	AVM Fritz!Card PCI/PCIv2/PnP	none
Typen-Nummern für Capi-Treiber:		
100	Generisches CAPI-Gerät ohne ISDN-Funktionalität, z.B. AVM Fritz!DSL SL	no parameter
101	AVM-B1 PCI	no parameter
102	AVM-B1 ISA	irq, io
103	AVM-B1/M1/M2 PCMCIA	no parameter
104	AVM Fritz!DSL	no parameter
105	AVM Fritz!PCI	no parameter

[14] Bei älteren Versionen wurde Typ 84 mit Typ 39 angedeutet.

Typ	Karte	Benötigte Parameter
106	AVM Fritz!PNP	irq, io (from isapnp setup)
107	AVM Fritz!Classic	irq, io
108	AVM Fritz!DSLv2	no parameter
109	AVM Fritz!USBv2	no parameter
110	AVM Fritz!DSL USB	no parameter
111	AVM Fritz!USB	no parameter
112	AVM Fritz!X USB	no parameter
113	AVM FRITZ!DSL USBv2	no parameter
114	AVM FRITZ!PCMCIA	no parameter
160	AVM Fritz!Box Remote-Capi	ip,port
161	Melware Remote CAPI (rcapi)	ip,port
Typen-Nummern für andere Treiber:		
201	ICN 2B	io, mem
Typen-Nummern für mISDN-Treiber (experimental):		
301	HFC-4S/8S/E1 multiport cards	no parameter
302	HFC-PCI based cards	no parameter
303	HFCS-USB Adapters	no parameter
304	AVM FritZ!Card PCI (v1 and v2) cards	no parameter
305	cards based on Infineon (former Siemens) chips: - Dialogic Diva 2.0 - Dialogic Diva 2.0U - Dialogic Diva 2.01 - Dialogic Diva 2.02 - Sedlbauer Speedwin - HST Saphir3 - Develo (former ELSA) Microlink PCI (Quickstep 1000) - Develo (former ELSA) Quickstep 3000 - Berkom Scitel BRIX Quadro - Dr.Neuhaus (Sagem) Niccy	no parameter
306	NetJet TJ 300 and TJ320 cards	no parameter
307	Sedlbauer Speedfax+ cards	no parameter
308	Winbond 6692 based cards	no parameter

Meine Karte ist eine Teles 16.3 NON-PNP ISA, also ist Type=3.

Für eine ICN-2B-Karte müssen IO und MEM gesetzt werden, zum Beispiel `ISDN_IO`='0x320', `ISDN_MEM`='0xd0000'.

Bei neueren Teles-PCI-Karten muss type=20 (statt 21) verwendet werden. Die Dinger melden sich bei "cat /proc/pci" mit "tiger" oder so ähnlich. Sonst kann ich nichts zu diesen Werten beitragen, sorry.

Um die ISDN-Typen 104 bis 114 verwenden zu können, ist es vorher nötig, die passenden Treiberdateien von `http://www.fli4l.de/download/stabile-version/avm-treiber/` herunterzuladen und in das fli4l-Verzeichnis zu entpacken. Da diese Treiber nicht der GPL unterliegen, können sie leider nicht mit dem ISDN Paket mitgeliefert werden.

Für die ISDN-Typen 81, 82, 109 bis 113 und 303 ist es nötig USB Support zu installieren und zu aktivieren. Siehe dazu USB - Support für USB-Geräte (Seite 249).

Um die ISDN-Typen 10, 22, 26, 39, 103 oder 114 verwenden zu können ist es nötig PCMCIA PC-Card Unterstützung zu installieren und zu aktivieren. Siehe dazu PCMCIA - PC-Card Unterstützung (Seite 199).

Tips zu den Typen-Nummern bekommt man auch über die i4l-FAQ oder Mailingliste, wenn man wirklich nicht weiß, was da für eine Karte im PC steckt.

Die Kartentypen, die mit „from isapnp setup“ gekennzeichnet sind, müssen mit dem PnP-Tool isapnp initialisiert werden - wenn es sich tatsächlich um eine PnP-Karte handelt. Siehe dazu die Beschreibung im Kapitel OPT_PNP - Installation von isapnp tools (Seite 79).

Der ISDN-Typ 0 wird dann benötigt, wenn man das ISDN-Paket installieren will ohne ISDN-Karte; z.B. um imond verwenden zu können bei einem Netzwerkrouter.

ISDN_DEBUG_LEVEL Dieses gibt den Debug-Level für den HiSaX-Treiber an. Der Debug-Level setzt sich dabei durch Addition der folgenden Werte zusammen (Zitat aus der Orginal-Doku):

Number	Debug-Information
1	Link-level <–> hardware-level communication
2	Top state machine
4	D-Channel Q.931 (call control messages)
8	D-Channel Q.921
16	B-Channel X.75
32	D-Channel l2
64	B-Channel l2
128	D-Channel link state debugging
256	B-Channel link state debugging
512	TEI debug
1024	LOCK debug in callc.c
2048	More debug in callc.c (not for normal use)

Die Standardeinstellung (`ISDN_DEBUG_LEVEL`=’31’) sollte den meisten reichen.

ISDN_VERBOSE_LEVEL Hiermit kann man die “Geschwätzigkeit” des ISDN-Subsystems im fli4l-Kernel einstellen. Jeder Verbose-Level schließt die Level mit niedrigerer Nummer mit ein. Die Verbose-Level sind:

’0’	keine zusätzlichen Informationen
’1’	Es wird protokolliert, was eine ISDN-Verbindung ausgelöst hat
’2’ und ’3’	Anrufe werden protokolliert
’4’ und mehr	Die Datentransferrate wird regelmäßig protokolliert.

Die Meldungen werden über das Kernel-Logging-Interface ausgegeben, erscheinen also bei aktiviertem `OPT_SYSLOGD` (Seite 76) dort.

Wichtig: *Sollen Anrufe mit telmond protokolliert werden, diesen Wert nicht kleiner als 2 einstellen, da telmond sonst die Informationen fehlen, um Anrufe zu protokollieren.*

Standardeinstellung: ISDN_VERBOSE_LEVEL='2'

ISDN_FILTER Aktiviert den Filtermechanismus des Kerns, um ein ordnungsgemäßes Auflegen nach dem angegebenen Hangup-Timeout zu gewährleisten. Siehe http://www.fli4l.de/hilfe/howtos/basteleien/hangup-problem-loesen/ für genauere Informationen.

ISDN_FILTER_EXPR Hier steht der zu nutzende Filter, wenn ISDN_FILTER auf 'yes' gesetzt ist.

4.13.3. OPT_ISDN_COMP (EXPERIMENTAL)

Mit OPT_ISDN_COMP='yes' werden die LZS- und die BSD-Kompression aktiviert. Das Kompressionspaket hat freundlicherweise Arwin Vosselman (E-Mail: arwin(at)xs4all(dot)nl) zusammengestellt. Dieses Zusatzpaket hat Experimental-Status.

Standard-Einstellung: OPT_ISDN_COMP='no'

Hier im Einzelnen die erforderlichen Parameter für LZS-Kompression:

ISDN_LZS_DEBUG (EXPERIMENTAL) Debug-Level-Einstellung:

- '0' keine Debugging Information
- '1' normale Debugging Information
- '2' erweiterte Debugging Information
- '3' schwere Debugging Information (inkl. Dumping der Datenpakete)

Standard-Einstellung: ISDN_LZS_DEBUG='1'

Wer bei Problemen mit der Komprimierung noch mehr Debugmeldungen sehen möchte, setzt diese Variable auf '2'.

ISDN_LZS_COMP (EXPERIMENTAL) Stärke der Kompression (nicht Dekompression!). Bitte erst einmal auf dem Wert '8' stehen lassen. Werte von 0 bis 9 sind möglich.

Grössere Zahlen geben bessere Kompression, 9 erzeugt aber übermässige CPU-Last.

Standard-Einstellung: ISDN_LZS_COMP='8'

ISDN_LZS_TWEAK (EXPERIMENTAL) Auch diese Variable erst einmal auf '7' stehen lassen.

Standard-Einstellung: ISDN_LZS_TWEAK='7'

Außer diesen 3 Werten muss noch die Variable ISDN_CIRC_x_FRAMECOMP angepasst werden, s. nächstes Kapitel.

4.13.4. ISDN-Circuits

In der fli4l-Konfiguration können mehrere Verbindungen über ISDN definiert werden. Davon sind maximal 2 Verbindungen auch zur gleichen Zeit über eine ISDN-Karte möglich.

Die Definition solcher Verbindungen geschieht über sogenannte Circuits. Dabei wird pro Verbindung ein Circuit verwendet.

In der Beispiel-Datei config.txt sind zwei solcher Circuits definiert:

- Circuit 1: Dialout über Internet-By-Call-Provider Microsoft Network, Sync-PPP

- Circuit 2: Dialin/Dialout zu einem ISDN-Router (das könnte beispielsweise auch fli4l sein) über Raw-IP, z.B. als Zugang zum Firmen-Netz von irgendwo. Bei mir konkret ist das eine Linux-Box mit isdn4linux als "Gegner".

Soll der fli4l-Router lediglich als Internet-Gateway dienen, ist nur ein Circuit notwendig. Ausnahme: Man will die Least-Cost-Router-Features von fli4l nutzen. Dann sind sämtliche erlaubten Circuits für verschiedene Zeitbereiche zu definieren, siehe unten.

ISDN_CIRC_N Gibt die Anzahl der verwendeten ISDN-Circuits an. Wird fli4l lediglich als ISDN-Anrufmonitor eingesetzt, ist einzustellen:

```
ISDN_CIRC_N='0'
```

Soll der fli4l-Router lediglich als einfaches ISDN-Gateway in das Internet verwendet werden, reicht ein Circuit. Ausnahme: LC-Routing, siehe unten.

ISDN_CIRC_x_NAME Hier sollte ein Name für den Circuit vergeben werden - max. 15 Stellen lang. Dieser wird dann im imon-Client `imonc.exe` statt der angewählten Telefonnummer gezeigt. Erlaubte Zeichen sind die Buchstaben 'A' bis 'Z' (Klein- und Großschreibung), die Zahlen '0' bis '9' und der Bindestrich '-'., wie z.B.

```
ISDN_CIRC_x_NAME='msn'
```

Der Name kann außerdem im Paketfilter oder bei OpenVPN benutzt werden. Wenn z.B. der Paketfilter einen ISDN Circuit regeln soll, muß ein 'circuit_' dem Circuit Namen vorangesetzt werden. Heißt ein ISDN Circuit z.B. 'willi', so wird daraus folgendes im Paketfilter:

```
PF_INPUT_3='if:circuit_willi:any prot:udp 192.168.200.226 192.168.200.254:53 ACCEPT'
```

ISDN_CIRC_x_USEPEERDNS Hiermit wird festgelegt, ob die vom Internet-Provider bei der Einwahl übergebenen Nameserver für die Dauer der Onlineverbindung in die Konfigurationsdatei des lokalen Nameservers eingetragen werden sollen. Sinnvoll ist die Nutzung dieser Option also nur bei Circuits für Internet-Provider. Inzwischen unterstützen fast alle Provider diese Art der Übergabe.

Nachdem die Nameserver-IP-Adressen übertragen wurden, werden die in der base.txt unter *DNS_FORWARDERS* eingetragenen Nameserver aus der Konfigurationsdatei des lokalen Nameservers entfernt und die vom Provider vergebenen IP-Adressen als Forwarder eingetragen. Danach wird der lokale Nameserver veranlaßt, seine Konfiguration neu einzulesen. Dabei gehen bis dahin aufgelöste Namen nicht aus dem Nameserver-Cache verloren.

Diese Option bietet den Vorteil, immer mit den am nächsten liegenden Nameservern arbeiten zu können, sofern der Provider die korrekten IP-Adressen übermittelt - dadurch geht die Namensauflösung schneller.

Im Falle eines Ausfalls eines DNS-Servers beim Provider werden in der Regel die übergebenen DNS-Server-Adressen sehr schnell vom Provider korrigiert.

Trotz allem ist vor jeder ersten Einwahl die Angabe eines gültigen Nameservers in *DNS_FORWARDERS* der base.txt zwingend erforderlich, da sonst die erste Anfrage nicht korrekt aufgelöst werden kann. Außerdem wird beim Beenden der Verbindung die originale Konfiguration des lokalen Nameservers wieder hergestellt.

Standard-Einstellung: ISDN_CIRC_x_USEPEERDNS='yes'

ISDN_CIRC_x_TYPE ISDN_CIRC_x_TYPE gibt den Typ der x-ten IP-Verbindung an. Dabei sind folgende Werte möglich:

'raw' RAW-IP
'ppp' Sync-PPP

In den meisten Fällen wird PPP verwendet, jedoch ist Raw-IP etwas effizienter, da hier der PPP-Overhead entfällt. Eine Authentifizierung ist zwar bei Raw-IP nicht möglich, es kann jedoch über die Variable ISDN_CIRC_x_DIALIN (s.u.) eine Zugangsbeschränkung auf ganz bestimmte ISDN-Nummern (Stichwort "Clip") eingestellt werden. Wird ISDN_CIRC_x_TYPE auf 'raw' gestellt wird analog zu den PPP up/down Scripten in /etc/ppp ein raw up/down Script ausgeführt.

ISDN_CIRC_x_BUNDLING Für die ISDN-Kanalbündelung wird das verbreitete MPPP-Protokoll nach RFC 1717 verwendet. Damit gelten folgende Einschränkungen, die aber in der Praxis meist nicht relevant sind:

- Nur mit PPP-Verbindungen möglich, nicht bei Raw-Circuits
- Kanalbündelung nach neuerer RFC 1990 (MLP) ist nicht möglich

Der 2. Kanal kann entweder mit dem Client imonc manuell hinzugeschaltet werden oder über die Bandbreitenanpassung automatisch aktiviert werden, siehe auch die Beschreibung zu ISDN_CIRC_x_BANDWIDTH.

Standard-Einstellung: ISDN_CIRC_x_BUNDLING='no'

Vorsicht: bei Verwendung von Kanalbündelung zusammen mit Kompression kann es zu Problemen kommen, siehe auch die Beschreibung zu ISDN_CIRC_x_FRAMECOMP.

ISDN_CIRC_x_BANDWIDTH Ist die ISDN-Kanalbündelung über ISDN_CIRC_x_BUNDLING='yes' aktiviert, kann mit dieser Variablen eine automatische Hinzuschaltung des 2. ISDN-Kanals eingestellt werden. Dabei sind 2 numerische Parameter anzugeben:

1. Schwellenwert in Byte/Sekunde (S)
2. Zeitintervall in Sekunden (Z)

Wird der Schwellenwert S für Z Sekunden überschritten, schaltet der Steuerprozeß imond den 2. Kanal automatisch hinzu. Wird der Schwellenwert S für Z Sekunden unterschritten, wird der 2. Kanal automatisch wieder deaktiviert. Die automatische Bandbreitenanpassung kann mit ISDN_CIRC_1_BANDWIDTH=" abgeschaltet werden. Dann ist lediglich eine manuelle Kanalbündelung über den Client imonc möglich.

Beispiele:

- ISDN_CIRC_1_BANDWIDTH='6144 30'

  Überschreitet die Transferrate den Wert von 6 kibibyte/second für 30 Sekunden, wird der 2. Kanal hinzugeschaltet.

- `ISDN_CIRC_1_BANDWIDTH='0 0'`

  Der zweite ISDN-Kanal wird sofort, spätestens 10 Sekunden nach dem Verbindungsaufbau hinzugeschaltet und bleibt solange bestehen, bis die komplette Verbindung beendet wird.

- `ISDN_CIRC_1_BANDWIDTH=''`

  Der zweite ISDN-Kanal kann nur manuell hinzugeschaltet werden, Voraussetzung ist jedoch weiterhin, dass `ISDN_CIRC_1_BUNDLING='yes'` eingestellt ist.

- `ISDN_CIRC_1_BANDWIDTH='10000 30'`

  Eigentlich sollte hiermit der 2. Kanal nach 30 Sekunden hinzugeschaltet werden, wenn während dieser Zeitspanne 10000 B/s erreicht werden. Dazu wird es aber nicht kommen, da mit ISDN nicht mehr als 8 kB/s erreicht werden können.

Ist `ISDN_CIRC_x_BUNDLING='no'` eingestellt, ist der Wert in der Variablen `ISDN_CIRC_x_BANDWIDTH` belanglos.

Standard-Einstellung: `ISDN_CIRC_x_BANDWIDTH=''`

ISDN_CIRC_x_LOCAL In dieser Variablen wird die lokale IP-Adresse auf der ISDN-Seite hinterlegt.

Bei dynamischer Adresszuweisung sollte dieser Wert **leer** sein. Beim Verbindungsaufbau wird dann die IP-Adresse ausgehandelt. In den meisten Fällen vergeben Internet-Provider diese Adresse dynamisch. Soll jedoch eine fest vergebene IP-Adresse verwendet werden, ist diese hier einzutragen. Diese Variable ist optional und muss bei Bedarf in das Konfigfile eingetragen werden.

ISDN_CIRC_x_REMOTE In dieser Variablen wird die entfernte IP-Adresse und die zugehörige Netzmaske auf der ISDN-Seite hinterlegt. Dazu muss die CIDR (Classles Inter-Domain Routing) Schreibweise benutzt werden. Details zu CIDR (Seite 41) ist in der Dokumentation des Baseispaketes bei IP_NET_x zu finden.

Bei dynamischer Adresszuweisung sollte dieser Wert **leer** sein. Beim Verbindungsaufbau wird dann die IP-Adresse ausgehandelt. In den meisten Fällen vergeben Internet-Provider diese Adresse dynamisch. Soll jedoch eine fest vergebene IP-Adresse verwendet werden, ist diese hier einzutragen. Diese Variable ist optional und muss bei Bedarf in das Konfigfile eingetragen werden.

Die angegebene Netzmaske wird bei der Configuration des Interfaces nach der Einwahl verwendet. Während dieser Configuration wird auch eine Route zum Host erzeugt, in den man sich einwählt. Da man diese Route in der Regel nicht braucht, ist es günstig, hier nur eine Route direkt zum Einwahlrechner zu erzeugen. Dazu setzt man die Netzmaske auf /32, indem man hier 32 als Anzahl der gesetzten Bits in der Netzmaske spezifiziert. Für Details siehe Kapitel: Technische Details zum Dialin (Seite 393).

ISDN_CIRC_x_MTU ISDN_CIRC_x_MRU Mit diesen optionalen Variablen können die sog. **MTU** (maximum transmission unit) und die **MRU** (maximum receive unit) eingestellt werden. Optional bedeutet, die Variable muß nicht in der Konfigurationsdatei stehen, sie ist bei Bedarf durch den Benutzer einzufügen!
Normal beträgt die MTU 1500 und die MRU 1524. Diese Einstellung sollte nur in Sonderfällen geändert werden!

ISDN_CIRC_x_CLAMP_MSS Hier sollte man ein yes setzen, wenn man synchrones ppp verwendet (ISDN_CIRC_x_TYPE='ppp') und eines der folgenden Symptome auftritt:

- der Webbrowser verbindet sich mit dem Webserver, aber es wird keine Seite angezeigt und kommt auch keine Fehlermeldung; es passiert einfach nichts mehr,
- das Versenden kleiner E-Mails funktioniert, bei großen gibt es Probleme oder
- ssh funktioniert, scp hängt nach dem initialen Verbindungsaufbau.

Provider, bei denen solche Probleme auftreten, sind z.B. Compuserve und andere Mediaways basierte Zugänge.

Standard-Einstellung: ISDN_CIRC_x_CLAMP_MSS='no'

ISDN_CIRC_x_HEADERCOMP Mit ISDN_CIRC_x_HEADERCOMP='yes' kann die Van-Jacobson-Komprimierung oder Headerkomprimierung eingestellt werden. Nicht alle Provider unterstützen das. Sollte es daher bei eingeschalteter Komprimierung zu Einwahlproblemen kommen, sollte ISDN_CIRC_x_HEADERCOMP='no' eingestellt werden.

Standard-Einstellung: ISDN_CIRC_x_HEADERCOMP='yes'

ISDN_CIRC_x_FRAMECOMP (EXPERIMENTAL) Dieser Parameter wird nur berücksichtigt, wenn OPT_ISDN_COMP='yes' eingestellt wird. Er regelt die Frame-Komprimierung.

Folgende Werte sind möglich:

'no'	Keine Frame-Komprimierung
'default'	LZS according RFC1974(std) and BSDCOMP 12
'all'	Negotiate lzs and bsdcomp
'lzs'	Negotiate lzs only
'lzsstd'	LZS according RFC1974 Standard Mode ("Sequential Mode")
'lzsext'	LZS according RFC1974 Extended Mode
'bsdcomp'	Negotiate bsdcomp only
'lzsstd-mh'	LZS Multihistory according RFC1974 Standard Mode ("Sequential Mode")

Welcher Wert für den jeweiligen Provider verwendet werden kann, muss ausprobiert werden. So weit bekannt geht bei T-Online nur 'lzsext'. Bei den meisten anderen Providern sollte man mit 'default' auskommen.

Vorsicht: Bei verwendung von Kanalbündelung in zusammenhang mit 'lzsext' kann es zu Problemen kommen. Diese Probleme sind, so weit bekannt, Einwahlserverspezifisch und damit meistens Providerspezifisch. Es können aber bei einem Provider auch verschiedene Typen Einwahlserver im Einsatz sein, es kann in dem Fall zu Unterschieden zwischen Einwahlknoten kommen.

'lzsstd-mh' ist für Router-zu-Routerbetrieb (r2r) gedacht. Das Verfahren wird von Providern nicht eingesetzt aber bringt bei Verwendung zwischen zwei fli4l-Router erhebliche Verbesserung bei gleichzeitigen Übertragung von mehreren Dateien. Die Headerkompression ist dazu erforderlich und wird deshalb automatisch eingeschaltet.

ISDN_CIRC_x_REMOTENAME Diese Variable ist normalerweise lediglich bei der Konfiguration von fli4l als Einwahlrouter relevant. Hier kann ein Name des Remote-Hosts eingetragen werden, muß aber nicht.

Standard-Einstellung: ISDN_CIRC_x_REMOTENAME=''

ISDN_CIRC_x_PASS Hier sind die Provider-Daten einzutragen. Im Beispiel oben handelt es sich um die Daten des Providers Microsoft Network.

`ISDN_CIRC_x_USER` enthält die Benutzerkennung, `ISDN_CIRC_x_PASS` das Password.

WICHTIG: Für einen T-Online-Zugang ist folgendes zu beachten:

Der Username AAAAAAAAAAAATTTTTT#MMMM setzt sich aus der zwölfstelligen Anschlußkennung, der T-Online-Nummer und der Mitbenutzernummer zusammen. Hinter der T-Online-Nummer muß ein '#' angegeben werden, wenn die Länge der T-Online-Nummer kürzer als 12 Zeichen ist.

Sollte dies in Einzelfällen nicht zum Erfolg führen (offenbar abhängig von der Vermittlungsstelle), muß zusätzlich zwischen der Anschlußkennung und der T-Online-Nummer ein weiteres '#'-Zeichen eingefügt werden.

Ansonsten (T-Online-Nr ist 12stellig) sind keine '#'-Zeichen anzugeben.

Beispiel: `ISDN_CIRC_1_USER`='123456#123'

Bei Raw-IP-Circuits haben diese Variablen keine Bedeutung.

ISDN_CIRC_x_ROUTE_N Die Anzahl der Routen die dieser ISDN Circuit bedient. Wenn dieser Circuit eine Default-Route definiert muss der Eintrag auf '1' gesetzt werden.

ISDN_CIRC_x_ROUTE_X Die Route oder die Routen für diesen Circuit. Für einen Internet-Zugang sollte man hier im ersten Eintrag '0.0.0.0/0' (default route) angeben. Das Format ist immer 'network/netmaskbits'. Eine Hostroute würde z.B. so aussehen: '192.168.199.1/32'. Bei Einwahl in den Firmen- oder Uni-Router ist lediglich das oder die Netze anzugeben, die man dort erreichen will, z.B.

```
ISDN_CIRC_%_ROUTE_N='2'
ISDN_CIRC_%_ROUTE_1='192.168.8.0/24'
ISDN_CIRC_%_ROUTE_2='192.168.9.0/24'
```

Bei mehreren Netzen müssen diese jeweils in einen eigenen Eintrag, also für jede Route muss eine ISDN_CIRC_x_ROUTE_y=" Zeile angelegt werden.

Möchte man die LC-Routing-Features von fli4l nutzen, kann *mehreren* Circuits eine Default-Route zugewiesen werden. Welcher Circuit dann tatsächlich verwendet wird, wird über `ISDN_CIRC_x_TIMES` eingestellt, siehe unten.

ISDN_CIRC_x_DIALOUT `ISDN_CIRC_x_DIALOUT` gibt die zu wählende Telefonnummer an. Es ist möglich, hier mehrere Nummern anzugeben (falls eine besetzt ist, wird die nächste angewählt) - die Trennung erfolgt dabei durch Leerzeichen. Laut Berichten in der Newsgroup dürfen maximal fünf Nummern angegeben werden.

ISDN_CIRC_x_DIALIN Soll der Circuit (auch) zum Einwählen genutzt werden, wird in `ISDN_CIRC_x_DIALIN` die Rufnummer des Anrufenden eingesetzt - und zwar mit Vorwahl, aber *ohne* die erste 0. Bei Anschlüssen hinter Telefonanlagen kann dies anders sein, eventuell müssen dann eine oder sogar zwei führende Nullen angegeben werden.

Soll es mehreren Teilnehmern ermöglicht werden, sich über diesen Circuit einzuwählen, können diese Nummern durch Leerzeichen getrennt angegeben werden. Besser ist es aber,

jedem Gegner einen extra Circuit zuzuweisen. Sonst könnte es bei Einwahl von zwei Gegnern zu gleicher Zeit (ist über die 2 ISDN-Kanäle durchaus möglich) auf demselben Circuit zu Kollisionen bzgl. IP-Adressen kommen.

Falls der Anrufende keine Rufnummer überträgt, hier eine '0' setzen. Aber Vorsicht: damit wird jedem eine Anwahl gestattet, der keine Rufnummer überträgt!

Möchte man eine Einwahl unabhängig von der MSN des Anrufenden realisieren, ist als Wert '*' anzugeben.

In den beiden letzten Fällen ist ein Authentifizierungsverfahren (siehe `ISDN_CIRC_x_AUTH`) unumgänglich.

ISDN_CIRC_x_CALLBACK Einstellung Callbackverfahren, mögliche Werte:

'in'	fli4l wird angerufen und ruft zurück
'out'	fli4l ruft an, hängt jedoch wieder ein und wartet auf Rückruf
'off'	kein Callback
'cbcp'	CallBack Control Protocol
'cbcp0'	CallBack Control Protocol 0
'cbcp3'	CallBack Control Protocol 3
'cbcp6'	CallBack Control Protocol 6

Bei den CallBack Control Protokolle (auch 'Microsoft CallBack' genannt) ist cbcp6 das meist übliche Protokoll.

Standard-Wert: 'off'

ISDN_CIRC_x_CBNUMBER Hier kann man für die Protokolle cbcp, cbcp3 und cbcp6 eine Rückrufnummer einsetzen (bei cbcp3 Pflicht).

ISDN_CIRC_x_CBDELAY Diese Variable gibt eine Verzögerung in Sekunden an, die bei Callback gewartet werden soll. Je nachdem, in welcher Richtung der Callback erfolgen soll, hat diese Variable eine etwas andere Bedeutung:

- `ISDN_CIRC_x_CALLBACK`='in':

  Wird fli4l angerufen und soll zurückrufen, ist `ISDN_CIRC_x_CBDELAY` die Wartezeit, bis der Rückruf erfolgen soll. Ein guter Erfahrungswert ist `ISDN_CIRC_x_CBDELAY`='3'. Je nach "Gegner" kann aber auch ein geringerer Wert funktionieren, welches dann den Verbindungsaufbau beschleunigen kann.

- `ISDN_CIRC_x_CALLBACK`='out':

  In diesem Fall gibt `ISDN_CIRC_x_CBDELAY` die Zeit an, wie lange das "Anklingeln des Gegners" erfolgen soll, bis auf den Rückruf gewartet wird. Auch hier ist `ISDN_CIRC_x_CBDELAY`='3' ein guter Erfahrungswert. Was mir dazu aufgefallen ist: Bei Fernverbindungen muß man bis zu 3 Sekunden "klingeln" lassen, bevor der andere Router überhaupt den Anruf sieht. Bei Ortsverbindungen kann meist dieser Wert kleiner gewählt werden. Im Zweifel: Ausprobieren!

Ist die Variable `ISDN_CIRC_x_CALLBACK`='off' eingestellt, wird `ISDN_CIRC_x_CBDELAY` ignoriert. Auch beim CallBack Control Protocol hat diese Variable keine Bedeutung.

ISDN_CIRC_x_EAZ Im Beispiel ist die MSN (hier EAZ genannt) auf 81330 gesetzt. Hier sollte die eigene MSN *ohne* Vorwahl eingetragen werden.

Bei Anschlüssen hinter einer Telefonanlage mit Anlagenanschluss ist meistens nur die Durchwahl anzugeben. Ich habe aber auch schon gelesen, dass eine '0' weiterhelfen kann, wenn es Probleme mit der verwendeten Telefonanlage geben sollte.

Dieses Feld kann auch leer sein. Dies soll bei besonders hartnäckigen TK-Anlagen helfen. Um Feedback wird an dieser Stelle gebeten.

ISDN_CIRC_x_SLAVE_EAZ Ist der fli4l-Router am internen S0-Bus einer Telefonanlage angeschlossen und möchte man Kanalbündelung verwenden, ist bei manchen Telefonanlagen die Angabe einer 2. Durchwahlnummer für den Slave-Kanal einzutragen.

Im Normalfall kann diese Variable jedoch leer bleiben.

ISDN_CIRC_x_DEBUG Soll ipppd zusätzliche Debug-Informationen ausgeben, muss man `ISDN_CIRC_x_DEBUG` auf 'yes' setzen. In diesem Fall schreibt ipppd zusätzlichen Informationen über die syslog-Schnittstelle.

WICHTIG: Damit diese auch über syslogd ausgegeben werden, muss die Variable `OPT_SYSLOGD` (Siehe OPT_SYSLOGD - Programm zum Protokollieren von Systemfehlermeldungen (Seite 76)) ebenso auf 'yes' gesetzt sein.
Weil manche Meldungen über klog ausgegeben werden sollte man beim Debuggen von ISDN auch `OPT_KLOGD` (Siehe OPT_KLOGD - Kernel-Message-Logger (Seite 78)) auf 'yes' setzen.

Bei Raw-IP-Circuits hat `ISDN_CIRC_x_DEBUG` keine Bedeutung.

ISDN_CIRC_x_AUTH Wird dieser Circuit auch zum Einwählen verwendet und soll eine Authentifizierung über PAP oder CHAP vom einwählenden "Gegner" gefordert werden, ist `ISDN_CIRC_x_AUTH` auf 'pap' oder 'chap' zu setzen - und *nur* dann. Anderenfalls immer leer lassen!

Grund: Ein angewählter Internet-Provider wird es immer ablehnen, sich selbst auszuweisen! Ausnahmen bestätigen die Regel, wie ich erst kürzlich in der i4l-Mailingliste las ...

Als Benutzername und Passwort werden die Einträge von `ISDN_CIRC_x_USER` und `ISDN_CIRC_x_PASS` benutzt.

Bei Raw-IP-Circuits hat diese Variable keine Bedeutung.

ISDN_CIRC_x_HUP_TIMEOUT Mit der Variablen `ISDN_CIRC_x_HUP_TIMEOUT` wird die Zeit gesteuert, nach der der fli4l-Rechner die Verbindung zum Provider beenden soll, wenn nichts mehr über die Leitung geht. Im Beispiel wird die Verbindung nach 40 Sekunden Idle-Time abgebaut, um Geld zu sparen. Bei einem erneuten Zugriff in's Netz wird die Verbindung in Sekundenschnelle wieder aufgebaut. Sinnvoll bei Providern, die sekundengenau abrechnen!

Man sollte zumindest in der Testphase das automatische Wählen/Einhängen des fli4l-Routers beobachten (entweder auf der Console oder im imon-Client für Windows). Nicht, dass durch eine fehlerhafte Konfiguration der ISDN-Anschluss zur Standleitung wird.

Wird der Wert auf '0' gestellt, wird keine Idle-Zeit mehr berücksichtigt, d.h. fli4l hängt von sich aus die Leitung nicht mehr ein. Bitte mit Vorsicht anwenden.

ISDN_CIRC_x_CHARGEINT Charge-Interval: Hier ist der Zeittakt in Sekunden anzugeben. Dieser wird dann für die Kosten-Berechnung verwendet.

Die meisten Provider rechnen minutengenau ab. In diesem Fall ist der Wert '60' richtig. Compuserve verwendet einen 3-Minuten-Takt (Stand Juni 2000), also `ISDN_CIRC_x_CHARGEINT`='180'. Bei Providern mit sekundengenauer Abrechnung (z.B. Planet-Interkom) setzt man besser `ISDN_CIRC_x_CHARGEINT` auf '1'.

Erweiterung für `ISDN_CIRC_x_CHARGEINT` >= 60 Sekunden:

Wurde `ISDN_CIRC_x_HUP_TIMEOUT` Sekunden lang kein Traffic bemerkt, wird ca. 2 Sekunden vor Ablauf des Taktes eingehängt. Die vom Provider berechnete Zeit wird also fast komplett ausgenutzt. Ein wirklich tolles Feature von isdn4linux!

Bei sekundengenau abgerechneten Verbindungen hat das natürlich keinen Sinn - daher gilt diese Regelung erst ab Zeittakten von 60 Sekunden.

ISDN_CIRC_x_TIMES Hier werden die Zeiten angegeben, wann dieser Circuit aktiviert werden soll und wann er wieviel kostet. Dadurch wird es möglich, zu verschiedenen Zeiten verschiedene Circuits mit Default-Routen zu verwenden (Least-Cost-Routing). Dabei kontrolliert der Daemon imond die Routen-Zuweisung.

Aufbau der Variablen:

```
ISDN_CIRC_x_TIMES='times-1-info [times-2-info] ...'
```

Jedes Feld times-?-info besteht aus 4 Unterfeldern - durch Doppelpunkt (':') getrennt.

1. Feld: W1-W2

   Wochentag-Zeitraum, z.B. Mo-Fr oder Sa-Su usw. Schreibweise in Englisch oder deutsch möglich. Soll ein einzelner Wochentag eingetragen werden, ist zu W1-W1 schreiben, also z.B. Su-Su.

2. Feld: hh-hh

   Stunden-Bereich, z.B. 09-18 oder auch 18-09. 18-09 ist gleichbedeutend mit 18-24 plus 00-09. 00-24 meint den ganzen Tag.

3. Feld: Charge

   Hier werden in Euro-Werten die Kosten pro Minute angegeben, z.B. 0.032 für 3.2 Cent pro Minute. Diese werden unter Berücksichtigung der Taktzeit umgerechnet für die tatsächlich anfallenden Kosten, welche dann im imon-Client angezeigt werden.

4. Feld: LC-Default-Route

   Der Inhalt kann Y oder N sein. Dabei bedeutet:

   - Y: Der angegebene Zeitbereich wird beim LC-Routing als Default-Route verwendet. Wichtig: In diesem Fall muss auch `ISDN_CIRC_x_ROUTE`='0.0.0.0/0' sein!
   - N: Der angegebene Zeitbereich dient nur zum Berechnen von Kosten, er wird beim automatischen LC-Routing jedoch nicht weiter verwendet.

Beispiel:

```
ISDN_CIRC_1_TIMES='Mo-Fr:09-18:0.049:N Mo-Fr:18-09:0.044:Y Sa-Su:00-24:0.044:Y'
ISDN_CIRC_2_TIMES='Mo-Fr:09-18:0.019:Y Mo-Fr:18-09:0.044:N Sa-Su:00-24:0.044:N'
```

Die Bedeutung ist dabei wie folgt: Circuit 1 (Provider Planet-Interkom) soll abends an Werktagen und komplett am Wochenende verwendet werden, jedoch tagsüber an Werktagen der Circuit 2 (Provider Compuserve).

Wichtig: *Die bei `ISDN_CIRC_x_TIMES` angegebenen Zeiten muessen die ganze Woche abdecken. Ist das nicht der Fall, kann keine gültige Konfiguration erzeugt werden.*

Wenn die Zeitbereiche aller LC-Default-Route-Circuits ("Y") zusammengenommen nicht die komplette Woche beinhalten, gibt's zu diesen Lückenzeiten keine Default-Route. Damit ist dann Surfen im Internet zu diesen Zeiten ausgeschlossen!

Beispiel:

```
ISDN_CIRC_1_TIMES='Sa-Su:00-24:0.044:Y Mo-Fr:09-18:0.049:N Mo-Fr:18-09:0.044:N'
ISDN_CIRC_2_TIMES='Sa-Su:00-24:0.044:N Mo-Fr:09-18:0.019:Y Mo-Fr:18-09:0.044:N'
```

Hier wurde für die Werktage von 18-09 Uhr "N" gesetzt. Zu diesen Zeiten gibt es keine Route in's Internet: Surfen verboten!

Noch ein ganz einfaches Beispiel:

```
ISDN_CIRC_1_TIMES='Mo-Su:00-24:0.0:Y'
```

für diejenigen, die eine Flatrate nutzen.

Und noch eine letzte Bemerkung zum LC-Routing:

Deutsche Feiertage werden wie Sonntage behandelt.

4.13.5. OPT_TELMOND - telmond-Konfiguration

Mit `OPT_TELMOND` kann man einstellen, ob der telmond-Server aktiviert werden soll. Dieser horcht auf eingehende Telefonanrufe und teilt über TCP-Port 5001 die anrufende und die angerufene Telefonnummer mit. Diese Information kann vom Windows- und Unix/Linux-Client imonc abgefragt und angezeigt werden (s.a. Kapitel "Client-/Server-Schnittstelle imond").

Zwingende Voraussetzung ist hierfür die Installation einer ISDN-Karte und die Konfiguration von `OPT_ISDN` und den dazugehörenden Konfigurations-Variablen.

Im laufenden Betrieb kann die korrekte Funktion von telmond unter Linux/Unix/Windows überprüft werden mit:

```
telnet fli4l 5001
```

Dann sollte der letzte Anruf gezeigt und anschließend die telnet-Verbindung sofort wieder geschlossen werden.

Der Port 5001 ist nur vom LAN aus erreichbar. Standardmäßig wird ein Zugriff von außen über die Firewall-Konfiguration abgeblockt. Möchte man jedoch auch den Zugriff innerhalb des LANs anders regeln, kann dies über die weitere telmond-Konfigurationsvariablen eingestellt werden, siehe unten.

Standard-Einstellung: `START_TELMOND`='yes'

TELMOND_PORT TCP/IP-Port, auf dem telmond auf Verbindungen horcht. Der Standardwert '5001' sollte nur in Ausnahmefällen geändert werden.

TELMOND_LOG Bei `TELMOND_LOG='yes'` werden sämtliche einkommenden Telefonanrufe in der Datei /var/log/telmond.log gespeichert. Der Inhalt der Datei kann mit dem imond-Client imonc unter Unix/Linux und Windows abgefragt werden.

Abweichende Pfade bzw. nach Clients aufgeteilte Log-Dateien können weiter unten konfiguriert werden.

Standard-Einstellung: `TELMOND_LOG='no'`

TELMOND_LOGDIR Ist das Protokollieren eingeschaltet, kann über `TELMOND_LOGDIR` ein alternatives Verzeichnis statt /var/log angegeben werden, z.B. '/boot'. Dann wird die LOG-Datei telmond.log auf dem Bootmedium angelegt. Dazu muß dann das Bootmedium auch Read/Write "gemounted" sein. Wird hier 'auto' angegeben, befindet sich das Logfile je nach Konfiguration unter /boot/persistent/isdn oder an einem anderen durch `FLI4L_UUID` bestimmten Pfad. Wenn /boot nicht Read/Write gemountet ist, wird das File in /var/run angelegt.

TELMOND_MSN_N Sollen bestimmte Anrufe nur auf einigen PC-Clients im imonc sichtbar werden, kann ein Filter eingestellt werden, mit dem Anrufe auf spezielle MSNs nur für diese PC-Clients protokolliert werden.

Ist so etwas notwendig, z.B. in einer WG, wird die Variable `TELMOND_MSN_N` auf die Anzahl der MSN-Filter eingestellt.

Standard-Einstellung: `TELMOND_MSN_N='0'`

TELMOND_MSN_x Für jeden MSN-Filter ist eine Liste von IP-Adressen anzugeben, für welche die Anrufe auf eingetragene MSN sichtbar werden sollen.

Die Variable `TELMOND_MSN_N` bestimmt die Anzahl solcher Konfigurationen, siehe oben.

Der Aufbau der Variblen ist:

```
TELMOND_MSN_x='MSN IP-ADDR-1 IP-ADDR-2 ...'
```

Ein einfaches Beispiel:

```
TELMOND_MSN_1='123456789 192.168.6.2'
```

Soll ein Anruf auf eine bestimmte MSN auf mehreren Rechnern sichtbar werden, z.B. Fax, sind die IP-Adressen der Rechner hintereinander anzugeben, z.B.

```
TELMOND_MSN_1='123456789 192.168.6.2 192.168.6.3'
```

TELMOND_CMD_N Sobald ein Telefonanruf (Voice) auf einer bestimmten MSN hereinkommt, können optional bestimmte Kommandos auf dem fli4l-Router ausgeführt werden. Mit `TELMOND_CMD_N` gibt man die Anzahl der konfigurierten Kommandos an.

TELMOND_CMD_x Mit TELMOND_CMD_1 bis TELMOND_CMD_n können Kommandos angegeben werden, welche ausgeführt werden, wenn ein Telefonanruf eintrifft.

Die Variable TELMOND_CMD_N bestimmt die Anzahl solcher Kommandos, siehe oben.

Die Variable hat folgenden Aufbau:

```
MSN CALLER-NUMBER  COMMAND ...
```

Dabei ist die MSN ohne Vorwahl einzutragen. Als CALLER-NUMBER gibt man die komplette Telefonnummer - also mit Vorwahl - an. Schreibt man als Wert für CALLER-NUMBER ein einfaches Sternchen (*), wird von telmond die Telefonnummer des Anrufers nicht ausgewertet.

Hier ein Beispiel:

```
TELMOND_CMD_1='1234567 0987654321 sleep 5; imonc dial'
TELMOND_CMD_2='1234568 * switch-on-coffee-machine'
```

Im ersten Fall wird die Kommandofolge "sleep 5; imonc dial" durchgeführt, wenn der Anrufer mit der Telefonnummer 0987654321 die MSN 1234567 anruft. Tatsächlich werden hier 2 Kommandos ausgeführt. Zunächst wird 5 Sekunden gewartet, damit der ISDN-Kanal wieder frei wird, auf dem der Anruf hereinkam. Anschließend wird der fli4l-Client imonc mit dem Argument "dial" gestartet. imonc gibt dieses Kommando 1:1 an den Server imond weiter, welcher dann auf dem Default-Circuit eine Netzverbindung herstellt, z.B. ins Internet. Welche Kommandos das imonc-Client-Programm an den Server imond weitergeben kann, ist im Kapitel "Client-/Server-Schnittstelle imond" erklärt. Damit diese Einstellung funktioniert, muss OPT_IMONC aus dem Paket "tools" installiert sein.

Das zweite Kommando "switch-on-coffee-machine" wird ausgeführt, wenn ein Anruf auf der MSN 1234568 hereinkommt, unabhängig, woher der Anruf kam. Natürlich gibt es das Kommando "switch-on-coffee-machine" (noch) nicht für fli4l!

Beim Aufruf der Kommandos können folgende Platzhalter verwendet werden:

%d	date	Datum
%t	time	Uhrzeit
%p	phone	Telefonnummer des Anrufers
%m	msn	Eigene MSN
%%	percent	das Prozentzeichen selbst

Diese Daten können dann von den aufgerufenen Programmen weiter verwendet werden, z.B. zum Verschicken per E-Mail.

TELMOND_CAPI_CTRL_N Wenn Sie einen CAPI-fähigen ISDN-Adapter oder eine Remote-CAPI (Typ 160 oder 161) verwenden, kann es sein, dass Sie die CAPI-Controller, an denen telmond auf Anrufe horcht, genauer konfigurieren wollen. Beispielsweise bietet die Fritz!Box Zugriff auf teilweise bis zu fünf verschiedene Controller an, von denen manche sich nicht unterscheiden (siehe hierzu die Informationen unter http://www.wehavemorefun.de/fritzbox/CAPI-over-TCP#Virtuelle_Controller). Um die zu verwendenden Controller einzuschränken, können Sie hier die Anzahl der zu nutzenden Controller angeben. In

der nachfolgenden Array-Variable `TELMOND_CAPI_CTRL_%` können Sie dann angeben, welche Controller genutzt werden sollen.

Wenn Sie diese Variable nicht nutzen, horcht telmond auf *allen* verfügbaren CAPI-Controllern.

TELMOND_CAPI_CTRL_x Wenn sie `TELMOND_CAPI_CTRL_N` ungleich Null gesetzt haben, müssen Sie in den Einträgen dieses Arrays die Indizes der CAPI-Controller angeben, an denen telmond auf eingehende Anrufe horchen soll.

Beispiel für die Remote-CAPI einer Fritz!Box mit "echtem" ISDN-Anschluss:

```
TELMOND_CAPI_CTRL_N='2'
TELMOND_CAPI_CTRL_1='1' # horche auf eingehende ISDN-Anrufe
TELMOND_CAPI_CTRL_2='3' # horche auf Anrufe auf dem internen S0-Bus
```

Beispiel für die Remote-CAPI einer Fritz!Box mit analogem Anschluss und SIP-Weiterleitung:

```
TELMOND_CAPI_CTRL_N='2'
TELMOND_CAPI_CTRL_1='4' # horche auf eingehende analoge Anrufe
TELMOND_CAPI_CTRL_2='5' # horche auf eingehende SIP-Anrufe
```

4.13.6. OPT_RCAPID - Remote CAPI Dämon

Dieses OPT konfiguriert auf dem fli4l-Router das Programm rcapid, das Zugriff auf die ISDN-CAPI-Schnittstelle des Routers übers Netzwerk anbietet. Geeignete Anwendungen können somit die ISDN-Karte des Routers übers Netzwerk so nutzen, als ob sie lokal zur Verfügung stünde. Die Funktionalität ähnelt somit dem Paket "mtgcapri". Der Unterschied besteht jedoch darin, dass "mtgcapri" nur Windows-Systeme als Klienten unterstützt, während die Netzwerk-Schnittstelle des rcapid nach Kenntnis des Autors zur Zet nur Linux-Systeme nutzen können. Insofern ergänzen sich in gemischten Umgebungen mit Windows- und Linux-Systemen beide Pakete ideal.

Konfiguration des Routers

OPT_RCAPID Diese Variable aktiviert das Anbieten der auf dem Router verfügbaren ISDN-CAPI für entfernte Klienten. Mögliche Werte sind "yes" und "no". Wird diese Variable auf "yes" gesetzt, wird der Internet-Dämon inetd so konfiguriert, dass auf Anfragen am rcapid-Port 6000 der rcapid-Dämon gestartet wird (der Port kann mit Hilfe der Variablen `RCAPID_PORT` geändert werden).

Beispiel: `OPT_RCAPID='yes'`

RCAPID_PORT Diese Variable enthält den TCP-Port, den der rcapid-Dämon benutzen soll.

Standard-Einstellung: `RCAPID_PORT='6000'`

Konfiguration der Linux-Klienten

Um auf einem Linux-Rechner die entfernte CAPI-Schnittstelle nutzen zu können, muss die modulare libcapi20-Bibliothek verwendet werden. Aktuelle Linux-Distributionen installieren bereits eine solche CAPI-Bibliothek (z.B. Debian Wheezy). Falls nicht, können die Quellen von http://ftp.de.debian.org/debian/pool/main/i/isdnutils/isdnutils_3.25+dfsg1.orig.tar.bz2 heruntergeladen werden; nach dem Auspacken und dem Wechsel ins Verzeichnis "capi20" kann die CAPI-Bibliothek mit dem üblichen Dreierschritt "./configure", "make" und "sudo make install" übersetzt und installiert werden. Ist die Bibliothek erst einmal installiert, muss man nur noch in der Konfigurationdatei `/etc/capi20.conf` vermerken, auf welchem Rechner das rcapid-Programm läuft. Ist der Router beispielsweise unter dem Namen "fli4l" erreichbar, sieht die Konfigurationsdatei folgendermaßen aus:

```
REMOTE fli4l 6000
```

Das war's! Ist auf dem Linux-Klienten das Programm "capiinfo" installiert (Teil des capi4k-utils-Pakets vieler Distributionen), dann kann man sofort die entfernte CAPI-Schnittstelle testen:

```
kristov@peacock ~ $ capiinfo
Number of Controllers : 1
Controller 1:
Manufacturer: AVM Berlin
CAPI Version: 1073741824.1229996355
Manufacturer Version: 2.2-00  (808333856.1377840928)
Serial Number: 0004711
BChannels: 2
[...]
```

Unter "Number of Controllers" wird die Anzahl der ISDN-Karten vermerkt, die auf dem Klienten nutzbar sind. Steht hier "0", dann funktioniert zwar die Verbindung zum rcapid-Programm, aber auf dem Router wird/werden die ISDN-Karte(n) nicht erkannt. Funktioniert die Verbindung zum rcapid-Programm gar nicht (z.B. weil `OPT_RCAPID` auf "no" steht), dann erscheint die Fehlermeldung "capi not installed - Connection refused (111)". In diesem Fall sollte man die Konfiguration noch einmal überprüfen.

4.14. OpenVPN - VPN-Support

Ab Version 2.1.5 ist das OpenVPN Paket fester Bestandteil von fli4l.

Wichtig: *Für die Nutzung von OpenVPN über das Internet, ist in jedem Fall eine Flatrate oder eine volumenbasierte Abrechnung notwendig! Wenn der fli4l-Router permanent eingeschaltet bleibt, wird die Verbindung nicht getrennt, da permanent Daten (wenn auch nur ein paar Bytes alle paar Sekunden) übertragen werden. Mit der Nutzung eines der VPN Pakete und der Verwendung eines VPN Tunnel über das Internet, legt der fli4l-Router nicht mehr auf und es entstehen hohe Kosten, wenn keine Flatrate oder ein volumenbasiertes Abrechnungsmodel benutzt wird! Gleiches gilt natürlich für eine ISDN Wählleitung.*

Neben OpenVPN gibt es in der opt-Datenbank http://www.fli4l.de/download/zusatzpakete/ noch das VPN Paket OPT_PoPToP.

Die Entscheidung für eine VPN Lösung hängt in erster Linie von der Sicherheit und der Funktion der eingesetzten Lösung ab. Aussagen zur Sicherheit der hier angebotenen VPN Lösungen gibt das fli4l Team bewusst nicht ab, verweist dafür auf folgende Webseiten und Berichte:

Linux-Magazin Ausgabe Januar 2004

http://diswww.mit.edu/bloom-picayune/crypto/14238

http://sites.inka.de/bigred/archive/cipe-l/2003-09/msg00263.html

Zur Funktionalität kann das fli4l Team aber eine klare Aussage treffen. In diesem Punkt ist OpenVPN der klare Gewinner gegenüber CIPE und poptop. OpenVPN unterstüzt neben einem Tunnel- und Bridgemodus auch Datenkompression und läuft im Gegensatz zu CIPE wesentlich stabiler auf dem fli4l-Router. Außerdem gibt es für OpenVPN auch eine Windows Version, die ab Windows 2000 eingesetzt werden kann. Einziger Nachteil von OpenVPN ist seine Größe im opt-Archiv gegenüber CIPE und die fehlende OpenVPN Unterstützung für fli4l Version 2.0.x.

4.14.1. OpenVPN - Einführendes Beispiel

Um den Einstieg in die Konfiguration zu erleichtern, vorab ein kleines Beispiel. Es sollen zwei Netze, die beide einen fli4l-Router einsetzen, über das Internet verbunden werden. Dazu wird von OpenVPN auf den zwei fli4l-Routern ein verschlüsselter Tunnel eingerichtet, durch den die Computer aus den enfernten Netzen miteinander kommunizieren können. Dabei spielen die im Bild 4.1 gezeigten Konfigurationsvariablen eine Rolle.

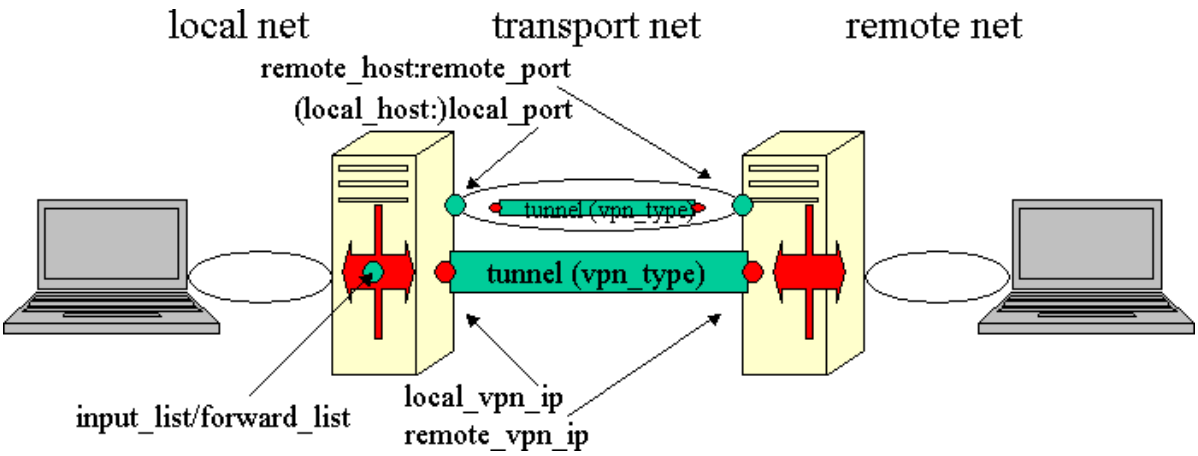


Abbildung 4.1.: VPN-Konfigurationsbeispiel — Tunnel zwischen zwei Routern

local net, remote net repräsentieren die beiden Netze, die über den Tunnel miteinander verbunden werden sollen. Die beiden zu verbindenden Netze müssen in unterschiedlichen TCP/IP Netzen sein und dürfen sich in ihren Netzmasken auch nicht überschneiden. Die Einstellungen von `IP_NET_x` (Seite 41) in der jeweiligen base.txt Konfigurationsdateien dürfen also nicht gleich sein. Es ist also mit einem VPN Tunnel nicht möglich zwei Netze miteinander zu verbinden, die beide das IP Netz 192.168.6.0/24 benutzen.

transport net das Transport Netzwerk besteht aus zwei Elementen:

- der Verbindung zwischen zwei OpenVPN-Daemons, beschrieben durch *remote_host:remote_port* und *(local_host:)local_port*. Das entspricht den

OpenVPN Einstellungen `OPENVPN_x_REMOTE_HOST`, `OPENVPN_x_REMOTE_PORT`, `OPENVPN_x_LOCAL_HOST` und `OPENVPN_x_LOCAL_PORT`.

- und dem Tunnel, über den die Verbindung zwischen den OpenVPN-Daemons etabliert wird, beschrieben durch *local_vpn_ip/remote_vpn_ip*. Dies entspricht dann wieder `OPENVPN_x_LOCAL_VPN_IP` und `OPENVPN_x_REMOTE_VPN_IP`. Die beiden VPN IP-Adressen dürfen sich dabei in keinem anderen, den beiden Routern bekannten Netzen liegen.

input_list, forward_list Pakete, die über den Tunnel gehen sollen, müssen zuerst durch den Paketfilter. Dieser erlaubt standardmäßig nur ICMP-Nachrichten (z. B. ping), die man zum Testen des Tunnels verwenden kann. Alles andere muß erst explizit erlaubt werden, im einfachsten Falle durch

```
OPENVPN_x_PF_INPUT_POLICY='ACCEPT'
OPENVPN_x_PF_FORWARD_POLICY='ACCEPT'
```

Bitte denken Sie daran, dass das komplette „Freigeben“ einer VPN Verbindung sicherheitstechnisch sehr bedenklich ist. Benutzen Sie lieber die tmpl: Syntax des Paketfilters, um nur gezielt die Dienste freizugeben, die Sie auch benötigen.

Mehr Einstellungen sind für einen einfachen VPN Tunnel nicht notwendig. Alle weiteren Einstellmöglichkeiten behandeln erweiterte Funktionen oder sind für spezielle Anwendungsfälle gedacht. Sie sollten mit diesen erweiterten Einstellungen erst dann arbeiten, weil der VPN Tunnel mit den minimalen Einstellungen erfolgreich aufgebaut werden kann.

4.14.2. OpenVPN - Konfiguration

Da OpenVPN ziemlich komplex ist, beginnen wir mit der Erklärung der zwingend notwendigen Angaben für jede VPN Verbindung. Erst wenn der fli4l-Router mit diesen Einstellungen eine Verbindung aufgebaut hat, sollten Sie sich daran wagen die erweiterten Konfigurationsmöglichkeiten von OpenVPN zu nutzen.

OPT_OPENVPN Default: `OPT_OPENVPN='no'`

Mit `'yes'` wird das OpenVPN Paket aktiviert. Die Einstellung `'no'` deaktiviert das OpenVPN Paket komplett.

OPENVPN_N Default: `OPENVPN_N='0'`

Wieviele OpenVPN Konfigurationen sind in der Konfigurationsdatei aktiv?

OPENVPN_x_REMOTE_HOST Default: `OPENVPN_x_REMOTE_HOST="`

Die IP-Adresse oder eine DNS-Adresse der OpenVPN Gegenstelle. Bei einem Roadwarrior (Seite 196) muss diese Zeile komplett fehlen. Wird diese Einstellung weggelassen, wartet OpenVPN auf einen Verbindungsaufbau und versucht nicht selbstständig die Verbindung aufzubauen.

OPENVPN_x_REMOTE_HOST_N Default: `OPENVPN_x_REMOTE_HOST_N='0'`

Bei der Benutzung von dynamischen DNS Diensten passiert es leider ab und an, dass ein Dienst nicht 100% zuverlässig funktioniert. Daher macht es in diesen Fällen Sinn, einfach zwei oder mehr DynDNS Dienste zu benutzen und seine aktuelle IP-Adresse bei allen Diensten gleichzeitig zu registrieren. Damit OpenVPN in diesem Fall auch alle DynDNS Namen durchgehen kann, muss hier noch die Liste der *zusätzlichen* DNS Namen eingegeben werden. Zusammen mit `OPENVPN_x_REMOTE_HOST` ergibt sich dann die Liste der DynDNS Adressen, die OpenVPN in zufälliger Reihenfolge zu kontaktieren versucht. Der Eintrag `OPENVPN_x_REMOTE_HOST` muss also weiterhin vorhanden sein!

OPENVPN_x_REMOTE_HOST_x Default: `OPENVPN_x_REMOTE_HOST_x="`

Es gilt die gleiche Beschreibung wie unter `OPENVPN_x_REMOTE_HOST` (Seite 175).

OPENVPN_x_REMOTE_PORT Default: `OPENVPN_x_REMOTE_PORT="`

Jede OpenVPN Verbindung braucht eine auf dem fli4l-Router bisher nicht benutzte Portadresse. Es empfiehlt sich, einen Port oberhalb von 10000 zu nehmen, da dort normalerweise keine häufig benutzen Ports liegen. Wenn Sie eine Verbindung für eine Gegenstelle bereitstellen wollen, die eine wechselnde IP-Adresse hat und über keine DynDNS Adresse verfügt, lassen Sie diesen Eintrag genau wie `OPENVPN_x_REMOTE_HOST` komplett weg.

OPENVPN_x_LOCAL_HOST Default: `OPENVPN_x_LOCAL_HOST="`

Gibt an, an welche IP-Adresse OpenVPN gebunden werden soll. Bei Verbindungen über das Internet sollte dieser Eintrag leer bleiben oder komplett weggelassen werden. Wird hier eine IP-Adresse angegeben, horcht OpenVPN nur auf dieser IP-Adresse auf eingehende Verbindungsanfragen. Wenn Sie eine WLAN Verbindung absichern wollen, sollten Sie hier die IP-Adresse der WLAN Karte vom fli4l-Router eintragen.

OPENVPN_x_LOCAL_PORT Default: `OPENVPN_x_LOCAL_PORT="`

Gibt die Portnummer an, auf der der lokale OpenVPN Daemon horcht. Für jede OpenVPN Einstellung benötigen Sie einen dafür reservierten Port, d.h. dieser Port kann nur von dieser einer OpenVPN Verbindung benutzt werden und darf auch von keiner anderen Software auf dem fli4l-Router benutzt werden. Die Einstellungen `OPENVPN_x_REMOTE_PORT` und `OPENVPN_x_LOCAL_PORT` jeder OpenVPN Verbindung müssen zusammenpassen! Wenn Sie auf einer Seite des Tunnel `OPENVPN_x_REMOTE_PORT='10111'` setzen *muss* die andere zwingend auf `OPENVPN_x_LOCAL_PORT='10111'` gesetzt werden.

Nochmal: Es ist sehr wichtig, diese Einstellungen auf die jeweilige OpenVPN Gegenstelle anzupassen, sonst ist eine Verbindung zwischen den OpenVPN Partnern nicht möglich.

Damit OpenVPN auf eingehende Verbindungen horchen kann, öffnet OpenVPN selbstständig die Ports im Paketfilter, die unter `OPENVPN_x_LOCAL_PORT` angegeben werden. Wenn dies nicht gewünscht wird, können Sie dies unter `OPENVPN_DEFAULT_OPEN_OVPNPORT` (Seite 183) anpassen. Es ist *nicht* notwendig den Eintrag `OPENVPN_DEFAULT_OPEN_OVPNPORT='yes'` zu setzen, da das die Standardeinstellung ist!

Es ist nicht möglich OpenVPN auf Ports kleiner als 1025 horchen zu lassen. Wenn Sie z.B. einen OpenVPN als tcp-server auf Port 443 (der https Port) konfigurieren wollen, müssen Sie den Port 443 per Paketfilter an einen Port über 1024 weiterleiten. Wenn Sie

z.B. den OpenVPN auf Port 5555 horchen lassen und den Port 443 weiterleiten wollen, muss folgendes in die PF_PREROUTING eingetragen werden.

```
PF_PREROUTING_5='tmpl:https dynamic REDIRECT:5555'
```

OPENVPN_x_SECRET Default: OPENVPN_x_SECRET="

OpenVPN benötigt zum Verschlüsseln der OpenVPN Verbindung ein sogenanntes Keyfile. Dieses Keyfile kann unter Windows oder Linux direkt mit OpenVPN erzeugt werden. Für Anfänger bietet es sich an, die Windows Software von OpenVPN zu installieren oder die OpenVPN WebGUI zu benutzen. Wenn Sie OpenVPN unter Windows nicht einsetzen wollen, sondern nur die für OpenVPN benötigen Keyfiles erstellen wollen, reicht es, die Punkte *OpenVPN User-Space Components*, *OpenSSL DDLs*, *OpenSSL Utilities*, *Add OpenVPN to PATH* und *Add Shortcuts to OpenVPN* zu installieren. Mit dem Menüpunkt *Generate a static OpenVPN key*, den Sie im Startmenü unter OpenVPN finden, können dann die benötigten Keydateien erzeugt werden. Nach dem Aufruf des Menüpunktes kommt die Meldung „Randomly generated 2048 bit key written to C:/Programme/OpenVPN/config/key.txt“. Die erstellte key.txt Datei ist das benötigte Keyfile. Kopieren Sie diese Datei einfach in das Verzeichnis <config>/etc/openvpn und bennenen Sie die key.txt entsprechend um, so dass der Dateiname aussagekräftig wird. Sie können ein Keyfile auch automatisch vom fli4l-Router erstellen lassen, wenn Sie OPENVPN_CREATE_SECRET auf 'yes' stellen und den fli4l-Router neu starten. Wenn Sie also das erste Mal OpenVPN konfigurieren, tragen Sie alle Daten in die Konfigdatei ein und setzen entweder OPENVPN_DEFAULT_CREATE_SECRET (Seite 182) auf 'yes', wenn Sie gleich für alle OpenVPN Verbindungen neue Keyfiles erzeugen wollen oder nur für die entsprechende OpenVPN Verbindung OPENVPN_x_CREATE_SECRET auf 'yes'. Nach dem Start des fli4l-Routers werden dann die oder das Keyfile(s) automatisch erzeugt und in /etc/openvpn mit dem hier angegebenen Namen abgelegt. Das oder die Keyfile(s) kann dann per scp kopiert oder mit einer Diskette übertragen werden. Sie müssen nach dem Erstellen der Schlüsseldateien die Einstellung wieder auf 'no' setzen, das fli4l Bootmedium neu erzeugen und die neu erzeugte Konfiguration starten. Bleibt die Einstellung auf 'yes' werden bei jedem Start des fli4l-Routers neue Schlüsseldateien erzeugt aber kein OpenVPN Daemon gestartet. Es also kann kein Tunnel aufgebaut werden. Sie können OPENVPN_x_CREATE_SECRET auch auf 'webgui' setzen, wenn Sie die WebGUI verwenden möchten um Keyfile(s) zu generieren. Dazu müssen Sie in der WebGUI in die Detailansicht der Verbindung(en) gehen und den Punkt Keymanagement auswählen. Genaueres dazu im Abschnitt 4.14.6

Tipp: Mit dem Kommando

```
openvpn --genkey --secret <dateiname>
```

können Sie ein Keyfile auf dem fli4l-Router auch von Hand erstellen.

Die Keyfiles müssen in das Verzeichnis <config>/etc/openvpn kopiert werden, wie in folgendem Bild zu sehen ist. Der Dateiname des Keyfiles ohne den Pfad muss anschließend in OPENVPN_x_SECRET hinterlegt werden. Dann werden die Keyfiles beim Erstellen des opt-Archives mit eingepackt.

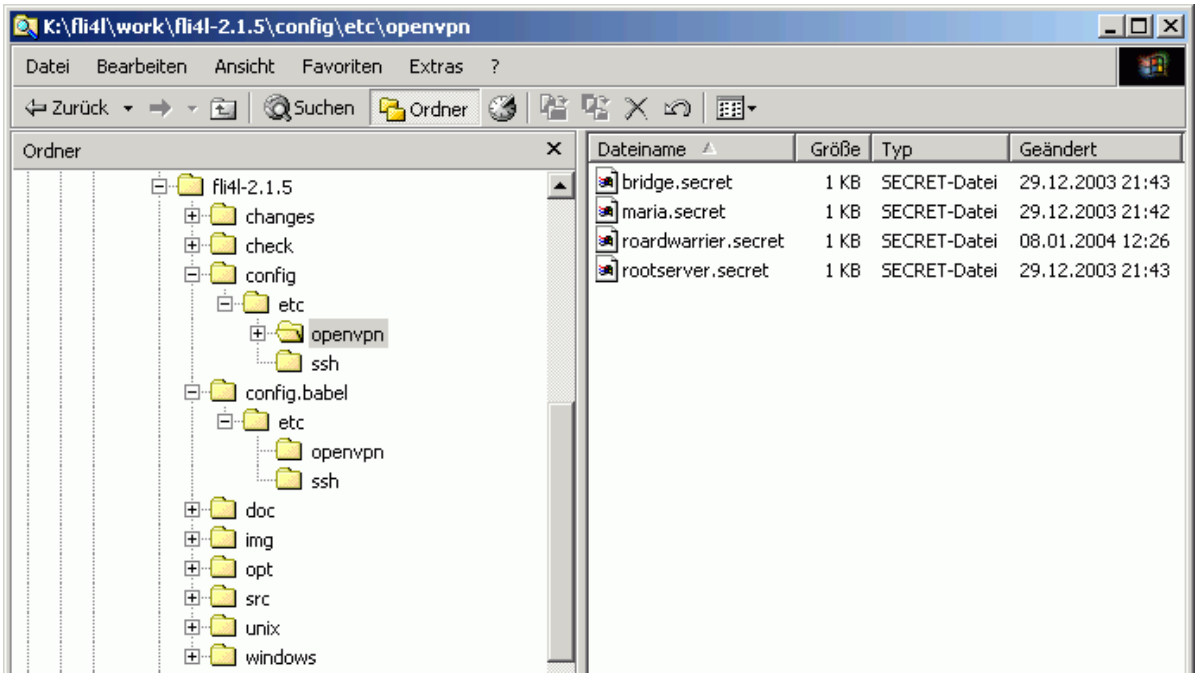


Abbildung 4.2.: fli4l config Directory mit OpenVPN *.secret Dateien

OPENVPN_x_TYPE Default: `OPENVPN_x_TYPE="`

Eine OpenVPN Verbindung kann entweder als Tunnel oder als Bridge benutzt werden. Über einen OpenVPN Tunnel kann ausschliesslich IP-Traffic geroutet werden. Über eine Bridge werden Ethernetframes übertragen, also nicht nur IP-Traffic, sondern z.B. auch IPX oder NetBEUI. Wenn OpenVPN als Transport für Ethernetframes benutzt werden soll, wird in jedem Fall noch das advanced_networking Paket benötigt. Bitte bedenken Sie, dass die Benutzung von Bridging über eine DSL Leitung sehr langsam werden kann!

4.14.3. OpenVPN - Bridgekonfiguration

Wenn Sie OpenVPN für eine Bridge benutzen wollen, sind folgende Einträge gültig. Bitte denken Sie daran, dass bei der Benutzung einer Bridge über das Internet der entstehende Broadcasttraffic unter Umständen schon eine relativ hohe Bandbreite benötigt.

Denken Sie daran, dass die folgenden Einstellungen nur gültig sind, wenn der `OPENVPN_x_TYPE` (Seite 178) für diese OpenVPN Verbindung auf `'bridge'` eingestellt wurde! Ausserdem wird eine konfigurierte Bridge aus dem advanced_networking Paket benötigt, an die sich die VPN Verbindung hängen kann.

OPENVPN_x_BRIDGE Default: `OPENVPN_x_BRIDGE="`

Hier wird der Name der Bridge angegeben, an die sich diese OpenVPN Verbindung hängen soll. Wenn also in `BRIDGE_DEV_x_NAME='cuj-br'` steht und sich diese OpenVPN Verbindung an diese Bridge hängen soll, muss hier ebenfalls `'cuj-br'` angegeben werden.

OPENVPN_x_BRIDGE_COST Default: `OPENVPN_x_BRIDGE_COST="`

Wenn Sie STP (siehe http://de.wikipedia.org/wiki/Spanning_Tree oder die Dokumentation im advanced_networking Paket) benutzen, können Sie hier die Kosten der Verbindung angeben.

OPENVPN_x_BRIDGE_PRIORITY Default: `OPENVPN_x_BRIDGE_PRIORITY="`

Wenn Sie STP (siehe http://de.wikipedia.org/wiki/Spanning_Tree oder die Dokumentation im advanced_networking Paket) benutzen, können Sie hier die Priorität der Verbindung angeben.

4.14.4. OpenVPN - Tunnelkonfiguration

OPENVPN_x_REMOTE_VPN_IP Default: `OPENVPN_x_REMOTE_VPN_IP="`

Diese Einstellung ist nur gültig, wenn der OPENVPN_x_TYPE (Seite 178) für diese OpenVPN Verbindung auf `'tunnel'` einstellt wurde!

Die VPN IP-Adresse der OpenVPN Gegenstelle. Die VPN IP-Adressen werden nur zum Routen benötigt und können fast frei gewählt werden. Es gelten dabei folgende Einschränkungen für die Auswahl der VPN IP-Adressen:

- Die IP-Adresse darf an keiner Stelle im lokalen Netz benutzt werden. Sie darf also nicht im Subnetz des fli4l-Routers liegen.
- Die IP-Adresse darf für kein lokales Netzwerkdevice benutzt werden.
- Die IP-Adresse darf nicht zu einem Netzwerk gehören, das mit `IP_ROUTE_x` geroutet wird.
- Die IP-Adresse darf nicht zu einem Netzwerk gehören, das mit `ISDN_CIRC_ROUTE_x` geroutet wird.
- Die IP-Adresse darf nicht zu einem Netzwerk gehören, das mit `CIPE_ROUTE_x` geroutet wird.
- Die IP-Adresse darf nicht zu einem Netzwerk gehören, das mit `OPENVPN_ROUTE_x` geroutet wird.
- Die IP-Adresse darf nicht zu einem Netzwerk gehören, das auf irgendeine andere Weise zum fli4l Netzwerk gehört oder vom fli4l-Router geroutet wird.

Wie Sie sehen darf die VPN IP-Adresse nirgends sonst benutzt werden. Bevor Sie mit der Konfiguration von OpenVPN beginnen, sollten Sie sich ein Netz suchen, was von keinem Netzwerk benutzt wird, in das Sie eine VPN Verbindung aufbauen wollen. Das Netzwerk sollte auch unbedingt zu einem der privaten Netze gehören (siehe http://ftp.univie.ac.at/netinfo/rfc/rfc1597.txt).

OPENVPN_x_LOCAL_VPN_IP Default: `OPENVPN_x_LOCAL_VPN_IP="`

Diese Einstellung ist nur gültig, wenn der OPENVPN_x_TYPE (Seite 178) für diese OpenVPN Verbindung auf `'tunnel'` einstellt wurde.

Die IP-Adresse, die das lokale OpenVPN tunX Device bekommt. Für die Auswahl der IP-Adresse gelten die gleichen Einschränkungen wie bei OPENVPN_x_REMOTE_VPN_IP (Seite 179).

Es ist übrigens möglich, für alle lokalen OpenVPN Verbindungen die gleiche IP-Adresse bei OPENVPN_x_LOCAL_VPN_IP zu benutzen. So ist es problemlos möglich, dass ein Host in einem VPN immer die gleiche IP-Adresse benutzt. Das vereinfacht die Paketfilterregeln unter Umständen drastisch.

OPENVPN_x_IPV6 Default: OPENVPN_x_IPV6='no'

Hiermit kann der native IPv6-Support von OpenVPN eingeschaltet werden. Da dieser Code noch recht neu ist, ist der Support als experimentell zu bezeichnen. Damit das Ganze einen Effekt hat, muss OPT_IPV6 aktiviert und konfiguriert sein. Bei OPENVPN_x_IPV6='no' und/oder OPT_IPV6='no' werden die IPv6 relevanten Variablen ignoriert.

ACHTUNG!!! Zur Zeit gibt es hier keine Überprüfung ob sich die Angaben mit anderen Teilen der Konfiguration überschneiden! Dies gilt für OPENVPN_x_LOCAL_VPN_IPV6, OPENVPN_x_REMOTE_VPN_IPV6 und OPENVPN_x_ROUTE_x.

OPENVPN_x_REMOTE_VPN_IPV6 Default: OPENVPN_x_REMOTE_VPN_IPV6="

Für die IPv6 gilt das Gleiche wie für die OPENVPN_x_REMOTE_VPN_IP (Seite 179).

```
OPENVPN_X_REMOTE_IPV6='FD00::1'
```

OPENVPN_x_LOCAL_VPN_IPV6 Default: OPENVPN_x_LOCAL_VPN_IPV6="

Für die IPv6 gilt das Gleiche wie für die OPENVPN_x_LOCAL_VPN_IP (Seite 179). Wird hier kein Subnetz gesetzt wird automatisch /64 als Subnetz genutzt.

```
OPENVPN_X_LOCAL_IPV6='FD00::2/112'
```

OPENVPN_x_ROUTE_N Default: OPENVPN_x_ROUTE_N="

Diese Einstellung ist nur gültig, wenn der OPENVPN_x_TYPE (Seite 178) für diese OpenVPN Verbindung auf 'tunnel' einstellt wurde.

Die angegebenen Routen werden automatisch von OpenVPN gesetzt, sobald OpenVPN gestartet wird. Es können bis zu 50 Netze über eine OpenVPN Verbindung geroutet werden. Sie müssen aber für jedes zu routende Netzwerk einen OPENVPN_x_ROUTE_x Eintrag erzeugen.

Bitte beachten Sie, dass Sie notwendige Paketfilterregeln in der OPENVPN_PF_FORWARD_x OPENVPN_PF_INPUT_x bzw OPENVPN_PF6_FORWARD_x OPENVPN_PF6_INPUT_x bzw selber setzen müssen. OpenVPN erlaubt nur ICMP über die VPN Verbindungen und verbietet allen anderen Datenverkehr. Details finden Sie unter OPENVPN_x_PF_INPUT_N (Seite 189) und OPENVPN_x_PF_FORWARD_N (Seite 190) bzw. unter OPENVPN_x_PF6_INPUT_N (Seite 190) und OPENVPN_x_PF6_FORWARD_N (Seite 190) bzw.

OPENVPN_x_ROUTE_x Default: OPENVPN_x_ROUTE_x="

Sie müssen hier die Netze angeben, die Sie über die OpenVPN Gegenstelle erreichen wollen. Sind hinter der OpenVPN Gegenstelle z.B. die Netzwerke 192.168.33.0/24 und

172.18.0.0/16 erreichbar und sollen diese über den OpenVPN Tunnel erreicht werden, müssen Sie diese beiden Netze jeweils einzeln unter OPENVPN_x_ROUTE_x eintragen. Es können hier auch Hostrouten (/32) eingetragen werden.

Wenn die Defaultroute über einen OpenVPN Tunnel gesetzt werden soll, geben sie bitte 0.0.0.0/0 bzw. ::/0 für IPv6 und ein optionales Flag als Route an. Auch hier gilt, das für IPv6 Routen OPT_IPv6 aktiv sein muss, die lokale und remote IPv6-Adresse für den Tunnel müssen gesetzt sein und OPENVPN_x_IPV6 auf yes stehen. OpenVPN kennt verschiedene Möglichkeiten die Defaultroute einzurichten, die sie mit dem Flag auswählen können. Jede Methode, die Defaultroute einzurichten, hat ihre Vor- und Nachteile. Im Moment unterstützt OpenVPN folgende Flags:

local Das *local* Flag sollten sie wählen, wenn die OpenVPN Gegenstelle innerhalb eines direkt von ihrem fli4l-Router erreichbaren Subnetz liegt. Das ist z.B. oft bei einer Defaultroute mit OpenVPN über WLAN der Fall.

def1 Mit diesem Flag werden zusätzliche zu einer Hostroute an die OpenVPN Gegenstelle zwei neue Routen, 0.0.0.0/1 und 128.0.0.0/1, eingetragen. Diese Routen übernehmen die Funktion einer Defaultroute und über diese Routen wird dann der komplette (verschlüsselte) Traffic an die OpenVPN Gegenstelle (die noch über die Hostroute erreichbar ist) geroutet.

Lassen Sie das optionale Flag weg wählt OpenVPN die Methode aus, wie die Defaultroute umgesetzt wird. Die Auswahl der Methode erfolgt dabei über die OpenVPN Version, im Moment wird als Standardeinstellung *local* benutzt.

```
OPENVPN_1_ROUTE_N='3'
OPENVPN_1_ROUTE_1='192.168.33.0/24'
OPENVPN_1_ROUTE_2='172.18.0.0/16'
OPENVPN_1_ROUTE_3='2001:db8:/32'
```

OpenVPN - Delegation von DNS und Reverse-DNS

OPENVPN_x_DOMAIN Default: OPENVPN_x_DOMAIN="

Über diesen Parameter gibt man die Remote Domain an. Diese Variable kann mehrere Domains enthalten die durch Leerzeichen getrennt angegeben werden müssen. Wenn nur dieser Parameter gesetzt ist (ohne Angabe eines zusätzlichen DNS-Servers) wird davon ausgegangen, dass der DNS Server auf der IP des gegenüberliegenden Tunnelendes lauscht (siehe OPENVPN_x_REMOTE_VPN_IP (Seite 179)). Dazu müssen auf dem Remote Router natürlich eingehende DNS-Anfragen erlaubt werden. (z.B. via OPENVPN_x_INPUT_y='tmpl:dns ACCEPT')

OPENVPN_x_ROUTE_x_DOMAIN Default: OPENVPN_x_ROUTE_x_DOMAIN="

Den verschiedenen Subnetzen können auch verschiedene Domains zugeordnet sein. Hier kann man pro OPENVPN_x_ROUTE_y eine entsprechende Domain konfigurieren. Sollte ein dazugehöriges OPENVPN_x_ROUTE_y_DNSIP existieren, so wird dieser Server benutzt, andernfalls der unter OPENVPN_x_DNSIP angegebene. Die Wirkung ist dann die selbe wie mit OPENVPN_x_DOMAIN, allerdings eignet sich diese Methode auch zur Dokumentation.

OPENVPN_x_DNSIP Default: `OPENVPN_x_DNSIP="`

Ist der Tunnelendpunkt nicht der zuständige DNS-Server, so kann hier die IP des zuständigen DNS-Servers angegeben werden. Ist nichts angegeben, wird die unter `OPENVPN_x_REMOTE_VPN_IP` (Seite 179) angegebene IP benutzt.

OPENVPN_x_ROUTE_x_DNSIP Default: `OPENVPN_x_ROUTE_x_DNSIP="`

Verschiedene geroutete Subnetze können auch durch verschiedene DNS-Server abgedeckt sein - hier kann man pro `OPENVPN_x_ROUTE_x` (Seite 180) einen eigenen zuständigen Server definieren.

4.14.5. Experteneinstellungen

Die in diesem Kapitel beschriebenen Einstellungen sind alle optional und sollten nur verändert werden, wenn die OpenVPN Verbindung mit den Standardeinstellungen funktioniert und Optimierungen (beispielsweie ein anderer Verschlüsselungsalgorithmus) vorgenommen werden sollen.

Alle nachfolgend beschriebenen `OPENVPN_DEFAULT_` Einstellungen sind optional, d.h. diese Optionen brauchen nicht in die openvpn.txt Konfigurationsdatei geschrieben werden. Fehlt der entsprechende Eintrag in der openvpn.txt Datei, wird vom OpenVPN Startskript der hier angegebene Defaultwert benutzt. Wenn Sie nicht vorhaben die Standardwerte der Vorgaben zu änden, schreiben Sie diese auch nicht in die openvpn.txt Konfigurationsdatei!

allgemeine Einstellungen

OPENVPN_DEFAULT_CIPHER Default: `OPENVPN_DEFAULT_CIPHER='BF-CBC'`

Eine der verfügbaren Verschlüsselungsmethoden. Die Verschlüsselungsmethode `'BF-CBC'` wird von allen OpenVPN Versionen (auch nicht fli4l spezifischen Versionen) als Standardeinstellung benutzt.

OPENVPN_DEFAULT_COMPRESS Default: `OPENVPN_DEFAULT_COMPRESS='yes'`

OpenVPN benutzt eine adaptive LZO Datenkomprimierung, um die Bandbreite einer Verbindung zu erhöhen. Adaptiv bedeutet, dass OpenVPN selbstständig erkennt, wenn z.B. bereits gepackte ZIP Dateien über eine OpenVPN Verbindung geschickt werden. In einem solchen Fall wird die Datenkomprimierung solange abgeschaltet, bis wieder Daten übertragen werden, die auch von einer Datenkomprimierung profitieren. Es gibt fast nie einen Grund die Datenkomprimierung zu deaktiveren, da dadurch die Bandbreite quasi kostenlos erhöht wird. Einziger Nachteil der Datenkomprimierung ist eine geringe Erhöhung der Latenzzeit von wenigen Millisekunden. Für Online-Games via VPN, bei denen die Reaktionszeit (”guter” ping) entscheidend ist, wäre es also unter Umständen sinnvoll die Datenkomprimierung abzuschalten.

OPENVPN_DEFAULT_CREATE_SECRET Default: `OPENVPN_DEFAULT_CREATE_SECRET='no'`

Mit dieser Einstellung erstellt OpenVPN automatisch Keyfiles beim Start des fli4l-Routers. Die entsprechende OpenVPN Verbindung wird allerdings nicht gestartet. Für Details lesen Sie bitte den Punkt `OPENVPN_x_SECRET` (Seite 177) nach.

OPENVPN_DEFAULT_DIGEST Default: `OPENVPN_DEFAULT_DIGEST='SHA1'`

Hier wird eine der verfügbaren Prüfsummen eingetragen. OpenVPN nutzt als Standardeinstellung die Prüfsummenmethode `'SHA1'`.

OPENVPN_DEFAULT_FLOAT Default: `OPENVPN_DEFAULT_FLOAT='yes'`

Bei OpenVPN Verbindungen mit Gegenstellen, die eine DynDNS Adresse benutzen, ist es jederzeit möglich, dass sich die IP-Adresse der OpenVPN Gegenstelle ändert. Damit OpenVPN auch die geänderte IP-Adresse akzeptiert, muss `OPENVPN_DEFAULT_FLOAT` auf `'yes'` eingestellt werden. Mit der Einstellung `'no'` wird die Änderung der IP-Adresse nicht erlaubt. Das ist in der Regel nur bei WLAN Verbindungen oder OpenVPN Verbindungen mit Gegenstellen, die eine statische IP-Adresse (wie z.B. die rootserver diverser Anbieter) haben, sinnvoll. Sie können diese Einstellung auch wie alle anderen `OPENVPN_DEFAULT_` Einstellungen für eine bestimmte OpenVPN Verbindung überschreiben.

OPENVPN_DEFAULT_KEYSIZE Default: `OPENVPN_DEFAULT_KEYSIZE="`

Die Schlüssellänge (=KEYSIZE) hängt von der verwendeten Verschlüsselungsmethode ab. Ändern Sie diese Einstellung nur, wenn Sie mit einer OpenVPN Gegenstelle zusammenarbeiten müssen, die nicht den Standardwert benutzt oder deren Einstellung Sie nicht beeinflussen können. Können Sie die Schlüssellänge selbst bestimmen, sollte dieser Wert immer leer bleiben. OpenVPN wendet dann eine optimale Schlüssellänge für die jeweilige Verschlüsselungmethode an.

OPENVPN_DEFAULT_OPEN_OVPNPORT Default: `OPENVPN_DEFAULT_OPEN_OVPNPORT='yes'`

Damit eine OpenVPN Gegenstelle mit Ihnen Kontakt aufnehmen kann, müssen Sie den Paketfilter Ihres fli4l-Routers entsprechend anpassen. In der Regel müssen Sie also auf allen TCP oder UDP Ports, je nach `OPENVPN_x_PROTOCOL` Einstellung, auf denen ein OpenVPN horcht, die `PF_INPUT_x` (Seite 44) in der base.txt anpassen. Mit der Einstellung `'yes'` werden diese Paketfilterregeln automatisch generiert. Bei einzelnen Verbindungen kann es aber durchaus Sinn machen, diese Einstellung auf `'no'` zu setzen und selber entsprechende Paketfilterregeln zu definieren.

OPENVPN_DEFAULT_ALLOW_ICMPPING Default: `OPENVPN_DEFAULT_ALLOW_ICMPPING='yes'`

Mit `'yes'` wird der Paketfilter für die entsprechende Verbindung so konfiguriert, dass ping Datenpakete den Filter passieren dürfen. Wenn es keinen sehr wichtigen Grund gibt, sollte der ICMP Ping immer zugelassen werden. Diese Einstellung hat *nichts* mit der Pingoption von OpenVPN zu tun!

OPENVPN_DEFAULT_PF_INPUT_LOG Default: `OPENVPN_DEFAULT_PF_INPUT_LOG='BASE'`

Mit `'yes'` oder `'no'` wird eingestellt, ob der Paketfilter eingehende Datenpakete in der INPUT Liste, die für die entsprechende VPN Verbindung gedacht waren mitschreiben soll, wenn das Datenpaket abgelehnt wird. Mit der Einstellung `'BASE'` wird die Einstellung von `'PF_INPUT_LOG='` aus der base.txt übernommen.

OPENVPN_DEFAULT_PF_INPUT_POLICY Default: `OPENVPN_DEFAULT_PF_INPUT_POLICY='REJECT'`

Diese Einstellung entspricht `'PF_INPUT_POLICY='` (Seite 55) aus der base.txt. Zusätzlich zu den dort möglichen Einstellungen, kann mit `'BASE'` die Einstellung von `'PF_INPUT_POLICY='` aus der base.txt übernommen werden.

OPENVPN_DEFAULT_PF_FORWARD_LOG Default: `OPENVPN_DEFAULT_PF_FORWARD_LOG='BASE'`

Mit `'yes'` oder `'no'` wird eingestellt, ob der Paketfilter eingehende Datenpakete in der FORWARD Liste, die für die entsprechende VPN Verbindung gedacht waren mitschreiben soll, wenn das Datenpaket abgelehnt wird. Mit der Einstellung `'BASE'` wird die Einstellung von `'PF_FORWARD_LOG='` aus der base.txt übernommen.

OPENVPN_DEFAULT_PF_FORWARD_POLICY Default: `OPENVPN_DEFAULT_PF_FORWARD_POLICY='REJECT'`

Diese Einstellung entspricht `'PF_FORWARD_POLICY='` (Seite 56) aus der base.txt. Zusätzlich zu den dort möglichen Einstellungen kann mit `'BASE'` die Einstellung von `'PF_FORWARD_POLICY='` aus der base.txt übernommen werden.

OPENVPN_DEFAULT_PING Default: `OPENVPN_DEFAULT_PING='60'`

Um einen einmal aufgebauten Tunnel offen zu halten und zu erkennen, ob die OpenVPN Gegenstelle noch erreichbar ist, wird in dem angegebenen Intervall in Sekunden ein kryptografisch gesicherter ping über die Leitung geschickt. Mit der Einstellung `'off'` schickt OpenVPN kein ping über die Leitung. Mit der Einstellung `'off'` werden nur dann Daten über den VPN Tunnel geschickt wenn auch tatsächlich Nutzdaten über die VPN Tunnel übertragen werden.

OPENVPN_DEFAULT_RENEG_SEC Default: `OPENVPN_DEFAULT_RENEG_SEC='3600'`

Hiermit kann die Variable RENEG-SEC aus OpenVPN angepasst werden, so dass gerade bei langsamen DSL-Verbindungen (Oder ISDN) es nicht zu einem Timeout und Abbruch der Verbindung kommt.

OPENVPN_DEFAULT_PING_RESTART Default: `OPENVPN_DEFAULT_PING_RESTART='180'`

Wird in dem angegebenen Intervall in Sekunden kein OpenVPN ping erfolgreich über die VPN Verbindung geschickt oder werden keine anderen Daten übertragen, wird die entsprechende OpenVPN Verbindung neu gestartet. Der Wert von `OPENVPN_DEFAULT_PING_RESTART` muss immer größer sein als der Wert von `OPENVPN_DEFAULT_PING`. Die Einstellung `'off'` unterbindet den automatischen Neustart einer OpenVPN Verbindung.

OPENVPN_DEFAULT_RESOLV_RETRY Default: `OPENVPN_DEFAULT_RESOLV_RETRY='infinite'`

Sind in `OPENVPN_x_REMOTE_HOST` oder `OPENVPN_x_LOCAL_HOST` DNS Namen statt IP-Adressen hinterlegt, dann müssen die Namen beim Start einer OpenVPN Verbindung erst zu IP-Adressen aufgelöst werden. Sollte dies fehlschlagen, versucht OpenVPN für die angegebene Zeit in Sekunden die DNS Adresse erneut aufzulösen. Gelingt dies schlußendlich nicht in der vorgegebenen Zeit, kommt keine OpenVPN Verbindung zustande. Mit der Angabe `'infinite'` (= unendlich) wird unendlich lange versucht, die DNS Auflösung vorzunehmen. Diese Einstellung sollte bis auf besondere Fälle nicht verändert werden!

OPENVPN_DEFAULT_RESTART Default: `OPENVPN_DEFAULT_RESTART='ip-up'`

Nach einer Trennung der Verbindung ist es sinnvoll, dass der entsprechende OpenVPN Tunnel sofort neu gestartet wird, damit die Unterbrechung des Tunnels möglichst kurz ist. Für alle OpenVPN Verbindungen, die über eine Wählverbindung wie DSL oder ISDN laufen, sollte hier `'ip-up'` eingetragen werden. OpenVPN Verbindungen über eine WLAN Strecke dagegen sollten hier `'never'` eintragen. In diesem Fall wird die Verbindung nicht

nach einer Neueinwahl wieder gestartet, da ja die WLAN Verbindung unabhängig von einer Einwahl ist. Wenn ein OpenVPN Tunnel über eine ISDN Wählverbindung mit `ISDN_CIRC_x_TYPE='raw'` aufgebaut werden soll, muss hier `'raw-up'` eingetragen werden.

OPENVPN_DEFAULT_PROTOCOL Default: `OPENVPN_DEFAULT_PROTOCOL='udp'`

Mit dieser Variablen wird festgelegt, welche Protokollvariante als Default benutzt werden soll. Normalerweise ist UDP eine sehr gute Wahl, allerdings gibt es manchmal nur die Möglichkeit mit TCP zu arbeiten. Der dadurch entstehende Overhead ist allerdings beachtlich. Mögliche Einstellungen sind `'udp'`, `'udp6'` `'tcp-server'`, `'tcp-server6'`, `'tcp-client'` oder `'tcp-client6'`. Die `'tcp-server'` oder `'tcp-client'` Einstellungen sind in der Regel nur dann sinnvoll, wenn ein VPN Tunnel durch div. andere Paketfilter oder andere Tunnel aufgebaut werden soll. Wenn Sie keine Spezialfälle behandeln müssen, sollten Sie *immer* den Standardwert `'udp'` benutzen. Mit angehängtem '6' ist der Tunnel IPv6 fähig (WAN) und kann über ein IPv6-Internet erreicht werden.

OPENVPN_DEFAULT_START Default: `OPENVPN_DEFAULT_START='always'`

Eine OpenVPN Verbindung kann entweder immer (`='always'`) oder später von Hand (`='on-demand'`) gestartet werden. Sie können also bestimmte OpenVPN Verbindungen erst bei Bedarf über die OpenVPN WebGUI (siehe 4.14.6) starten. Alternativ ist der Start aber auch jederzeit über die fli4l Konsole möglich. Melden Sie sich dazu auf der fli4l Konsole an und führen Sie folgende Befehle direkt auf der Konsole aus:

```
cd /etc/openvpn
openvpn --config name.conf --daemon openvpn-name
```

Damit wird ein OpenVPN Tunnel gestartet, der ab sofort im Hintergrund läuft. Anstelle `name.conf` nehmen Sie natürlich den Namen Ihrer Konfigurationsdatei in dem `/etc/openvpn` Verzeichnis.

OPENVPN_DEFAULT_VERBOSE Default: `OPENVPN_DEFAULT_VERBOSE='2'`

Diese Variable gibt an, wie gesprächig OpenVPN sein soll. Wenn eine VPN Verbindung einwandfrei läuft, ist es möglich diesen Wert auf `'0'` zu setzen, um alle Meldungen zu unterbinden. Für die ersten Tests ist ein Wert von `'3'` sinnvoll. Noch größere Werte erhöhen die Debugmeldungen und helfen unter Umständen Fehler zu finden. Der Maximalwert beträgt `'11'`.

OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE Default:
`OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE='100'`

Diese Variable gibt an, wie viele Log Zeilen gespeichert werden sollen. Dieses Log kann dann über die WebGUI (Seite 191) abgefragt werden.

OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS Default:
`OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS='no'`

Mit dieser Variablen wird eingestellt, ob beim Empfang doppelter Pakete eine Warnung im Log ausgegeben werden soll, da dies Hinweise auf ein Sicherheitsproblem im Netzwerk

sein können. Insbesondere bei schwachen WLAN-Verbindungen, kann es aber häufig passieren, dass Pakete doppelt gesendet werden. Dann ist es sinnvoll die Warnungen auszustellen, damit diese nicht das Log füllen. Die Einstellung dieser Variablen hat *keinen* Einfluss auf die Sicherheit der OpenVPN Verbindung.

OPENVPN_DEFAULT_MSSFIX Default: `OPENVPN_DEFAULT_MSSFIX="`

Mit der MSSFIX Einstellung wird die Größe der TCP Datenpakete über die VPN Verbindung vorgegeben. Mit `OPENVPN_DEFAULT_MSSFIX='0'` wird diese Option ausgeschaltet. Wird eine Fragmentgröße angegeben und der MSSFIX Eintrag leer gelassen, wird automatisch die Fragmentgröße benutzt. Diese Einstellung funktioniert nur, wenn `OPENVPN_x_PROTOCOL='udp'` gesetzt wird.

OPENVPN_DEFAULT_FRAGMENT Default: `OPENVPN_DEFAULT_FRAGMENT='1300'`

Aktiviert die interne Fragmentierung von OpenVPN mit einer Paketgröße von x Bytes. Diese Einstellung funktioniert nur, wenn `OPENVPN_x_PROTOCOL='udp'` gesetzt wird.

Mit `OPENVPN_DEFAULT_FRAGMENT='0'` wird die Fragmentierung komplett deaktiviert.

OPENVPN_DEFAULT_TUN_MTU Default: `OPENVPN_DEFAULT_TUN_MTU='1500'`

Stellt die MTU des virtuellen OpenVPN Adapters auf x Bytes ein. Diese Option sollte nur geändert werden, wenn man weiss was man macht. Es ist meistens sinnvoller, erst mit den Fragment oder MSSFIX Optionen zu arbeiten.

OPENVPN_DEFAULT_TUN_MTU_EXTRA Default: `OPENVPN_DEFAULT_TUN_MTU_EXTRA="`

Wenn bei `OPENVPN_x_PROTOCOL='bridge'` eingestellt wird, werden 32 Bytes als extra Speicher für die Verwaltung der Puffers für das tap Gerät reserviert. Bei `OPENVPN_x_PROTOCOL='tunnel'` wird kein extra Speicher reserviert. Diese Einstellung wirkt sich nur auf den Speicherbedarf im Router aus und hat keinen Einfluss auf die Datenmenge die über den Tunnel verschickt wird.

OPENVPN_DEFAULT_LINK_MTU Default: `OPENVPN_DEFAULT_LINK_MTU="`

Stellt die MTU der OpenVPN Verbindung auf x Bytes ein. Diese Option sollte nur benutzt werden, wenn man weiss was man macht. Es ist meistens sinnvoller erst mit den Fragment oder MSSFIX Optionen zu arbeiten.

OPENVPN_DEFAULT_SHAPER Default: `OPENVPN_DEFAULT_SHAPER="`

Begrenzt die *ausgehende* Bandbreite des Tunnel auf die angegebene Anzahl von Bytes pro Sekunde. Möglich sind Werte von 100 Bytes bis zu 100000000 Bytes. Bei Werten bis 1000 Bytes pro Sekunde sollten Sie die MTU der Verbindung reduzieren, sonst steigen die ping Zeiten stark an. Wenn Sie einen Tunnel auf eine Bandbreite in beide Richtungen begrenzen wollen, müssen Sie dies auf beiden Seiten getrennt einstellen.

In der aktuellen OpenVPN Version funktioniert das Shaping nicht korrekt, d.h. die Übertragungsrate durch einen mittels Shapping konfigurierten Tunnel ist unter Umständen extrem schwankend bzw. der Durchsatz bricht total ein. Das Problem kann je nach eingesetzer Hardware auftreten zu komplett unterschiedlichen Verhalten führen. Im Moment sollte die Shappingfunktion mit Vorsicht behandelt werden und im Zweifel bei jeder Änderung ausgiebig getestet werden.

OPENVPN_EXPERT Default: `OPENVPN_EXPERT='no'`

Der Expert-Mode erlaubt Ihnen, native Openvpn Konfigurationsdateien zu nutzen. Diese müssen im Konfigurationsordner unter etc/openvpn abgelegt werden. Alle dort liegenden Dateien werden auf den Router übertragen.

Der Expert-Mode ignoriert die restlichen Konfigurationseinstellungen. Deshalb muss `OPENVPN_N='0'` eingestellt werden.

Der Expert-Mode richtet keinerlei Firewall-Regeln ein. Diese müssen Sie manuell in die base.txt eintragen.

verbindungsspezifische Einstellungen

Die folgenden OpenVPN Optionen gelten nur für die jeweilige OpenVPN Verbindung. Auch hier gibt es nur wenige Angaben, die zwingend sind. Die meisten Optionen können einfach weggelassen werden. Für alle Defaultwerte gilt, dass diese von der jeweils gleichlautenden `OPENVPN_DEFAULT_x` Einstellung übernommen werden. Wenn Sie also den entsprechenden `OPENVPN_DEFAULT_` Wert ändern, gilt dieser Defaultwert für alle OpenVPN Verbindungen, die den allgemeinen Defaultwert nicht überschreiben.

OPENVPN_x_NAME Default: `OPENVPN_x_NAME="`

Legt einen bis zu 16 Zeichen langen Namen für die jeweilige OpenVPN Verbindung fest. Unter diesem Namen wird die Konfigurationsdatei im Verzeichnis /etc/openvpn (mit dem Anhang .conf) angelegt. Außerdem erscheint der Name im syslog. Wenn Sie beispielsweise den Namen `'peter'` eintragen, taucht später im syslog der Eintrag 'openvpn-peter' auf. Damit können Sie die verschiedenen OpenVPN Verbindungen gut auseinanderhalten. Der Name darf Buchstaben, Zahlen und das '-' Zeichen enthalten.

OPENVPN_x_ACTIV Default: `OPENVPN_x_ACTIV='yes'`

Wenn man eine OpenVPN Verbindung deaktivieren, aber die Konfiguration nicht löschen will, kann diese OpenVPN Verbindung mit der Einstellung 'no' deaktiviert werden. Die Konfigurationsdaten werden dann zwar in der rc.cfg aufgenommen, aber es wird keine entsprechende OpenVPN Verbindung erzeugt.

OPENVPN_x_CHECK_CONFIG Default: `OPENVPN_x_CHECK_CONFIG='yes'`

Die erweiterten Prüfungen von OpenVPN sind in seltenen Fällen zu streng. Wenn beispielsweise eine ISDN Backupverbindung die gleichen Routingeinträge benutzt wie eine Verbindung, die über das Internet läuft, wird die erweiterte Prüfung diese Verbindungen mit einer Fehlermeldung bedenken. In diesem Fall sollte bei der Backupverbindung die erweitere Prüfung deaktiviert werden. Dazu setzen Sie `OPENVPN_x_CHECK_CONFIG='no'`, um die Prüfungen für diese Verbindung auszuschalten.

OPENVPN_x_CIPHER Default siehe: `OPENVPN_DEFAULT_CIPHER`

Siehe `OPENVPN_DEFAULT_CIPHER` (Seite 182). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_COMPRESS Default siehe: `OPENVPN_DEFAULT_COMPRESS`

Siehe `OPENVPN_DEFAULT_COMPRESS` (Seite 182). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_CREATE_SECRET Default siehe: OPENVPN_DEFAULT_CREATE_SECRET='no'

Siehe OPENVPN_x_SECRET (Seite 177). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_DIGEST Default siehe: OPENVPN_DEFAULT_DIGEST

Siehe OPENVPN_DEFAULT_DIGEST (Seite 183). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_FLOAT Default siehe: OPENVPN_DEFAULT_FLOAT

Siehe OPENVPN_DEFAULT_FLOAT (Seite 183). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_KEYSIZE Default siehe: OPENVPN_DEFAULT_KEYSIZE

Siehe OPENVPN_DEFAULT_KEYSIZE (Seite 183). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_ISDN_CIRC_NAME Default OPENVPN_x_ISDN_CIRC_NAME=”

Gibt an über welchen ISDN Circuit diese OpenVPN Verbindung aufgebaut wird. Hier wird der Name des entsprechenden ISDN Circuits eingetragen, der mit ISDN_CIRC_x_NAME=" (Seite 161) definiert wird. Der entsprechende ISDN Circuit muss dabei vom Typ 'raw' sein.

OPENVPN_x_PING Default siehe: OPENVPN_DEFAULT_PING

Siehe OPENVPN_DEFAULT_PING (Seite 184). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_PROTOCOL Default: OPENVPN_x_PROTOCOL='udp'

Gibt an mit welchem Protokoll der OpenVPN Tunnel aufgebaut werden soll. Mögliche Einstellungen sind 'udp', 'tcp-server' oder 'tcp-client'. Die 'tcp-server' oder 'tcp-client' Einstellungen sind in der Regel nur dann sinnvoll, wenn ein VPN Tunnel durch diverse andere Paketfilter oder andere Tunnel aufgebaut werden soll. Wenn Sie keine Spezialfälle behandeln müssen, sollten Sie *immer* den Standardwert 'udp' benutzen.

OPENVPN_x_RESOLV_RETRY Default siehe: OPENVPN_DEFAULT_RESOLV_RETRY

Siehe OPENVPN_DEFAULT_RESOLV_RETRY (Seite 184). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_PING_RESTART Default siehe: OPENVPN_DEFAULT_PING_RESTART

Siehe OPENVPN_DEFAULT_PING_RESTART (Seite 184). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_START Default siehe: OPENVPN_DEFAULT_START

Siehe OPENVPN_DEFAULT_START (Seite 185). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_VERBOSE Default siehe: OPENVPN_DEFAULT_VERBOSE

Siehe OPENVPN_DEFAULT_VERBOSE (Seite 185). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_MANAGEMENT_LOG_CACHE Default siehe: OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE

Siehe OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE (Seite 185). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_MUTE_REPLAY_WARNINGS Default siehe: OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS

Siehe OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS (Seite 185). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_RESTART Default siehe: OPENVPN_DEFAULT_RESTART

Siehe OPENVPN_DEFAULT_RESTART (Seite 184). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_ALLOW_ICMPPING Default siehe: OPENVPN_DEFAULT_ALLOW_ICMPPING

Siehe OPENVPN_DEFAULT_ALLOW_ICMPPING (Seite 183). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_OPEN_OVPNPORT Default siehe: OPENVPN_DEFAULT_OPEN_OVPNPORT

Siehe OPENVPN_DEFAULT_OPEN_OVPNPORT (Seite 183). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_PF_INPUT_LOG Default siehe: OPENVPN_DEFAULT_PF_INPUT_LOG

Siehe OPENVPN_DEFAULT_PF_INPUT_LOG (Seite 183). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_PF_INPUT_POLICY Default siehe: OPENVPN_DEFAULT_PF_INPUT_POLICY

Siehe OPENVPN_DEFAULT_PF_INPUT_POLICY (Seite 183). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_PF_INPUT_N Default: OPENVPN_x_PF_INPUT_N='0'

Gibt die Anzahl der folgenden OPENVPN_x_PF_INPUT_x= Einträge an.

OPENVPN_x_PF_INPUT_x Default: OPENVPN_x_PF_INPUT_x="

Wie im base Paket stehen hier die Anweisungen für den Paketfilter. Es wird genau die gleiche Syntax wie in base.txt benutzt. Auch tmpl: und Hostaliase sind möglich. Zusätzlich gibt es noch die Möglichkeit, einige spezielle symbolische Namen zu benutzen. Es werden folgende symbolische Namen unterstützt:

VPNDEV Entspricht dem aktuellen tun Device der jeweiligen OpenVPN Verbindung.

LOCAL-VPN-IP Setzt die IP-Adresse von OPENVPN_x_LOCAL_VPN_IP ein.

REMOTE-VPN-IP Setzt die IP-Adresse von OPENVPN_x_REMOTE_VPN_IP ein.

REMOTE-NET Setzt die IP-Adresse von OPENVPN_x_REMOTE_VPN_IP ein und zusätzlich noch alle Netze die mit OPENVPN_x_ROUTE_x angegeben wurden.

OPENVPN_x_PF_FORWARD_LOG Default siehe: OPENVPN_DEFAULT_PF_FORWARD_LOG

Siehe OPENVPN_DEFAULT_PF_FORWARD_LOG (Seite 184). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_PF_FORWARD_POLICY Default siehe: OPENVPN_DEFAULT_PF_FORWARD_POLICY

Siehe OPENVPN_DEFAULT_PF_FORWARD_POLICY (Seite 184). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_PF_FORWARD_N Default: OPENVPN_x_PF_FORWARD_N='0'

Gibt die Anzahl der folgenden OPENVPN_x_PF_FORWARD_x= Einträge an.

OPENVPN_x_PF_FORWARD_x Default: OPENVPN_x_PF_FORWARD_x="

Siehe OPENVPN_x_PF_INPUT_x (Seite 189).

OPENVPN_x_PF_PREROUTING_N Default: OPENVPN_x_PF_PREROUTING_N='0'

Gibt die Anzahl der folgenden OPENVPN_x_PF_PREROUTING_x= Einträge an.

OPENVPN_x_PF_PREROUTING_x Default: OPENVPN_x_PF_PREROUTING_x="

Siehe OPENVPN_x_PF_INPUT_x (Seite 189).

OPENVPN_x_PF_POSTROUTING_N Default: OPENVPN_x_PF_POSTROUTING_N='0'

Gibt die Anzahl der folgenden OPENVPN_x_PF_POSTROUTING_x= Einträge an.

OPENVPN_x_PF_POSTROUTING_x Default: OPENVPN_x_PF_POSTROUTING_x="

Ab fli4l Revision 3.5.0 (oder 3.5.0-rev18133 für tarball Benutzer) ergibt sich eine Änderung im Verhalten. War es früher möglich Einträge in der Form von

```
OPENVPN_1_PF_POSTROUTING_1='MASQUERADE'
```

anzugeben wird ab jetzt die Angabe einer Quell und Zieladresse erzwungen. Dies ist notwendig geworden, weil die POSTROUTING Regeln sonst nicht im vollem Umfang genutzt werden konnten. In den meisten Fällen reicht es einfach die Regeln und IP_NET_x (Seite 41) und REMOTE-NET zu ergänzen.

Siehe OPENVPN_x_PF_INPUT_x (Seite 189).

OPENVPN_x_PF6_INPUT_N Default: OPENVPN_x_PF6_INPUT_N='0'

Gibt die Anzahl der folgenden OPENVPN_x_PF6_INPUT_x= Einträge an.

OPENVPN_x_PF6_INPUT_x Default: OPENVPN_x_PF6_INPUT_x="

Wie im IPv6 Paket stehen hier die Anweisungen für den Paketfilter. Es wird genau die gleiche Syntax wie in ipv6.txt benutzt. Auch tmpl: und Hostaliase sind möglich. Zusätzlich gibt es noch die Möglichkeit, einige spezielle symbolische Namen zu benutzen. Siehe dazu OPENVPN_x_PF_INPUT_x (Seite 189)

OPENVPN_x_PF6_FORWARD_N Default: OPENVPN_x_PF6_FORWARD_N='0'

Gibt die Anzahl der folgenden OPENVPN_x_PF6_FORWARD_x= Einträge an.

OPENVPN_x_PF6_FORWARD_x Default: `OPENVPN_x_PF6_FORWARD_x="`

Siehe `OPENVPN_x_PF6_INPUT_x` (Seite 190).

OPENVPN_x_MSSFIX Default siehe: `OPENVPN_DEFAULT_MSSFIX`

Siehe `OPENVPN_DEFAULT_MSSFIX` (Seite 186). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_FRAGMENT Default siehe: `OPENVPN_DEFAULT_FRAGMENT`

Siehe `OPENVPN_DEFAULT_FRAGMENT` (Seite 186). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_TUN_MTU Default siehe: `OPENVPN_DEFAULT_TUN_MTU`

Siehe `OPENVPN_DEFAULT_TUN_MTU` (Seite 186). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_TUN_MTU_EXTRA Default siehe: `OPENVPN_DEFAULT_TUN_MTU_EXTRA`

Siehe `OPENVPN_DEFAULT_TUN_MTU_EXTRA` (Seite 186). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_LINK_MTU Default siehe: `OPENVPN_DEFAULT_LINK_MTU`

Siehe `OPENVPN_DEFAULT_LINK_MTU` (Seite 186). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

OPENVPN_x_SHAPER Default siehe: `OPENVPN_DEFAULT_SHAPER="`

Siehe `OPENVPN_DEFAULT_SHAPER` (Seite 186). Im Gegensatz zu der Default Einstellung wirkt diese Einstellung nur auf diese OpenVPN Verbindung.

4.14.6. OpenVPN - WebGUI

Seit der Version 2.1.10 ist es möglich über eine WebGUI, die konfigurierten OpenVPN Verbindungen zu starten, stoppen und andere grundlegende Funktionen auszuführen. Dazu wird das mini_httpd Paket benötigt. Zudem muss die Variable `OPENVPN_WEBGUI` in der openvpn.txt auf 'yes' gesetzt werden. Dann wird der fli4l Weboberfläche der Menüpunkt OpenVPN hinzugefügt. Wählt man diesen Menüpunkt an, erscheint eine Übersicht über die konfigurierten OpenVPN Verbindungen, deren Status und die jeweils zur Verfügung stehenden Aktionen (siehe Abbildung 4.3).

OpenVPN - WebGUI - Verbindungsübersicht

Status: Der Status einer Verbindung wird mit Ampelmännchen symbolisiert. Ein rotes Männchen bedeutet, dass der OpenVPN-Prozess nicht läuft, ein gelbes, dass der Prozess zwar läuft aber (noch) keine Verbindung zur Gegenstelle aufgebaut werden konnte und ein grünes, dass die Verbindung mit der Gegenstelle „steht“. Genauere Informationen über den Status erhält man als Tooltip über dem Ampelmännchen. Das kann insbesondere bei „gelbem“ Status aufschlussreich sein.

Abbildung 4.3.: Verbindungsübersicht

Name: In dieser Spalte steht der Name der OpenVPN-Verbindung wie er in der Konfiguration angegeben wurde. Ein Klick auf den Namen führt in eine Übersicht, in der genauere Informationen zu dieser Verbindung angezeigt werden. Dazu später mehr.

Aktion: Hier sind die zur Verfügung stehenden Aktionen als Buttons symbolisiert. Was sie jeweils bedeuten, erfährt man über Tooltips. Folgende Buttons gibt es:

OpenVPN - WebGUI - Detailansicht einer Verbindung

Statistik: Hier werden ein paar interessante Statistiken angezeigt. Die Statistik kann nur angezeigt werden, wenn die Verbindung gestartet und nicht angehalten ist.

Log: Zeigt die letzten 20 Zeilen des Verbindungslogs. Wenn Sie mehr Zeilen sehen möchten, können Sie auch deren Anzahl angeben und auf 'Anzeigen' klicken. Wenn Sie als Anzahl 'all' einngeben, wird das gesamte Log angezeigt. Dieser Reiter wird nur angezeigt, wenn die Verbindung gestartet ist.

Debug-Log: Zeigt die Ausgabe eines Startvorgangs. Die OpenVPN-Verbindung wird gestartet und die Ausgaben dabei angezeigt. Das ist dann nützlich, wenn die Verbindung über den Start Button nicht starten will und man dadurch kein normales Log zu sehen bekommt. Dieser Reiter wird nur angezeigt, wenn die Verbindung nicht gestartet ist.

Symbol	Erläuterung
	Der OpenVPN-Prozess wird gestartet und es wird versucht eine Verbindung aufzubauen.
	Der OpenVPN-Prozess wird beendet.
	Die Verbindung wird zurückgesetzt.
	Die Verbindung wird zurückgesetz und auf 'hold' geschaltet. Dann gehen keine Daten mehr über die Verbindung.
	Die Verbindung wird wieder freigegeben. Daten können wieder über die Verbindung fließen.

Tabelle 4.9.: Aktionen der OpenVPN-Webgui

Abbildung 4.4.: Detailansicht einer Verbindung (Keymanagement)

Paketfilter: Zeigt die Paketfilterkonfiguration an, die für diese Verbindung gültig ist. Der Paketfilter ist nur konfiguriert, wenn die Verbindung gestartet und als Tunnel eingerichtet ist.

Bridge: Zeigt die Konfiguration der Bridges auf dem Router an. Dieser Punkt nur angezeigt, wenn die Verbindung als Bridge eingerichtet ist.

Konfiguration: Über diesen Punkt kann die beim Booten generierte Konfiguration der Verbindung angeschaut werden.

Keymanagement: Über diesen Punkt kann für diese Verbindung ein Schlüssel erzeugt und auch heruntergeladen werden. (siehe Abbildung 4.4) Ist kein Schlüssel vorhanden (beim ersten Start) wird automatisch einer generiert und angezeigt. Er kann über das Download-Symbol direkt heruntergeladen, oder mit copy/paste in eine Textdatei übertragen werden. Um einen neu generierten Schlüssel auf dem Router zu speichern, ist auf das Disketten-Symbol zu klicken. Ein solcher Speichervorgang kann mit einem Klick auf das Wiederherstellen Symbol rückgängig gemacht werden.

Supportinformationen: Hier werden alle Dinge angezeigt, die bei Problemen relevant sein könnten. Sie können diese Informationen dann beispielsweise auf Anfrage per copy&paste in einen Artikel der Newsgroup übertragen.

4.14.7. OpenVPN - Zusammenarbeit unterschiedlicher OpenVPN Versionen

Bei der Zusammenarbeit unterschiedlicher OpenVPN Versionen muss darauf geachtet werden, dass diese unterschiedliche Standardwerte für die Parameter einer Verbindung benutzen. Das betrifft insbesondere die MTU, Fragment und MSSFIX Einstellungen. Wenn die entsprechenden Werte nicht „zusammenpassen“, ist ein Verbindungsaufbau nicht möglich oder die Verbindung funktioniert zwar mit einem Pingbefehl, bricht aber beispielsweise bei der Benutzung von ssh zusammen. Typische Fehlermeldungen in einem solchen Fall sind z. B.:

```
FRAG_IN error flags=0xfa2a187b: FRAG_TEST not implemented
FRAG_IN error flags=0xfa287f34: spurrious FRAG_WHOLE flags
```

Die entscheidenden Parameter für das Zustandekommen einer Verbindung sind folgende Einstellungen:

OPENVPN_x_TUN_MTU Der MTU Wert des TUN Geräte war bei OpenVPN 1.x auf 1300 eingestellt. Ab OpenVPN 2.0 wird 1500 als Standardwert angenommen.

OPENVPN_x_LINK_MTU Die Bytegröße der Verbindung der beiden OpenVPN Daemonen. Dieser Standardwert ist abhängig von der verwendeten OpenVPN Version und des Betriebssystems.

OPENVPN_x_FRAGMENT Datenpakete (egal ob UDP oder TCP), deren Größe über der Fragmentgrenze liegt, werden auf Datenpakete aufgeteilt, die nicht größer sind als die unter `OPENVPN_x_FRAGMENT` angegeben Bytegröße.

OPENVPN_x_MSSFIX Damit TCP Verbindungen, die über das VPN Daten austauschen, nach Möglichkeit die Datenpakete nicht fragmentieren müssen, kann hier eine gewünschte

maximale Größe der TCP Datenpakete vorgegeben werden. An diese Vorgabe halten sich dann aktuelle Betriebssysteme und ein aufwendiges fragmentieren der Datenpakete ist nicht notwendig.

Die unterschiedlichen OpenVPN Versionen benutzen folgende Werte als Standardwerte. Diese Werte müssen Sie beachten, wenn Sie mit OpenVPN Versionen Kontakt aufnehmen wollen, die nicht auf einem fli4l-Router laufen. Die Standardwerte auf dem fli4l-Router sind in der zweiten Tabelle aufgeführt.

OpenVPN Version/Option	1.xx	2.00
OPENVPN_x_TUN_MTU	1300	1500
OPENVPN_x_TUN_MTU_EXTRA	unbekannt	32
OPENVPN_x_FRAGMENT	unbekannt	nicht konfiguriert
OPENVPN_x_MSSFIX	nicht konfiguriert	1450

Tabelle 4.10.: Unterschiedliche MTU Parameter der unterschiedlichen OpenVPN Versionen.

fli4l Version/Option	bis einschließlich 2.1.8	ab 2.1.9
OPENVPN_x_TUN_MTU	1300	1500
OPENVPN_x_TUN_MTU_EXTRA	64	32
OPENVPN_x_FRAGMENT	nicht konfiguriert	1300
OPENVPN_x_MSSFIX	nicht konfiguriert	1300

Tabelle 4.11.: Unterschiedliche MTU Parameter der fli4l-Router Versionen.

Aufgrund dieser unterschiedlichen Einstellungen, sollten Sie die für Ihr Netzwerk passenden Standardwerte ermitteln und diese dann explizit in die config/openvpn.txt schreiben. Folgende Werte sind in den meisten Fällen gute Startwerte für die ersten Tests.

```
OPENVPN_DEFAULT_TUN_MTU='1500'
OPENVPN_DEFAULT_MSSFIX='1300'
OPENVPN_DEFAULT_FRAGMENT='1300'
```

Leider gibt es für fli4l Versionen vor 2.1.9 keine Möglichkeit den „tun-mtu“ Parameter direkt zu setzen. Allerdings läßt sich dieser Parameter indirekt über den `OPENVPN_x_LINK_MTU` beeinflussen. Der tun-mtu Wert ist ca. 45 Bytes kleiner als der bei `OPENVPN_x_LINK_MTU` angegebene Wert. Um den genauen Wert zu ermitteln, hilft nur ausprobieren.

4.14.8. OpenVPN - Beispiele

Einige Beispiele verdeutlichen die Konfiguration des OpenVPN Paketes.

Beispiel - Zwei Netze mit fli4l-Routern verbinden

Im ersten Beispiel werden zwei fli4l-Router miteinander verbunden. Die Netzwerke hinter den fli4l-Routern sollen dabei Zugriff auf das jeweils andere Netzwerk erhalten. In diesem Beispiel wollen Peter und Maria ihre Netzwerke über ihre fli4l-Router miteinander verbinden. Peter benutzt als privates Netz 192.168.145.0/24 und als DynDNS Adresse 'peter.eisfair.net'. Bei Maria sieht es ähnlich aus, nur benutzt sie das Netzwerk 10.23.17.0/24 und als DynDNS Adresse 'maria.eisfair.net'. Das sich beide unbegrenzt vertrauen, erlauben sie sich gegenseitig den kompletten Zugriff auf ihre jeweiligen Netze.

OpenVPN Option	Peter	Maria
OPENVPN_1_NAME=	'maria'	'peter'
OPENVPN_1_REMOTE_HOST=	'maria.eisfair.net'	'peter.eisfair.net'
OPENVPN_1_REMOTE_PORT=	'10000'	'10001'
OPENVPN_1_LOCAL_PORT=	'10001'	'10000'
OPENVPN_1_SECRET=	'pema.secret'	'pema.secret'
OPENVPN_1_TYPE=	'tunnel'	'tunnel'
OPENVPN_1_REMOTE_VPN_IP=	'192.168.200.202'	'192.168.200.193'
OPENVPN_1_LOCAL_VPN_IP=	'192.168.200.193'	'192.168.200.202'
OPENVPN_1_ROUTE_N=	'1'	'1'
OPENVPN_1_ROUTE_1=	'10.23.17.0/24'	'192.168.145.0/24'
OPENVPN_1_PF_INPUT_N=	'1'	'1'
OPENVPN_1_PF_INPUT_1=	'ACCEPT'	'ACCEPT'
OPENVPN_1_PF_FORWARD_N=	'1'	'1'
OPENVPN_1_PF_FORWARD_1=	'ACCEPT'	'ACCEPT'

Tabelle 4.12.: OpenVPN Konfiguration mit 2 fli4l-Routern

Beispiel - Zwei Netze mit einer Bridge verbinden

Im nächsten Beispiel wird eine Bridge über eine Funkverbindung aufgebaut. Bei einer Bridge kann der Paketfilter nicht sinnvoll konfiguriert werden, da dort nur Ethernetframes weitergeleitet werden, aber nicht unbedingt IP-Pakete. Bitte immer daran denken, dass bei einer Bridgekonfiguration ein gemeinsames Netz benutzt werden muss. Und es dürfen keine IP-Adressen doppelt vergeben werden.

OpenVPN Option	Peter	Maria
OPENVPN_2_NAME	'bridge'	'bridge'
OPENVPN_2_REMOTE_HOST	'10.1.0.1'	'10.2.0.1'
OPENVPN_2_REMOTE_PORT	'10005'	'10006'
OPENVPN_2_LOCAL_HOST	'10.2.0.1'	'10.1.0.1'
OPENVPN_2_LOCAL_PORT	'10006'	'10005'
OPENVPN_2_FLOAT	'no'	'no'
OPENVPN_2_RESTART	'never'	'never'
OPENVPN_2_SECRET	'bridge.secret'	'bridge.secret'
OPENVPN_2_TYPE	'bridge'	'bridge'
OPENVPN_2_BRIDGE	'pema-br'	'pema-br'

Tabelle 4.13.: OpenVPN Konfiguration mit 2 fli4l-Routern deren Netzwerk über eine Funkverbindung gebridgt wird.

Zusätzlich zu den Angaben für OpenVPN muss natürlich noch eine Bridge in advanced_networking konfiguriert werden und die base.txt so angepaßt werden, dass dort die Bridge und nicht eth0 als Netzwerkdevice für das interne Netzwerk benutzt wird. Hier nochmal die relevanten Auszüge aus der Konfiguration von advanced_networking und base.

Beispiel - Zugriff für einen Roadwarrior konfigurieren

Bei diesem Beispiel (Roadwarrior) wird über ein Notebook mit Windows XP und einem GPRS Zugang der Zugang zu dem LAN hinter dem fli4l-Router ermöglicht. Dazu wird auf dem Windows XP Notebook OpenVPN installiert und die entsprechende *.ovpn Datei angepaßt. Leider ist der tun/tap Treiber unter Windows nicht ganz so flexibel wie sein Unix Gegenstück. Daher müssen die Point-to-Pointadressen für die VPN IP-Adressen in einem 255.255.255.252 (oder /30) Netz liegen. Wenn der Roadwarrior nur auf Dienste im LAN hinter und auf dem

advanced_networking Option	Peter	Maria
OPT_BRIDGE_DEV	'yes'	'yes'
BRIDGE_DEV_BOOTDELAY	'no'	'no'
BRIDGE_DEV_N	'1'	'1'
BRIDGE_DEV_1_NAME	'pema-br'	'pema-br'
BRIDGE_DEV_1_DEVNAME	'br0'	'br0'
BRIDGE_DEV_1_DEV_N	'1'	'1'
BRIDGE_DEV_1_DEV_1_DEV	'eth0'	'eth0'

Tabelle 4.14.: OpenVPN Konfiguration mit 2 fli4l-Routern deren Netzwerk über eine Funkverbindung gebridgt wird. Die Konfiguration der Bridge in advanced_networking.

base Option	Peter	Maria
IP_NET_N	'1'	'1'
IP_NET_1	'192.168.193.254/24'	'192.168.193.1/24'
IP_NET_1_DEV	'br0'	'br0'

Tabelle 4.15.: OpenVPN Konfiguration mit 2 fli4l-Routern deren Netzwerk über eine Funkverbindung gebridgt wird. Die Konfiguration der Bridge in der Basiskonfiguration (base.txt).

fli4l-Router zugreifen soll und nicht selber angesprochen werden muss, ist die Angabe einer Route auf der fli4l Seite nicht notwendig. Der Roadwarrior kann bei Bedarf über seine virtuelle IP-Adresse (`OPENVPN_3_REMOTE_VPN_IP`) angesprochen werden. Wenn der Roadwarrior über eine feste IP-Adresse verfügt, könnte man auch alternativ eine Hostroute eintragen. Wenn der Roadwarrior z.B. die feste IP-Adresse 192.168.33.33 hat, könnte man folgendes noch in die fli4l openvpn.txt Konfigurationsdatei einfügen:

```
OPENVPN_3_ROUTE_N='1'
OPENVPN_3_ROUTE_1='192.168.33.33/32'
```

Mit der Paketfilterkonfiguration, die hier im Beispiel gezeigt wird erlauben wir wieder die komplette Kommunikation in beide Richtungen. Nur auf den fli4l-Router direkt kann der Roadwarrior nicht zugreifen. Das wäre z.B. notwendig, wenn der Roadwarrior den DNS Server auf dem fli4l-Router benutzen soll.

```
OPENVPN_3_PF_FORWARD_N='1'
OPENVPN_3_PF_FORWARD_1='ACCEPT'
```

Soll der Zugriff vom Roadwarrior auf den internen DNS Server auf dem fli4l-Router erlaubt werden, muss noch folgendes zur fli4l Konfiguration dazugeschrieben werden:

```
OPENVPN_3_PF_INPUT_N='1'
OPENVPN_3_PF_INPUT_1='if:VPNDEV:any tmpl:dns ACCEPT'
```

Beispiel - WLAN Verbindung absichern

In diesem Beispiel wird eine WLAN Verbindung mit Hilfe von OpenVPN abgesichert. Es wird davon ausgegangen, dass im fli4l-Router sowohl eine LAN als auch eine WLAN Karte verwendet wird, oder ein Accesspoint an einer zusätzlichen Netzwerkkarte im fli4l angeschlossen

OpenVPN Option fli4l-Router	roadwarrior
OPENVPN_3_NAME='roadwarrior'	remote peter.eisfair.net
OPENVPN_3_LOCAL_PORT='10011'	rport 10011
OPENVPN_3_SECRET='roadwarrior.secret'	secret roadwarrior.secret
OPENVPN_3_TYPE='tunnel'	dev tun
OPENVPN_3_REMOTE_VPN_IP='192.168.200.238'	
OPENVPN_3_LOCAL_VPN_IP='192.168.200.237'	ifconfig 192.168.200.238 192.168.200.237
OPENVPN_3_ROUTE_N='0'	
OPENVPN_3_PF_FORWARD_N='1'	
OPENVPN_3_PF_FORWARD_1='ACCEPT'	
	route 192.168.145.0 255.255.255.0
	comp-lzo
	persist-tun
	persist-key
	ping-timer-rem
	ping-restart 60
	proto udp
	tun-mtu 1500
	fragment 1300
	mssfix

Tabelle 4.16.: OpenVPN Konfiguration mit einem Windowsrechner über GPRS.

ist. Ziel soll es sein, dass ein WLAN Client ohne VPN-Verbindung lediglich Zugriff auf den VPN Port des fli4l-Routers hat. Erst nach dem erfolgreichen Verbinden mit OpenVPN, soll uneingeschränkter Austausch mit dem Kabelgebundenen LAN möglich sein. Es müssen dafür auch Änderungen am DNSMASQ DHCP Server durchgeführt werden. Außerdem wird das advanced_networking Paket benötigt. Einstellungen in base.txt: `IP_NET_1` ist dabei das kabelgebundene LAN und `IP_NET_2` das WLAN.

```
IP_NET_N='2'
IP_NET_1='192.168.3.254/24'
IP_NET_1_DEV='br0'
IP_NET_2='192.168.4.254/24'
IP_NET_2_DEV='eth2'
```

Die DHCP-Range ist nach Belieben einzustellen. Für `IP_NET_2` sind aber unbedingt folgende Einstellungen hinzuzufügen:

```
DHCP_RANGE_2_DNS_SERVER1='none'
DHCP_RANGE_2_NTP_SERVER='none'
DHCP_RANGE_2_GATEWAY='none'
```

Einstellung in advanced_networking.txt:

```
OPT_BRIDGE_DEV='yes'
BRIDGE_DEV_BOOTDELAY='yes'
BRIDGE_DEV_N='1'
BRIDGE_DEV_1_NAME='br'
BRIDGE_DEV_1_DEVNAME='br0'
BRIDGE_DEV_1_DEV_N='1'
BRIDGE_DEV_1_DEV_1_DEV='eth0'
```

OpenVPN Option Router	WLAN-Client
OPENVPN_4_NAME='wlan1'	
OPENVPN_4_LOCAL_HOST='192.168.4.254'	remote 192.168.4.254
OPENVPN_4_LOCAL_PORT='20001'	rport 20001
OPENVPN_4_SECRET='wlan1.secret'	secret wlan1.secret
OPENVPN_4_TYPE='bridge'	dev tap
OPENVPN_4_BRIDGE='br'	
OPENVPN_4_RESTART='never'	
OPENVPN_4_MUTE_REPLAY_WARNINGS='yes'	
	comp-lzo
	persist-tun
	persist-key
	ping-timer-rem
	ping-restart 60
	proto udp
	tun-mtu 1500
	fragment 1300
	mssfix

Tabelle 4.17.: OpenVPN Absicherung eines WLAN.

4.14.9. Weiterführende Links zum Thema OpenVPN

Abschliessend noch einige Links, die sich mit der Konfiguration von OpenVPN beschäftigen:
http://openvpn.net
http://de.wikipedia.org/wiki/OpenVPN
http://openvpn.se/
http://arnowelzel.de/wiki/de/fli4l/openvpn
http://wiki.freifunk.net/OpenVPN
http://w3.linux-magazine.com/issue/24/Charly.pdf
http://w3.linux-magazine.com/issue/25/WirelessLAN_Intro.pdf
http://w3.linux-magazine.com/issue/25/OpenVPN.pdf

4.15. PCMCIA - PC-Card Unterstützung

4.15.1. PCMCIA-Treiber

fli4l kann auch mit PCMCIA-Karten zusammenarbeiten. Bei `OPT_PCMCIA`='yes' werden die entsprechenden Basis-Treiber installiert. Welche konkreten Kartentreiber verwendet werden sollen, wird z.B. über `NET_DRV_x` (Seite 33) eingestellt.

PCMCIA_PCIC - PCMCIA Socket-Driver

Es kann dabei gewählt werden: 'i82365' oder 'tcic' für PCMCIA Bridges, sowie 'yenta_socket' und 'i82092' für Cardbus Bridges.

Standard-Einstellung: `PCMCIA_PCIC`='i82365'

PCMCIA_PCIC_OPTS - Optionen für den PCMCIA Socket-Driver

Standard-Einstellung: `PCMCIA_PCIC_OPTS`=''

Mögliche Einstellungen: poll_interval=n n in je 10 Millisekunden - Sinnvoller Wert: 1000 Stellt das Abfrageintervall für Kartenwechsel ein irq_list=x,y,z,... Eine Liste der zu verwendenden Interrupts

PCMCIA_MISC_N PCMCIA_MISC_x Anzahl der zusätzlich zu ladenden PCMCIA-Module: serial_cs für Modems und Combo-Karten parport_cs Druckerschnittstellen

4.16. PPP - Anbindung eines Rechners über serielle Schnittstelle

Mit der Einstellung `OPT_PPP='yes'` ist es möglich, einen weiteren PC über die serielle Schnittstelle anzubinden. Dieses kann zum Beispiel dann sinnvoll sein, wenn man ein Notebook, welches keine Netzwerkkarte hat, in das Netzwerk einbeziehen will. Im folgenden Text wird der über die serielle Schnittstelle angeschlossene Rechner der Client-PC genannt.

PPP_DEV Hier ist der serielle Port von fli4l anzugeben. Folgende Werte sind erlaubt:

'com1' COM1-Port (klein geschrieben!)
'com2' COM2-Port (klein geschrieben!)

PPP_SPEED Hier muss die Übertragungsrate (bit/sec) eingetragen werden. 38400 wird auch von alten Schnittstellen unterstützt. Eventuell kann es Probleme geben, wenn man die Rate auf 57600 oder gar 115200 bit/s einstellt.

```
Beispiel: PPP_SPEED='38400'
```

PPP_IPADDR PPP_PEER In `PPP_IPADDR` ist die IP-Adresse von fli4l auf COM-Port-Seite einzutragen, z.B. '192.168.4.1'. In Variable `PPP_PEER` wird die zu verwendende IP-Adresse des Client-PCs eingetragen, z.B. '192.168.4.2'.

PPP_NETWORK PPP_NETMASK In `PPP_NETWORK` ist das verwendete Netzwerk einzutragen und in Variable `PPP_NETMASK` die verwendete Netzwerkmaske. Diese beide Variablen werden vom Zusatzpaket 'samba_lpd' verwendet.

Wichtig: *Dabei ist folgendes zu beachten:*

1. Die IP-Adressen dürfen **nicht** aus dem Netzwerk-Adressbereich des Ethernet-LANs stammen, sondern es muss für die Point-To-Point-Konfiguration ein eigener Netzwerk-Adressbereich verwendet werden!
2. Damit der Client-PC auch eine Verbindung zum Internet aufnehmen kann, ist das Mini-PPP-Netzwerk ebenso zu maskieren wie das LAN. Dazu ist die Liste der zu maskierenden Netzwerke mittels der Variablen `PF_POSTROUTING_%` (Seite 60) zu erweitern (s. nächster Abschnitt).
3. Man sollte den Client-PC mit in die Host-Tabelle für den DNS-Server aufnehmen.

Das hat folgenden Grund:

Möchte man vom Client-PC telnet oder ftp zum fli4l-Router verwenden, machen die entsprechenden Daemons auf fli4l einen Reverse-DNS-Lookup, um festzustellen, wer denn da eine Verbindung wünscht. Ist der Client-PC nicht in der Host-Tabelle eingetragen, wird eine Verbindung in's Internet hergestellt, um den Namen des Clients herauszufinden. Und dieses kann durch Eintragen des Client-PCs in der Host-Tabelle vermeiden.

Beispiel für eine PPP-Konfiguration über die serielle Schnittstelle:

```
PPP_DEV='com1'
PPP_SPEED='38400'
PPP_IPADDR='192.168.4.1'
PPP_PEER='192.168.4.2'
PPP_NETWORK='192.168.4.0'
PPP_NETMASK='255.255.255.0'
```

und weiter in config/base.txt:

```
PF_POSTROUTING_N='2'
PF_POSTROUTING_1='192.168.6.0/24 MASQUERADE'
PF_POSTROUTING_2='192.168.4.0/24 MASQUERADE'
```

Das erste Netzwerk ist das Ethernet-LAN und das zweite das oben konfigurierte PPP-Netzwerk.

Als letztes ist noch die DNS-Konfiguration anzupassen, z.B.:

```
HOST_5='192.168.4.2 serial-pc'
```

Nicht vergessen, HOST_N (Seite 98) zu inkrementieren!

Ist der Client-PC ein Windows-Rechner, kann dort über die Konfiguration des DFÜ-Adapters eine PPP-Verbindung zum fli4l-Router konfiguriert werden.

Bei Verwendung eines Linux-Rechners erstellt man am besten folgendes Shellscript auf dem Client-PC (z.B. /usr/local/bin/ppp-on):

```
#! /bin/sh
dev='/dev/ttyS0'                    # COM1, für COM2: ttyS1
speed='38400'                       # Geschwindigkeit
options='defaultroute crtscts'      # Optionen
myip='192.168.4.2'                  # IP-Adresse Notebook
fli4lip='192.168.4.1'               # IP-Adresse fli4l-Router
pppd $dev $speed $options $myip:$fli4lip &
```

Sollte es es damit Probleme geben: man pppd

Auch muss der fli4l-Rechner als DNS-Server auf dem Client-PC eingetragen werden, wenn man mit dem Rechner eine Verbindung zum Internet wünscht. Es müssen dafür in /etc/resolv.conf des Client-PCs folgende zwei Zeilen eingetragen werden: die gewählte Domain und die Ethernet-IP-Adresse des fli4l-Router als Nameserver.

Beispiel:

```
search domain.de
nameserver 192.168.1.4
```

“domain.de” bzw. “192.168.1.4” sind durch die entsprechenden Werte zu ersetzen. Wichtig: Die IP-Adresse muß die der Ethernet-Karte des fli4l-Routers sein!

Als serielle Verbindung wird ein sogenanntes Nullmodemkabel (Seite 365) verwendet. Die Anschluß-Belegung ist im Anhang der Base-Dokumentation beschrieben.

Ein Howto, wie ein Windows-Client über serielles PPP angebunden werden kann, hat Oliver Walter verfasst:

http://www.fli4l.de/hilfe/howtos/basteleien/opt-ppp-howto/

4.17. PROXY - Verschiedene Proxy-Server

4.17.1. OPT_PRIVOXY - Ein Werbung-filternder HTTP-Proxy

Privoxy wird auf der offiziellen Privoxy-Homepage (http://www.privoxy.org/) als "Privacy Enhancing Proxy" (="privatsphärenerweiternder Proxy") beworben. Als sichbarer und erwünschenwerter Nebeneffekt ersetzt Privoxy Werbe-Banner und -Popups durch leere Bilder, verhindert das Speichern von ungewollten Cookies (kleine Datenpakete, mit denen eine Website einen bestimmten Surfer wiedererkennen kann) und verhindert die Anzeige von sogenannten Web-Bugs (das sind 1x1 Pixel große Bilder, die dazu benutzt werden, um das Surfverhalten von Benutzern auszuspähen).

Privoxy kann, während es läuft, ganz einfach über ein Webinterface konfiguriert und (de)aktiviert werden. Dieses Webinterface findet sich unter http://config.privoxy.org/ oder http://p.p/.

Eine so getätige Konfiguration überlebt keinen Neustart... (tobi

Privoxy ist eine konsequente Weiterentwicklung des Internet Junkbusters, der bis Version 2.1.0 in diesem Paket (http://www.junkbuster.com/) enthalten war. Die wichtigste Neuerung ist, dass alle Regeln für die Filterung in der zentralen Datei `default.action` definiert werden. Diese befindet sich bei fli4l im Verzeichnis `/etc/privoxy`. Der große Vorteil an dieser Methode ist, dass sich neue Versionen dieser Datei seperat von http://sourceforge.net/projects/ijbswa/files/ herunterladen lassen. So kann jeder fli4l-Benutzer die Datei selbst auf dem neusten Stand halten, ohne auf Updates von fli4l angewiesen zu sein. (Momentan befindet sich die Version 1.8 dieser Datei im Paket.)

PRIVOXY_MENU Fügt dem httpd-Menü einen Privoxy-Abschnitt hinzu.

PRIVOXY_N Gibt die Anzahl der Privoxy-Instanzen an, die gestartet werden sollen.

PRIVOXY_x_LISTEN Hier werden die IP-Adressen oder symbolischen Namen inklusive der Portnummer der Interfaces angegeben, auf denen Privoxy auf Verbindungen von Clients horchen soll. Es ist eine gute Idee, hier nur die Adressen der Interfaces anzugeben, denen man vertraut, da alle Rechner vollen Zugriff auf Privoxy haben (und auf den eventuell aktivierten Konfigurations-Editor). In der Regel ist die Vorgabe `IP_NET_1_IPADDR:8118` sinnvoll.

Auf hier angegebenen Adressen lauscht Privoxy und bietet seine Dienste an. 8118 ist der Standard-Port. Die Angabe hier muss man dann bei der Proxy-Konfiguration des jeweils verwendeten Internet-Browsers benutzen. Weitere Informationen zur Konfiguration von Internet Explorer und Netscape Navigator auf

http://www.privoxy.org/

Genaue URL

Als Proxy beim jeweiligen Browser muss der fli4l-Rechner angegeben werden, also das, was man bei HOSTNAME='fli4l' angegeben hat bzw. dessen IP (z.B 192.168.6.1), die man bei `HOST_x_IP`='192.168.6.1' angegeben hat. Zusammen mit dieser Port-Angabe hier hat man dann alle nötigen Daten, um seinen Webbrowser für die Nutzung von Privoxy zu konfigurieren.

PRIVOXY_x_ALLOW_N Gibt die Anzahl der Listeneinträge an.

PRIVOXY_x_ALLOW_x Die Liste der Netze und/oder IP-Adressen für die der Paketfilter geöffnet wird. Sinnvoll ist hier auch wieder die Vorgabe `IP_NET_1`.

PRIVOXY_x_ACTIONDIR Diese Variable gibt den Ort an, an dem die Privoxy-Regelsätze (die Dateien *default.action* und *user.action*) auf dem Router liegen sollen. Der angegebene Pfad wird relativ zum Wurzelverzeichnis ausgewertet. Diese Variable kann für zwei Dinge verwendet werden:

Verlagern der Standardregeln auf permanenten Speicher Gibt man als Verzeichnis einen Ort ausserhalb der Ram-Disk an, werden die Standardregelsätze beim erstmaligen Booten dorthin kopiert und dann von diesem Ort aus genutzt. Änderungen an diesen Regelsätzen überleben dann ein Reboot des Routers. Zu beachten ist, dass nach einem Update des Privoxy-Paketes diese Regeln immer noch Verwendung finden und evtl. mit dem aktuellen Paket kommende neuere Regelsätze ignoriert werden.

Verwenden eigener Regelsätze fli4l gestattet das Überschreiben der Standardregeln mit nutzerspezifischen Regeln. Dazu legt man im *config*-Verzeichnis ein Unterverzeichnis an (z.B. *etc/my_privoxy*; es darf nicht *etc/privoxy* heissen) und legt dort die eigenen Regeln ab.

Das Setzen dieser Variable ist optional.

PRIVOXY_x_HTTP_PROXY Möchte man zusätzlich zu Privoxy einen weiteren HTTP Proxy verwenden, der dann z.B. auch Webseiten zwischenspeichert, so kann man den hier angeben. Privoxy bedient sich dann dieses Proxys. So kann man die Vorteile mehrerer Proxys nutzen. Ein Eintrag könnte so aussehen:

```
PRIVOXY_1_HTTP_PROXY='mein.provider.de:8000'
```

Die Angabe ist optional.

PRIVOXY_x_SOCKS_PROXY Möchte man zusätzlich zu Privoxy einen weiteren SOCKS Proxy verwenden, kann man den hier angeben. Um die Privatspähre weiter zu erhöhen kann der Datenverkehr vom Privoxy beispielsweise durch das Tor Netzwerk geschickt werden. Für weitere Details zu Tor lesen Sie in der Tor Dokumentation (Seite 204) weiter. Ein Eintrag für die Nutzung von Tor könnte so aussehen:

```
PRIVOXY_x_SOCKS_PROXY='127.0.0.1:9050'
```

Die Angabe ist optional.

PRIVOXY_x_TOGGLE Diese Option aktiviert die Möglichkeit, den Proxy über das Webinterface ein- bzw. auszuschalten. Wird Privoxy ausgeschaltet, wirkt er als einfacher Forwarding-Proxy und ändert den Inhalt der übertragenen Seiten in keinster Weise. Es ist zu beachten, dass diese Einstellung für ALLE Benutzer des Proxys gilt, d.h. wenn ein Benutzer Privoxy abschaltet, ist Privoxy auch für die anderen Nutzer nur noch ein Weiterleitungs-Proxy.

PRIVOXY_x_CONFIG Diese Option ermöglicht den Benutzern des Proxys die interaktive Bearbeitung der Konfiguration über das Privoxy-Webinterface. Für weitere Details bitte ich auch hier, die Privoxy-Dokumentation zu konsultieren.

PRIVOXY_x_LOGDIR Mit dieser Option kann ein Logverzeichnis für Privoxy angegeben werden. Dies kann z.B. dann sinnvoll sein, wenn Website-Zugriffe der Benutzer geloggt werden sollen. Wird hier nichts angegeben (Standard), dann loggt nur die wichtigsten Meldungen auf die Konsole und ignoriert `PRIVOXY_LOGLEVEL`.

Hier kann auch 'auto' eingetragen werden, was den Log-Pfad auf das System-Verzeichnis für persistente Daten verlegt. Bitte darauf achten, daß in diesem Fall `FLI4L_UUID` korrekt konfiguriert wird, da mit großen Datenmengen zu rechnen ist und sonst /boot oder gar die Ram-Disk gefüllt wird.

PRIVOXY_x_LOGLEVEL Diese Option gibt an, was Privoxy in die Logdatei schreiben soll. Folgende Werte sind möglich, sie können durch Leerstelle getrennt angegeben werden, man kann sie aber auch addieren.

Wert	Was wird geloggt?
1	Jeden Request (GET/POST/CONNECT) ausgeben.
2	Status jeder Verbindung ausgeben
4	I/O-Status anzeigen
8	Header-Parsing anzeigen
16	**Alle** Daten loggen
32	Force-Feature debuggen
64	reguläre Ausdrücke debuggen
128	schnelle Weiterleitungen debuggen
256	GIF De-Animation debuggen
512	Common Log Format (zur Logfile-Analyse)
1024	Popup-Kill-Funktion debuggen
2048	Zugriffe auf den eingebauten Webserver loggen
4096	Startmeldungen und Warnungen
8192	Nicht-fatale Fehler

Um eine Logdatei im Common Logfile Format zu erstellen, sollte NUR der Wert 512 angegeben werden, da sonst die Logdatei durch andere Meldungen "verschmutzt" wird und somit nicht mehr problemlos ausgewertet werden kann.

Privoxy bietet sehr viele Konfigurationsmöglichkeiten. Diese können aber aus verständlichen Gründen nicht alle durch die Konfigurations-Datei von fli4l abgedeckt werden. Sehr viele dieser Optionen können im Webinterface von Privoxy eingestellt werden. Genauere Infos über den Aufbau dieser Dateien gibt es auf der Privoxy-Homepage. Die Konfigurationsdateien von Privoxy liegen unter <fli4l-Verzeichnis>/opt/etc/privoxy/. Es handelt sich hierbei um die Orginal-Dateien aus dem Privoxy-Paket, allerdings wurden, um Platz zu sparen, alle Kommentare entfernt.

4.17.2. OPT_TOR - Ein anonymes Kommunikationssystem für das Internet

Tor ist ein Werkzeug für eine Vielzahl von Organisationen und Menschen, die ihren Schutz und ihre Sicherheit im Internet verbessern wollen. Die Nutzung von Tor hilft Ihnen, das Browsen und Veröffentlichen im Web, Instantmessaging, IRC, SSH und anderen TCP basierende Anwendungen zu anonymisieren. Weiterhin bietet Tor eine Plattform auf der Softwareentwickler neue Anwendungen schaffen können die zu mehr Anonymität, Sicherheit und zum Schutz der Privatsphäre beitragen.

https://www.torproject.org/index.html.de

TOR_LISTEN_N

TOR_LISTEN_x Hier werden die IP-Adressen oder symbolischen Namen inklusive der Portnummer der Interfaces angegeben, auf denen Tor auf Verbindungen von Clients horchen soll. Es ist eine gute Idee, hier nur die Adressen der Interfaces anzugeben, denen man vertraut, da alle Rechner vollen Zugriff auf Tor haben (und auf den eventuell aktivierten Konfigurations-Editor). In der Regel ist die Vorgabe `IP_NET_1_IPADDR:9050` sinnvoll.

Auf hier angegebenen Adressen lauscht Tor und bietet seine Dienste an. 9050 ist der Standard-Port. Die Angabe hier muss man dann bei der Proxy-Konfiguration des jeweils verwendeten Programms benutzen.

Als Proxy beim jeweiligen Programm muss der fli4l-Rechner angegeben werden, also das, was man bei HOSTNAME='fli4l' angegeben hat bzw. dessen IP (z.B 192.168.6.1), die man bei `HOST_x_IP`='192.168.6.1' angegeben hat. Zusammen mit dieser Port-Angabe hier hat man dann alle nötigen Daten, um sein Programm für die Nutzung von Tor zu konfigurieren.

TOR_ALLOW_N Gibt die Anzahl der Listeneinträge an.

TOR_ALLOW_x Die Liste der Netze und/oder IP-Adressen für die der Paketfilter geöffnet wird. Sinnvoll ist hier auch wieder die Vorgabe `IP_NET_1`.

TOR_CONTROL_PORT Hier kann angegeben werden auf welchem TCP Port Tor einen Kontrollzugang über das Tor Control Protocol öffnen soll. Die Angabe ist optional. Wird nichts angegeben wird diese Funktion deaktiviert.

TOR_CONTROL_PASSWORD Hier kann ein Passwort für den Kontrollzugang angegeben werden.

TOR_DATA_DIR Diese Angabe ist optional. Wird nichts angegeben, wird der Standardordner /etc/tor verwendet

TOR_HTTP_PROXY Soll Tor die Anfragen an einen HTTP-Proxy weiterleiten, kann man den hier angeben. Tor bedient sich dann dieses Proxys. So kann man die Vorteile mehrerer Proxys nutzen. Ein Eintrag könnte so aussehen:

```
TOR_HTTP_PROXY='mein.provider.de:8000'
```

Die Angabe ist optional.

TOR_HTTP_PROXY_AUTH Eine eventuell notwendige Authentifizierung für den Proxy kann hier in der Form Benutzername:Passwort eingetragen werden.

TOR_HTTPS_PROXY Hier kann ein HTTPS-Proxy eingetragen werden. Siehe dazu auch `TOR_HTTP_PROXY`.

TOR_HTTPS_PROXY_AUTH Siehe dazu `TOR_HTTP_PROXY_AUTH`.

TOR_LOGLEVEL Diese Option gibt an, was Tor in die Logdatei schreiben soll. Folgende Werte sind möglich: debug, info, notice, warn oder err Die Werte debug und info sollten aus Sicherheitsgründen möglichst nicht verwendet werden.

TOR_LOGFILE Falls Tor statt ins syslog in eine Datei loggen soll, kann diese hier angegeben werden.

Hier kann auch 'auto' eingetragen werden, was den Log-Pfad auf das System-Verzeichnis für persistente Daten verlegt. Bitte darauf achten, daß in diesem Fall `FLI4L_UUID` korrekt konfiguriert wird, da mit großen Datenmengen zu rechnen ist und sonst /boot oder gar die Ram-Disk gefüllt wird.

4.17.3. OPT_SS5 - Ein Socks4/5 Proxy

Für einige Programme wird ein Socks-Proxy benötigt. SS5 stellt diese Funktionalität bereit.
http://ss5.sourceforge.net/

SS5_LISTEN_N

SS5_LISTEN_x Hier werden die IP-Adressen oder symbolischen Namen inklusive der Portnummer der Interfaces angegeben, auf denen SS5 auf Verbindungen von Clients horchen soll. Es ist eine gute Idee, hier nur die Adressen der Interfaces anzugeben, denen man vertraut, da alle Rechner vollen Zugriff auf SS5 haben (und auf den eventuell aktivierten Konfigurations-Editor). In der Regel ist die Vorgabe `IP_NET_1_IPADDR:8050` sinnvoll.

Auf hier angegebenen Adressen lauscht SS5 und bietet seine Dienste an. 8050 ist der Standard-Port. Die Angabe hier muss man dann bei der Proxy-Konfiguration des jeweils verwendeten Programms benutzen.

Als Proxy beim jeweiligen Programm muss der fli4l-Rechner angegeben werden, also das, was man bei HOSTNAME='fli4l' angegeben hat bzw. dessen IP (z.B 192.168.6.1), die man bei `HOST_x_IP`='192.168.6.1' angegeben hat. Zusammen mit dieser Port-Angabe hier hat man dann alle nötigen Daten, um sein Programm für die Nutzung von SS5 zu konfigurieren.

SS5_ALLOW_N Gibt die Anzahl der Listeneinträge an.

SS5_ALLOW_x Die Liste der Netze und/oder IP-Adressen für die der Paketfilter geöffnet wird. Sinnvoll ist hier auch wieder die Vorgabe `IP_NET_1`.

4.17.4. OPT_TRANSPROXY (EXPERIMENTELL) - Transparenter HTTP-Proxy

Transproxy ist ein „transparenter" Proxy, also ein Programm, dass es ermöglicht, alle HTTP-Abfragen, die über den Router laufen, abzufangen und an einen normalen HTTP-Proxy, z.B. Privoxy, weiterzuleiten. Um diese Funktionalität zu erreichen, muss der Paketfilter HTTP-Anfragen, die eigentlich ins Internet gehen sollen, an Transproxy weiterreichen, welcher diese weiter aufbereitet und an den anderen HTTP-Proxy weitergibt. iptables bietet zur Unterstützung dieser Funktion die Aktion „REDIRECT":

```
PF_PREROUTING_1='tmpl:http IP_NET_1 REDIRECT:8081'
```

Diese Regel würde alle HTTP-Pakete aus dem ersten definierten Netz (normalerweise das interne LAN) an Transproxy auf Port 8081 weiterleiten.

TRANSPROXY_LISTEN_N

TRANSPROXY_LISTEN_x Hier werden die IP-Adressen oder symbolischen Namen inklusive der Portnummer der Interfaces angegeben, auf denen Transproxy auf Verbindungen von Clients horchen soll. Hier müssen alle Interfaces angegeben werden, für die im Paketfilter Pakete auf Transproxy umgelenkt werden. Mit der Vorgabeeinstellung `any:8081` hört Transproxy auf allen Interfaces.

TRANSPROXY_TARGET_IP

TRANSPROXY_TARGET_PORT Mit diesen Optionen wird festgelegt, an welchen Dienst eingehende HTTP-Anfragen umgeleitet werden. Dies kann ein beliebiger Standard-HTTP-Proxy (Squid, Privoxy, Apache, etc.) auf einem beliebigen anderen Rechner (oder auch auf fli4l selbst) sein. Hier ist darauf zu achten, dass der Proxy sich nicht im Bereich der durch den Paketfilter umgeleiteten HTTP-Anfragen befindet, da sonst eine Schleife entsteht.

TRANSPROXY_ALLOW_N

TRANSPROXY_ALLOW_x Die Liste der Netze und/oder IP-Adressen für die der Paketfilter geöffnet wird. Dies sollte die gleichen Netze abdecken, die auch im Paketfilter umgeleitet werden. Werden hier keine Bereiche angegeben, müssen die Angaben von Hand in der Paketfilter-Konfiguation vorgenommen werden.

4.17.5. OPT_SIPPROXY (EXPERIMENTELL) - Proxy für Session Initiation Protocol

Möchte man mehrere SIP-Anwendungen (egal ob Ekiga, x-lite oder Hardware SIP-Telefone) hinter einem Router betreiben, so kann es vorkommen, dass Netzwerk-Ports speziell weitergeleitet werden müssen, da sonst die Verbindungen nicht so funktionieren, wie sie sollten.

Um dies zu vermeiden, kann man einen speziellen SIP Proxy-Server nutzen. Es werden derzeit (fli4l V4.0.0) mehrere solche Proxys evaluiert. Sollte jemand über eine Empfehlung verfügen so darf er sich gerne an das Team wenden!

4.17.6. OPT_IGMPPROXY - Proxy für Internet Group Management Protocol

Die Deutsche Telekom AG bietet mittlerweile seit einigen Jahren VDSL25/50 (Bandbreite: 25/50 MBit/s) zusammen mit Entertain-Paketen an. Damit besteht die Möglichkeit, Fernsehen über Internet (IPTV) zu empfangen.

Die Verteilung von IPTV erfolgt als Multicast, d.h. von einem Punkt zu einer (geschlossenen) Gruppe. Zur Organisation vom Multicast-Gruppen ist das Netzwerkprotokoll IGMP (Internet Group Management Protocol) notwendig. IGMP (http://de.wikipedia.org/wiki/IGMP) bietet die Möglichkeit, dynamisch Multicast-Gruppen zu verwalten. Die Verwaltung findet nicht in der Sende-Station statt, sondern in den Routern, an denen Empfänger einer Multicast-Gruppe direkt angeschlossen sind. IGMP bietet Funktionen, mit denen eine Station einem Router mitteilt, dass sie Multicast-IP-Pakete einer bestimmten Multicast-Gruppe empfangen will.

Die mitgelieferten Speedport-Router (derzeit W700V/W701V/W722) unterstützen IGMP. Wer fli4l für IPTV statt Speedport-Router nutzen will, benötigt einen IGMP-Proxy (http://sourceforge.net/projects/igmpproxy/) auf dem fli4l-Router. OPT_IGMPPROXY ist ein IGMP-Proxy für fli4l.

Diese Dokumentation zum OPT_IGMP Paket beschreibt die Konfiguration von fli4l, um VDSL und IPTV mit der mitgelieferten Set-Top-Box (STB) X300T/X301T bzw. MR-303 hinter einem fli4l-Router zu betreiben. In dieser Beschreibung erfolgt die Installation von IPTV über eine zusätzliche Netzwerkkarte.

Voraussetzung

Die Deutsche Telekom hat VDSL als VLAN eingeführt. In der Einführungsphase (Startnetz) wurde nur ein VLAN-Tag (ID7) verwendet, über den der gesamte Traffic floss. Nach der Umstellung (Zielnetz) auf zwei VLAN-Tags (ID7, ID8) bleibt der Internet Traffic auf ID7 und der neue ID8 wird ausschließlich für den IPTV Multicast-Traffic verwendet. Die Umstellung des VDSL Betriebs auf das Zielnetz (zwei VLAN Tags ID7/ID8) ist nach derzeitigem Stand größtenteils abgeschlossen.

Hardware (neben Set-Top-Box und VDSL-Modem):

- HW für fli4l: Für VDSL 25/50 sollte es besser kein 486er mehr sein. Falls es zu Bild-/Tonstörungen kommt kann das daran liegen, dass die eingesetzte HW zu wenig Leistung hat.
- High-End-NICs (Beispiele: 3Com, Intel Pro100). Realtek-Chipsätze stellen eher das Low-End-Spektrum dar.

Software:

- Paket: advanced_networking
- Paket: dhcp_client (für Zielnetz und Verwendung von ID8)

Die Anpassung der Konfigurationsdateien (base.txt, dsl.txt, advanced_networking.txt, dhcp_client.txt, dns_dhcp.txt) wird im Folgenden beschrieben.

Hardware Setup

Die Empfehlung für den Speedport-Router, die IPTV STB ohne weitere Netzwerk-Elemente direkt an den Router anzuschließen, gilt natürlich auch für den fli4l-Router. Falls dennoch Netzwerk-Knoten (Hub, Switch, Bridge, Gateway, Router) zwischen IPTV Box und Router geschaltet werden, sollten diese multicastfähig sein, um Störungen zu vermeiden.

Im Heimnetz werden i.d.R. keine Switche verwendet, die virtuelle Netze (VLAN) voneinander trennen, um den restlichen Verkehr (ID7) vom IPTV Multicast-Traffic (ID8) zu entlasten.

Deshalb wird hier als HW-Konfiguration eine separate NIC (Network Interface Card = LAN- bzw. Ethernet-Karte) im fli4l verwendet, um die Set-Top-Box (STB) direkt mit dem fli4l zu verbinden und das restliche Heimnetz vom Multicast-Traffic zu entlasten und alle o.g. Probleme auszuschließen. Wer die 'Single'-NIC-Methode bevorzugt sollte selbst wissen was er tut (das wird hier nicht weiter beschrieben).

Anbei ein Diagramm, wie im genannten Beispiel der fli4l-Router vom Standard-Router zum Router mit 3 NIC's migriert wird:

- Standard-Konfiguration:
  - eth0 wird als NIC für das interne home/office LAN in base.txt eingetragen
  - In dsl.txt wird als DSL-Interface eth1 angegeben

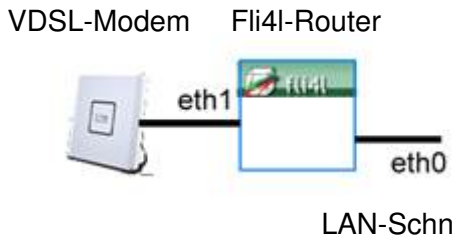


Abbildung 4.5.: Fli4l in der Standardkonfiguration

- Erweiterte Konfiguration mit zusätzlicher IPTV NIC:
  - Nach Einbau der zusätzlichen NIC in den fli4l-Router wird diese in base.txt als eth2 eingetragen.

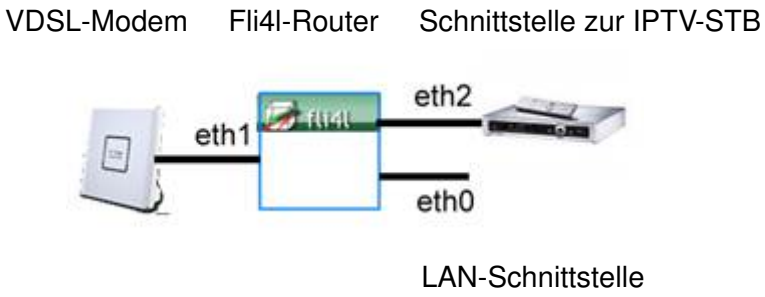


Abbildung 4.6.: Fli4l in der IPTV-Konfiguration

VLAN-Konfiguration

Eines vorweg: OPT_IGMP ist nicht auf VLAN angewiesen. Vielmehr wird VLAN derzeit von der Deutschen Telekom für VDSL verwendet und muss dafür vom Router unterstützt werden. Ob VLAN für den Internet-Betrieb auch bei anderen Providern (Arcor, Alice, etc...) benötigt wird, ist uns nicht bekannt.

Um VDSL25/50 von T-Home für den Internet-Betrieb zum Laufen zu bringen, muss die NIC zum VDSL-Modem zwingend als VLAN-Interface konfiguriert werden - siehe auch (http://www.fli4l.de/fileadmin/doc/deutsch/html/fli4l-3.4.0/node30.html).

Ein Hinweis für alle, die nur das 'normale DSL' der Telekom, also ADSL, ADSL2, ADSL2+ haben: VLAN wird nur von VDSL benötigt, nicht aber vom 'normalen DSL'. Die VLAN-Konfiguration wird deshalb mit dem 'normalen DSL' nicht funktionieren.

Bei dem Vorhandensein von zwei VLAN-Tags (Zielnetz, siehe oben) wird der traffic wie folgt aufgeteilt:

- VLAN ID7: Internet-Traffic

- VLAN ID8: IPTV Multicast-Traffic

Damit läuft Internet-Traffic unabhängig vom IPTV-Traffic. Wesentlicher Unterschied ist, dass für VLAN ID7 eine PPPoE-Einwahl erforderlich ist. VLAN ID8 wird über einen DHCP-Server ohne Einwahl zur Verfügung gestellt. In dieser Zielarchitektur gibt es keine Zwangstrennung nach 24h mehr.

Für VLAN ist folgende Konfiguration erforderlich (NICs wie im Abschnitt Hardware-Setup angegeben):

advanced_networking.txt

```
VLAN_DEV_N='2'
VLAN_DEV_1_DEV='eth1'     # interface of VDSL-Modem; example: eth1
                          # In unserem Beispiel geht 'eth1' zum VDSL-Modem
VLAN_DEV_1_VID='7         # ID7 to support VLAN for internet
VLAN_DEV_2_DEV='eth1'     # interface of VDSL-Modem; example: eth1
                          # In unserem Beispiel geht 'eth1' zum VDSL-Modem
VLAN_DEV_2_VID='8'        # ID8 to support VLAN for IPTV
```

Die Virtual-NIC eth1.7 muss in die DSL-Konfiguration eintragen werden:

dsl.txt

```
PPPOE_ETH='eth1.7'        # eth<nummer der karte zum vdsl-modem>.7'
                          # Bsp 'eth1.7'
```

Für die Virtual-NIC eth1.8 benötigen wir einen dhcp_client, da VLAN ID8 über einen DHCP-Server ohne Einwahl zur Verfügung gestellt wird.

dhcp_client

```
OPT_DHCP_CLIENT='yes'
DHCP_CLIENT_TYPE='dhcpcd'
DHCP_CLIENT_INTERFACES='IP_NET_3_DEV' # listen on interface eth1.8
DHCP_CLIENT_USEPEERDNS='no'
DHCP_CLIENT_HOSTNAME=''
```

Seit Fli4l V3.3 kann für das Interface nicht mehr `eth1.8` angegeben werden, sondern es muss der Eintrag `IP_NET_x_DEV` verwendet werden, der für das Interface in base.txt definiert wurde; hier `IP_NET_3_DEV`.

Optional:
Falls die verwendete NIC mit der MTU-Größe Probleme hat, muss der MTU-Wert über den Parameter DEV_MTU angepasst werden. Im Test zeigten Intel Pro/100 (e100) und auch eine 3-Com-Karte keine Probleme, andere User berichten, dass bei der 3Com '3c59x' der MTU-Wert auf 1496 angepasst werden muss.

```
DEV_MTU_1=''              # Adjust MTU size of NIC on VDSL-Modem
                          # Example: DEV_MTU_1='eth1 1496'
```

Jetzt sind noch die Konfigurationsdateien base.txt und dns_dhcp.txt anzupassen, wie im nächsten Kapitel beschrieben.

Konfiguration einer zusätzlichen NIC für IPTV

In base.txt und dns_dhcp.txt muss die Konfiguration für VLAN und für die zweite NIC angepasst werden.

Zweite NIC für IPTV eintragen:

```
NET_DRV_N='2'
NET_DRV_1='via-rhine'     # 1. NIC für als LAN-Schnittstelle
NET_DRV_2='3c59x'         # 2. NIC - hier 3Com für IPTV SetTopBox
```

Jetzt muss der Adressraum für die zweite NIC festgelegt werden. Hier wird fürs LAN 192.168.2.0/24 und für die zweite NIC 192.168.3.0/24 verwendet. Außerdem werden Einträge für die Virtual-NICs eth1.7 und eth1.8 benötigt:

```
IP_NET_N='4'
IP_NET_1='192.168.2.1/24'           # home/office LAN
IP_NET_1_DEV='eth0'
IP_NET_2='192.168.3.1/24'           # iptv LAN
IP_NET_2_DEV='eth2'
IP_NET_3='dhcp'                     # dhcp client - IP ueber dhclient
IP_NET_3_DEV='eth1.8'
IP_NET_3_MAC='00:40:63:da:cf:32'    # neue MAC/nicht MAC von eth1
IP_NET_4='dhcp'                     # eth1.7 zum modem
IP_NET_4_DEV='eth1.7'
IP_NET_4_MAC='00:40:63:da:cf:33'    # neue MAC/nicht MAC von eth1
```

Wichtig ist auch die Änderung die MAC-Adressen für eth1.7 und eth1.8, welche nicht mit eth1 übereinstimmen dürfen, da sonst – abhängig vom VDSL-Net der DTAG – ggf. Störungen nach der Zwangstrennung auftreten können.

Für die neue NIC muss der Zugriff auf das Internet natürlich genauso funktionieren, wie für die erste NIC. Dazu sind weitere Einstellungen notwendig:

```
PF_INPUT_1='IP_NET_1 ACCEPT'
PF_INPUT_2='IP_NET_2 ACCEPT'
PF_INPUT_3='any 224.0.0.0/4 ACCEPT'
[...]
PF_FORWARD_3='any 224.0.0.0/4 ACCEPT'
PF_FORWARD_5='IP_NET_1 ACCEPT'
PF_FORWARD_6='IP_NET_2 ACCEPT'
[...]
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'
PF_POSTROUTING_2='IP_NET_2 MASQUERADE'
```

Damit später auch eine dynamische DHCP-Adressierung an der neuen IPTV-NIC klappt und die SetTop-Box mit einem Namen angesprochen werden kann, sind noch folgende Einstellungen in dns_dhcp.txt erforderlich:

```
HOST_10_NAME='igmp'
HOST_10_IP4='192.168.3.1'
HOST_11_NAME='iptv'
HOST_11_IP4='192.168.3.4'
HOST_11_MAC='00:D0:E0:93:49:34'          # MAC Adr T-Home X300T
[...]
DHCP_RANGE_2_NET='IP_NET_2'
DNSDHCP_RANGE_2_START='192.168.3.10'
DNSDHCP_RANGE_2_END='192.168.3.20'
DNSDHCP_RANGE_2_DNS_SERVER1=''
DNSDHCP_RANGE_2_DNS_SERVER2=''
DNSDHCP_RANGE_2_NTP_SERVER=''
DNSDHCP_RANGE_2_GATEWAY=''
```

Am Besten ist es, nach der Konfiguration der neuen NIC, diese erst einmal an einen PC zu hängen um zu sehen ob man über die neue NIC auch ins Internet kommt. Ist der Test erfolgreich, sollte die neue zweite NIC richtig konfiguriert sein.

IGMP-Funktion

Beim Booten des fli4l-Routers werden die Parameter der config-Datei proxy.txt in die Konfigurationsdatei /etc/igmpproxy.conf geschrieben, welche beim Start des Programms igmpproxy eingelesen werden.

Entgegen den früheren opt_igmp Versionen, wird der IGMP-Proxy jetzt einmalig beim Booten des Routers gestartet und läuft dann solange eine physikale Verbindung zum Internet besteht. Der IGMP-Proxy wird weder durch die 24h Zwangstrennung, noch durch ein manuelles trennen und verbinden des Internet-Traffics beeinflusst.

IGMP-Konfiguration

OPT_IGMPPROXY Mit `'yes'` wird das IGMP Proxy Paket aktiviert. Die Einstellung `'no'` deaktiviert das Paket komplett.

IGMPPROXY_DEBUG Mit `'yes'` können Meldungen des IGMP Proxies ins syslog ausgegeben werden.

IGMPPROXY_DEBUG2 Mit `'yes'` kann das Loglevel des IGMP Proxies erhöht werden.

IGMPPROXY_QUICKLEAVE_ON Mit Quickleave kann die Last im Upstream-Link gesenkt werden. Falls der Parameter mittels `'yes'` eingeschaltet wird, führt dies dazu, dass der Multicast nach einem Kanalwechsel schneller abbestellt und so die Last im Downstream gesenkt wird, indem sich der IGMP-Proxy wie ein Receiver verhält.

Gibt es 2 STBen und sehen diese dasselbe Programm, dann kann es (mit Quickleave = yes) passieren, dass beim Umschalten des Programms von einer STB bei der zweiten STB das Programm unterbrochen wird. Beim Einsatz von nur einer STB kann Quickleave gefahrlos eingeschaltet werden (yes).

```
IGMPPROXY_QUICKLEAVE_ON='yes'       # Quickleave-Modus einschalten
                                    # yes or no; Default: yes
```

IGMPPROXY_UPLOAD_DEV Für den IPTV-Betrieb benötigt der IGMP-Proxy ein Upstream- und ein Downstream-Interface. Das Upstream-Interface ist die Schnittstelle mit der NIC, an der das VDSL-Modem hängt. Diese sollte i.d.R. immer gleich bleiben.

Mit der Trennung von IPTV auf ID8 muss natürlich auch in der Konfiguration für den IGMP-Proxy eth1.8 statt bisher ppp0 eingetragen werden, womit die Umstellung Startznetz (nur ID7) auf das Zielnetz (mit ID7/8) komplett ist.

```
IGMPPROXY_UPLOAD_DEV='eth1.8'       # Upstream Interface; Default: ppp0
                                    # eth1.8 für T-Home/VDSL mit id7/id8
```

IGMPPROXY_DOWNLOAD_DEV Die Schnittstelle des Downstream-Interfaces (NIC zur IPTV SetTop-Box) ist hier abhängig von der HW-Konfiguration einzutragen. Für fli4l mit zweiter NIC – wie in diesem Dokument beschrieben - ist eth2 das Interface zur SetTop-Box.

```
IGMPPROXY_DOWNLOAD_DEV='eth2'       # Downstream Interface
```

IGMPPROXY_ALT_N Mit diesem Parameter wird die Anzahl der Adressbereiche für Multicast Streams festgelegt.

IGMPPROXY_ALT_NET_x Mit dem Parameter IGMPPROXY_ALT_NET werden Adressbereiche für Multicast-Traffic festgelegt, welche Ihren Ursprung außerhalb des Heim-Netzwerks haben, sowie der lokale Adressbereich, an dem die STB hängt.

```
IGMPPROXY_ALT_N='3'                         # Anzahl der Multicast Sourcen
IGMPPROXY_ALT_NET_1='239.35.0.0/16'         # IPTV streams - immer benoetigt
IGMPPROXY_ALT_NET_2='217.0.119.0/24'        # Erforderlich fuer T-Home
IGMPPROXY_ALT_NET_3='193.158.34.0/23'       # Erforderlich fuer T-Home
                                            # vor Mai 2013 '193.158.35.0/24'
# IGMPPROXY_ALT_NET_4='192.168.3.0/24'      # Adressraum IPTV SetTop-Box/nicht
                                            # mehr notwendig
```

IGMPPROXY_WLIST_N Mit diesem Parameter wir die Anzahl der Whitelists für die IGMP Reports festgelegt.

IGMPPROXY_WHLIST_NET_x :

Bei IGMPv3 können alle Adressen in einem Report zusammengefasst werden (http://grinch.itg-em.de/entertain/artikel/zielnetzarchitektur-und-igmpproxy/). Dieser wird dann komplett ignoriert, was dazu führt, dass der IGMP Querier irgendwann sämtlichen Multicasttraffic abschaltet, da er denkt, dieser würde nicht mehr benötigt. Um dies zu verhindern wurde die Konfiguration von whitelists erlaubt. Nur Multicastgruppen in dieser Liste werden dann auch auf WAN-Seite angefordert.

```
IGMPPROXY_WLIST_N='1'                       # Anzahl der Multicast Sourcen
IGMPPROXY_WHLIST_NET_1='239.35.0.0/16'      # IPTV streams - immer benoetigt
                                            # siehe oben
```

Änderungen in anderen Config-Dateien

Seit Revision 32955 ist es nicht mehr notwendig, die Firewall für den IGMP Proxy und die Multicast Streams anzupassen, wenn in der base.txt die Standardregeln für die Firewall aktiviert sind (`PF_INPUT_ACCEPT_DEF='yes'` und `PF_FORWARD_ACCEPT_DEF='yes'`). Das Startskript fügt dann automatisch die notwendigen Regeln ein, wenn `OPT_IGMPPROXY='yes'` gesetzt ist.

Im Detail handelt es sich um zwei Regeln, die in der INPUT Kette eingetragen werden, damit die eingehenden Nachrichten den IGMP Proxy erreichen:

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target   prot opt in   out   source         destination
 [...]
    0     0 ACCEPT   all  --  *    *     0.0.0.0/0      224.0.0.1     \
    /* automatically added for IGMP Proxy */

    0     0 ACCEPT   all  --  *    *     0.0.0.0/0      224.0.0.22    \
    /* automatically added for IGMP Proxy */
 [...]
```

Und außerdem eine Regel in der FORWARD Kette, die erlaubt, dass die eingehenden Multicast Streams auch zum Media Receiver weitergeleitet werden:

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target   prot opt in   out   source         destination
 [...]
    0     0 ACCEPT   all  --  *    *     0.0.0.0/0      239.35.0.0/16  \
    /* automatically added for IPTV streams */
 [..]
```

Sofern man die Standardregeln nicht aktiviert sind, müssen manuell mindestens die folgenden Regeln eingefügt werden.

```
PF_INPUT_x='any 224.0.0.1/32 ACCEPT'
PF_INPUT_x='any 224.0.0.22/32 ACCEPT'
[...]
PF_FORWARD_x='any 239.35.0.0/16 ACCEPT'
```

Hinweis: Gegenüber früheren Versionen der Dokumentation wurden die Regeln auf die tatsächlich notwendigen Netze eingeschränkt. Sofern das IPTV nicht laufen sollte, nehmen wir gerne Hinweise auf zusätzlich notwendige Netze entgegen.

WICHTIG! Ab Ende Mai 2013 hat die Telekom für Entertain neue (classless) Routen eingeführt (http://www.onlinekosten.de/forum/showthread.php?t=116415&page=38). Der Grund dafür scheint wohl zu sein, das jetzt mehr als 256 Sender bzw. Adressen benutzt werden. Damit sendet der DHCP-Server dhcp-routen, die nicht mehr im bisherigen Subnetz enthalten sind. Solange die Telekom nicht ihr iptv-server-subnetz (193.158.34.0/23) wechselt, kann eine statische route fuer das vlan8-interface anlegt werden, um diese Änderung zu berücksichtigen, da sonst multicast nicht mehr funktioniert.

Lösung: In base.txt muss ein zusätzliche Route angegeben werden.

```
IP_ROUTE_N='1'
IP_ROUTE_1='193.158.34.0/23 eth1.8'
```

4.17.7. OPT_STUNNEL - Tunneln von Verbindungen über SSL/TLS

Das Programm "stunnel" erlaubt es, ansonsten unverschlüsselte Verbindungen in einem verschlüsselten SSL/TLS-Tunnel zu kapseln. Dies ermöglicht sicheren Datenaustausch über sonst unsichere Klartext-Protokolle. Auf Grund der Möglichkeiten des SSL/TLS-Protokolls sind verschiedene Formen der Client/Server-Validierung möglich.

Konfiguration

OPT_STUNNEL Diese Variable aktiviert die Unterstützung für SSL/TLS-Tunnel.

Standard-Einstellung: `OPT_STUNNEL='no'`

Beispiel: `OPT_STUNNEL='yes'`

STUNNEL_DEBUG Mit dieser Variable kann eingestellt werden, wie sehr "stunnel" seine Arbeitsweise protokolliert. Mögliche Einstellungen umfassen "yes" (alles wird protokolliert), "no" (Warnungen und Fehler werden protokolliert) oder ein Wert zwischen null und sieben, der die maximale Schwere der zu protokollierenden Meldungen angibt, wobei null für allerdringendste Meldungen und sieben für Debug-Meldungen steht. Die Einstellung "yes" entspricht der maximalen Schwere sieben, die Einstellung "no" entspricht der maximalen Schwere vier.

Standard-Einstellung: `STUNNEL_DEBUG='no'`

Beispiel 1: `STUNNEL_DEBUG='yes'`

Beispiel 2: `STUNNEL_DEBUG='5'`

STUNNEL_N Diese Variable konfiguriert die Anzahl der Tunnel-Instanzen. Jede Tunnel-Instanz "horcht" auf einem Netzwerkport "A", verbindet sich bei einer eingehenden Verbindung mit einem anderen Netzwerkport "B" (der durchaus auch auf einer ganz anderen Maschine liegen kann) und leitet jeglichen Datenverkehr von "A" nach "B" weiter. Ob die Daten, die bei "A" ankommen, via SSL/TLS verschlüsselt sind, von "stunnel" entschlüsselt und dann nach "B" unverschlüsselt weitergeleitet werden oder umgekehrt, entscheidet sich durch die Variable `STUNNEL_x_CLIENT` (Seite 215).

Standard-Einstellung: `STUNNEL_N='0'`

Beispiel: `STUNNEL_N='2'`

STUNNEL_x_NAME Der Name des jeweiligen Tunnels. Er muss unter allen konfigurierten Tunneln eindeutig sein.

Beispiel: `STUNNEL_1_NAME='imond'`

STUNNEL_x_CLIENT Über diese Variable kann eingestellt werden, welche Teile der Kommunikation via SSL/TLS verschlüsselt werden. Es gibt zwei Möglichkeiten:

- *Client-Modus:* Der Tunnel erwartet von außen unverschlüsselte Daten und schickt diese verschlüsselt an das andere Ende des Tunnels. Dies entspricht der Einstellung `STUNNEL_x_CLIENT='yes'`.
- *Server-Modus:* Der Tunnel erwartet von außen via SSL/TLS verschlüsselte Daten und schickt diese entschlüsselt an das andere Ende des Tunnels. Dies entspricht der Einstellung `STUNNEL_x_CLIENT='no'`.

Tunnel im Client-Modus eignen sich also vor allem für Verbindungen, die "nach außen", also z. B. ins (ungeschützte) Internet gehen, da die Daten vor dem Verlassen des lokalen Netzwerks verschlüsselt werden. Dafür muss die Gegenstelle aber natürlich auch einen Server anbieten, der via SSL/TLS verschlüsselte Daten erwartet. Beispielsweise kann so ein E-Mail-Client im LAN, der nur unverschlüsseltes POP3 "spricht", einen POP3-over-SSL-Dienst im Internet nutzen.[15]

Tunnel im Server-Modus eignen sich umgekehrt für Verbindungen, die "von außen", also z. B. aus dem (ungeschützten) Internet kommen, bei denen die Daten verschlüsselt ankommen. Wenn der eigentliche Dienst auf Server-Seite jedoch kein SSL/TLS versteht, müssen die Daten vorher entsprechend entschlüsselt werden. Beispielsweise kann so der Zugriff auf die Web-GUI des fli4l über via SSL/TLS verschlüsseltes HTTP (HTTPS) erfolgen, indem man auf dem fli4l einen Tunnel konfiguriert, der via SSL/TLS verschlüsselte HTTP-Daten auf Port 443 empfängt, diese entschlüsselt und dann an den lokalen Web-Server `mini_httpd`, der auf Port 80 horcht, weiterleitet.

Die Konfigurationen für diese Anwendungsfälle werden weiter hinten vorgestellt.

Beispiel: `STUNNEL_1_CLIENT='yes'`

STUNNEL_x_ACCEPT Hiermit wird festgelegt, auf welchem Port (und an welcher Adresse) der Tunnel auf eingehende Verbindungen "lauscht". Es gibt prinzipiell zwei Möglichkeiten:

- Der Tunnel soll an *allen* Adressen (auf allen Schnittstellen) lauschen. Hierfür muss "any" verwendet werden.
- Der Tunnel soll nur an bestimmten Adressen lauschen. Dies wird mit Hilfe einer entsprechenden Referenz auf das konfigurierte IP-Subnetz eingestellt, beispielsweise `IP_NET_1_IPADDR` (für IPv4) oder `IPV6_NET_2_IPADDR` (für IPv6).

Des Weiteren *muss* hinter die Adresse der Port stehen, wobei der Port von der Adresse mit Hilfe eines Doppelpunktes (":") abgetrennt ist.

Beispiel 1: `STUNNEL_1_ACCEPT='any:443'`

Beispiel 2: `STUNNEL_1_ACCEPT='IP_NET_1_IPADDR:443'`

Beispiel 3: `STUNNEL_1_ACCEPT='IPV6_NET_2_IPADDR:443'`

Zu bedenken ist, dass die Verwendung von `IP_NET_x_IPADDR` bzw. `IPV6_NET_x_IPADDR` das Layer-3-Protokoll (IPv4 oder IPv6) festlegt; diese Wahl *muss* zu den Belegungen der Variablen `STUNNEL_x_ACCEPT_IPV4` und `STUNNEL_x_ACCEPT_IPV6` passen. Sie können also nicht IPv6 für den Tunnel mit Hilfe von `STUNNEL_1_ACCEPT_IPV6='no'` deaktivieren und dann mit Hilfe von `STUNNEL_1_ACCEPT='IPV6_NET_2_IPADDR:443'` an einer IPv6-Adresse lauschen; dies gilt analog für die umgekehrte Konstellation (`STUNNEL_1_ACCEPT_IPV4='no'` und die Verwendung von `IP_NET_x_IPADDR`). Des Weiteren hängt die Bedeutung von "any" von den aktivierten Layer-3-Protokollen (IPv4 oder IPv6) ab: Es wird natürlich nur auf Adressen gelauscht, die zu den via `STUNNEL_x_ACCEPT_IPV4` und `STUNNEL_x_ACCEPT_IPV6` aktivierten Layer-3-Protokollen gehören.

STUNNEL_x_ACCEPT_IPV4 Mit dieser Variable wird eingestellt, ob das IPv4-Protokoll für *eingehende* Verbindungen des Tunnels genutzt werden soll. Typischerweise ist dies

[15] vgl. http://de.wikipedia.org/wiki/POP3S

der Fall, und diese Variable sollte den Wert "yes" enthalten. Die Belegung mit "no" stellt sicher, dass der Tunnel nur eingehende IPv6-Verbindungen akzeptiert. Dies erfordert jedoch eine valide IPv6-Konfiguration (siehe hierzu die Dokumentation des ipv6-Pakets).

Standard-Einstellung: `STUNNEL_x_ACCEPT_IPV4='yes'`

Beispiel: `STUNNEL_1_ACCEPT_IPV4='no'`

STUNNEL_x_ACCEPT_IPV6 Analog zu `STUNNEL_x_ACCEPT_IPV4` wird mit dieser Variable eingestellt, ob das IPv6-Protokoll für eingehende Verbindungen des Tunnels genutzt werden soll. Typischerweise ist das der Fall, wenn Sie die generelle Nutzung des IPv6-Protokolls mit Hilfe von `OPT_IPV6='yes'` aktiviert haben. Die Belegung mit "no" stellt sicher, dass der Tunnel nur eingehende IPv4-Verbindungen akzeptiert.

Standard-Einstellung: `STUNNEL_x_ACCEPT_IPV6=<Wert von OPT_IPV6>`

Beispiel: `STUNNEL_1_ACCEPT_IPV6='no'`

STUNNEL_x_CONNECT Hiermit wird festgelegt, welches Ziel der SSL/TLS-Tunnel hat. Es gibt prinzipiell drei Möglichkeiten, wobei bei jeder der drei Möglichkeiten noch ein mit ":" abgetrennter Port angehängt werden muss:

- Eine numerische IPv4- oder IPv6-Adresse.

  Beispiel 1: `STUNNEL_1_CONNECT='192.0.2.2:443'`
- Der DNS-Name von einem internen Host.

  Beispiel 2: `STUNNEL_1_CONNECT='@webserver:443'`
- Der DNS-Name von einem externen Host.

  Beispiel 3: `STUNNEL_1_CONNECT='@www.example.com:443'`

Wird ein interner Host eingetragen, der sowohl eine IPv4- als auch eine IPv6-Adresse besitzt, dann wird die IPv4-Adresse bevorzugt. Wird ein externer Host eingetragen, der sowohl eine IPv4- als auch eine IPv6-Adresse besitzt, dann hängt das verwendete Layer-3-Protokoll davon ab, welche Adresse als erstes vom DNS-Resolver zurückgegeben wird.

STUNNEL_x_OUTGOING_IP Mit dieser optionalen Variable kann die *lokale* Adresse für die *ausgehende* Verbindung des Tunnels angegeben werden. Dies ist nur dann sinnvoll, wenn das Ziel des Tunnels über mehrere Schnittstellen (Routen) erreicht werden kann, also wenn man z. B. zwei Internet-Anbindungen nutzt. Normalerweise muss diese Variable nicht gesetzt werden.

Beispiel: `STUNNEL_1_OUTGOING_IP='IP_NET_1_IPADDR'`

STUNNEL_x_DELAY_DNS Wird diese optionale Variable auf "yes" gesetzt, so wird ein in `STUNNEL_x_CONNECT` verwendeter externer DNS-Name erst dann in eine Adresse umgewandelt, wenn die *ausgehende* Tunnelverbindung aufgebaut wird, also wenn der erste Client sich lokal mit der eingehenden Seite des Tunnels verbunden hat. Dies ist dann nützlich, wenn das Ziel des Tunnels ein Rechner ist, der nur über einen dynamischen DNS-Namen erreicht werden kann und die Adresse hinter diesem Namen häufiger wechselt, oder auch wenn eine aktive Einwahl bereits beim Starten von "stunnel" verhindert werden soll.

Standard-Einstellung: `STUNNEL_x_DELAY_DNS='no'`

Beispiel: `STUNNEL_1_DELAY_DNS='yes'`

STUNNEL_x_CERT_FILE Diese Variable enthält den Dateinamen des Zertifikats, das für den Tunnel verwendet werden soll. Für Server-Tunnel (`STUNNEL_x_CLIENT='no'`) ist dies das Server-Zertifikat, das vom Client ggfs. gegen eine "Certificate Authority" (CA) validiert wird. Für Client-Tunnel (`STUNNEL_x_CLIENT='yes'`) ist dies ein (in der Regel optionales) Client-Zertifikat, das vom Server ggfs. gegen eine CA validiert wird.

Das Zertifikat muss im so genannten PEM-Format vorliegen und muss unterhalb von `<config-Verzeichnis>/etc/stunnel/` abgespeichert werden. Nur der Dateiname muss in dieser Variable gespeichert werden, nicht der Pfad.

Für einen Server-Tunnel ist ein Zertifikat zwingend erforderlich!

Beispiel: `STUNNEL_1_CERT_FILE='myserver.crt'`

STUNNEL_x_CERT_CA_FILE Diese Variable enthält den Dateinamen des CA-Zertifikats, das für die Validierung des Zertifikats der Gegenstelle verwendet werden soll. Typischerweise validieren Clients das Zertifikat des Servers, andersherum ist dies jedoch genauso möglich. Einzelheiten zur Validierung lesen Sie bitte in der Beschreibung der Variable `STUNNEL_x_CERT_VERIFY` (Seite 218) nach.

Das Zertifikat muss im so genannten PEM-Format vorliegen und muss unterhalb von `<config-Verzeichnis>/etc/stunnel/` abgespeichert werden. Nur der Dateiname muss in dieser Variable gespeichert werden, nicht der Pfad.

Beispiel: `STUNNEL_1_CERT_CA_FILE='myca.crt'`

STUNNEL_x_CERT_VERIFY Diese Variable steuert die Validierung des Zertifikats der Gegenstelle. Es gibt fünf Möglichkeiten:

- *none:* Das Zertifikat der Gegenstelle wird überhaupt nicht valididert. In diesem Falle kann die Variable `STUNNEL_x_CERT_CA_FILE` leer bleiben.
- *optional:* Stellt die Gegenstelle ein Zertifikat zur Verfügung, so wird es gegen das CA-Zertifikat geprüft, das mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` konfiguriert wird. Stellt die Gegenstelle *kein* Zertifikat zur Verfügung, so ist dies kein Fehler und die Verbindung wird dennoch akzeptiert. Diese Einstellung ist nur sinnvoll für Server-Tunnel, weil Client-Tunnel *immer* ein Zertifikat vom Server erhalten.
- *onlyca:* Das Zertifikat der Gegenstelle wird gegen das CA-Zertifikat geprüft, das mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` konfiguriert wird. Sendet die Gegenstelle kein Zertifikat oder passt es nicht zur konfigurierten CA, wird die Verbindung abgewiesen. Dies ist nützlich, wenn eine eigene CA verwendet wird, da man dann alle potentiellen Gegenstellen kennt.
- *onlycert:* Das Zertifikat der Gegenstelle wird mit dem Zertifikat verglichen, das mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` konfiguriert wird. Es wird *nicht* gegen ein CA-Zertifikat geprüft, sondern es wird sichergestellt, dass die Gegenstelle *genau* das passende (Server- oder Client-)Zertifikat sendet. Die Datei, die mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` referenziert wird, enthält in diesem Fall also kein CA-, sondern ein Host-Zertifikat. Diese Einstellung stellt sicher, dass sich wirklich nur eine bestimmte und bekannte Gegenstelle verbinden darf (Server-Tunnel) bzw. eine Verbindung nur zu einer bekannten Gegenstelle (Client-Tunnel) aufgebaut wird. Dies ist für Peer-to-Peer-Verbindungen zwischen Hosts nützlich, die man beide unter Kontrolle hat, für die man aber keine eigene CA verwendet.

- *both:* Das Zertifikat der Gegenstelle wird mit dem Zertifikat verglichen, das mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` konfiguriert wird, *und* es wird zusätzlich sichergestellt, dass es zu einem CA-Zertifikat passt. Die Datei, die mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` referenziert wird, enthält in diesem Fall also *sowohl* ein CA- *als auch* ein Host-Zertifikat. Es handelt sich also um eine Kombination der Einstellungen *onlycert* und *onlyca*. Im Vergleich zur Einstellung *onlycert* werden somit Verbindungen abgelehnt, falls das CA-Zertifikat abgelaufen ist (auch wenn das Zertifikat der Gegenstelle ansonsten passt).

Standard-Einstellung: `STUNNEL_x_CERT_VERIFY='none'`

Beispiel: `STUNNEL_1_CERT_VERIFY='onlyca'`

Anwendungsbeispiel 1: Zugang zur fli4l-WebGUI via SSL/TLS

Mit diesem Beispiel wird die fli4l-WebGUI um einen SSL/TLS-Zugang erweitert.

```
OPT_STUNNEL='yes'
STUNNEL_N='1'

STUNNEL_1_NAME='http'
STUNNEL_1_CLIENT='no'
STUNNEL_1_ACCEPT='any:443'
STUNNEL_1_ACCEPT_IPV4='yes'
STUNNEL_1_ACCEPT_IPV6='yes'
STUNNEL_1_CONNECT='127.0.0.1:80'
STUNNEL_1_CERT_FILE='server.pem'
STUNNEL_1_CERT_CA_FILE='ca.pem'
STUNNEL_1_CERT_VERIFY='none'
```

Anwendungsbeispiel 2: Via SSL/TLS gesicherte Kontrolle von zwei entfernten fli4l-Routern via imonc

Mit diesem Beispiel werden die bekannten Schwachstellen des imonc/imond-Protokolls (Senden von Passwörtern im Klartext) für WAN-Verbindungen umgangen. (Die LAN-Verbindung zum Tunnel kann natürlich weiterhin abgehört werden!)

Konfiguration des lokalen fli4l im LAN (Client-Tunnel):

```
OPT_STUNNEL='yes'
STUNNEL_N='2'

STUNNEL_1_NAME='remote-imond1'
STUNNEL_1_CLIENT='yes'
STUNNEL_1_ACCEPT='any:50000'
STUNNEL_1_ACCEPT_IPV4='yes'
STUNNEL_1_ACCEPT_IPV6='yes'
STUNNEL_1_CONNECT='@remote1:50000'
STUNNEL_1_CERT_FILE='client.pem'
STUNNEL_1_CERT_CA_FILE='ca+server1.pem'
STUNNEL_1_CERT_VERIFY='both'

STUNNEL_2_NAME='remote-imond2'
```

```
STUNNEL_2_CLIENT='yes'
STUNNEL_2_ACCEPT='any:50001'
STUNNEL_2_ACCEPT_IPV4='yes'
STUNNEL_2_ACCEPT_IPV6='yes'
STUNNEL_2_CONNECT='@remote2:50000'
STUNNEL_2_CERT_FILE='client.pem'
STUNNEL_2_CERT_CA_FILE='ca+server2.pem'
STUNNEL_2_CERT_VERIFY='both'
```

Konfiguration des ersten entfernten fli4l (Server-Tunnel):

```
OPT_STUNNEL='yes'
STUNNEL_N='1'

STUNNEL_1_NAME='remote-imond'
STUNNEL_1_CLIENT='no'
STUNNEL_1_ACCEPT='any:50000'
STUNNEL_1_ACCEPT_IPV4='yes'
STUNNEL_1_ACCEPT_IPV6='yes'
STUNNEL_1_CONNECT='127.0.0.1:5000'
STUNNEL_1_CERT_FILE='server1.pem'
STUNNEL_1_CERT_CA_FILE='ca+client.pem'
STUNNEL_1_CERT_VERIFY='both'
```

Konfiguration des zweiten entfernten fli4l (Server-Tunnel):

```
OPT_STUNNEL='yes'
STUNNEL_N='1'

STUNNEL_1_NAME='remote-imond'
STUNNEL_1_CLIENT='no'
STUNNEL_1_ACCEPT='any:50000'
STUNNEL_1_ACCEPT_IPV4='yes'
STUNNEL_1_ACCEPT_IPV6='yes'
STUNNEL_1_CONNECT='127.0.0.1:5000'
STUNNEL_1_CERT_FILE='server2.pem'
STUNNEL_1_CERT_CA_FILE='ca+client.pem'
STUNNEL_1_CERT_VERIFY='both'
```

Eine Verbindung zu dem entfernten “imond” wird aufgebaut, indem eine Verbindung zum lokalen fli4l auf Port 50000 (erster entfernter fli4l) bzw. 50001 (zweiter entfernter fli4l) initiiert wird. Dieser fli4l verbindet sich dann via SSL/TLS-Tunnel mit dem jeweiligen entfernten fli4l, der wiederum die Daten über eine dritte (Host-interne) Verbindung letztlich an den entfernten “imond” weiterleitet. Die Einstellungen für die Validierung stellen sicher, dass jeder fli4l jeweils nur den anderen fli4l als Verbindungspartner akzeptiert.

4.18. QoS - Quality of Service

Mit QoS kann man die verfügbare Bandbreite regulieren und zum Beispiel auf verschiedene Ports, IP’s und noch einiges mehr zu verteilen.

Ein Modem verwaltet eine Packet-Queue (Queue = Schlange, Reihe (in einer Schlange stehen)) in der Pakete gespeichert werden, die die verfügbare Bandbreite überschreiten. Bei DSL-Modems zum Beispiel, sind diese sehr groß. Das hat den Vorteil, dass recht gleichmäßig die

maximale Bandbreite ausgenutzt werden kann. Denn schickt der Router an das Modem für kurze Zeit (sehr kurz) weniger Pakete, dann hat das Modem noch immer Pakete in der Queue, die es abzuarbeiten gilt. So eine Queue ist sehr simpel gehalten, denn dort geht alles der Reihe nach, ist eben ein faires Modem :-D

Und hier kommt dann QoS ins Spiel. QoS verwaltet auch eine Packet-Queue, nur eben im Router selber und dort hat man die Möglichkeit König zu sein, eben zu entscheiden welches Paket zu erst darf und welche Pakete sich noch ein bischen zurückhalten müssen. Wenn alles richtig konfiguriert wurde, dann sendet QoS die Pakete gerade eben so schnell, dass sie nicht in die Packet-Queue des Modems landen. Das wäre so, als hätte man die Queue vom Modem in den Router geholt.

Noch etwas allgemeines zu den Geschwindigkeitseinheiten: QoS unterstützt Mibit/s (mebibit/s) und Kibit/s (kibibit/s), wobei gilt 1Mibit = 1024Kibit.

4.18.1. Konfiguration

OPT_QOS Hier ist yes zusetzen wenn man das `OPT_QOS` einsetzen will und no wenn man das Gegenteil beabsichtigt

QOS_INTERNET_DEV_N Die Anzahl der Devices, die Daten ins Internet routen.

QOS_INTERNET_DEV_x Hier sollte die Liste der Devices eingetragen werden, über die Daten ins Internet übertragen werden. Beispiele:

`QOS_INTERNET_DEV_N='3'`	Anzahl der Geräte
`QOS_INTERNET_DEV_1='ethX'`	für Kabel und sonstige Ethernet-Verbindungen
`QOS_INTERNET_DEV_2='ppp0'`	für DSL über PPPoE
`QOS_INTERNET_DEV_3='ipppX'`	für ISDN

Das ISDN-Device für den ersten Circuit dürfte das ippp0 lauten, für den 2. ippp1. Wenn aber für den ersten Circuit Kanalbündelung aktiviert wurde, dann heißt der 2. Kanal des 1. Circuit ippp1 und der 2. Circuit ippp2. Man sollte QOS mit ISDN nur dann nutzen, wenn Kanalbündelung für den benutzten Circuit deaktiviert ist.

QOS_INTERNET_BAND_DOWN Maximale Downstreambandbreite des Internetzugangs. Siehe weiter oben: Hinweis zu den Geschwindigkeitseinheiten (Seite 221).

Hinweis: Für zeitkritische Aufgaben, wie das bevorzugen von ACK-Paketen, ist es nötig die Bandbreite nicht höher zu setzen als wirklich vorhanden, da man sonst zwar innerhalb der Packet-Queue auf dem Router die Pakete sortiert, dies dann aber nicht ganz korrekt gemacht wird und letztendlich doch wieder in der Packet-Queue des Modems aufgehalten werden. Möglich ist es außerdem, das die vom Provider angegebene Bandbreite nicht hundertprozentig mit der wirklich verfügbaren übereinstimmt, es könnte ein bischen mehr oder auch weniger sein. Da ist also ausprobieren angesagt.

QOS_INTERNET_BAND_UP Maximale Upstreambandbreite des Internet-Zugangs. Siehe Hinweis zu den Geschwindigkeitseinheiten unter `OPT_QOS`.

Hinweis: Siehe Hinweis bei `QOS_INTERNET_BAND_DOWN`.

QOS_INTERNET_DEFAULT_DOWN Hier ist die Standardklasse für Pakete anzugeben, die aus dem Internet kommen. Alle Pakete, die nicht durch einen Filter in eine Klasse gesteckt wurden, landen dann in der angegebenen Klasse.

Wurde keine Klasse eingerichtet, für die die Variable

```
QOS_CLASS_x_DIRECTION='down'
```

gesetzt wurde, so setzt man:

```
QOS_INTERNET_DEFAULT_DOWN='0'
```

Beispiel:

Es wurden 2 Klassen eingerichtet und ein Filter steckt alle Pakete, die z.B. an eine bestimmte IP-Adresse geschickt wurden in die 1. von den beiden. Alle anderen Pakete sollen in die 2. Klasse gesteckt werden. Folglich müßte hier

```
QOS_INTERNET_DEFAULT_DOWN='2'
```

eingetragen werden.

Es ist darauf zu achten, dass für `QOS_INTERNET_DEFAULT_DOWN` eine Klasse angegeben wird, deren `QOS_CLASS_x_DIRECTION` Variable das Argument down enthält.

QOS_INTERNET_DEFAULT_UP Hier ist die Standardklasse für Pakete anzugeben, die in das Internet gehen. Alle Pakete, die nicht durch einen Filter in eine Klasse gesteckt wurden, landen dann in der angegebenen Klasse.

Wurde keine Klasse eingerichtet, für die die Variable

```
QOS_CLASS_x_DIRECTION='up'
```

gesetzt wurde, so setzt man:

```
QOS_INTERNET_DEFAULT_UP='0'
```

Das ganze funktioniert analog zu `QOS_INTERNET_DEFAULT_DOWN`.

Es ist darauf zu achten, dass für `QOS_INTERNET_DEFAULT_UP` eine Klasse angegeben wird, deren `QOS_CLASS_x_DIRECTION` Variable das Argument up enthält.

QOS_CLASS_N Hier ist die gewünschte Anzahl der Klassen (engl. Class) anzugeben.

QOS_CLASS_x_PARENT Mit dieser Variable kann man Klassen verschachteln. Man gibt hier immer die Nummer der Vaterklasse an. Die Bandbreite die der Vaterklasse zugeteilt wurde, kann dann unter den Unterklassen weiter aufgeteilt werden. Die maximale Verschachtelungstiefe beträgt hier 8 Ebenen, wobei das Interface selber schon eine Ebene darstellt, es bleiben also maximal 7 konfigurierbar.

Soll die Klasse keine Unterklasse sein, so gibt man hier folgendes an:

```
QOS_CLASS_x_PARENT='0'
```

Ihr wird dann je nachdem zu welcher Richtung sie gehört (siehe QOS_CLASS_x_PORT_TYPE), maximal die in QOS_CLASS_x_PORT_TYPE oder QOS_INTERNET_BAND_DOWN angegebene Bandbreite zugeteilt.

Wichtig: Falls hier nicht '0' angegeben wird, so ist darauf zu achten, dass die Vaterklasse vorher definiert wird (auf die Nummerierung bezogen).

QOS_CLASS_x_MINBANDWIDTH Bandbreite, die man der Klasse zusprechen will. Man könnte hier auch von einem Verhältnis sprechen. Siehe Hinweis zu den Geschwindigkeitseinheiten unter OPT_QOS.

Beispiel: Man hat eine Klasse, dessen Bandbreite auf 128Kibit/s beschränkt ist.:

```
QOS_CLASS_1_MINBANDWIDTH='128Kibit/s'
QOS_CLASS_1_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_1_PARENT='0'
```

Weiterhin hat man 3 Klassen dessen QOS_CLASS_x_MINBANDWIDTH- und QOS_CLASS_x_MAXBANDWIDTH-Einstellungen wie folgt aussehen und alle Unterklassen unserer ersten Klasse sind:

```
QOS_CLASS_2_PARENT='1'
QOS_CLASS_2_MINBANDWIDTH='60Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='128Kibit/s'

QOS_CLASS_3_PARENT='1'
QOS_CLASS_3_MINBANDWIDTH='40Kibit/s'
QOS_CLASS_3_MAXBANDWIDTH='128Kibit/s'

QOS_CLASS_4_PARENT='1'
QOS_CLASS_4_MINBANDWIDTH='28Kibit/s'
QOS_CLASS_4_MAXBANDWIDTH='128Kibit/s'
```

Alle Unterklassen besitzen die selbe (oder auch keine) Priorität (siehe QOS_CLASS_x_PRIO). Wird nun auf jede dieser 3 Klassen mehr Verkehr produziert als in ihrer jeweiligen QOS_CLASS_x_MINBANDWIDTH angegeben, so bekommt jede Klasse entsprechend ihrer QOS_CLASS_x_MINBANDWIDTH-Einstellung Bandbreite zugewiesen. Wenn aber z.B. Klasse 2 nur 20Kibit/s an Verkehr produziert, dann läßt dies Klasse ja 40Kibit/s "übrig". Dieser Überschuß wird im Verhältnis 40/28 unter Klasse 3 und 4 aufgeteilt. Jede Klasse selber ist durch QOS_CLASS_x_MAXBANDWIDTH auf 128Kibit/s beschränkt und da sie alle Unterklassen einer auf 128Kibit/s beschränkten Klasse sind, können sie auch alle zusammen nicht mehr als 128Kibit/s konsumieren.

QOS_CLASS_x_MAXBANDWIDTH Bandbreite, die man der Klasse maximal zuteilen will. Es macht keinen Sinn einen niedrigeren Wert als der in QOS_CLASS_x_MINBANDWIDTH einzutragen. Gibt man hier nichts an, so nimmt diese Variable automatisch den Wert von QOS_CLASS_x_MINBANDWIDTH an. Eine solche Klasse kann dann natürlich keine überschüssige Bandbreite beanspruchen.

Siehe Hinweis zu den Geschwindigkeitseinheiten unter OPT_QOS.

QOS_CLASS_x_DIRECTION Mit dieser Variable wird angegeben, zu welcher Richtung die Klasse gehört. Soll sie zur Regulierung des Upstreams benutzt werden, so ist hier

```
QOS_CLASS_x_DIRECTION='up'
```

anzugeben, für den Downstream analog:

```
QOS_CLASS_x_DIRECTION='down'
```

QOS_CLASS_x_PRIO Hier wird geregelt, welche Priorität ein Klasse hat. Je niedriger die Nummer, desto höher die Priorität. Erlaubt sind werte zwischen 0 und 7. Wenn die Variable leer gelassen wird, so kommt das dem setzen einer 0 gleich.

Wenn eine Priorität gesetzt wird, dann wird darüber bestimmt, welcher Klasse zuerst Überschüssige Bandbreite angeboten wird. Um das klar zu machen, ändern wir das Beispiel aus `QOS_CLASS_x_MINIMUMBANDWIDTH` leicht ab: An der ersten Klasse wird nichts verändert. Die Klassen 2-4 bekommen eine Priorität zugewiesen:

```
QOS_CLASS_2_PARENT='1'
QOS_CLASS_2_MINBANDWIDTH='60Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_2_PRIO='1'

QOS_CLASS_3_MINBANDWIDTH='40Kibit/s'
QOS_CLASS_3_PARENT='1'
QOS_CLASS_3_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_3_PRIO='1'

QOS_CLASS_4_PARENT='1'
QOS_CLASS_4_MINBANDWIDTH='28Kibit/s'
QOS_CLASS_4_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_4_PRIO='2'
```

Wie in dem Ursprungsbeispiel konsumiert Klasse 2 nur 20Kibit/s und läßt somit einen Überschuß von 40Kibit/s übrig. Klasse 3 und 4 wollen noch immer mehr Bandbreite als überhaupt verfügbar. Da nun aber Klasse 3 eine höhere Priorität als Klasse 4 hat, darf sie den Überschuß von 40Kibit/s vertilgen.

Angenommen Klasse 3 braucht aber nur 20Kibit/s des ursprünglichen Überschusses von 40Kibit/s, dann bekommt Klasse 4 die restlichen 20Kibit/s.

Nehmen wir nochmals etwas anderes an: Klasse 4 verbraucht gar keine Bandbreite und Klasse 2 und 3 wollen mehr als es überhaupt gibt. Dann bekommt jede erstmal ihre in `QOS_CLASS_x_MINBANDWIDTH` angegebene Bandbreite und der Rest wird unter ihnen im 60/40 Verhältnis aufgeteilt, da beide Klassen die selbe Priorität haben.

Wie man also sieht beeinflußt `QOS_CLASS_x_PRIO` nur, wie ein eventueller Bandbreitenüberschuß aufgeteilt wird.

QOS_CLASS_x_LABEL Mit dieser optionalen Variable kann ein Label für die Klasse gesetzt werden. Dieses wird bei aktivem OPT_RRDTOOL zur Beschriftung der Graphen von QOS genutzt.

QOS_FILTER_N Gewünschte Anzahl der der Filter angeben.

Zu den Filtern allgemein läßt sich noch folgendes sagen: Die Argumente von verschiedenen Variablen sind UND-verknüpft, mehrere Argumente der selben Variable sind ODER-verknüpft. Soll heißen: Wird zum Beispiel in einem und dem selben Filter nach einer IP-Adresse und einem Port gefiltert, so werden nur Pakete herausgefiltert und in die Zielklasse(n) gesteckt, die auf beides gleichzeitig zutreffen.

Ein weiteres Beispiel: In einem und dem selben Filter sind zwei Ports (21 und 80) und eine IP-Adresse angegeben. Ein Datenpaket kann natürlich nicht von zwei Ports gleichzeitig kommen. Es verhält sich dann so: Der Filter filtert Pakete heraus, die entweder Port 21 benutzen und gleichzeitig die IP-Adresse, oder von Port 80 kommen und gleichzeitig von der IP-Adresse.

Wichtig: Es kommt auf die Reihenfolge der Filter an!

Ein Beispiel: Man möchte den Verkehr, der über den Port 456 läuft, für **alle** Rechner in Klasse A stopfen. Zusätzlich möchte man alle Pakete an den Rechner mit der IP 192.168.6.5 - bis auf die Pakete über Port 456 - in Klasse B laufen lassen. Richte ich nun den Filter mit der IP als erstes ein, dann landen alle Pakete - auch die über Port 456 laufen - in Klasse B und ein nachfolgender Filter für den Port 456 ändert auch nichts daran. Der Filter für den Port 456 muß also noch vor dem Filter mit der IP 192.168.6.5 stehen.

QOS_FILTER_x_CLASS Mit dieser Variable stellt ihr ein, in welche Klasse das Paket, auf den dieser Filter zutrifft, gesteckt werden soll. Möchte man zum Beispiel die gefilterten Pakete in die Klasse, die mit den Variabeln `QOS_CLASS_25_MINBANDWIDTH` spezifiziert wurde, stecken, so müßte man hier

```
QOS_FILTER_x_CLASS='25'
```

setzen.

Mit `QOS_CLASS_x_DIRECTION` gibt man ja an, ob eine Klasse nun zum Up- oder Downstream gehört. Wenn nun ein Filter gesetzt wird, der gefilterte Pakete zum Beispiel in eine Upstream-Klasse wandern läßt, dann werden auch nur Pakete aus dem Upstream von diesem Filter gefiltert und in die angegebene Klasse gesteckt. `QOS_CLASS_x_DIRECTION` bestimmt also in welcher "Richtung" gefiltert wird.

Seit Version 2.1 ist es nun auch möglich mehr als eine Zielklasse anzugeben. Möchte man zum Beispiel den Verkehr über Port 456 sowohl für den Upstream als auch für den Downstream klassifizieren, so würde man hier

```
QOS_FILTER_x_CLASS='4 25'
```

angeben, wobei zum Beispiel Klasse Nummer 4 die Upstreamklasse ist und 25 die Downstreamklasse. Es macht keinen Sinn hier jeweils mehr als ein Up- und Downstreamklassen anzugeben, somit wird man auch nie mehr als zwei Zielklassen eintragen.

QOS_FILTER_x_IP_INTERN Hier können IP-Adressen und IP-Bereiche aus den internen Netzwerkwerken, nach denen gefiltert werden soll, angegeben werden. Sie sind durch Leerzeichen zu trennen und können frei kombiniert werden.

Das könnte zum Beispiel so aussehen:

```
QOS_FILTER_x_IP_INTERN='192.168.6.0/24 192.168.5.7 192.168.5.12'
```

Hier werden alle Adressen in der Form 192.168.6.X gefiltert und zusätzlich noch die IPs 192.168.5.7 und 192.168.5.12.

Diese Variable darf auch leer gelassen werden.

Wird diese Variable gleichzeitig mit `QOS_FILTER_x_IP_EXTERN` genutzt, so wird nur Verkehr gefiltert, der zwischen den in `QOS_FILTER_x_IP_INTERN` und `QOS_FILTER_x_IP_EXTERN` angegebenen IPs oder IP-Bereichen stattfindet.

Wichtig: *Falls zusätzlich durch `QOS_FILTER_x_OPTION` nach ACK, TOSMD, TOSMT, TOSMR oder TOSMC gefiltert wird und die Variable `QOS_CLASS_x_DIRECTION` der Zielklasse 'down' entspricht, dann wird diese Variable ignoriert.*

QOS_FILTER_x_IP_EXTERN Hier können IP-Adressen und IP-Bereiche aus dem externen Netzwerke (welches über `QOS_INTERNET_DEV` angebunden ist), nach denen gefiltert werden soll, angegeben werden. Sie sind durch Leerzeichen zu trennen und können frei kombiniert werden. Das ganze funktioniert analog zu `QOS_FILTER_x_IP_INTERN`.

Diese Variable darf auch leer gelassen werden.

Wichtig: *Falls zusätzlich durch `QOS_FILTER_x_OPTION` nach ACK, TOSMD, TOSMT, TOSMR oder TOSMC gefiltert wird und die Variable `QOS_CLASS_x_DIRECTION` der Zielklasse 'down' entspricht, dann wird diese Variable ignoriert.*

QOS_FILTER_x_PORT Hier können Ports und Portranges angegeben werden, getrennt durch Leerzeichen und dürfen frei kombiniert werden. Falls die Variable leer ist, werden Übertragungen über sämtliche Ports limitiert.

Zu dem Format von Portranges: Möchte man nach Ports von 5000 bis 5099 filtern, so würde das folgender Maßen aussehen:

`QOS_FILTER_x_PORT`='5000-5099'

Ein weiteres Beispiel: Man möchte den Verkehr über die Ports 20 bis 21, 137 bis 139 und Port 80 filtern und in die selbe Klasse stecken lassen. Das sähe dann so aus:

```
QOS_FILTER_x_PORT='20-21 137-139 80'
```

Diese Variable darf auch leer gelassen werden.

Wichtig:

- Wenn nach Ports gefilter wird, muß auch `QOS_FILTER_x_PORT_TYPE` etsprechend gesetzt werden.
- Wenn zusätzlich durch `QOS_FILTER_x_OPTION` nach ACK, TOSMD, TOSMT, TOSMR oder TOSMC gefiltert wird, dann werden Portranges ignoriert.

QOS_FILTER_x_PORT_TYPE Das setzen dieser Variable ist nur wichtig im Zusammenhang mit `QOS_FILTER_x_PORT` und muß auch nur dann gesetzt werden (wird ansonsten ignoriert).

Da sich die Ports beim Clientbetrieb von den Ports beim Serverbetrieb unterscheiden, muss angegeben werden ob der Port des Servers oder Clients gemeint ist. Als Bezugspunkte gelten hier die Rechner aus dem eigenen Netz.

Folgende Einstellungen sind Möglich:

```
QOS_FILTER_x_PORT_TYPE='client'
QOS_FILTER_x_PORT_TYPE='server'
```

Seit Version 2.1 ist auch die Kombination der beiden Argumenten möglich, um sowohl den Verkehr über den angegebenen Port aus dem eigenen Netz, als auch den Verkehr über selbigen Port aus dem Internet in die selbe Klasse zu stecken:

```
QOS_FILTER_x_PORT_TYPE='client server'
```

Dies entspricht der Erstellung von zwei ähnlichen Filtern, bei denen `QOS_FILTER_x_PORT_TYPE` einmal auf Client und einmal auf Server gesetzt wurde.

QOS_FILTER_x_OPTION Mit dieser Variable kann man weitere Eigenschaften für den Filter aktiveren. Es darf hier höchstens eines der folgenden Argumente angegeben werden, denn eine Kombination dieser in ein und dem selben Filter macht keinen Sinn. Hingegen ist es sehr wohl möglich und auch teilweise sinnvoll, dass zum Beispiel ein Filter für ACK-Pakete und ein 2. Filter für TOSMD-Pakete ihre Pakete in die selbe Zielklasse leiten (siehe `QOS_FILTER_x_CLASS`).

ACK Acknowledgement-Pakete.

Ein Paket das auf diese Option zutrifft, wird als Bestätigung für ein Datenpaket zurückgesendet. Wenn ihr z.B. einen großen Download am laufen habt, dann kommen bei euch viele Datenpakete an und für jedes muß eine Antwort gesendet werden, dass das Datenpaket angekommen ist. Lassen diese Bestätigungspakete auf sich warten, so so wartet der Datenversender ab, bis diese eingetroffen sind, bevor er neue Datenpakete sendet, was euch nicht so richtig schmeckt.

Das ganze ist insbesondere wichtig bei asymetrischen Verbindungen (ungleiche Up/Downstream-Bandbreite), wie sie bei den meisten privaten DSL-Angeboten üblich sind. Wird der meist relativ kleine Upstream an seine Grenzen gefahren, so stapeln sich die Pakete vor dem Ausgang förmlich auf und irgendwo in diesem riesigen Haufen sitzen hier und da die kleinen Bestätigungspakete. Im Normalfall geht das dann hübsch der Reihe nach. Bis das Bestätigungspaket dann an der Reihe ist, kann es gut sein, dass es so lange gedauert hat, dass unser Datenversender eine kleine Pause einlegt und was wie gesagt nicht gut für den Downstream ist.

Wir müssen also dafür sorgen, dass die Bestätigungspakete auf die Überholspur kommen, so dass sie in windeseile an allen “normalen” Paketen vorbeihuschen, damit sie auch noch rechtzeitig beim Datenversender ankommen. Wie sich dies Option sinnvoll mit einer Klasse kombinieren läßt, wird bei den Anwendungsbeispielen erläutert.

ICMP Ping-Pakete (Protokoll ICMP)

Ping-Pakete werden dazu benutzt, die Zeit zu messen, die ein Paket von A nach B braucht. Wenn ihr also ordentlich angeben wollt, dann gebt den Ping-Paketen z.B. eine höhere Priorität. Das hat jetzt nichts mit dem Spielen im Internet selber zu tun, also nicht denken nur weil ihr den Ping-Pakete den Vorrang gibt, dass ihr super niedrige Pingzeiten im Spiel bekommt...

IGMP IGMP-Pakete (Protokoll IGMP)

Wenn IP-TV benutzt wird, ist es sinnvoll, das IGMP Protokoll zu filtern und zu priorisieren.

TCPSMALL Kleine TCP Pakete

Durch diesen Filter können ausgehende HTTP(s)-Requests gefiltert priorisiert werden. Eine Kombination mit einem Zielport ist möglich und sinnvoll. Größe der TCP Pakete: max. 800 Byte.

TCP TCP-Pakete (Protokoll TCP)

Es wird nur nach Paketen gefiltert, die das Protokoll TCP benutzen.

UDP UDP-Pakete (Protokoll UDP)

Es wird nur nach Paketen gefiltert, die das Protokoll UDP benutzen.

TOS* Type of Service

TOS steht für "Type of Service". Eine Applikation kann für jedes Paket was es verschickt eines der 4 TOS-Bits setzen. Damit wird angegeben welche Behandlung für die Pakete vorgesehen sind. So kann z.B. SSH TOS-Minimum-Delay für das Versenden der ein und Ausgabe setzen und TOS-Maximum-Troughput für das Versenden von Dateien. Generell benutzen Linux/Unix Programme diese Bits öfter als Windowsprogramme. Außerdem kann man z.B. auch it der Firewall die TOS-Bits für bestimmte Pakete setzen. Letztendlich kommt es dann aber auf die Router auf der Strecke an, ob die TOS-Bits beachtet werden, oder nicht. Wirklich von Interesse für einen fli4l sind aber eigentlich nur die TOS-Bits Minimum-Delay und Maximum-Throughput.

TOSMD - TOS Minimum-Delay Wird für Dienste benutzt, bei denen es wichtig ist, dass Pakete möglichst ohne Zeitverzögerung weitergeleitet werden. Empfohlen wird dieses TOS-Bit für FTP (Kontrolldaten), Telnet und SSH.

TOSMT - TOS Maximum-Troughput Wird für Dienste benutzt, bei denen es wichtig ist, dass große Datenmengen mit hoher Geschwindigkeit weitergeleitet werden. Empfohlen wird dieses TOS-Bit für FTP-Data und WWW.

TOSMR - TOS Maximum-Reliability Wird benutzt, wenn es wichtig ist, das man eine gewisse Sicherheit hat, dass die Daten an ihr Ziel gelangen, ohne das ein erneutes senden nötig ist. Empfohlen wird dieses TOS-Bit für SNMP und DNS.

TOSMC - TOS Minimum-Cost Wird benutzt, wenn es wichtig ist die Kosten der Datenübertragung zu Minimieren. Empfohlen wird dieses TOS-Bit für NNTP und SMTP.

DSCP* Differentiated Services Code Point

Mit DSCP bezeichnet man die Markierung nach RFC 2474. Dieses Verfahren hat 1998 die TOS Markierung weitestgehend abgelöst.

Die Filter auf DSCP-Klassen können wie folgt konfiguriert werden:

```
QOS_FILTER_x_OPTION='DSCPef'
QOS_FILTER_x_OPTION='DSCPcs3'
```

Bitte beachten, dass DSCP groß und die Klasse kleingeschrieben wird.

Es können folgende Klassen verwendet werden:

af11-af13, af21-af23, af31-af33, af41-af43, cs1-cs7, ef und be (Standard)

4.18.2. Anwendungsbeispiele

Wie konfiguriert man `OPT_QoS` nun genau? Dies wird nun an einigen Beispielen gezeigt:

- Beispiel 1: Ein einfaches Beispiel mit dem Ziel die Bandbreite auf 3 Rechner zu verteilen.
- Beispiel 2: Ein Beispiel mit dem Ziel die Bandbreite auf 2 Rechner zu verteilen und die jeweiligen Bandbreiten pro Rechner wiederum noch ein zweites mal aufzuteilen auf einen Port und den restlichen Verkehr des jeweiligen Rechners.
- Beispiel 3: Ein Beispiel, welches die allgemeine Funktionsweise von QoS versucht nahezubringen.
- Beispiel 4: Beispielkonfiguration für das Bevorteilen von ACK-Paketen, damit der Downstream bei gleichzeitig starkem Upstream nicht einbricht.

Beispiel 1

Ein einfaches Beispiel mit dem Ziel die Bandbreite auf 3 Rechner zu verteilen.

Dazu erstellen wir 4 Klassen (siehe `QOS_CLASS_N` und folgende) mit den jeweiligen Geschwindigkeiten (siehe `QOS_CLASS_x_MINBANDWIDTH` / `QOS_CLASS_x_MINBANDWIDTH`) und hängen sie an die Klasse 0 (siehe `QOS_CLASS_x_PARENT`) also direkt an das Interface für “up” bzw. “down” (siehe `QOS_CLASS_x_DIRECTION`).

Die vierte Klasse ist nur für eventuelle Besucher und bekommt weniger Bandbreite zugeteilt. Mit `QOS_INTERNET_DEFAULT_DOWN`=’4’ lassen wir in diesem Fall allen nicht gefilterten Verkehr in die vierte “Gast”-Klasse wandern. Da wir aber selten Gäste haben und die Bandbreite für die anderen 3 Klassen jeweils die selbe beträgt, bekommt jeder Rechner 1/3 der gesamten Bandbreite, effektiv also 256Kibit/s.

Mit dieser Konfiguration haben wir allerdings erst das Grundgerüst erstellt. Jetzt müssen wir noch sagen welcher Verkehr durch welche Klasse geregelt werden soll.

Dazu benutzen wir Filter, welche den Verkehr den einzelnen Klassen zuordnen. Wir erstellen also 3 Filter für die 3 Rechner (siehe `QOS_FILTER_N` und folgende) und ordnen jeden Filter einer Klasse zu (siehe `QOS_FILTER_x_CLASS`). Jetzt können wir mit `QOS_FILTER_x_IP_INTERN`, `QOS_FILTER_x_IP_INTERN`, `QOS_FILTER_x_PORT`, `QOS_FILTER_x_PORT` und `QOS_FILTER_x_OPTION` bestimmen was durch die jeweilige Klasse zu der der Filter gehört geregelt werden soll.

Nennen wir das Interface 0 und die 3 Klassen 1, 2 und 3 und die 3 Filter F1, F2 und F3 ergibt sich das in Abbildung 4.7 dargestellte Szenario.

Die Konfiguration sieht dann so aus:

Drei Rechner nach IP gefiltert die je 1/3 bekommen falls kein Gast anwesend ist:

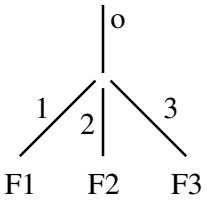


Abbildung 4.7.: Beispiel 1

```
OPT_QOS='yes'
QOS_INTERNET_DEV_N='1'
QOS_INTERNET_DEV_1='ppp0'
QOS_INTERNET_BAND_DOWN='768Kibit/s'
QOS_INTERNET_BAND_UP='128Kibit/s'
QOS_INTERNET_DEFAULT_DOWN='4'
QOS_INTERNET_DEFAULT_UP='0'

QOS_CLASS_N='4'

QOS_CLASS_1_PARENT='0'
QOS_CLASS_1_MINBANDWIDTH='232Kibit/s'
QOS_CLASS_1_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_1_DIRECTION='down'
QOS_CLASS_1_PRIO=''

QOS_CLASS_2_PARENT='0'
QOS_CLASS_2_MINBANDWIDTH='232Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_2_DIRECTION='down'
QOS_CLASS_2_PRIO=''

QOS_CLASS_3_PARENT='0'
QOS_CLASS_3_MINBANDWIDTH='232Kibit/s'
QOS_CLASS_3_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_3_DIRECTION='down'
QOS_CLASS_3_PRIO=''

QOS_CLASS_4_PARENT='0'
QOS_CLASS_4_MINBANDWIDTH='72Kibit/s'
QOS_CLASS_4_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_4_DIRECTION='down'
QOS_CLASS_4_PRIO=''


QOS_FILTER_N='3'

QOS_FILTER_1_CLASS='1'
QOS_FILTER_1_IP_INTERN='192.168.0.2'
QOS_FILTER_1_IP_EXTERN=''
QOS_FILTER_1_PORT=''
QOS_FILTER_1_PORT_TYPE=''
```

```
QOS_FILTER_1_OPTION=''

QOS_FILTER_2_CLASS='2'
QOS_FILTER_2_IP_INTERN='192.168.0.3'
QOS_FILTER_2_IP_EXTERN=''
QOS_FILTER_2_PORT=''
QOS_FILTER_2_PORT_TYPE=''
QOS_FILTER_2_OPTION=''

QOS_FILTER_3_CLASS='3'
QOS_FILTER_3_IP_INTERN='192.168.0.4'
QOS_FILTER_3_IP_EXTERN=''
QOS_FILTER_3_PORT=''
QOS_FILTER_3_PORT_TYPE=''
QOS_FILTER_3_OPTION=''
```

Die Option `QOS_INTERNET_DEFAULT_UP` wurde auf 0 gesetzt da der Upstream nicht beschränkt werden soll.

Beispiel 2

Ein Beispiel mit dem Ziel die Bandbreite auf 2 Rechner zu verteilen und die jeweiligen Bandbreiten pro Rechner wiederum noch ein zweites mal aufzuteilen auf einen Port und den restlichen Verkehr des jeweiligen Rechners.

Dazu erstellen wir erst einmal wieder 2 Klassen mit den jeweiligen Gesamtgeschwindigkeit und hängen sie direkt an das Interface für “up” bzw. “down” (siehe erstes Beispiel). Jetzt erstellen wir für den ersten Rechner an der ersten Klasse zwei weitere Klassen. Die Klassen werden genau so erstellt wie die beiden ersten Klassen direkt am Interface, allerdings mit einer Besonderheit: `QOS_CLASS_x_PARENT` ist jetzt nicht 0, sondern die Nummer der Klasse an die die Klassen angehängt werden sollen. Ist dies z. B. `QOS_CLASS_1`, so muss man jetzt `QOS_CLASS_1` von der Klasse die angehängt werden soll auf 1 setzen. Das gleiche wird für den zweiten Rechner auch gemacht. Man hängt wieder zwei Klassen an die Klasse für den zweiten Rechner. Dies kann man nun nicht nur für zwei Rechner machen, sondern für so viele wie man möchte. Auch kann man so viele Unterklassen an einer Klasse erstellen wie man möchte.

Hiermit haben wir wieder das Grundgerüst erstellt und müssen nun mit den Filtern den Verkehr den einzelnen Klassen zuordnen. (siehe erstes Beispiel)

Wir erstellen also 2 Filter für den ersten Rechner und 2 Filter für den zweiten Rechner. Jeweils einen Filter für den Port und einen Filter für den restlichen Verkehr vom Rechner. Hierbei ist unbedingt auf die Reihenfolge zu achten. Als erstes jeweils nur den Port und danach den Rest. Anders herum würde ja schon der Filter für den Rest alles einer Klasse zuordnen.

Nennen wir das Interface 0 und die 6 Klassen 1, 2, 3, 4, 5 und 6 und die 4 Filter F1, F2, F3 und F4 ergibt sich das in Abbildung 4.7 dargestellte Szenario.

Die Konfiguration sieht dann so aus:

Zwei Klassen für 2 Rechner die je 1/2 bekommen, und zwar vom Interface, mit jeweils 2 Klassen für einen Port der 2/3 bekommt und den Rest der 1/3 bekommt, und zwar jeweils von der Vaterklasse:

```
OPT_QOS='yes'
QOS_INTERNET_DEV_N='1'
```

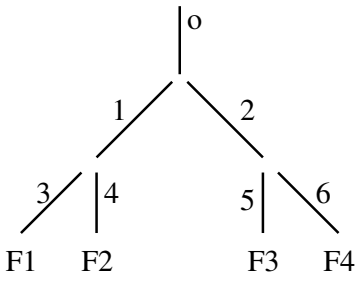


Abbildung 4.8.: Beispiel 2

```
QOS_INTERNET_DEV_1='ppp0'
QOS_INTERNET_BAND_DOWN='768Kibit/s'
QOS_INTERNET_BAND_UP='128Kibit/s'
QOS_INTERNET_DEFAULT_DOWN='7'
QOS_INTERNET_DEFAULT_UP='0'

QOS_CLASS_N='6'

QOS_CLASS_1_PARENT='0'
QOS_CLASS_1_MINBANDWIDTH='384Kibit/s'
QOS_CLASS_1_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_1_DIRECTION='down'
QOS_CLASS_1_PRIO=''

QOS_CLASS_2_PARENT='0'
QOS_CLASS_2_MINBANDWIDTH='384Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_2_DIRECTION='down'
QOS_CLASS_2_PRIO=''

QOS_CLASS_3_PARENT='1'
QOS_CLASS_3_MINBANDWIDTH='256Kibit/s'
QOS_CLASS_3_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_3_DIRECTION='down'
QOS_CLASS_3_PRIO=''

QOS_CLASS_4_PARENT='1'
QOS_CLASS_4_MINBANDWIDTH='128Kibit/s'
QOS_CLASS_4_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_4_DIRECTION='down'
QOS_CLASS_4_PRIO=''

QOS_CLASS_5_PARENT='2'
QOS_CLASS_5_MINBANDWIDTH='256Kibit/s'
QOS_CLASS_5_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_5_DIRECTION='down'
QOS_CLASS_5_PRIO=''

QOS_CLASS_6_PARENT='2'
QOS_CLASS_6_MINBANDWIDTH='128Kibit/s'
```

```
QOS_CLASS_6_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_6_DIRECTION='down'
QOS_CLASS_6_PRIO=''

QOS_FILTER_N='4'

QOS_FILTER_1_CLASS='3'
QOS_FILTER_1_IP_INTERN='192.168.0.2'
QOS_FILTER_1_IP_EXTERN=''
QOS_FILTER_1_PORT='80'
QOS_FILTER_1_PORT_TYPE='client'
QOS_FILTER_1_OPTION=''

QOS_FILTER_2_CLASS='4'
QOS_FILTER_2_IP_INTERN='192.168.0.2'
QOS_FILTER_2_IP_EXTERN=''
QOS_FILTER_2_PORT=''
QOS_FILTER_2_PORT_TYPE=''
QOS_FILTER_2_OPTION=''

QOS_FILTER_3_CLASS='5'
QOS_FILTER_3_IP_INTERN='192.168.0.3'
QOS_FILTER_3_IP_EXTERN=''
QOS_FILTER_3_PORT='80'
QOS_FILTER_3_PORT_TYPE='client'
QOS_FILTER_3_OPTION=''

QOS_FILTER_4_CLASS='6'
QOS_FILTER_4_IP_INTERN='192.168.0.3'
QOS_FILTER_4_IP_EXTERN=''
QOS_FILTER_4_PORT=''
QOS_FILTER_4_PORT_TYPE=''
QOS_FILTER_4_OPTION=''
```

Bei diesem Beispiel wurde die Option `QOS_INTERNET_DEFAULT_DOWN` so gewählt, dass der Verkehr, welcher nicht durch einen Filter einer Klasse zugeordnet wird, in eine nicht existierende Klasse gesteckt wird. Einfach aus dem Grund um das Beispiel zu vereinfachen und weil in dem Beispiel davon ausgegangen wird dass es keinen Rest gibt. Verkehr der in eine nicht existierende Klasse geleitet wird, wird nur sehr langsam weiter geleitet. Wenn es einen Rest gibt, ist also unbedingt darauf zu achten, dass dieser in eine eigene Klasse gesteckt wird, die auch existiert.

Die Option `QOS_INTERNET_DEFAULT_UP` wurde auf 0 gesetzt da der Upstream nicht beschränkt werden soll.

Beispiel 3

Ein Beispiel, welches die allgemeine Funktionsweise von QoS versucht nahezubringen.

In Abbildung 4.9 ist noch einmal die Aufteilung aus dem zweiten Beispiel zu sehen, allerdings mit einer Erweiterung. An die Beiden Unterklassen der zweiten Klasse sind jeweils noch zwei weitere Unterklassen Angehängt. Es ist also möglich noch tiefer zu verschachteln. Es ist möglich, tiefer zu verschachteln als auf diesem Bild, die momentane Grenze liegt hier bei 8

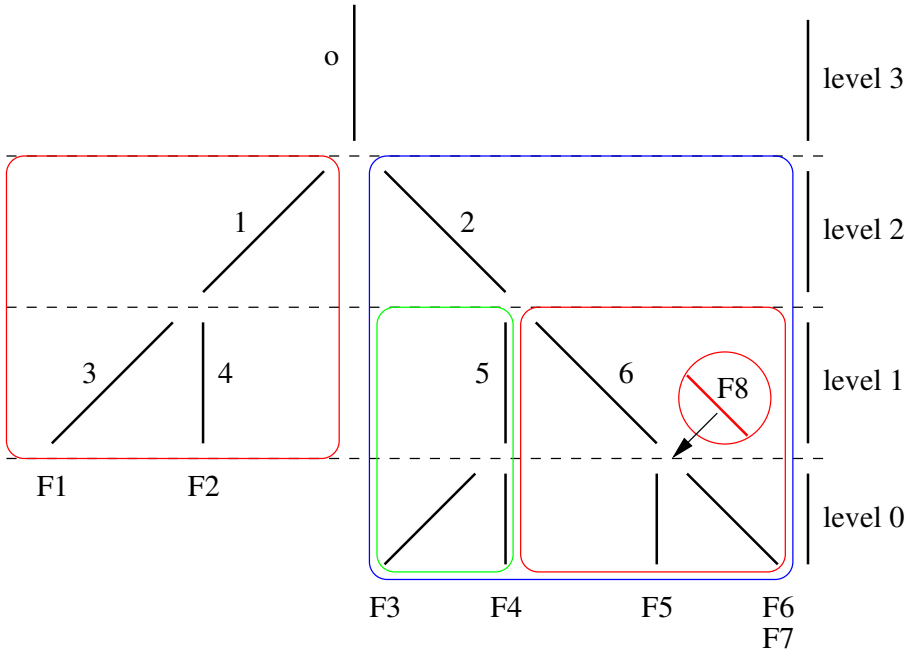


Abbildung 4.9.: Beispiel 3

Stufen, also man darf maximal nur 7 Weitere Stufen nach dem Interface erstellen, danach ist schluss. In die "Breite" ist allerdings keine Grenze gesetzt. Man kann also an eine Unterklasse innerhalb einer Stufe so viele Klassen anhängen wie man möchte.

Auf diesem Bild ist ausserdem noch zu erkennen, dass es auch möglich ist mehr als einen Filter an eine Klasse zu hängen, so wie es an der Klasse 10 geschehen ist. Aber auch bei den Filtern bleibt zu beachten dass es Momentan nicht möglich ist einen Filter mitten in den "Baum" zu heangen, so wie es mit F8 geschehen sollte.

Schauen wir uns nun als letztes noch den Sinn von Klassen und Unterklassen an. Klassen dienen dazu die Geschwindigkeit einer Verbindung einzustellen und zu regeln. Die Verteilung der Geschwindigkeit erfolgt wie bei `QOS_CLASS_x_MINBANDWIDTH` beschrieben. Allerdings kann dies einen Nachteil haben wenn man z.b. alle Klassen an eine Klasse hängt. Möchte man z.b. einem Rechner die hälfte der bandbreite geben und dem zweiten ebenfalls die hälfte allerdings aufgeteilt auf 2/3 http und 1/3 Rest also jeweils 2/6 und 1/6 vom ganzen. So geschieht nun folgendes: bei Vollast bekommt jeder seine hälfte. Ueberträgt der zweite jedoch nichts über http so bleibt ja 2/6 über. Diese 2/6 bekommt jedoch nun nicht der 2. Rechner alleine, sondern es wird nach dem beschriebenen Verfahren aufgeteilt. Um dieses zu verhindern erstellt man Unterklassen. Der Verkehr einer Klasse wird somit erst an die Unklassen verteilt, erst wenn diese nicht den Kompletten Verkehr beanspruchen wird der Rest an andere Klassen Verteilt. In dem Bild sind jeweils die Bereiche eingekreist welche zusammengehoren. Rot = 1, Blau = 2, Grün = 5 und Orange = 6.

Beispiel 4

Beispielkonfiguration für das Priorisieren von ACK-Paketen, damit der Downstream bei gleichzeitig starkem Upstream nicht einbricht:

```
OPT_QOS='yes'
QOS_INTERNET_DEV_N='1'
QOS_INTERNET_DEV_1='ppp0'
QOS_INTERNET_BAND_DOWN='768Kibit/s'
QOS_INTERNET_BAND_UP='128Kibit/s'
QOS_INTERNET_DEFAULT_DOWN='0'
QOS_INTERNET_DEFAULT_UP='2'
```

Hier konfigurieren wir ppp0 als Internetdevice (DSL) und geben die für TDSL (und einiger anderer Provider) übliche Up/Downstreambandbreiten an. Eventuell ist es nötig, dass wir die Upstreambandbreite noch um das eine oder andere Kibibit herabsetzen, das muß man ausprobieren.

Da wir keine Klassen für den Downstream einrichten wollen, setzen wir

```
QOS_INTERNET_DEFAULT_DOWN='0'
```

Für den Upstream soll die Klasse mit der Nummer 2 die Standardklasse sein. Das Netzwerkdevice ist eth0 und auf 10Mibit/s eingestellt.

```
QOS_CLASS_N='2'

QOS_CLASS_1_PARENT='0'
QOS_CLASS_1_MINBANDWIDTH='127Kibit/s'
QOS_CLASS_1_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_1_DIRECTION='up'
QOS_CLASS_1_PRIO=''
```

Dies ist die Klasse in die wir unsere ACK-(Bestätigungs-)Pakete hineinstecken wollen. Die ACK-Pakete sind recht klein und benötigen deswegen nur recht wenig Bandbreite. Trotzdem wollen wir sie eigentlich in keinster Weise einschränken und teilen ihr 127Kibit/s zu. 1Kibit/s lassen wir übrig für den Rest.

```
QOS_CLASS_2_PARENT='0'
QOS_CLASS_2_MINBANDWIDTH='1Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_2_DIRECTION='up'
QOS_CLASS_2_PRIO=''
```

In diese Klasse soll dann der Rest (alles außer ACK-Pakete) hineingesteckt werden. Die Bandbreite, die wir dieser Klasse zuteilen sind die verbleibenden 1Kibit/s (128-127=1). Wir begrenzen sie aber auch nicht auf 1Kibit/s, begrenzt wird die Klasse durch den Eintrag

```
QOS_CLASS_2_MAXBANDWIDTH='128Kibit/s'
```

Da unsere erste Klasse die zugeteilte Bandbreite wohl kaum ausnutzen wird, bleibt also immer etwas übrig und das was übrig bleibt schnappt sich dann die zweite. Wenn man den Upstream noch weiter aufteilen möchte (was meist der Fall ist), so sind alle weiteren Klassen unter diese Klasse zu “hängen”. Dabei muß natürlich auch `QOS_INTERNET_DEFAULT_UP` entsprechend angepaßt werden.

```
QOS_FILTER_N='1'

QOS_FILTER_1_CLASS='1'
QOS_FILTER_1_IP_INTERN=''
QOS_FILTER_1_IP_EXTERN=''
QOS_FILTER_1_PORT=''
QOS_FILTER_1_PORT_TYPE=''
QOS_FILTER_1_OPTION='ACK'
```

Dieser Filter filtert alle Pakete, die auf die Option ACK zutreffen, also ACK-Pakete. Durch den Eintrag `QOS_FILTER_1_CLASS`='1' erreichen wir, dass diese gefilterten Pakete in die 1. Klasse gesteckt werden.

Zum Testen muß sucht man sich am besten eine gute oder mehrere Up- und Downloadquellen aus, von denen man weiß, dass sie sowohl den Up- als auch den Downstream voll auslasten können und läßt die Kabel glühen. Dabei sollte man einen Blick auf die Trafficanzeige des ImonC werfen. Am besten führt man das ganze auch mal ohne QoS durch.

Der Downstream sollte gar nicht oder wesentlich weniger stark einbrechen als ohne diese Konfiguration. Wie schon gesagt kann man die Lage noch verbessern, in dem man die Upstreambandbreite in Kibibit-Schritten herabsetzt und dann die Auswirkungen beobachtet. Bei mir wurde zum Beispiel das Optimum bei 121Kibit/s erreicht (kein Einbruch des Downstreams mehr). Dabei sind natürlich auch die MAXBANDWIDTH- und MINBANDWIDTH-Werte der Klassen entsprechend anzupassen.

4.19. SSHD - Secure Shell, Secure Copy

Eine Secure-Shell bietet die Möglichkeit, eine verschlüsselte Verbindung mit dem fli4l-Router aufzunehmen. Außerdem können mit dem Secure-Copy-Befehl Dateien verschlüsselt auf den fli4l-Router übertragen werden. Wird zusätzlich eine Public Key Anmeldung (Seite 239) benutzt, können Befehle auf dem fli4l-Router und Dateiübertragungen auch scriptgesteuert ausgeführt werden. Ab der Version 2.1.7 gibt es nur noch einen SSH2 Server.

4.19.1. Installation des Secure-Shell-Dienstes

OPT_SSHD Standard-Einstellung: `OPT_SSHD='no'`

Soll der Zugriff auf den Router mittels ssh ermöglicht werden, bedarf es der Änderung auf von `OPT_SSHD` auf `'yes'`. Dies installiert den ssh-Server Dropbear auf dem fli4l-Router. Dies ermöglicht auch das Kopieren von Dateien auf den Router.

SSHD_ALLOWPASSWORDLOGIN Standard-Einstellung: `SSHD_ALLOWPASSWORDLOGIN='yes'`

Wird `SSHD_ALLOWPASSWORDLOGIN` auf `'no'` eingestellt, ist die Anmeldung mit ssh über ein Passwort auf dem fli4l-Router nicht mehr möglich. Die Anmeldung kann dann nur noch mittels privatem/öffentlichem Schlüsselpaar (private/public key) erfolgen. Dies setzt voraus, dass ein öffentlicher Schlüssel (Seite 239) auf dem Router hinterlegt ist.

SSHD_CREATEHOSTKEYS Standard-Einstellung: `SSHD_CREATEHOSTKEYS='no'`

Ein ssh-Server benötigt einen sogenannten Hostkey, der weltweit einmalig sein sollte, damit sich der ssh-Server eindeutig gegenüber einem ssh-Client identifizieren kann. Das

sshd opt-Paket liefert zwar einen Hostkey mit, um das erste Einloggen auf dem fli4l-Router per ssh-Client zu erlauben, aber der mitgelieferte Hostkey sollte so schnell wie möglich durch einen selbst generierten, nur Ihnen bekannten Hostkey ersetzt werden. Die Generierung eines eigenen Hostkeys ist deshalb so wichtig, weil nur auf diese Weise Schutz gegen so genannte Man-in-the-Middle-Attacken möglich ist. Ihr ssh-Client bemerkt es, wenn ein Cracker vorgibt, Ihr fli4l-Router zu sein, da dem Cracker dessen Hostkey nicht bekannt ist. Ihr ssh-Client warnt Sie daraufhin mit einer Meldung, dass der Hostkey sich geändert hat.

Die Erzeugung Ihres eigenen Hostkeys geschieht vollkommen automatisch, sobald Sie die Einstellung `SSHD_CREATEHOSTKEYS` auf `'yes'` setzen. Dieser Vorgang ist sehr rechenintensiv und kann deshalb die Bootzeit um mehrere Minuten verlängern. Wenn der fli4l-Router mit aktiviertem `SSHD_CREATEHOSTKEYS` Eintrag startet, wird ein (oder mehrere) Hostkey(s) in dem Verzeichnis `/tmp/ssh` erzeugt. Die Dateien die dort stehen, kopieren Sie in das Verzeichnis `etc/ssh` unterhalb Ihres config Verzeichnisses (auf dem Rechner, auf dem sie fli4ls Bootmedium erzeugen). In meinem Fall sieht ein Directorylisting des config.babel Verzeichnisses so aus:

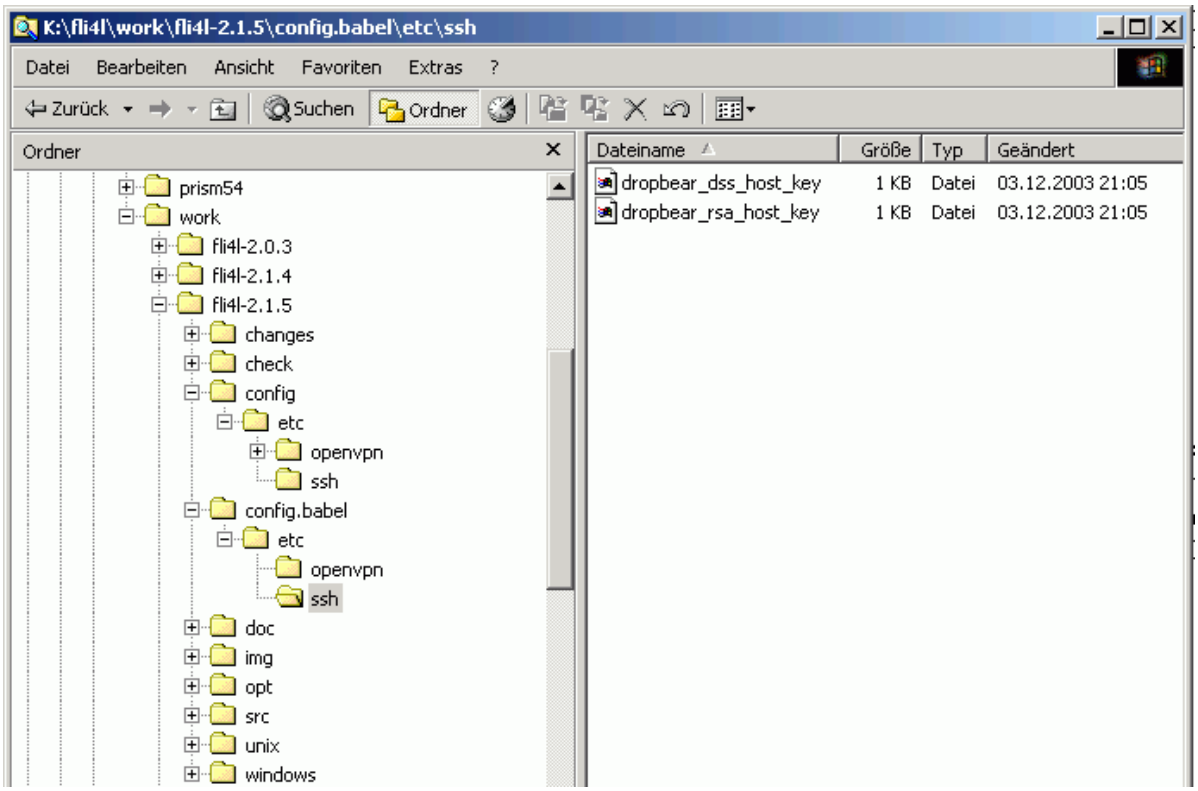


Abbildung 4.10.: Verzeichnisstruktur fli4l

Beachten Sie, dass unterhalb der `config.babel` Verzeichnis erst das Verzeichnis etc kommt und darunter dann das Verzeichnis ssh. Und genau dorthin muss der oder die eben erzeugte(n) Hostkey(s) kopiert werden. Ab der fli4l Version 2.1.5 werden Dateien die unterhalb Ihres config Verzeichnisses stehen vorrangig vor den Dateien aus dem opt Verzeichnis behandelt. Dadurch werden bei dem nächsten Update Ihres fli4l-Routers die Dateien aus

dem Verzeichnis `config/etc/ssh` eingebunden und nicht die Dateien, die im Verzeichnis `opt/etc/ssh` stehen. So ist es möglich für jeden fli4l-Router, den Sie konfigurieren, einen eigenen Hostkey zu benutzen. Wenn Sie die fli4l-Routerdateien erzeugen, erscheint ziemlich zum Schluss die Meldung „appending config specific files to opt.img ...“. Dort werden dann alle Dateien aufgelistet, die aus Ihrem config Verzeichnis kommen und nicht aus dem opt Verzeichnis.

```
#
# appending config specific files to opt.img ...
#
etc/ssh/dropbear_dss_host_key
etc/ssh/dropbear_rsa_host_key
```

Wenn Sie einen neuen Hostkey erzeugt haben, setzen Sie danach den Wert `SSHD_CREATEHOSTKEYS` wieder auf `'no'`, damit die Startskripte des fli4l-Routers nicht ständig einen neuen Hostkey generieren.

Wenn Sie sich nach dem Update des Hostkey auf Ihrem fli4l-Router anmelden, wird eine (je nach Programm unterschiedliche) Warnmeldung von Ihrem ssh-Client ausgegeben, die Sie auf einen geänderten Hostkey hinweist. Das ist normal, da Sie ja gerade den von fli4l mitgelieferten Hostkey gegen den von Ihnen erzeugten Hostkey ausgetauscht haben. Befolgen Sie die Hinweise Ihres ssh-Client, wie Sie den geänderten Hostkey permanent übernehmen können. Sollten Sie diese Warnmeldung zu einem späteren Zeitpunkt noch einmal bekommen, sollten Sie in jedem Fall prüfen, warum diese Warnung ausgegeben wurde und nicht einfach blind den geänderten Hostkey akzeptieren.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
ca:a4:ab:e7:af:d8:68:05:d3:1f:e6:15:08:d6:ed:36.
Please contact your system administrator.
Add correct host key in /home/babel/.ssh/known_hosts to get rid of this message.
Offending key in /home/babel/.ssh/known_hosts:7
Password authentication is disabled to avoid man-in-the-middle attacks.
```

SSHD_PORT Standard-Einstellung: `SSHD_PORT='22'`

Mit `SSHD_PORT` kann abweichend vom Standard ein Port angegeben werden, auf dem der ssh-Server laufen soll.

Möchte man den ssh-Zugang auch von außen erlauben, ist `INPUT_ACCEPT_PORT_x` (Seite 44) anzupassen.

Die Befehle, um von einem Unix-/Linux-Rechner über das SSH-Protokoll auf fli4l zuzugreifen, lauten:

- ssh - Secure Shell
- scp - Secure Copy

Entsprechende Programme für Windows sind ebenso verfügbar, s. auch:
http://www.chiark.greenend.org.uk/~sgtatham/putty/
http://winscp.net/eng/docs/lang:de
http://www.tectia.com/de/de.iw3

SSHD_PUBLIC_KEY_N Standard-Einstellung: `SSHD_PUBLIC_KEY_N='0'`

`SSHD_PUBLIC_KEY_N` beschreibt die Anzahl der öffentlichen Schlüssel, die auf den fli4l-Router kopiert werden sollen.

SSH gestattet die Authentifizierung mit Hilfe von asymmetrischen Verschlüsselungsverfahren. Dabei erfolgt die Authentifizierung anstatt über Nutzername und Passwort über Nutzername und einem Public-/Privatekey. Damit kann man sich die Eingabe eines Passwortes sparen. Das Schlüsselpaar erzeugt man mit Hilfe von ssh-keygen (oder puttygen, wenn putty unter Windows als ssh-Client eingesetzt wird). Optional kann beim Schlüsselerzeugen eine Passphrase (also ein Passwort, das man braucht, wenn man den Schlüssel benutzen will) vergeben werden, welche die Sicherheit noch zusätzlich erhöht. Benutzt man Passphrases sollte man über den Einsatz eines Schlüsselagenten nachdenken (siehe ssh-agent oder pageant).

Wichtig: *Der private Teil des Schlüsselpaares, ist so sorgfältig zu behandeln wie ein Passwort, da er die gleiche Funktion erfüllt. Der private Teil des Schlüssel wird bei dem ssh-Client hinterlegt. Der öffentliche Teil des Schlüssel wird für den fli4l-Router gebraucht und mit `SSHD_PUBLIC_KEY_x` oder `SSHD_PUBLIC_KEYFILE_x` zur Verfügung gestellt.*

Für weitere Informationen siehe die manual Pages von ssh und Konsorten bzw. die Dokumentation zu putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/).

SSHD_PUBLIC_KEY_x Für jeden Nutzer, der über ssh Zugang zum fli4l-Router erlangen möchte, kann hier der öffentliche Teil des Schlüssel angegeben werden. Am einfachsten geht das per Cut-and-Paste aus einem Terminalfenster heraus. Das könnte z.B. in etwa wie folgt aussehen:

```
SSHD_PUBLIC_KEY_1='1024 ... nutzername@hostname'
```

Wichtig: *Der Schlüssel enthält keine Zeilenumbrüche. Bei Cut-and-Paste aus puttygen heraus werden aber eventuell selbige eingefügt. Diese Zeilenumbrüche müssen wieder entfernt werden.*

SSHD_PUBLIC_KEYFILE_N Standard-Einstellung: `SSHD_PUBLIC_KEYFILE_N='0'`

Anstatt den Inhalt des öffentlichen Teil des Schlüssel in die sshd.txt Datei zu kopieren, können Sie den öffentlichen Teil des Schlüssel auch direkt in das opt-Archiv kopieren lassen. Das funktioniert genauso wie bei `SSH_CREATEHOSTKEYS` beschrieben wurde. Kopieren Sie Ihren öffentlichen Teil des Schlüssel in das Verzeichnis `<config>/etc/ssh`.

SSHD_PUBLIC_KEYFILE_x Der Dateiname des öffentlichen Teil des Schlüssels im `<config>/etc/ssh` Verzeichnis.

```
SSHD_PUBLIC_KEYFILE_1='root@fli4l'
```

SSH_CLIENT_PRIVATE_KEYFILE_N Standard-Einstellung:
`SSH_CLIENT_PRIVATE_KEYFILE_N='0'`

Wenn Sie mit dem ssh oder plink Client private Schlüssel zur Anmeldung an einen ssh Server benutzen wollen können Sie diese in das Verzeichnis <config>/etc/ssh kopieren. Das funktioniert genauso wie bei SSH_CREATEHOSTKEYS beschrieben wurde. Kopieren Sie Ihren privaten Teil des Schlüssel in das Verzeichnis <config>/etc/ssh. Private Schlüssel im OpenSSH Format werden automatisch bei jedem Startvorgang von fli4l ins das dropbear Format konvertiert.

SSH_CLIENT_PRIVATE_KEYFILE_x Der Dateiname des privaten Teil des Schlüssels im <config>/etc/ssh Verzeichnis.

```
SSHD_PRIVATE_KEYFILE_1='babel@rootserver'
```

4.19.2. Installation des dbclients

OPT_SSH_CLIENT Standard-Einstellung: `OPT_SSH_CLIENT='no'`

Wenn man einen reinen ssh2/scp Client benutzen möchte, kann man den dbclient von dropbear durch Setzen von `OPT_SSH_CLIENT='yes'` aktivieren. Dieser Client hat den Vorteil, dass er sich viel Programmcode mit dem dropbear ssh Server teilt. Dadurch wird sehr viel Platz im OPT–Archiv gespart. Der dbclient ist weitgehend kompatibel mit dem ssh/scp Client, die Befehlsparameter sind ähnlich. Es wird auch ein symbolischer Link auf /usr/bin/ssh bzw. /usr/bin/scp angelegt, damit ein gewohntes ssh <host> bzw. scp <source> <target> funktioniert.

Wenn man die dbclient bekannten Hostkeys permanent speichern will muss man die Datei known_hosts auf dem Verzeichnis /.ssh auf dem fli4l–Router in das config/etc/ssh kopieren. Das geschieht ähnlich wie mit einem erzeugten Hostkey. In dem folgenden Beispiel ist das ausgepackte fli4l Verzeichnis (in der die fli4l-Bootmedium erzeugt wird) in /home/babel/fli4l-3.10.4 zu finden. Die Konfigurationsdateien liegen alle im Verzeichnis config.babel.

```
cd /home/babel/fli4l-3.10.4
mkdir -p config.babel/etc/ssh
scp fli4l:/.ssh/* config.babel/etc/ssh
```

4.19.3. Installation des plink Clients

OPT_PLINK_CLIENT Standard-Einstellung: `OPT_PLINK_CLIENT='no'`

Installiert auf dem fli4l-Router einen ssh1/ssh2/telnet Client. Das plink Programm ist die Unixversion des bekannten PuTTY Programms für Windows. Ein Aufruf von plink auf dem fli4l-Router gibt eine Hilfeseite für die Benutzung von plink aus.

Wenn man die plink bekannten Hostkeys permanent speichern will muss man die Datei sshhostkeys auf dem Verzeichnis /.putty auf dem fli4l–Router in das config/etc/plink kopieren. Das geschieht ähnlich wie mit einem erzeugten Hostkey. In dem folgenden

Beispiel ist das ausgepackte fli4l Verzeichnis (in der das fli4l-Bootmedium erzeugt wird) in /home/babel/fli4l-3.10.4 zu finden. Die Konfigurationsdateien liegen alle im Verzeichnis config.babel.

```
cd /home/babel/fli4l-3.10.4
mkdir -p config.babel/etc/plink
scp fli4l:/.putty/* config.babel/etc/plink
```

4.19.4. Installation des sftp-server

OPT_SFTPSERVER Standard-Einstellung: `OPT_SFTPSERVER='no'`

Installiert auf dem fli4l-Router einen sftp-server.

4.19.5. Literatur

Dropbear SSH2 Site: http://matt.ucc.asn.au/dropbear/dropbear.html
Erste Version der Dokumentation von Claas Hilbrecht <babel@fli4l.de>, im April 2004

4.20. TOOLS - Zusätzliche Werkzeuge zum Debugging

Das Paket TOOLS liefert eine Reihe von Unix Programmen, die zumeist für Administrations- und Debugzwecke gedacht sind. Andere Programme wie wget werden z.B. dafür verwendet, die erste (Werbungs-)Seite einiger Provider abzufangen. Mit dem Wert 'yes' wird das jeweilige Programm mit auf den fli4l-Router kopiert. Die Standardeinstellung ist 'no'. Die Programme werden nur kurz vorgestellt, wie sie zu bedienen sind, entnehme man bitte den man Pages einer beliebigen Unix/ Linux Distribution oder online unter: http://www.linuxmanpages.com

4.20.1. Netzwerk-Tools

OPT_DIG Schweizer Taschenmesser fürs DNS

Der Befehl dig erlaubt es, vielfältige DNS-Abfragen durchzuführen.

OPT_FTP FTP-Client

Mit dem Programm ftp können eine FTP-Verbindung zu einem FTP-Server aufgebaut und Dateien zwischen Router und FTP-Server übertragen werden.

FTP_PF_ENABLE_ACTIVE Die Einstellung `FTP_PF_ENABLE_ACTIVE='yes'` fügt dem Paketfilter eine Regel hinzu, die auf dem Router initiiertes aktives FTP ermöglicht. Bei `FTP_PF_ENABLE_ACTIVE='no'` muss eine solche Regel (falls gewünscht) manuell zum `PF_OUTPUT_%`-Array hinzugefügt werden, ein Beispiel ist in diesem Abschnitt (Seite 69) zu finden.

Passives FTP ist immer möglich, hierfür ist weder diese Variable noch eine explizite Paketfilter-Regel notwendig.

OPT_IFTOP Netzwerküberwachung

Mit dem Programm iftop wird eine Auflistung alle aktiven Netzwerkverbindung und deren Durchsatz direkt auf dem fli4l angezeigt.

Das Programm iftop wird nach dem Anmelden auf dem fli4l-Router durch Eingabe von iftop gestartet.

OPT_IMONC Textorientiertes Steuerprogramm für imond

Dieses Programm liefert ein textorientiertes Frontend für den Router, um den imond zu steuern.

OPT_IPERF Performancemessung im Netzwerk

Mit dem Programm iperf kann eine Performancemessung des Netzwerks durchgeführt werden. Dazu wird das Programm auf den beiden beteiligten Testsystemen gestartet. Auf dem Server wird das Programm mit

```
fli4l-server 3.10.4~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

gestartet. Der Server wartet dann auf eine Verbindung vom Client. Der Client wird durch

```
fli4l-client 3.10.4~# iperf -c 1.2.3.4
------------------------------------------------------------
Client connecting to 1.2.3.4, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 1.2.3.5 port 50311 connected with 1.2.3.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec    985 MBytes    826 Mbits/sec
```

gestartet. Sofort startet die Performancemessung und zeigt die ersten Ergebnisse an. iperf kennt noch eine Reihe weiterer Optionen, für Details schauen Sie sich bitte die Informationen auf der Homepage http://iperf.sourceforge.net/ an.

OPT_NETCAT Übertragen von Daten an TCP basierte Server

OPT_NGREP Ein grep der direkt auf einem Netzwerkdevice arbeiten kann.

OPT_NTTCP Netzwerktest

Mit dem Programm NTTCP kann man die Netzwerkgeschwindigkeit testen. Dazu wird auf einer Seite ein Server gestartet und auf einer anderen Seite ein entsprechende Client.

Den Server startet man durch Eingabe von nttcp -i -v. Der Server wartet dann auf eine Testanforderung des Clients. Um jetzt z.B die Geschwindigkeit zu testen gibt man auf dem Client nttcp -t <IP Adresse des Servers> ein.

So sieht ein gestarteter nttcp Server aus:

```
fli4l-server 3.10.4~# nttcp -i -v
nttcp-l: nttcp, version 1.47
nttcp-l: running in inetd mode on port 5037 - ignoring options beside -v and -p
```

So sieht ein Test mit einem nttcp Client aus:

```
fli4l-client 3.10.4~# nttcp -t 192.168.77.77
l~~8388608~~~~4.77~~~~0.06~~~~~14.0713~~~1118.4811~~~~2048~~~~429.42~~~34133.3
1~~8388608~~~~4.81~~~~0.28~~~~~13.9417~~~~239.6745~~~~6971~~~1448.21~~~24896.4
```

Die Hilfeseite von nttcp zeigt alle weiteren Parameter:

```
Usage: nttcp [local options] host [remote options]
       local/remote options are:
        -t      transmit data (default for local side)
        -r      receive data
        -l#     length of bufs written to network (default 4k)
        -m      use IP/multicasting for transmit (enforces -t -u)
        -n#     number of source bufs written to network (default 2048)
        -u      use UDP instead of TCP
        -g#us   gap in micro seconds between UDP packets (default 0s)
        -d      set SO_DEBUG in sockopt
        -D      don't buffer TCP writes (sets TCP_NODELAY socket option)
        -w#     set the send buffer space to #kilobytes, which is
                dependent on the system - default is 16k
        -T      print title line (default no)
        -f      give own format of what and how to print
        -c      compares each received buffer with expected value
        -s      force stream pattern for UDP transmission
        -S      give another initialisation for pattern generator
        -p#     specify another service port
        -i      behave as if started via inetd
        -R#     calculate the getpid()/s rate from # getpid() calls
        -v      more verbose output
        -V      print version number and exit
        -?      print this help
        -N      remote number (internal use only)
        default format is: %9b%8.2rt%8.2ct%12.4rbr%12.4cbr%8c%10.2rcr%10.1ccr
```

OPT_RTMON Installiert ein tool, dass Änderungen der Routingtabelle überwacht. Primäre Verwendung: Debugging

OPT_SOCAT Das Programm "socat" ist quasi eine verbesserte und mit mehr Funktionen "vollgestopfte" Version des "netcat"-Programms (Seite 242). Mit "socat" können nicht nur diverse Netzwerk-Verbindungen aufgebaut bzw. entgegengenommen werden, sondern auch Daten an UNIX-Sockets, Geräte, FIFOs etc. gesandt bzw. von dort ausgelesen werden. Insbesondere können Quellen und Ziele *verschiedener* Typen miteinander verbunden werden: Ein Beispiel wäre etwa ein via TCP auf einem Port horchender Netzwerk-Server, der empfangene Daten in einen lokalen FIFO schreibt bzw. Daten aus dem FIFO ausliest und diese dann übers Netzwerk an den Client schickt. Siehe http://www.dest-unreach.org/socat/doc/socat.html für mehr Informationen sowie Anwendungsbeispiele.

OPT_TCPDUMP debug

Mit dem Programm tcpdump kann Netzwerkverkehr beobachtet, ausgewertet mitgeschnitten werden. Mehr dazu unter z.B. Google mit den Suchworten “tcpdump man“

tcpdump <parameter>

OPT_WGET http/ftp Client

Mit dem Programm wget können Daten von einem Webserver im Batch abgerufen werden. Praktisch ist aber (und deswegen ist wget im fli4l-Paket dabei), dass man damit Umlenkungen des Providers auf den eigenen Webserver nach einem Verbindungsaufbau auf einfache Weise abfangen kann, z.B. für Freenet. Wie das geht, hat Steffen Peiser in einem Mini-HowTo erklärt.

Siehe: http://www.fli4l.de/hilfe/howtos/einsteiger/wget-und-freenet/

4.20.2. Die Hardware-Erkennung

Oftmals weiß man nicht genau, welche Hardware im eigenen Rechner steckt bzw. welche Treiber man nun genau für seine Netzwerkkarte oder seinen USB-Chipsatz verwenden soll. Die Hardware kann an der Stelle helfen. Sie liefert eine Liste von Geräten im Rechner und wenn möglich den dazugehörenden Treiber. Man kann dabei auswählen, ob die Erkennung gleich beim Booten erfolgen soll (was sich vor einer Erstinstallation empfiehlt) oder später bei laufendem Rechner bequem über das Web-Interfaces getriggert werden soll. Die Ausgabe könnte dabei z.B. wie folgt aussehen:

```
fli4l 3.10.4 # cat /bootmsg.txt
#
# PCI Devices and drivers
#
Host bridge: Advanced Micro Devices [AMD] CS5536 [Geode companion] Host Bridge (rev 33)
Driver: 'unknown'
Entertainment encryption device: Advanced Micro Devices [AMD] Geode LX AES Security Block
Driver: 'geode_rng'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: Atheros Communications, Inc. AR5413 802.11abg NIC (rev 01)
Driver: 'unknown'
ISA bridge: Advanced Micro Devices [AMD] CS5536 [Geode companion] ISA (rev 03)
Driver: 'unknown'
IDE interface: Advanced Micro Devices [AMD] CS5536 [Geode companion] IDE (rev 01)
Driver: 'amd74xx'
USB Controller: Advanced Micro Devices [AMD] CS5536 [Geode companion] OHC (rev 02)
Driver: 'ohci_hcd'
USB Controller: Advanced Micro Devices [AMD] CS5536 [Geode companion] EHC (rev 02)
Driver: 'ehci_hcd'
```

Hier stecken also im wesentlichen 3 Netzwerkkarten drin, die vom 'via_rhine'-Treiber verwaltet werden und eine Atheros-Wlan-Karte, die vom madwifi-Treiber verwaltet wird (der Name wird noch nicht korrekt aufgelöst).

OPT_HW_DETECT Diese Variable sorgt dafür, dass die für die Hardware-Erkennung Dateien auf dem Router landen. Man kann sich die Ergebnisse dann entweder nach dem Booten auf der Konsole ansehen, wenn man `HW_DETECT_AT_BOOTTIME` auf 'yes' gesetzt hat oder im Web-Interface ansehen, wenn man `OPT_HTTPD` (Seite 131) auf 'yes' gesetzt hat. Im Web-Interface kann man sich natürlich auch den Inhalt von '/bootmsg.txt' ansehen, wenn man schon einen funktionierenden Netzzugang hat.

HW_DETECT_AT_BOOTTIME Startet die Hardware-Erkennung beim Booten. Die Erkennung läuft im Hintergrund (sie dauert ein wenig) und schreibt dann ihre Ergebnisse auf die Konsole und nach '/bootmsg.txt'.

OPT_LSPCI Auflisten aller PCI-Geräte

OPT_I2CTOOLS Tools für I^2C Zugriffe.

OPT_IWLEEPROM Tool zum Zugriff auf das EEPROM von Intel und Atheros WLAN Karten.

Wird benötigt um z.B. bei ath9k Karten die Reg-Domain passend zu setzen (siehe `http://blog.asiantuntijakaveri.fi/2014/08/one-of-my-atheros-ar9280-minipcie-cards.html`).

OPT_ATH_INFO Tool zur Hardwarediagnose von WLAN Karten mit Atheros Chipsatz.

Mithilfe dieses Tools können z.B. bei ath5k WLAN Karten detaillierte Informationen über die verwendete Hardware gewonnen werden. Dazu gehören z.B. der verwendete Chipsatz oder Angaben zur Kalibrierung.

4.20.3. Dateien-Tools

OPT_E3 Ein Editor für fli4l

Dies ist ein sehr kleiner, in Assembler geschriebener Editor. Er stellt verschiedene Editor-Modi zur Verfügung, die andere („große") Editoren nachstellen. Um einen bestimmten Modus zu wählen, reicht es e3 mit dem richtigen Befehl zu starten. Eine Kurzübersicht der Tastenbelegung bekommt man, wenn man e3 ohne Parameter startet oder Alt+H drückt (außer im VI-Modus, dort muß man im CMD-Modus „:h" eintippen). Zu beachten ist auch, dass das Caret-Zeichen (^) für die Ctrl-/Strg-Taste steht.

Befehl	Modus
e3 / e3ws	WordStar, JOE
e3vi	VI, VIM
e3em	Emacs
e3pi	Pico
e3ne	NEdit

OPT_MTOOLS Die mtools stellen eine Reihe von DOS-ähnlichen Befehlen zum vereinfachten Umgang (Kopieren, Formatieren, etc.) mit DOS-Datenträgern bereit.

Die genaue Syntax der Befehle kann in der Dokumentation von mtools nachgeschlagen werden:
http://www.gnu.org/software/mtools/manual/mtools.html

OPT_SHRED Installiert das Programm *shred* auf dem Router, ein Programm zum gründlichen Löschen von Blockgeräten.

OPT_YTREE Datei-Manager

Installiert Datei-Manager Ytree auf dem Router.

4.20.4. Entwickler-Tools

OPT_OPENSSL Mit dem Programm openssl können z.B. Test der Cryptobeschleuniger durchgeführt werden.

openssl speed -evp des -elapsed openssl speed -evp des3 -elapsed openssl speed -evp aes128 -elapsed

OPT_STRACE debug

Mit dem Programm strace können die Funktionsaufrufe, der Ablauf eines Programmes beobachtet werden

strace <programm>

OPT_REAVER Brute force Angriff aus Wifi WPS PINs

Testet alle möglichen WPS PINS aus um das WPA Passwort zu ermitteln. Details für die Verwendung auf der Kommandozeile bitte unter http://code.google.com/p/reaver-wps/ nachlesen.

OPT_VALGRIND Installiert Valgrind auf dem Router.

4.21. UMTS - Anbindung mittels UMTS an das Internet

Anbindung eines fli4l mittels UMTS an das Internet. Für den Betrieb sind unter anderem auch weitere optionale Pakete erforderlich.

4.21.1. Konfiguration

OPT_UMTS Standard-Einstellung: `OPT_UMTS`='no'

'yes' aktiviert das Paket.

UMTS_DEBUG Standard-Einstellung: `UMTS_DEBUG`='no'

Soll pppd zusätzliche Debug-Informationen ausgeben, muss man `UMTS_DEBUG` auf 'yes' setzen. In diesem Fall schreibt pppd zusätzlichen Informationen über die syslog-Schnittstelle.

WICHTIG: Damit diese auch über syslogd ausgegeben werden, muss die Variable `OPT_SYSLOGD` (s.o.) ebenso auf 'yes' gesetzt sein.

Einwahldaten einiger deutscher Netzbetreiber/Provider

Anbieter	APN	Benutzername	Passwort
T-Mobile	internet.t-mobile	beliebig	beliebig
Vodafone	web.vodafone.de	beliebig	beliebig
E-Plus	internet.eplus.de	eplus	gprs
O2 (Vertragskunden)	internet	beliebig	beliebig
O2 (Prepaid-Kunden)	pinternet.interkom.de	beliebig	beliebig
Alice	internet.partner1	beliebig	beliebig

UMTS_PIN Standard-Einstellung: `UMTS_PIN`='disabled'

Pin für die SIM-Karte

Erlaubt sind eine 4stellige Nummer oder das Wort 'disabled'

UMTS_DIALOUT Standard-Einstellung: `UMTS_DIALOUT`='*99***1#'

Wählparameter zum Herstellen der Verbindung

UMTS_GPRS_UMTS Standard-Einstellung: `UMTS_GPRS_UMTS`='both'

Welche Übertragungsart soll genutzt werden

Erlaubte Werte (both, gprs, umts)

UMTS_APN Standard-Einstellung: `UMTS_APN`='web.vodafone.de'

UMTS_USER Standard-Einstellung: `UMTS_USER`='anonymer'

UMTS_PASSWD Standard-Einstellung: `UMTS_PASSWD`='surfer'

Hier werden die für die Einwahl nötigen Daten angegeben.

Es sind Benutzerkennung und Passwort für den jeweils benutzten Provider anzugeben. `UMTS_USER` enthält die Benutzerkennung, `UMTS_PASSWD` das Passwort.

Für einige deutsche Netzbetreiber/Provider lauten die APNs (Einwahlknoten)

- http://www.teltarif.de/mobilfunk/internet/einrichtung.html

UMTS_NAME Standard-Einstellung: `UMTS_NAME`='UMTS'

Hier sollte ein Name für den Circuit vergeben werden - max. 15 Stellen lang. Dieser wird im imon-Client imonc angezeigt. Leerstellen (Blanks) sind nicht erlaubt.

UMTS_HUP_TIMEOUT Standard-Einstellung: `UMTS_TIMEOUT`='600'

Hier kann die Zeit in Sekunden angegeben werden, nach welcher die Verbindung beendet werden soll, wenn nichts mehr über die UMTS-Verbindung läuft. Dabei steht ein Timeout von '0' für kein Timeout.

UMTS_TIMES Standard-Einstellung: `UMTS_TIMES`='Mo-Su:00-24:0.0:Y'

Die hier angegebenen Zeiten bestimmen, wann dieser Circuit aktiviert werden soll und wann er wieviel kostet. Dadurch wird es möglich, zu verschiedenen Zeiten verschiedene Circuits mit Default-Routen zu verwenden (Least-Cost-Routing). Dabei kontrolliert der Daemon imond die Routen-Zuweisung.

UMTS_CHARGEINT Standard-Einstellung: `UMTS_CHARGEINT='60'`

Charge-Interval: Hier ist der Zeittakt in Sekunden anzugeben. Dieser wird dann für die Kosten-Berechnung verwendet.

UMTS_USEPEERDNS Standard-Einstellung: `UMTS_USEPEERDNS='yes'`

Soll der DNS des Providers verwenden werden.

UMTS_FILTER Standard-Einstellung: `UMTS_FILTER='yes'`

fli4l legt automatisch auf, wenn während der über hangup timeout angegebenen Zeit keine Daten über das ppp0-Interface gehen. Leider wertet das Interface auch Datentransfers mit, die von außen kommen, z.B. durch Verbindungsversuche eines P2P-Clients wie eDonkey. Da man heutzutage eigentlich pemanent von anderen kontaktiert wird, kann es passieren, dass fli4l die UMTS-Verbindung nie beendet.

Hier hilft die Option `UMTS_FILTER`. Setzt man es auf yes, wird nur noch Verkehr gewertet, der von der eigenen Maschine generiert wird und externer Traffic wird komplett ignoriert. Da von draußen reinkommender Traffic in der Regel dazu führt, dass der Router oder dahinter liegende Rechner reagieren, indem sie z.B. Verbindungswünsche ablehnen, werden zusätzlich noch einige rausgehende Pakete ignoriert.

UMTS_ADAPTER (optional)

Hier wird eingetragen ob es sich um eine PCMCIA-Karte, einen USB-Adapter oder um ein per USB-Kabel angeschlossenes Telefon handelt.

Bei nichtvorhandensein der Variable werden nur die benötigten Dateien für einen USB-Adapter kopiert.

Erlaubte Werte: (pcmcia,usbstick,usbphone)

Alle folgenden Variablen sind optional und nur notwendig wenn die automatische Erkennung versagt.

UMTS_IDVENDOR (optional) `UMTS_IDVENDOR='xxxx'`

Hersteller ID nach Einschalten des Adapters

UMTS_IDDEVICE (optional) `UMTS_IDDEVICE='xxxx'`

Produkt ID nach Einschalten des Adapters

Angabe der folgenden beiden Parameter nur notwendig, wenn sich eine ID ändert nach der Initialisierung

UMTS_IDVENDOR2 (optional) `UMTS_IDVENDOR2='xxxx'`

Hersteller ID nach Initialisierung des Adapters

UMTS_IDDEVICE2 (optional) `UMTS_IDDEVICE2='xxxx'`

Produkt ID nach Initialisierung des Adapters

UMTS_DRV (optional) `UMTS_DRV='xxxx'`

Treiber zum Ansteuern Adapters, wenn nicht angegeben wird 'usbserial' genommen

UMTS_SWITCH (optional) `UMTS_SWITCH='-v 0x0af0 -p 0x6971 -M 555...000 -s 10'`

Parameter für usb-modeswitch zum Initialisieren des Modems. (siehe Website usb-modeswitch) Es sollten bis auf wenige Ausnahmen alle auf der Website genannten Modems automatisch erkannt werden.

- http://www.draisberghof.de/usb_modeswitch/

UMTS_DEV (optional)

Bei Problemen kann hier die Datenschnittstelle für den pppd angegeben werden. Für die Adapter sind das meist folgende:

```
ttyUSB0 für usbstick
ttyS2   für pcmcia
ttyACM0 für usbphone
```

UMTS_CTRL (optional)

Einige Adapter haben mehrere Schnittstellen, über die das Modem gesteuert wird. Ist nur eine vorhanden können Statusinformationen nur im 'Offline'-Zustand ausgelesen werden. Bei einer Option Fusion UMTS Quad lautet die Schnittstelle zB: ttyUSB2.

4.21.2. Beispielkonfiguration für RRDTOOL

Je nach Hardware ist es möglich, über OPT_HTTPD (Seite 131) die Signalstärke und Bitfehler anzeigen zu lassen. Ausserdem kann der Verlauf der Signalstärke bzw. Bitfehlerrate mittels OPT_RRDTOOL aufgezeichnet werden. Bei mancher Hardware ist es nicht so sinnvoll, da Statusinformationen nur während des Offline-Zustandes ausgelesen werden können. Als Quelle für RRD ist dabei 'umts' anzugeben.

Beispielkonfig für rrdtool:

```
RRDTOOL_x_SOURCE='umts'
RRDTOOL_x_LABEL='UMTS Status'
RRDTOOL_x_OPTIONS_N='2'
RRDTOOL_x_OPTIONS_1='signal'
RRDTOOL_x_OPTIONS_1_LABEL='Signalstärke'
RRDTOOL_x_OPTIONS_2='error'
RRDTOOL_x_OPTIONS_2_LABEL='Bitfehler'
```

4.22. USB - Support für USB-Geräte

OPT_USB Hier wird die grundsätzliche Unterstützung von USB-Geräten ein- beziehungsweise ausgeschaltet. Erst wenn hier 'yes' eingetragen wird, können USB-Geräte überhaupt verwendet werden. Sollten Sie also in der base.txt ein USB-Gerät ausgewählt haben, so müssen Sie hier zwingend 'yes' eintragen. Andernfalls wird das Gerät nicht verwendet.

Mit der Aktivierung ist auch der Support für USB Sticks, externen Laufwerken und Tastaturen eingeschaltet.

Standard-Einstellung: `OPT_USB='no'`

USB_EXTRA_DRIVER_N Anzahl der zusätzlich zu ladenden Treiber.

Standard-Einstellung: `USB_EXTRA_DRIVER_N='0'`

USB_EXTRA_DRIVER_x Treiber der geladen werden soll.

Mögliche Werte im Moment

- printer - Unterstützung für USB-Drucker
- belkin_sa - USB Belkin Serial converter
- cyberjack - REINER SCT cyberJack pinpad/e-com USB Chipcard Reader
- digi_acceleport - Digi AccelePort USB-2/USB-4 Serial Converter
- empeg - USB Empeg Mark I/II
- ftdi_sio - USB FTDI Serial Converter
- io_edgeport - Edgeport USB Serial
- io_ti - Edgeport USB Serial
- ipaq - USB PocketPC PDA
- ir-usb - USB IR Dongle
- keyspan - Keyspan USB to Serial Converter
- keyspan_pda - USB Keyspan PDA Converter
- kl5kusb105 - KLSI KL5KUSB105 chipset USB->Serial Converter
- kobil_sct - KOBIL USB Smart Card Terminal (experimental)
- mct_u232 - Magic Control Technology USB-RS232 converter
- omninet - USB ZyXEL omni.net LCD PLUS
- pl2303 - Prolific PL2303 USB to serial adaptor
- visor - USB HandSpring Visor / Palm OS
- whiteheat - USB ConnectTech WhiteHEAT

Standard-Einstellung: `USB_EXTRA_DRIVER_x=''`

USB_EXTRA_DRIVER_x_PARAM Parameter für den zusätzlichen Treiber. Im Normalfall muss hier nichts eingegeben werden.

Standard-Einstellung: `USB_EXTRA_DRIVER_x_PARAM=''`

USB_MODEM_WAITSECONDS Standard-Einstellung: `USB_MODEM_WAITSECONDS='21'`

Leider brauchen das Eagle und das Speedtouch USB Modem eine halbe Ewigkeit bis sie bereit sind. In den meisten Fällen reichen die 21 Sekunden, die als Standardeinstellung genommen werden, für die Initialisierung aus. Manchmal hat man das Glück das man den Wert auch halbieren kann und sein Eagle oder Speedtouch USB Modem bereits nach 10 Sekunden einsatzbereit ist, dann kann man hier halt 10 Sekunden eintragen. Wenn man Pech hat muss man den Wert erhöhen. Hier hilft leider nur probieren und austesten.

4.22.1. Probleme mit USB-Geräten

Es kann bei einigen USB-Geräten zu Problemen kommen. Das kann verschiedene Ursachen haben, wie Beispielsweise der Treiber-Software oder dem USB-Controller.

In der vorliegenden Version funktioniert das Eagle-USB-DSL-Modem nur dann, wenn es auch an den Splitter angeschlossen ist. Sollte das nicht der Fall sein, wird das entsprechende eth-Device nicht generiert. Das hat zur Folge, das das Modem nicht verwendbar ist. Schliessen Sie also bitte vorher das Modem an die Telefonleitung an.

4.22.2. Hinweise zur Benutzung

Es ist darauf zu achten, dass die USB-Unterstützung hardwareseitig aktiviert ist. Insbesondere bei Onboard-USB-Kontrollern ist das wichtig. So wird z. B. ein WRAP ohne USB-Anschluss ausgeliefert. USB kann hier durch ein Zusatzmodul nachgerüstet werden und ist aus diesem Grund im BIOS standardmäßig deaktiviert.

4.22.3. Mounten von USB-Geräten

Eingesteckte USB-Geräte werden zwar automatisch erkannt, müssen aber 'von Hand' sowohl an- als auch abgemeldet werden. Beim Einstecken z. B. eines USB-Stick wird dieser als SCSI-Device erkannt. Aus diesem Grund erfolgt der Zugriff über das Device sd# bei SuperFloppy-Geräten bzw. über sd#<Partitionsnummer> bei Geräten mit einer Partitionstabelle. USB-Sticks werden wie Festplatten behandelt, also bei zwei USB-Anschlüssen sda1 und sdb1 angesprochen. USB-Floppies hingegen werden durch sda bzw. sdb angesprochen, also ohne Angabe einer Partitionsnummer.

Somit kann ein USB-Stick durch das Kommando
mount /dev/sda1 /mnt
nach /mnt gemountet werden. Analog dazu durch
mount /dev/sdb1 /mnt
für das zweite USB-Gerät. Die Geräte werden in der Reihenfolge des Einsteckens benannt, also erstes USB-Gerät = sda, zweites USB-Gerät = sdb etc. pp. Es lässt sich somit nicht fix definieren, welcher der USB-Ports welche 'Bezeichnung' hat, da diese von der Reihenfolge des Einsteckens der Geräte abhängt. Die Abmeldung der angemeldeten USB-Geräte erfolgt durch
umount /mnt

Bei gleichzeitiger Verwendung mehrerer USB-Geräte sollte es unbedingt vermieden werden, alles in ein Ziel zu mounten. Aus diesem Grund bietet es sich an, unterhalb von /mnt weitere Verzeichnisse anzulegen, in welche die Geräte dann gemountet werden können. Dies kann z. B. wie folgt erledigt werden:
mkdir /mnt/usba mkdir /mnt/usbb
Beim Mounten der Geräte werden dann diese Verzeichnisse als Ziel angegeben:
mount /dev/sda1 /mnt/usba mount /dev/sdb1 /mnt/usbb
Somit ist der Inhalt der USB-Geräte unter /mnt/usba bzw. /mnt/usbb zu finden. Die Abmeldung erfolgt dann durch
umount /mnt/usba umount /mnt/usbb
Wenn mehrere Partitionen je USB-Gerät existieren, müssen die Verzeichnisse unterhalb von /mnt entsprechend strukturiert werden.

4.23. WLAN - Wireless-LAN Unterstützung

Achten Sie in jedem Fall darauf, dass Sie beim Einsatz von PCI Karten ein Mainboard benutzen, was mindestens die PCI 2.2 Spezifikationen erfüllt. Auf älteren Mainboard die nur PCI 2.1 oder älter unterstützen kann es zu den unterschiedlichsten Fehler kommen. Entweder startet der Computer gar nicht (er läßt sich nicht einmal einschalten), oder die WLAN-Karte wird beim PCI Scan nicht gefunden.

WLAN-Karten werden in der base.txt `IP_NET_X_DEV` mit wlanX angesprochen. Wenn nur eine WLAN-Karte im System ist, hat diese also den Namen wlan0.

4.23.1. WLAN-Konfiguration

OPT_WLAN Standard-Einstellung: `OPT_WLAN`='no'

Aktiviert das Wireless LAN Option Pack.

WLAN_WEBGUI Standard-Einstellung: `WLAN_WEBGUI`='yes'

Aktiviert das Webinterface für das Wireless LAN Option Pack.

WLAN_REGDOMAIN Mit dieser Variable kann man die Landesspezifischen Einstellungen anpassen. Gültige Werte sind ISO 3166-1 alpha-2 Ländercodes wie z.B. 'DE' In verschiedenen Ländern gelten verschiedene Vorgaben für die Kanalauswahl und Sendeleistungen.

WLAN_N Anzahl der voneinander unabhängigen WLAN-Konfigurationen. Steht hier eine '1' so ist das Verhalten wie in früheren Versionen von fli4l.

WLAN_x_MAC MAC-Adresse der WLAN-Karte in dieser Schreibweise:

XX:XX:XX:XX:XX:XX

Jedes X ist ein Hex-Digit der Mac-Adresse der Karte, für die diese Konfiguration gelten soll. Sollte keine der hier eingetragenen Mac-Adressen zu einer Karte passen, so wird die Konfiguration `WLAN_1_*` auf diese Karte angewandt und es wird eine Warnmeldung ausgegeben, die auf den Umstand hinweist. Die Warnmeldung enthält die festgestellte MAC-Adresse der Karte. Diese ist in der Konfiguration einzutragen, damit auch das Web-Interface problemlos funktionieren kann.

WLAN_x_MAC_OVERRIDE Ändert die MAC-Adresse der WLAN-Karte damit man als Client an ein WLAN mit MAC-Filter verbinden kann ohne dort den Filter anpassen zu müssen. Hilfreich bei WAN-Anbindungen, die z.B. auf die MAC-Adresse eines gelieferten WLAN-USB-Sticks gebunden sind.

WLAN_x_ESSID Die SSID ist der Name für das Funknetzwerk. Die auch "Network Name" genannte Zeichenfolge kann bis zu 32 Zeichen lang sein. Sie wird im AP eines WLAN konfiguriert und von allen Clients, die darauf Zugriff haben sollen, eingestellt. Auch bei Ad-Hoc muß die SSID auf allen teilnehmenden Nodes identisch sein.

WLAN_x_MODE Stellt den zu verwendenden WLAN-Modus der Karte ein.

Standard-Einstellung: `WLAN_x_MODE`='ad-hoc'

Mögliche Werte:

ad-hoc	für ein Funknetz ohne Access-Point
managed	gemanagedtes Funknetz mit mehreren Zellen
master	die WLAN-Karte arbeitet als Access-Point

`WLAN_x_MODE`='master' funktioniert nur mit einem geeigneten WLAN-Treiber.

WLAN_x_NOESSID Ermöglicht das Abschalten der ESSID in den Beacon Frames. Nur möglich mit Treiber hostap_* und Firmware >= 1.6.3 im WLAN_MODE='master'

Dieses Feature ist optional und muß manuell zur config/wlan.txt hinzugefügt werden.

WLAN_x_CHANNEL Setzt den Übertragungskanal des Netzwerks.

Standard-Einstellung: `WLAN_x_CHANNEL`='1'

Mögliche Werte: 1-13 und 36,40,44,48,52,56,60,64,100,104,108,112,116,120,124,128,132,136,140

Bitte lesen sie die Dokumentation Ihrer WLAN-Karte um herauszufinden, welche Kanäle in Ihrem Land erlaubt sind. Sollten sie hier einen nicht erlaubten Kanal einstellen, so sind sie alleine dafür verantwortlich. In Deutschland sind die Kanäle 1-13 im Frequenzband 2,4 GHz (Modi: b und g) erlaubt. Die Kanäle im Bereich 36-140 (siehe oben) sind im 5 GHz zulässig.

Desweiteren ist der Wert '0' erlaubt, falls `WLAN_x_MODE`='managed' gesetzt ist. Dadurch wird kein expliziter Kanal eingestellt, sondern der AP auf allen verfügbaren Kanälen gesucht. Man kann dem Kanal-Wert auch einen Buchstaben a,b oder g anhängen (z.B. 5g), welcher dann den gewünschten Betriebsmodus/Frequenzband auswählt.

Ein angehängtes 'n' oder 'N' selektiert bei entsprechenden WLAN-Karten die Nutzung von 802.11n. Kleingeschrieben bedeutet: 20 MHz Kanalbreite, grossgeschrieben: 40 MHz Kanalbreite.

Großschreibung bei a/b/g sorgt bei einigen (aktuell nur ath_pci) Treibern dafür, dass proprietäre WLAN-Turbos aktiviert werden. Diese Option ist experimentell und kann auch wieder entfernt werden.

WLAN_x_RATE Setzt die Übertragungsgeschwindigkeit des Netzwerks.

Standard-Einstellung: `WLAN_x_RATE`='auto'

Mögliche Werte: 1,2,5.5,11,auto - Angaben in Megabit/s je nach Karte können auch noch diese Raten ausgewählt werden: 6,9,12,18,24,36,48 und 54. Bei manchen 54 MBit-Karten kann die Rate nicht angegeben werden. Hier ist dann 'auto' einzutragen.

WLAN_x_RTS Aktiviert RTS/CTS Handshake. Diese Option ist in grossen Wlans mit vielen sendenden Clients nuetzlich wenn sich die Clients gegenseitig nicht hoeren koennen sondern nur den AP. Ist diese Option aktiviert sendet der Client vor jedem Sendevorgang ein RTS mit der Bitte um Erlaubnis zum Senden und bekommt ein CTS, die Erlaubnis zum Senden, vom AP zurueck. Damit weiss jeder Client dass ein Client sendet auch wenn er diesen Client nicht hoert. Hierdurch werden Kollisionen vermindert weil sicher gestellt ist dass immer nur ein Client sendet. Diese Option macht nur unter der oben beschriebenen Situation Sinn weil sie zusaetzlichen overhead hinzufuegt und somit die Gesamtbandbreite verringert. Durch die Verringerung von Kollisionen kann sich die Bandbreite jedoch wieder erhoehen.

Dieses Feature ist optional und muß manuell zur config/wlan.txt hinzugefügt werden.

WLAN_x_ENC_N (Überholt) Legt die Anzahl der Wireless Encryption Key's fest (WEP).

Mögliche Werte: 0-4

WLAN_x_ENC_x (Überholt) Setzt die Wireless Encryption Keys.

Mögliche Werte:

XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XX	128 Bit Hex-Key (X=0-F)
XXXX-XXXX-XX	64 Bit Hex-Key (X=0-F)
s:<5 Zeichen>	64 Bit
s:<6-13 Zeichen>	128 Bit
P:<1-64 Zeichen>	128 Bit

Das Verfahren der Key-Vergabe mit s:Text ist _nicht_ mit der Passphrase der Windows-Treiber kompatibel. Hier bitte einen Hex-Key verwenden! Unter Windows wird der Hex-Key meist _ohne_ die Bindestriche '-' verwendet. Die Angabe mittels P:<Text> ist kompatibel zur Passphrase der meisten Windows WLAN-Treiber (wenn nicht allen) aber _nur_ im 128 Bit Modus. Linux erlaubt es, verschiedene Schlüssellängen zu mischen. Windows-Treiber jedoch in der Regel _nicht_!

WLAN_x_ENC_ACTIVE (Überholt) Legt den aktiven Wireless Encryption Key fest.

Mögliche Werte: 1-4

Diese Variable ist aufzunehmen, wenn `WLAN_x_ENC_N` > 0 gesetzt wird. Ansonsten optional.

WLAN_x_ENC_MODE (Überholt) Aktiviert den Encryption Mode.

Mögliche Werte:

on/off	mit oder ohne Veschlüsselung
open	nimmt auch unverschlüsselte Pakete an
restricted	nimmt nur veschlüsselte Pakete an

Sinnvoller Wert: 'restricted'

Dieses Feature ist optional und muß manuell zur config/wlan.txt hinzugefügt werden. Ist die Variable nicht vorhanden, so wird als Default 'off' angenommen, wenn kein WEP-Key definiert ist und 'restricted' wenn mindestens ein Key definiert ist.

WLAN_x_WPA_KEY_MGMT Will man statt WEP-Verschlüsselung WPA verwenden, stellt man hier den gewünschten WPA-Modus ein. Momentan wird nur WPA-PSK unterstützt, also WPA mit einem Client und Access-Point vorab bekannten Schlüssel. Dieser Schlüssel sollte sorgfältig gewählt werden und nicht zu kurz sein, da er ansonsten auch anfällig gegen Wörterbuchattacken ist.

Unterstützt werden im *managed*-Mode alle vom Wpa-Supplicanten (`http://hostap.epitest.fi/wpa_supplicant/` und im *master*-Mode alle vom Hostapd (`http://hostap.epitest.fi/hostapd/`) unterstützten Karten.

Erfolgreich getestet wurden bereits Karten basierend auf den Chipsätzen von Atheros und vom hostap-Treiber unterstützte Karten (sowohl im managed als auch im master mode). Theoretisch ist auch noch Unterstützung für atmel-Karten und einige andere möglich. Hier müssen die Ersteller der entsprechenden Opts aber ihre Opt-Pakete noch entsprechend anpassen.

WLAN_x_WPA_PSK Hier wird der Schlüssel angegeben, der zur Kommunikation zwischen Client und Access-Point verwendet werden soll. Dieser Schlüssel wird in Form einer Passphrase (eines Satzes) angegeben, die mindestens 16 Zeichen lang sein muß und bis zu 63 Zeichen lang sein kann. Folgende Zeichen werden unterstützt:

a-z A-Z 0-9 ! # $ % & ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~

WLAN_x_WPA_TYPE Zur Auswahl stehen hier 1 für WPA1, 2 für den WPA2 (IEEE 802.11i) Modus und 3 für beide - der Client kann dann entscheiden ob er WPA1 oder WPA2 nutzen möchte. Wenn die WLAN Hardware den Standard unterstützt, so ist dem sicheren WPA2 Verfahren den Vorzug zu geben.

WLAN_x_WPA_ENCRYPTION Die Verschlüsselungsprotokolle TKIP und die erweiterte Version CCMP (AES-CTR/CBC-MAC Protocol, manchmal auch nur AES genannt) stehen hier zur Verfügung. CCMP wird eventuell nicht von älterer WLAN-Hardware unterstützt. Es können auch beide gemeinsam angegeben werden.

WLAN_x_WPA_DEBUG (Experimentell) Bei Problemen mit der WPA-Anbindung kann man diese Variable auf 'yes' setzen, um den zuständigen daemon zu umfangreicheren Ausgaben zu veranlassen. Diese kann man dann zur Diagnose der Probleme verwenden.

WLAN_x_AP Registriert diese Node bei einem Access-Point.

Hier ist die MAC-Adresse des Access-Points anzugeben. Wenn man bereits den WLAN-Mode "master" ausgewählt hat, ist dieser Eintrag leer zu lassen. Diese Option ist nur dann sinnvoll, wenn der fli4l den AP nicht von selber finden kann oder an einen bevorzugten Access-Point gebunden werden soll. Nur zum Einsatz in WLAN-Mode "managed" gedacht.

Dieses Feature ist optional und muß manuell zur config/wlan.txt hinzugefügt werden.

WLAN_x_ACL_POLICY Policy der ACL.

Standard-Einstellung: `WLAN_x_ACL_POLICY`='allow'

Beschreibt eine Aktion, der die angegebenen MAC-Adressen unterliegen:

deny	Keine der aufgelisteten MAC-Adressen erhält Zugang
allow	Nur aufgelistete MAC-Adressen erhalten Zugang
open	Alle MAC-Adressen erhalten unabhängig vom Filter Zugang

Leider werden WLAN_ACL's aktuell nur von einem Treiber sauber unterstützt: hostap_* Als Alternative bieten sich die in 3.0.x deutlich erweiterten Firewall-Möglichkeiten an.

WLAN_x_ACL_MAC_N AP-ACLs - Einschränkung der erlaubten WLAN-Stationen.

Standard-Einstellung: `WLAN_x_ACL_MAC_N`='0'

Eine Zahl größer 0 aktiviert die Access Control List (der MAC-Adressenfilter) und gibt die Anzahl der ACL-Einträge an. Die Access Control List ist eine Liste von MAC-Adressen, denen der Zugang zum Access Point (AP) erlaubt/verboten wird. Anzahl der Mac-Adressen, die definiert werden.

WLAN_x_ACL_MAC_x Mac-Adressen in der Form: 00:00:E8:83:72:92

WLAN_x_DIVERSITY Hiermit kann man einstellen, ob manuelle Antennen-Diversity aktiviert wird.

Standard-Einstellung: `WLAN_x_DIVERSITY`='no' (automatische Wahl)

WLAN_x_DIVERSITY_RX Auswahl der Empfangsantenne.

Standard-Einstellung: `WLAN_x_DIVERSITY_RX`='1'

0 = Automatische Auswahl
1 = Antenne 1
2 = Antenne 2

WLAN_x_DIVERSITY_TX Auswahl der Sendeantenne.

Standard-Einstellung: `WLAN_x_DIVERSITY_TX`='1'

WLAN_x_WPS Aktiviert den WPS-Support. Push-Button und PIN ist möglich. Es ist sinnvoll, WLAN_WEBGUI zu aktivieren, es sei denn es ist nur die Steuerung per Commandline gewünscht.

Standard-Einstellung: `WLAN_x_WPS`='no'

WLAN_x_PSKFILE Bei aktiviertem PSKFILE können neben dem unter `WLAN_x_WPA_PSK` konfigurierten Preshared Key auch weitere Client-bezogene Keys genutzt werden. Aktuell nutzt die Funktion `WLAN_x_WPS` dieses File um darüber konfigurierten Clients individuelle Keys zu geben.

Wird das File abgeschaltet sind auch bisher mit aktiviertem File verbundene WPS-Clients nicht mehr in der Lage mit dem AccessPoint zu verbinden.

WPS-Clients, die mit abgeschaltetem File verbunden wurden, sind davon nicht betroffen.

Standard-Einstellung: `WLAN_x_PSKFILE`='yes'

WLAN_x_BRIDGE Alternativ zur Angabe im Paket ADVANCED_NETWORKING kann hier umgekehrt angegeben werden an welche Bridge das WLAN gebunden werden soll.

Beispiel: `WLAN_x_BRIDGE`='br0'

Achtung: Entweder in Advanced-Network oder hier angeben! Nicht an beiden Stellen!

4.23.2. Beispiele

Anbindung an einen Access Point via WPA

```
OPT_WLAN='yes'
WLAN_N='1'
WLAN_1_MAC='00:0F:A3:xx:xx:xx'
WLAN_1_ESSID='foo'
WLAN_1_MODE='managed'           # Anbindung an Access Point
WLAN_1_CHANNEL='1'
WLAN_1_RATE='auto'
#
# WPA Konfiguration
#
WLAN_1_ENC_N='0'                # kein WEP
```

```
WLAN_1_WPA_KEY_MGMT='WPA-PSK'   # WPA pre shared key
WLAN_1_WPA_TYPE='1'             # WPA 1
WLAN_1_WPA_ENCRYPTION='TKIP'
WLAN_1_WPA_PSK='Deine gute Passphrase (16-63 Zeichen)'
#
# irrelevant im WPA Kontext
#
WLAN_1_ENC_N='0'
WLAN_1_ENC_ACTIVE='1'
WLAN_1_ACL_POLICY='allow'
WLAN_1_ACL_MAC_N='0'
```

Access Point mit WPA2 Verschlüsselung

```
OPT_WLAN='yes'
WLAN_N='1'
WLAN_1_MAC='00:0F:A3:xx:xx:xx'
WLAN_1_ESSID='foo'
WLAN_1_MODE='master'            # Access Point
WLAN_1_CHANNEL='1g'             # Channel 1, Modus 'g' auf einer
                                # Atheros-Karte
WLAN_1_RATE='auto'
#
# WPA Konfiguration
#
WLAN_1_ENC_N='0'                # kein WEP
WLAN_1_WPA_KEY_MGMT='WPA-PSK'   # WPA pre shared key
WLAN_1_WPA_TYPE='2'             # WPA 2
WLAN_1_WPA_ENCRYPTION='CCMP'
WLAN_1_WPA_PSK='Deine gute Passphrase (16-63 Zeichen)'
#
# MAC basierte Zugriffkontrolle auf AP
#
WLAN_1_ACL_POLICY='allow'
WLAN_1_ACL_MAC_N='0'
#
# irrelevant im WPA Kontext
#
WLAN_1_ENC_ACTIVE='1'
```

Access Point mit WEP Verschlüsselung

```
OPT_WLAN='yes'
WLAN_N='1'
WLAN_1_MAC='00:0F:A3:xx:xx:xx'
WLAN_1_ESSID='foo'
WLAN_1_MODE='master'            # Access Point
WLAN_1_CHANNEL='1'
WLAN_1_RATE='auto'
#
# WEP Konfiguration
#
WLAN_1_WPA_KEY_MGMT=''          # kein WPA
```

```
WLAN_1_ENC_N='4'                # 4 WEP-Keys
WLAN_1_ENC_1='...'
WLAN_1_ENC_2='...'
WLAN_1_ENC_3='...'
WLAN_1_ENC_4='...'
WLAN_1_ENC_ACTIVE='1'           # erster Schlüssel ist aktiv
#
# MAC basierte Zugriffkontrolle auf AP
#
WLAN_1_ACL_POLICY='allow'
WLAN_1_ACL_MAC_N='0'
#
# irrelevant für WEP Konfiguration
#
WLAN_1_WPA_TYPE='2'
WLAN_1_WPA_ENCRYPTION='CCMP'
WLAN_1_WPA_PSK='...'
```

4.23.3. Virtual Accesspoint (VAP)(Experimentell)

Bestimmte WLAN-Karten (Treiber: ath_pci, ath5k, ath9k, ath9k_htc) können auf bis zu 4 virtuelle WLAN-Karten aufgeteilt werden. (VAP)

Die WLAN-Konfiguration der virtuellen AP kann beliebig sein bis auf folgende Bedingung: Gleich sein muss: Kanal und MAC-Adresse. Anhand der mehrfach verwendeten MAC-Adresse, wird die Karte identifiziert, die aufgesplittet werden soll. Bei mehreren verbauten Karten kann dies auch mehrfach gemacht werden.

Das Basis-Device wird weiterhin wlan0 heissen (bei einer WLAN-Karte). Bei VAP dann wlan0v2 usw. Zum binden an eine Bridge bitte hier WLAN_x_BRIDGE='br0' usw. verwenden!.

Das aktuelle Maximum ist: je nach Karte und Treiber bis zu 8x Master.

4.23.4. Zeitgesteuertes ein– und ausschalten mit easycron

Mittels *easycron* (Seite 125), einem anderen Paket, kann das WLAN ein– und ausgeschaltet werden.

```
EASYCRON_N='2'
EASYCRON_1_CUSTOM  = ''     # Jeden Abend um 24 Uhr ausschalten
EASYCRON_1_COMMAND = '/usr/sbin/wlanconfig.sh wlan0 down'
EASYCRON_1_TIME    = '* 24 * * *'

EASYCRON_2_CUSTOM  = ''     # und um 8 Uhr wieder an.
EASYCRON_2_COMMAND = '/usr/sbin/wlanconfig.sh wlan0'
EASYCRON_2_TIME    = '* 8 * * *'
```

4.23.5. Spendenhinweis

Durch die großzügige Spende von 2 Ralink 2500 basierten WLAN-Karten können WLAN-Karten mit dem RT25xx Chipsatz mit fli4l in den Modi ad-hoc und managed verwendet werden. Als Treiber ist in der base.txt hierzu rt2500 auszuwählen. Die Karten wurden gespendet von:

Computer Contor, Pilgrimstein 24a, 35037 Marburg

4.24. SRC - Das fli4l-Buildroot

Dieses Kapitel ist hauptsächlich für Entwickler interessant, die Binärprogramme oder Linux-Kernel für den fli4l übersetzen wollen. Wenn Sie fli4l nur als Router einsetzen und keine Pakete für den fli4l anbieten wollen, die eigene Binärprogramme benötigen, können Sie dieses Kapitel komplett überspringen.

Generell ist für die Übersetzung von Programmpaketen für den fli4l ein Linux-System erforderlich. Eine Übersetzung unter anderen Betriebssystemen (Microsoft Windows, OS X, FreeBSD etc.) wird *nicht* unterstützt.

Die Anforderungen an ein Linux-System zur fli4l-Entwicklung sind wie folgt:

- GNU gcc und g++ in der Version 2.95 oder neuer
- GNU gcc-multilib (je nach Hostsystem nötig)
- GNU binutils (enthält den Binder sowie andere, notwendige Programme)
- GNU make in der Version 3.81 oder neuer
- GNU bash
- libncurses5-dev für `fbr-make *-menuconfig` (je nach Hostsystem nötig)
- die Programme sed, awk, which, flex, bison und patch
- die Programme makeinfo (Paket texinfo) und msgfmt (Paket gettext)
- die Programme tar, cpio, gzip, bzip2 und unzip
- die Programme wget, rsync, svn und git
- die Programme perl und python

Im Folgenden repräsentieren **fett** gedruckte Zeichen zu tätigende Eingaben, das ↵-Zeichen steht für die Eingabetaste Ihrer Tastatur und schließt einzugebende Befehle ab.

4.24.1. Eine Übersicht über die Quellen

Im `src`-Verzeichnis finden Sie folgende Unterverzeichnisse:

Verzeichnis	Inhalt
`fbr`	In diesem Verzeichnis befindet sich ein angepasstes Buildsystem, das auf dem Buildroot zur uClibc (aktuell in der Version 0.9.33.2) basiert. FBR steht hierbei für "fli4l-Buildroot". Damit ist es möglich, alle auf dem fli4l verwendeten Programme (Kernel, Anwendungen und Bibliotheken) neu zu übersetzen.

Verzeichnis	Inhalt
`fli4l`	Dieses Verzeichnis enthält die fli4l-spezifischen Quellen, nach Paketen geordnet. Alle Quellen, die in diesem Unterverzeichnis enthalten sind, wurden entweder speziell für die Verwendung mit fli4l geschrieben oder zumindest stark angepasst.
`cross`	In diesem Verzeichnis befinden sich Skripte, mit denen die Cross-Compiler erstellt werden können, die für das Übersetzen von mkfli4l für diverse Plattformen benötigt werden.

4.24.2. Übersetzen eines Programms für den fli4l

Im Unterverzeichnis "fbr" finden Sie das Skript `fbr-make`, das die Übersetzung aller Programme aus den Basispaketen für den fli4l-Router steuert. Dieses Skript kümmert sich um das Herunterladen und Übersetzen aller für den fli4l benötigten Binärdateien. Generell legt das Skript Dateien in dem Verzeichnis `~/.fbr` ab; existiert dieses noch nicht, wird es angelegt. (Dieses Verzeichnis kann mit Hilfe der Umgebungsvariable `FBR_BASEDIR` verändert werden, siehe unten.)

Hinweis: Während des Übersetzungsvorgangs wird viel Platz benötigt (momentan etwa 900 MiB für die heruntergeladenen Archive und knapp 30 GiB für die Zwischenergebnisse und die resultierenden Kompilate). Stellen Sie somit sicher, dass Sie unterhalb von `~/.fbr` über genug Platz verfügen! (Alternativ können Sie auch die `FBR_TIDY`-Option nutzen, siehe unten.)

Die Verzeichnisstruktur unterhalb von `~/.fbr` ist wie folgt:

Verzeichnis	Inhalt
`fbr-<branch>-<arch>`	Hierhin wird das uClibc-Buildroot entpackt. `<branch>` steht hierbei für den Entwicklungszweig (z.B. `trunk`), aus dem das FBR stammt. Ist der Ursprung des FBRs ein entpacktes `src`-Paket, wird `fbr-custom` benutzt. `<arch>` steht für die jeweilige Prozessorarchitektur (z.B. `x86` oder `x86_64`). Mehr zu diesem Verzeichnis steht weiter unten.
`dl`	Hier werden die heruntergeladenen Archive gespeichert.
`own`	Hier können eigene FBR-Pakete abgelegt werden, die ebenfalls übersetzt werden sollen.

Unterhalb des Buildroot-Verzeichnisses `~/.fbr/fbr-<branch>-<arch>/buildroot` sind die folgenden Verzeichnisse interessant:

Verzeichnis	Inhalt
`output/sandbox`	In diesem Verzeichnis gibt es für jedes FBR-Paket ein Unterverzeichnis, das die Dateien des FBR-Pakets nach dem Übersetzungsvorgang aufnimmt. In dem Verzeichnis `output/sandbox/<paket>/target` befinden sich dabei die Dateien, die für den fli4l-Router vorgesehen sind. In dem Verzeichnis `output/sandbox/<paket>/staging` hingegen befinden sich Dateien, die zum Übersetzen *anderer* FBR-Pakete, die dieses FBR-Paket benötigen, erforderlich sind.
`output/target`	In diesem Verzeichnis werden *alle* übersetzen Programme für den fli4l-Router abgelegt. Dieses Verzeichnis spiegelt somit die Verzeichnisstruktur auf dem fli4l-Router wider. Mit Hilfe von `chroot` kann man in dieses Verzeichnis wechseln und die übersetzten Programme ausprobieren.[16]

Allgemeine Einstellungen

Die Arbeitsweise von `fbr-make` kann durch verschiedene Umgebungsvariablen beeinflusst werden:

Variable	Beschreibung
`FBR`	Gibt den Pfad zum FBR explizit an. Standardmäßig wird der Pfad `~/.fbr/fbr-<branch>-<arch>` (s.o.) verwendet.
`FBR_BASEDIR`	Gibt den Basispfad zum FBR explizit an. Standardmäßig wird der Pfad `~/.fbr` (s.o.) verwendet. Diese Variable wird ignoriert, falls die Umgebungsvariable `FBR` gesetzt wird.
`FBR_DLDIR`	Gibt das Verzeichnis an, das die Quellarchive enthält. Standardmäßig wird der Pfad `${FBR}/../dl` (also z.B. `~/.fbr/dl`) verwendet.
`FBR_OWNDIR`	Gibt das Verzeichnis an, das die eigenen Pakete enthält. Standardmäßig wird der Pfad `${FBR}/../own` (also z.B. `~/.fbr/own`) verwendet.

[16] Dies ist an gewisse Voraussetzungen geknüpft, siehe hierzu den Abschnitt "Testen eines übersetzten Programms" (Seite 263).

Variable	Beschreibung
`FBR_TIDY`	Wenn diese Variable den Wert "y" enthält, werden Zwischenergebnisse, die während des Bauens der FBR-Pakete entstehen, unmittelbar nach der Installation in das Verzeichnis `output/target` gelöscht. Das spart viel Speicherplatz und ist eigentlich immer empfehlenswert, wenn man nach dem Bauen von Paketen nicht den Drang verspürt, in `output/build/...` hineinzuschauen. Falls diese Variable den Wert "k" enthält, werden nur die Zwischenergebnisse in den diversen Linux-Kernel-Verzeichnissen entfernt, weil dies verhältnismäßig viel Platz spart, ohne dass dadurch Funktionalität verloren geht. Alle anderen Belegungen (oder wenn die Variable gänzlich fehlt) sorgen dafür, dass alle Zwischenergebnisse erhalten bleiben.
`FBR_ARCH`	Diese Variable gibt die Prozessorarchitektur an, für die das FBR (bzw. einzelne FBR-Pakete) gebaut werden sollen. Fehlt sie, wird `x86` angenommen. Die unterstützten Architekturen sind weiter unten zu finden.

Momentan unterstützt das FBR die folgenden Architekturen:

Architektur	Beschreibung
`x86`	Intel x86-Architektur (32-Bit), auch IA-32 genannt.
`x86_64`	AMD x86-64-Architektur (64-Bit), von Intel auch Intel 64 oder EM64T genannt.

Übersetzen aller FBR-Pakete

Wenn Sie `fbr-make` mit dem Argument `world` ausführen, dauert es je nach verwendetem Rechner und verwendeter Internetanbindung mehrere Stunden, bis alle Quellarchive heruntergeladen und übersetzt worden sind.[17]

Übersetzen des Toolchains

Wenn Sie `fbr-make` mit dem Argument `toolchain` ausführen, werden alle FBR-Pakete heruntergeladen und übersetzt, die für das Bauen der eigentlichen fli4l-Binärprogramme benötigt werden (also Übersetzer, Binder, uClibc-Bibliothek etc.). Normalerweise wird dieses Kommando nicht benötigt, da alle FBR-Pakete vom Toolchain abhängig sind und somit diese Toolchain-Programme ohnehin heruntergeladen und gebaut werden.

Übersetzen eines einzelnen FBR-Pakets

Wollen Sie hingegen nur ein bestimmtes FBR-Paket übersetzen (etwa die Programme für ein selbst entwickeltes OPT), können Sie den Namen des FBR-Pakets bzw. die Namen mehrerer

[17] Das Herunterladen der Quellarchive wird natürlich nur einmal durchgeführt, so lange Sie das FBR nicht aktualisieren und dadurch neuere Paketversionen andere Quellarchive benötigen.

FBR-Pakete dem `fbr-make`-Programm mitgeben (etwa `fbr-make openvpn` zum Herunterladen und Übersetzen der OpenVPN-Programme). Dabei werden alle nötigen Abhängigkeiten ebenfalls heruntergeladen und übersetzt.

Erneutes Übersetzen eines einzelnen FBR-Pakets

Möchten Sie ein bestimmtes FBR-Paket erneut übersetzen (warum auch immer), müssen Sie zuerst die Informationen im FBR über den vorher stattgefundenen Übersetzungsvorgang entfernen. Dazu können Sie den Befehl `fbr-make <paket>-clean` (z.B. `fbr-make openvpn-clean`) verwenden. Dabei werden die Informationen all jener FBR-Pakete, die von dem angegebenen FBR-Paket abhängig sind, ebenfalls zurückgesetzt, so dass sie beim nächsten `fbr-make world` ebenfalls neu übersetzt werden.

Erneutes Übersetzen aller FBR-Pakete

Möchten Sie das komplette FBR neu übersetzen (z.B. weil Sie es als Benchmark-Programm für Ihr neues High-End-Entwicklersystem nutzen wollen ;-), können Sie mit Hilfe des Kommandos `fbr-make clean` nach der Bestätigung einer Sicherheitsabfrage alle Artefakte entfernen, die während des letzten FBR-Baus erzeugt worden sind.[18] Dies ist auch nützlich, um belegten Plattenspeicher freizugeben.

4.24.3. Testen eines übersetzten Programms

Ist ein Programm mit `fbr-make` übersetzt worden, kann es auf dem Entwicklungsrechner auch getestet werden. Ein solcher Test funktioniert natürlich nur, wenn die Prozessorarchitektur des Entwicklerrechners mit der Prozessorarchitektur des fli4ls, für welchen die Programme übersetzt werden sollen, übereinstimmt. (Es ist z.B. nicht möglich, `x86_64`-fli4l-Programme auf einem `x86`-Betriebssystem auszuführen.) Ist diese Voraussetzung erfüllt, kann man mit

```
chroot ~/.fbr/fbr-<branch>-<arch>/buildroot/output/target /bin/sh↵
```

in das fli4l-Zielverzeichnis wechseln und dort das/die übersetzte(n) Programm(e) direkt ausprobieren. Beachten Sie jedoch bitte, dass Sie für `chroot` Administrator-Rechte benötigen und daher je nach Vorliebe und Systemkonfiguration die Dienste von `sudo` oder `su` in Anspruch nehmen müssen! Auch müssen Sie zumindest das FBR-Paket `busybox` übersetzt haben (via `fbr-make busybox`), damit Sie in der `chroot`-Umgebung eine Shell zur Verfügung haben. Ein kleines Beispiel:

```
$ sudo chroot ~/.fbr/fbr-trunk-x86/buildroot/output/target /bin/sh↵
Passwort:(Ihr Passwort)↵


BusyBox v1.22.1 (fli4l) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# ls↵
THIS_IS_NOT_YOUR_ROOT_FILESYSTEM  mnt
bin                               opt
dev                               proc
```

[18] Es wird das gesamte Verzeichnis `~/.fbr/fbr-<branch>-<arch>/buildroot/output` entfernt.

```
etc                             root
home                            run
img                             sbin
include                         share
lib                             sys
lib32                           tmp
libexec                         usr
man                             var
media                           windows
# bc --version↵
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
# echo "42 - 23" | bc↵
19
#
```

4.24.4. Entwanzen eines übersetzten Programms

Macht ein Programm auf dem fli4l Probleme, mit anderen Worten: es stürzt ab, dann hat man die Möglichkeit, den Programmzustand unmittelbar vor dem Absturz nachträglich zu analysieren (auch "Post-Mortem Debugging" genannt). Hierzu muss man zuerst in der Konfiguration des `base`-Pakets `DEBUG_ENABLE_CORE='yes'` aktivieren. Wird dann beim Absturz ein Speicherabbild in `/var/log/dumps/core.<PID>` generiert, wobei "PID" die Prozessnummer des abgestürzten Prozesses ist, dann kann man den Zustand des Programms auf einem Linux-Rechner mit einem voll übersetzten FBR folgendermaßen analysieren. Im folgenden Beispiel ist das zu analysierende Programm `/usr/sbin/collectd`, das sich mit einem SIGBUS verabschiedet hatte. Das Speicherabbild liegt dabei in `/tmp/core.collectd`.

```
fli4l@eisler:~$ .fbr/fbr-trunk-x86/buildroot/output/host/usr/bin/i586-linux-gdb↵
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-unknown-linux-gnu --target=i586-buildr
oot-linux-uclibc".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) set sysroot /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target↵
(gdb) set debug-file-directory /project/fli4l/.fbr/fbr-trunk-x86/buildroot/outpu
t/debug↵
(gdb) file /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/usr/sbin/co
llectd↵
Reading symbols from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/u
sr/sbin/collectd...Reading symbols from /project/fli4l/.fbr/fbr-trunk-x86/buildr
oot/output/debug/.build-id/8b/28ab573be4a2302e1117964edede2e54ebbdbf.debug...don
e.
done.
(gdb) core /tmp/core.collectd↵
[New LWP 2250]
[New LWP 2252]
```

```
[New LWP 2259]
[New LWP 2257]
[New LWP 2255]
[New LWP 2232]
[New LWP 2235]
[New LWP 2238]
[New LWP 2242]
[New LWP 2244]
[New LWP 2245]
[New LWP 2231]
[New LWP 2243]
[New LWP 2251]
[New LWP 2248]
[New LWP 2239]
[New LWP 2229]
[New LWP 2249]
[New LWP 2230]
[New LWP 2247]
[New LWP 2233]
[New LWP 2256]
[New LWP 2236]
[New LWP 2246]
[New LWP 2240]
[New LWP 2241]
[New LWP 2237]
[New LWP 2234]
[New LWP 2253]
[New LWP 2254]
[New LWP 2258]
[New LWP 2260]
Failed to read a valid object file image from memory.
Core was generated by `collectd -f'.
Program terminated with signal 7, Bus error.
#0  0xb7705f5d in memcpy ()
   from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libc.so.0
(gdb) backtrace↵
#0  0xb7705f5d in memcpy ()
   from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libc.so.0
#1  0xb768a251 in rrd_write (rrd_file=rrd_file@entry=0x808e930, buf=0x808e268,
    count=count@entry=112) at rrd_open.c:716
#2  0xb76834f3 in rrd_create_fn (
    file_name=file_name@entry=0x808d2f8 "/data/rrdtool/db/vm-fli4l-1/cpu-0/cpu-i
nterrupt.rrd.async", rrd=rrd@entry=0xacff2f4c) at rrd_create.c:727
#3  0xb7683d7b in rrd_create_r (
    filename=filename@entry=0x808d2f8 "/data/rrdtool/db/vm-fli4l-1/cpu-0/cpu-int
errupt.rrd.async", pdp_step=pdp_step@entry=10, last_up=last_up@entry=1386052459,
    argc=argc@entry=16, argv=argv@entry=0x808cf18) at rrd_create.c:580
#4  0xb76b77fd in srrd_create (
    filename=0xacff33f0 "/data/rrdtool/db/vm-fli4l-1/cpu-0/cpu-interrupt.rrd.asy
nc",
    pdp_step=10, last_up=1386052459, argc=16, argv=0x808cf18) at utils_rrdcreate
.c:377
#5  0xb76b78cb in srrd_create_thread (targs=targs@entry=0x808bab8)
```

```
    at utils_rrdcreate.c:559
#6  0xb76b7a8f in srrd_create_thread (targs=0x808bab8) at utils_rrdcreate.c:491
#7  0xb7763430 in ?? ()
   from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libpthread
.so.0
#8  0xb775e672 in clone ()
   from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libpthread
.so.0
(gdb) frame 1↵
#1  0xb768a251 in rrd_write (rrd_file=rrd_file@entry=0x808e930, buf=0x808e268,
    count=count@entry=112) at rrd_open.c:716
716         memcpy(rrd_simple_file->file_start + rrd_file->pos, buf, count);
(gdb) print (char*) buf↵
$1 = 0x808e268 "RRD"
(gdb) print rrd_simple_file->file_start↵
value has been optimized out
(gdb) list↵
711         if((rrd_file->pos + count) > old_size)
712         {
713             rrd_set_error("attempting to write beyond end of file");
714             return -1;
715         }
716         memcpy(rrd_simple_file->file_start + rrd_file->pos, buf, count);
717         rrd_file->pos += count;
718         return count;       /* mimmic write() semantics */
719     #else
720         ssize_t   _sz = write(rrd_simple_file->fd, buf, count);
(gdb) list 700↵
695      * rrd_file->pos of rrd_simple_file->fd.
696      * Returns the number of bytes written or <0 on error.  */
697
698     ssize_t rrd_write(
699         rrd_file_t *rrd_file,
700         const void *buf,
701         size_t count)
702     {
703         rrd_simple_file_t *rrd_simple_file = (rrd_simple_file_t *)rrd_file->
pvt;
704     #ifdef HAVE_MMAP
(gdb) print *(rrd_simple_file_t *)rrd_file->pvt↵
$2 = {fd = 9, file_start = 0xa67d0000 <Address 0xa67d0000 out of bounds>,
  mm_prot = 3, mm_flags = 1}
```

Hier sieht man nach etwas “Wühlen”, dass sich in dem `rrd_simple_file_t`-Objekt ein ungültiger Zeiger befindet (“Address ... out of bounds”). Im weiteren Debugging-Verlauf wurde deutlich, dass ein gescheiterter `posix_fallocate`-Aufruf die Ursache für den Programmabsturz war.

Wichtig hierbei ist, dass *alle* anzugebenden Pfade voll qualifiziert sind (`/project/...`) und dass man auch keine “Abkürzungen” (etwa `~/...`) verwendet. Wenn man dies nicht beachtet, kann es passieren, dass `gdb` die Debug-Informationen zur Anwendung und/oder zu den verwendeten Bibliotheken nicht findet. Die Debug-Informationen sind nämlich ais Platzgründen nicht direkt in dem untersuchten Programm enthalten, sondern in einer separaten Datei unterhalb

des Verzeichnisses `~/.fbr/fbr-<branch>-<arch>/buildroot/output/debug/` gespeichert.

4.24.5. Informationen über das FBR

Anzeige der Hilfe

Was `fbr-make` alles für Sie tun kann, können Sie sich mit dem Kommando `fbr-make help` ausgeben lassen.

Anzeige von Programminformationen

Sie können sich alle verfügbaren FBR-Pakete sowie deren Versionen anschauen, indem Sie das Kommando `fbr-make show-versions` nutzen:

```
$ fbr-make show-versions↵
Configured packages

acpid 2.0.20
actctrl 3.25+dfsg1
add-days undefined
[...]
```

Anzeige von Bibliotheksabhängigkeiten

Mit Hilfe des Kommandos `fbr-make links-against <soname>` können Sie sich alle Dateien in `~/.fbr/fbr-<branch>-<arch>/buildroot/output/target` anzeigen lassen, die gegen eine Bibliothek mit dem Bibliotheksnamen `soname` gebunden sind. Um beispielsweise alle Programme und Bibliotheken zu identifizieren, welche die `libm` (Bibliothek mit mathematischen Funktionen) verwenden, nutzen Sie das Kommando `fbr-make links-against libm.so.0`, da `libm.so.0` der Bibliotheksname der libm-Bibliothek ist. Eine mögliche Ausgabe ist:

```
$ fbr-make links-against librrd_th.so.4↵
Executing plugin links-against
Files linking against librrd_th.so.4
collectd usr/lib/collectd/rrdcached.so
collectd usr/lib/collectd/rrdtool.so
rrdtool usr/bin/rrdcached
```

Dabei steht in der ersten Spalte der Paketname und in der zweiten der (relative) Pfad zu der Datei, die gegen die angegebenene Bibliothek gebunden ist.

Um den Bibliotheksnamen für eine Bibliothek herauszufinden, können Sie `readelf` wie folgt nutzen:

```
$ readelf -d ~/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libm-0.9.33.2.so |↵
> grep SONAME↵
 0x0000000e (SONAME)                     Library soname: [libm.so.0]
```

Anzeige von Versionsänderungen

(Nur) für fli4l-Team-Entwickler mit Schreibzugriff auf das fli4l-SVN-Repository ist das Kommando `fbr-make version-changes` interessant. Es listet alle FBR-Pakete auf, deren Version lokal modifiziert wurde, deren Version in der Arbeitskopie also von der Repository-Version

abweicht. Damit kann der Entwickler sich einen Überblick über aktualisierte FBR-Pakete verschaffen, etwa bevor er die Änderungen ins Repo schreibt. Eine mögliche Ausgabe ist:

```
$ fbr-make version-changes↵
Executing plugin version-changes
Package version changes
KAMAILIO: 4.0.5 --> 4.1.1
```

Hier sieht man sofort, dass das `kamailio`-FBR-Paket von der Version 4.0.5 auf die Version 4.1.1 aktualisiert worden ist.

4.24.6. Ändern der FBR-Konfiguration

Rekonfiguration des FBRs

Mit Hilfe des Kommandos `fbr-make buildroot-menuconfig` ist es zum einen möglich, die zu übersetzenden FBR-Pakete auszuwählen. Das ist nützlich, wenn Sie andere FBR-Pakete für den fli4l übersetzen möchten, die standardmäßig nicht aktiviert sind, aber im uClibc-Buildroot integriert sind, oder wenn Sie eigene FBR-Pakete aktivieren wollen. Zum anderen können andere, globale Eigenschaften des FBRs verändert werden, etwa die Version des verwendeten GCC-Compilers. Beim erfolgreichen Beenden des Konfigurationsmenüs wird die neue Konfiguration im Verzeichnis `src/fbr/buildroot/.config` gespeichert.

Beachten Sie jedoch bitte, dass solche Änderungen der Toolchain-Konfiguration offiziell *nicht* unterstützt werden, weil die resultierenden Binärprogramme mit hoher Wahrscheinlichkeit inkompatibel mit der offiziellen fli4l-Distribution werden. Wenn Sie also Binärprogramme für Ihr eigenes OPT benötigen und dieses OPT veröffentlichen wollen, sollten Sie keine Toolchain-Einstellung verändern!

Rekonfiguration der uClibc-Bibliothek

Mit dem Kommando `fbr-make uclibc-menuconfig` kann die Funktionalität der verwendeten uClibc-Bibliothek verändert werden. Beim erfolgreichen Beenden des Konfigurationsmenüs wird die neue Konfiguration in `src/fbr/buildroot/package/uclibc/uclibc.config` gespeichert.

Wie im letzten Abschnitt gilt auch hier: Eine Änderung ist mit hoher Wahrscheinlichkeit nicht kompatibel mit der offiziellen fli4l-Distribution und wird somit nicht unterstützt!

Rekonfiguration der Busybox

Mit Hilfe des Kommandos `fbr-make busybox-menuconfig` kann die Busybox in ihrer Funktionalität angepasst werden. Beim erfolgreichen Beenden des Konfigurationsmenüs wird die neue Konfiguration in `src/fbr/buildroot/package/busybox/busybox-<Version>.config` gespeichert.

Auch hier gilt: Eine Änderung ist höchstwahrscheinlich nicht kompatibel mit der offiziellen fli4l-Distribution und wird somit nicht unterstützt! Allenfalls das

Ergänzen der Busybox um neue Applets ist problemlos, solange Sie die so modifizierte Busybox nur auf Ihren fli4l-Routern einsetzen (und nicht vom Nutzer Ihres OPTs den Einsatz einer derart modifizierten Busybox verlangen).

Rekonfiguration der Linux-Kernelpakete

Mit `fbr-make linux-menuconfig` bzw. `fbr-make linux-<version>-menuconfig` kann die Konfiguration aller aktivierten Kernel-Pakete bzw. eines bestimmten Kernel-Pakets vorgenommen werden. Beim erfolgreichen Beenden des Konfigurationsmenüs wird die neue Konfiguration in `src/fbr/buildroot/linux/linux-<version>/dot-config-<arch>` gespeichert.[19]

Wie im letzten Abschnitt gilt auch hier: Eine Änderung ist höchstwahrscheinlich nicht kompatibel mit der offiziellen fli4l-Distribution und wird somit nicht unterstützt! Allenfalls das Ergänzen des Linux-Kernels um neue Module ist problemlos, solange Sie den so modifizierten Kernel nur auf Ihren fli4l-Routern einsetzen (und nicht vom Nutzer Ihres OPTs den Einsatz eines derart modifizierten Kernels verlangen).

4.24.7. Aktualisierung des FBRs

Jedem der beschriebenen Kommandos geht eine Prüfung des FBRs auf Aktualität voraus. Wird eine Diskrepanz zwischen den Quellen, in denen sich `fbr-make` befindet (also entpacktes `src`-Paket oder SVN-Arbeitskopie) und dem FBR in `~/.fbr/fbr-<branch>-<arch>/buildroot` entdeckt, wird letzteres auf den neuesten Stand gebracht. Dabei werden neu dazugekommene FBR-Pakete integriert sowie alte, nicht mehr enthaltene FBR-Pakete gelöscht. Auch die Konfigurationen werden verglichen: FBR-Pakete mit veränderter Konfiguration sowie alle davon abhängigen FBR-Pakete werden neu gebaut. Dies stellt sicher, dass das FBR auf Ihrem Computer immer dem der fli4l-Entwickler entspricht (mit Ausnahme Ihrer eigenen FBR-Pakete unterhalb von `~/.fbr/own/`). **Das bedeutet jedoch auch, dass Änderungen am offiziellen Teil der Buildroot-Konfiguration bei der nächsten Aktualisierung verloren gehen!** Auch deshalb ist eine Rekonfiguration des FBRs (s.o.) somit nicht zu empfehlen, zumindest nicht, wenn Sie mit `src`-Paketen anstatt mit SVN-Arbeitskopien arbeiten. (Bei der Aktualisierung einer SVN-Arbeitskopie werden Ihre lokalen Konfigurationsänderungen und die Änderungen im SVN-Repository zusammengeführt, so dass das Problem der verlorenen Konfiguration hier nicht auftritt.) Hingegen können Sie Ihre eigenen FBR-Pakete problemlos umkonfigurieren, ohne dass es bei einer Aktualisierung zu Datenverlusten kommt.

4.24.8. Eigene Programme ins FBR einbinden

Die Übersetzung der einzelnen FBR-Pakete wird über kleine Makefiles gesteuert. Will man eigene FBR-Pakete entwickeln, so muss man ein Makefile sowie eine Konfigurationsbeschreibung unter `~/.fbr/own/<paket>/` ablegen. Wie diese aufgebaut sind und wie man daraus abgeleitet eigene Makefiles schreiben kann, wird in der Dokumentation des uClibc-Buildroots unter http://buildroot.uclibc.org/downloads/manual/manual.html#adding-packages ausführlich beschrieben.

[19] Dies gilt nur für den Standard-Kernel. Für Varianten eines Kernelpakets wird stattdessen eine `diff`-Datei in `src/fbr/buildroot/linux/linux-<version>/linux-<version>_<variante>/dot-config-<arch>.diff` abgelegt.

5. Erzeugen der fli4l Archive/Bootmedien

Sind alle Konfigurationsarbeiten erledigt, können die fli4l Archive/Bootmedien, sei es eine bootfähige Compact-Flash, ein bootfähiges ISO-Image oder nur die zum Remote-Update benötigten Dateien, erstellt werden.

5.1. Erzeugen der fli4l Archive/Bootmedien unter Linux bzw. anderen Unix-Derivaten und Mac OS X

Dies geschieht mit Hilfe von Scripts (`.sh`), die im fli4l Wurzelverzeichnis zu finden sind.

`mkfli4l.sh`

Das Build-Script erkennt selbständig die unterschiedlichen Bootvarianten (Seite 25).
Der einfachste Aufruf sieht unter Linux so aus:

```
sh mkfli4l.sh
```

Die Aktionen des Build-Scripts werden durch drei Mechanismen gesteuert:

- Konfigurationsvariable `BOOT_TYPE` aus der `<config>/base.txt`
- Konfigurationsdatei `<config>/mkfli4l.txt`
- Parameter des Build-Scripts

An Hand der Konfigurationsvariable `BOOT_TYPE` (Seite 25) entscheidet sich, welche Aktion des Build-Scripts ausgeführt wird:

- Erstellen eines bootfähigen fli4l CD-ISO-Images
- Bereitstellen der fli4l Dateien, zwecks Remote-Update
- Erzeugen der fli4l Dateien und direktes Remote-Update per SCP
- usw.

Die Beschreibung der Variablen der Konfigurationsdatei `<config>/mkfli4l.txt` finden Sie im Kapitel Steuerungsdatei mkfli4l.txt (Seite 278).

5.1.1. Kommandozeilenoptionen

Der letzte Steuerungsmechanismus ist das Anhängen von Optionsparametern an den Aufruf des Build-Script auf der Kommandozeile. Die Steuerungsmöglichkeiten entsprechen denen der Steuerungsdatei `mkfli4l.txt`. Die Angabe von Optionsparametern überschreiben die Werte aus der Steuerungsdatei. Aus Komfortgründen unterscheiden sich die Namen der Optionsparameter von den Namen der Variablen aus der Steuerungsdatei. Es existiert teilweise eine Kurz- und eine Langform:

```
Usage: mkfli4l.sh [options] [config-dir]

-c, --clean         cleanup the build-directory
-b, --build <dir>   set build-directory to <dir> for the fli4l-files
-h, --help          display this usage
--batch             don't ask for user input

config-dir          set other config-directory - default is "config"

--hdinstallpath <dir> install a pre-install environment directly to
                    usb/compact flash device mounted or mountable to
                    directory <dir> in order to start the real installation
                    process directly from that device
                    device either has to be mounted and to be writable
                    for the user or it has to be mountable by the user
                    Do not use this for regular updates!

*** Remote-Update options
--remoteupdate        remote-update via scp, implies "--filesonly"
--remoteremount       make /boot writable before copying files and
                      read only afterwards
--remoteuser <name>   user name for remote-update - default is "fli4l"
--remotehost <host>   hostname or IP of remote machine - default
                      is HOSTNAME set in [config-dir]/base.txt
--remotepath <path>   pathname on remote maschine - default is "/boot"
--remoteport <portnr> portnumber of the sshd on remote maschine

*** Netboot options (only on Unix/Linux)
--tftpbootpath <path>   pathname to tftpboot directory
--tftpbootimage <name>  name of the generated bootimage file
--pxesubdir <path>      subdirectory for pxe files relative to tftpbootpath

*** Developer options
-u, --update-ver    set version to <fli4l_version>-rev<svn revision>
-v, --verbose       verbose - some debug-output
-k, --kernel-pkg    create a package containing all available kernel
                    modules and terminate afterwards.
                    set COMPLETE_KERNEL='yes' in config-directory/_kernel.txt
                    and run mkfli4l.sh again without -k to finish
```

```
--filesonly     create only fli4l-files - do not create a boot-media
--no-squeeze    don't compress shell scripts
--rebuild       rebuild mkfli4l and related tools; needs make, gcc
```

Eine HD-Vorinstallation einer passend formatierten (FAT16/FAT32) CompactFlash im USB-Cardreader oder eines USB-Sticks ist über die Option `--hdinstallpath <dir>` möglich. Dieses können Sie *auf eigenes Risiko* zur Installation auf eine CompactFlash oder einen USB-Stick benutzen. Hierbei werden auf die angegebene Partition die nötigen Dateien des fli4l kopiert. Sie rufen dazu zunächst im fli4l-Verzeichnis

```
sh mkfli4l.sh --hdinstallpath <dir>
```

auf. Dabei werden die fli4l Dateien auf eine CF-Card oder USB-Stick kopiert.

Um die nächsten Schritte ausführen zu können, sind folgende Voraussetzungen zu erfüllen:

- `chmod 777 /dev/brain`
- superuser-Rechte
- installiertes `syslinux`
- installiertes `fdisk`

Durch das Script erfolgt eine Kontrolle, ob dieser Datenträger tatsächlich ein USB-Laufwerk ist und die erste Partition eine FAT-Partition ist. Anschliessend werden der Bootloader und die nötigen Dateien auf den angegebenen Datenträger kopiert. Sie erhalten eine Meldung über den Erfolg oder Misserfolg.

Nach dem Build müssen Sie

```
syslinux --mbr /dev/brain

 # make partition bootable using fdisk
 #     p - print partitions
 #     a - toggle bootable flag, specify number of fli4l partition
 #         usually '1'
 #     w - write changes and quit
 fdisk /dev/brain

 # install boot loader
 syslinux -i /dev/brain
```

ausführen. Dann sollte die CF bzw. der USB-Stick bootfähig sein. Vergessen Sie nicht, den Datenträger auszuhängen (via `umount`).

Als letzter Optionsparameter kann ein alternatives Konfigurationverzeichnis übergeben werden. Das normale Konfigurationsverzeichnis heißt `config` und liegt direkt im fli4l Wurzelverzeichnis. An diesem Ort legen alle fli4l Pakete die Konfigurationsdateien ab. Möchte man mehr

als eine Konfiguration verwalten, so erstellt man sich ein weiteres Verzeichnis, z.B. `hd.conf`, legt dort eine Kopie der Konfigurationsdateien ab und verändert diese den Anforderungen entsprechend. Hier einige Beispiele:

```
sh mkfli4l.sh --filesonly hd.conf
sh mkfli4l.sh --no-squeeze config.test
```

5.2. Erzeugen der fli4l Archive/Bootmedien unter Windows

Es wird das Tool 'AutoIt3' verwendet (http://www.autoitscript.com/site/autoit/). Dieses ermöglicht eine 'grafische' Ausgabe, sowie Dialoge, mit denen die in den folgenden Abschnitten beschriebenen Variablen beinflusst werden können.

`mkfli4l.bat`

Das Build-Programm erkennt selbständig die unterschiedlichen Bootvarianten (Seite 25).

Der Aufruf von 'mkfli4l.bat' kann direkt aus dem Windows Explorer erfolgen, wenn man keine optionalen Parameter verwenden möchte.

Die Aktionen des Build-Programms werden durch verschiedene Mechanismen gesteuert:

- Konfigurationsvariable `BOOT_TYPE` aus der `<config>/base.txt`
- Konfigurationsdatei `<config>/mkfli4l.txt`
- Parameter des Build-Programmes
- Interaktive Einstellung in der GUI

An Hand der Konfigurationsvariable `BOOT_TYPE` (Seite 25) entscheidet sich, welche Aktion das Build-Programm ausführt:

- Erstellen eines bootfähigen fli4l CD-ISO-Images
- Bereitstellen der fli4l Dateien, zwecks Remote-Update
- Erzeugen der fli4l Dateien und direktes Remote-Update per SCP
- HD-pre-install einer passend formatierten CF im Cardreader
- usw.

Die Beschreibung der Variablen der Konfigurationsdatei `<config>/mkfli4l.txt` finden Sie im Kapitel Steuerungsdatei mkfli4l.txt (Seite 278).

5.2.1. Kommandozeilenoptionen

Ein weiterer Steuerungsmechanismus ist das Anhängen von Optionsparametern an den Aufruf des Build-Programms auf der Kommandozeile. Die Steuerungsmöglichkeiten entsprechen denen der Steuerungsdatei `mkfli4l.txt`. Die Angabe von Optionsparametern überschreiben die Werte aus der Steuerungsdatei. Aus Komfortgründen unterscheiden sich die Namen der Optionsparameter von den Namen der Variablen aus der Steuerungsdatei. Es existiert teilweise eine Kurz- und eine Langform:

```
Usage: mkfli4l.bat [options] [config-dir]

-c, --clean             cleanup the build-directory
-b, --build <dir>       sets build-directory to <dir> for the fli4l-files
-v, --verbose           verbose - some debug-output
    --filesonly         creates only fli4l-files - does not create a disk
    --no-squeeze        don't compress shell scripts
-h, --help              display this usage

config-dir              sets other config-directory - default is "config"

*** Remote-Update options
--remoteupdate          remote-update via scp, implies "--filesonly"
--remoteuser <name>     user name for remote-update - default is "fli4l"
--remotehost <host>     hostname or IP of remote machine - default
                        is HOSTNAME set in [config-dir]/base.txt
--remotepath <path>     pathname on remote maschine - default is "/boot"
--remoteport <portnr>   portnumber of the sshd on remote maschine

*** GUI-Options
--nogui                 disable the config-GUI
--lang                  change language
                        [deutsch|english|espanol|french|magyar|nederlands]
```

Als letzter Optionsparameter kann ein alternatives Konfigurationverzeichnis übergeben werden. Das normale Konfigurationsverzeichnis heißt `config` und liegt direkt im fli4l Wurzelverzeichnis. An diesem Ort legen alle fli4l Pakete die Konfirgurationsdateien ab. Möchte man mehr als eine Konfiguration verwalten, so erstellt man sich ein weiteres Verzeichnis, z.B. `hd.conf`, legt dort eine Kopie der Konfigurationsdateien ab und verändert diese den Anforderungen entsprechend. Hier einige Beispiele:

```
mkfli4l.bat hd.conf
mkfli4l.bat -v
mkfli4l.bat --no-gui config.hd
```

5.2.2. Konfigurationsdialog – Einstellung des Konfigurationsverzeichnis

Im Hauptfenster wird die Einstellung des Konfigurationsverzeichnis angezeigt und es kann ein Fenster geöffnet werden zur Auswahl des Konfigurationsverzeichnis.

Zu beachten ist, dass eine Änderung des ‘Config-Dir’ alle Optionen auf die Werte setzt, die in der dortigen Steuerungsdatei ‘mkfli4l.txt’ (Seite 278) gesetzt bzw. als Kommandozeilenparameter übergeben wurden.

Findet mkfli4l.bat kein Verzeichnis fli4l-x.y.z\config oder in dem Verzeichnis keine Datei mit dem Namen 'base.txt' öffnet sich sofort das Fenster zur Auswahl des Konfigurationsverzeichnis. Dieses ermöglicht es auf einfache Weise im fli4l-Verzeichnis mehrere Konfigurationen zu verwalten.

Beispiel:

```
fli4l-x.y.z\config
fli4l-x.y.z\config.fd
fli4l-x.y.z\config.cd
fli4l-x.y.z\config.hd
fli4l-x.y.z\config.hd-erstellen
```

5.2.3. Konfigurationsdialog – allgemeine Einstellungen

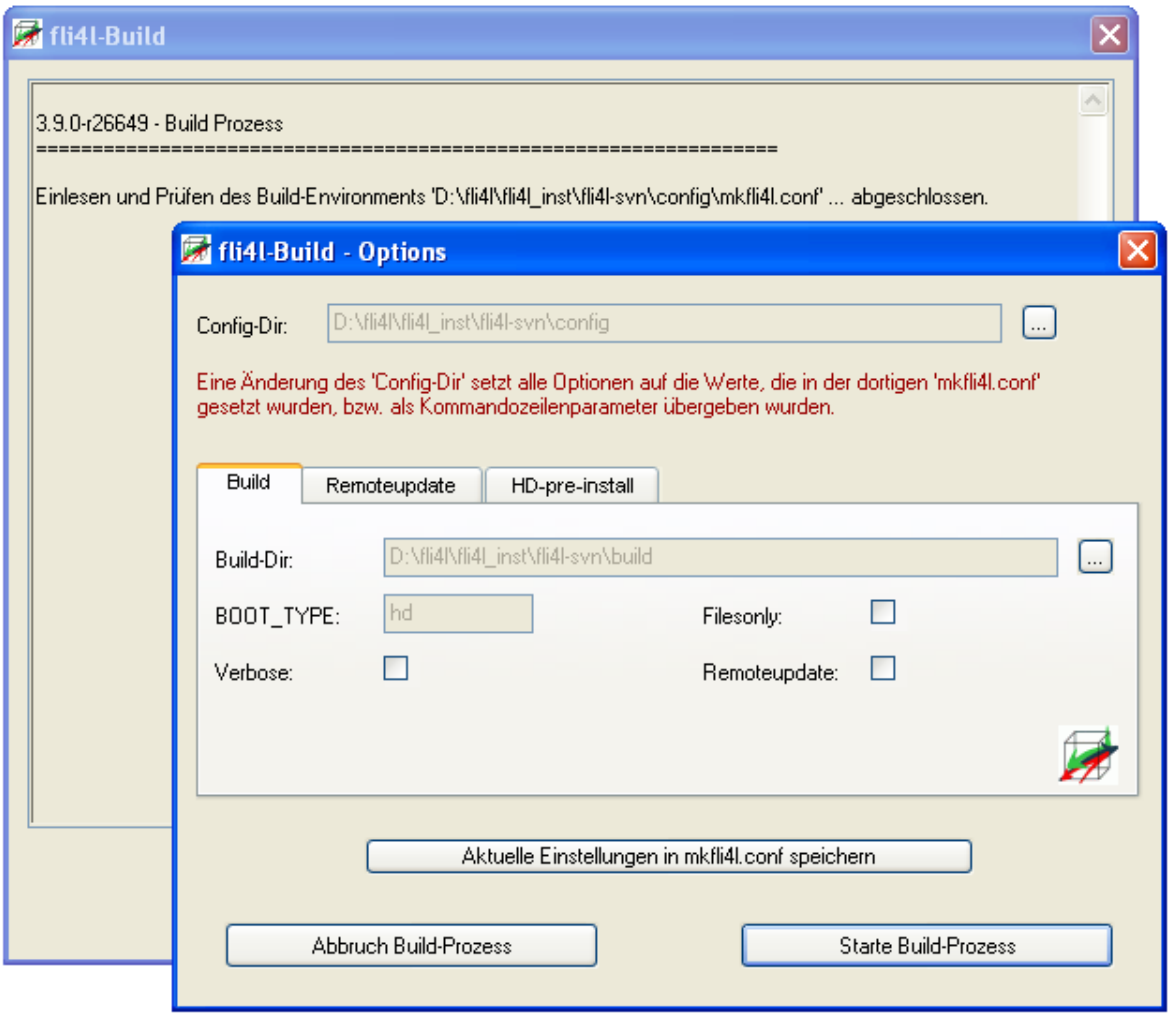


Abbildung 5.1.: Einstellungen

In diesem Dialog werden die Einstellungen für die Archiv/Bootmedienerstellung festgelegt:

- Build-Dir – Verzeichnis für die Archive/CD-Images/...
- `BOOT_TYPE` – Anzeige des verwendeten/eingestellen `BOOT_TYPE` – nicht änderbar
- Verbose – Aktivierung von zusätzlichen Ausgaben während der Erstellung
- Filesonly – es werden nur die Archive erstellt – kein bootmedium/kein Image
- Remoteupdate – Aktivierung des Remoteupdates per SCP

Mit der Schaltfläche **Aktuelle Einstellungen in mkfli4l.txt speichern** können die aktuell eingestellten Werte in der mkfli4l.txt gespeichert werden.

5.2.4. Konfigurationsdialog – Einstellungen für Remoteupdate

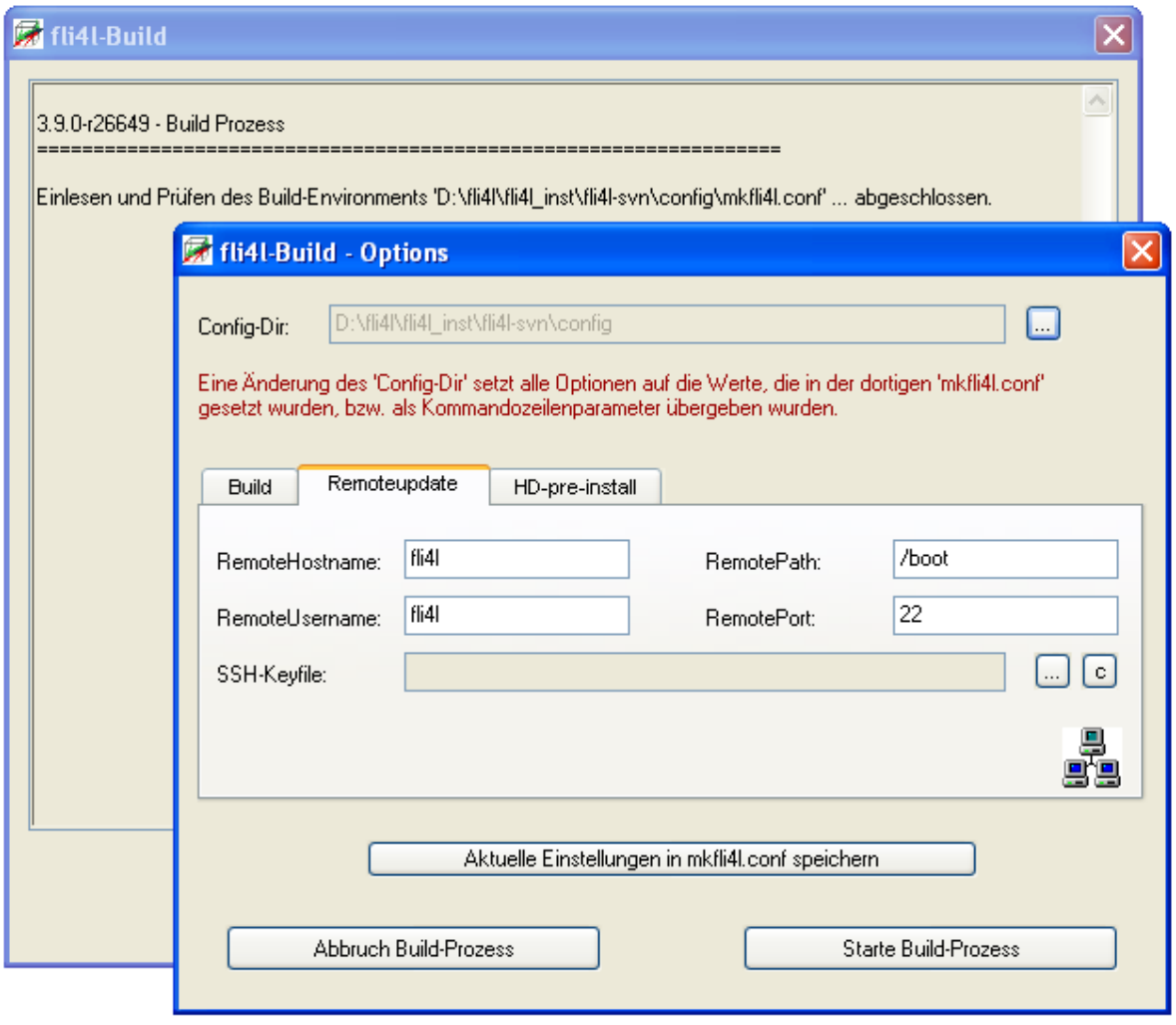


Abbildung 5.2.: Einstellungen für Remoteupdate

In diesem Dialog werden die Einstellungen für den Remoteupdate festgelegt:

- IP-Adresse oder Hostname
- Benutzername auf dem Remote-Host
- Remote-Pfad (default: /boot)
- Remote-Port (default: 22)
- zu verwendendes SSH-Keyfile (ppk-Format von Putty)

5.2.5. Konfigurationsdialog – Einstellungen für HD-pre-install

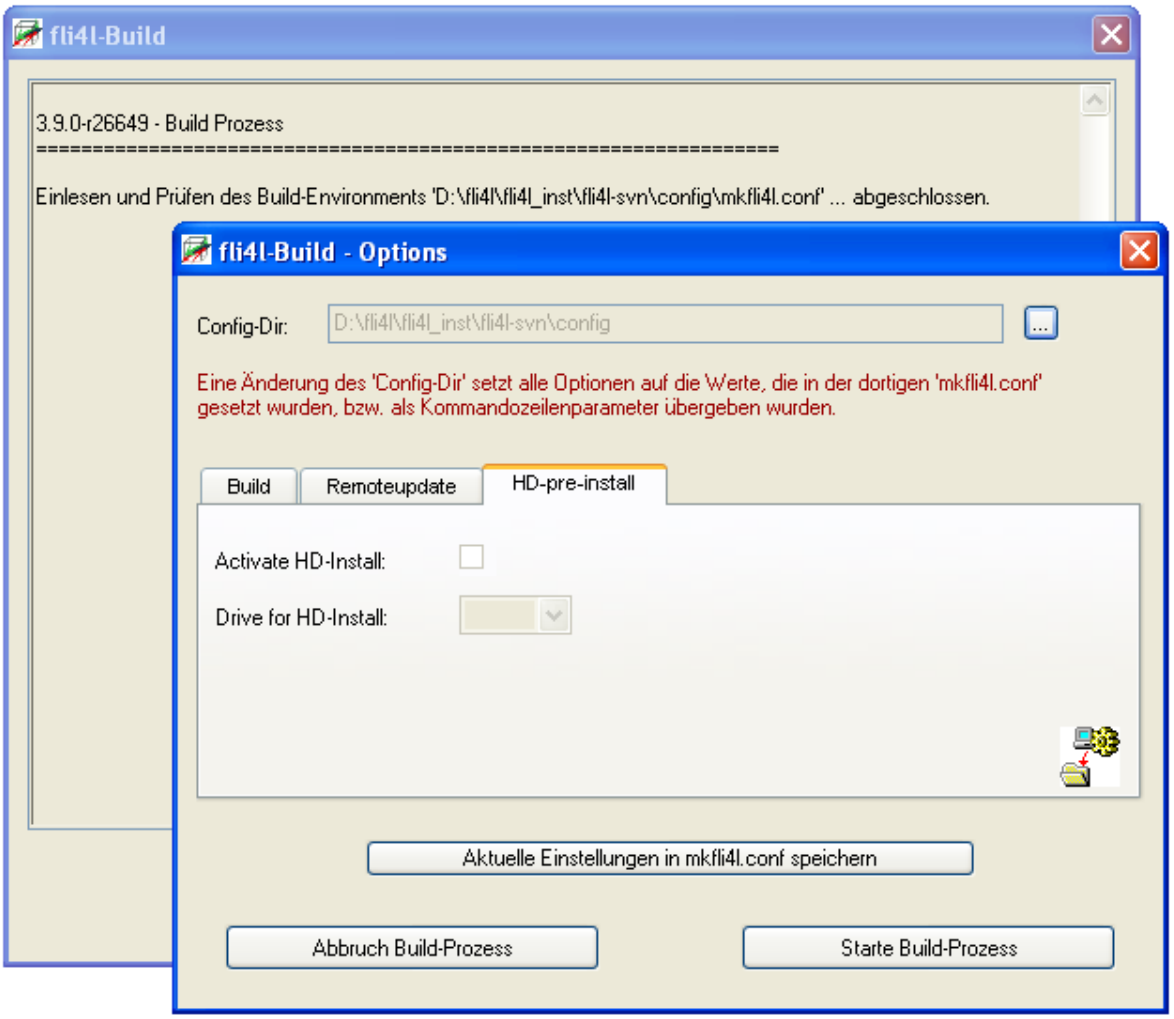


Abbildung 5.3.: Einstellungen für HD-pre-install

In diesem Dialog können die Optionen für den HD-pre-install auf einer entsprechend partitionierten und formatierten CompactFlash-Karte in einem USB-Reader eingestellt werden.
Mögliche Optionen:

- HD-pre-install aktivieren

- Laufwerksbuchstabe der CF-Karte

Hinweis zur Partionierung und Formatierung der CF: Für eine HD-Installation nach TYP A (siehe dazu Paket HD) muss auf der CF eine primäre aktive und formatierte FAT-Partition vorhanden sein. Möchte man weiterhin auch eine Datenpartiton benutzen, wird zusätzlich eine Linux-Partition, die mit dem Dateisystem ext3 formatiert ist, sowie die Datei `hd.cfg` auf der FAT-Partiton benötigt (hierzu sollten unbedingt die Hinweise im Paket HD beachtet werden).

5.3. Steuerungsdatei mkfli4l.txt

Seit fli4l-Version 2.1.9 existiert die Steuerungsdatei `<config>/mkfli4l.txt`. Durch sie werden z.B. vom Standard abweichende Verzeichnisse übergeben. Die Steuerungsdatei hat einen ähnlichen Aufbau wie die normalen fli4l Konfigurationsdateien. Alle Konfigurationsvariablen sind hier optional, d.h. sie müssen nicht in der Konfigurationsdatei vorkommen oder können als Kommentar gekennzeichnet werden.

BUILDDIR Standardwert: 'build'

Legt fest, in welchem Verzeichnis die fli4l Dateien erzeugt werden sollen. Ist die Variable undefiniert, setzt mkfli4l unter Windows 'build' relativ zum fli4l Wurzelverzeichnis ein und meint damit also das Verzeichnis `build` im fli4l Wurzelverzeichnis:

```
Pfad/fli4l-x.y.z/build
```

Unter *nix setzt mkfli4l `<config>/build` ein und legt damit die generierten Dateien zusammen mit der Konfiguration ab.

Die konfigurierten Pfade in `BUILDDIR` müssen der jeweiligen Logik von Windows oder *unix entsprechen. Werden relative Pfade gesetzt, wird der Pfad durch den Buildprozess passend zu Windows oder *unix konvertiert.

VERBOSE Standardwert: `VERBOSE='no'`

Mögliche Werte sind `'yes'` oder `'no'`. Steuert die *Geschwätzigkeit* des Build Prozesses.

FILESONLY Standardwert: `FILESONLY='no'`

Mögliche Werte `'yes'` oder `'no'`. Hiermit kann das Erstellen eines Boot-Mediums abgeschaltet werden, es werden also nur die Dateien erzeugt –

REMOTEUPDATE Standardwert: `REMOTEUPDATE='no'`

Mögliche Werte `'yes'` oder `'no'`. Aktiviert das automatische Übertragen der erstellten Dateien mittels SCP auf den Router. Dieses setzt ein installiertes Paket SSHD (Seite 236) mit aktiviertem `scp` voraus. Siehe dazu auch die folgenden Variablen.

REMOTEHOSTNAME Standardwert: `REMOTEHOSTNAME="`

Gibt den Ziel-Hostnamen für den SCP Datentransfer an. Sollte kein Name angegeben sein, wird dieser der Variable `HOSTNAME` (Seite 25) entnommen.

REMOTEUSERNAME Standardwert: `REMOTEUSERNAME='fli4l'`

Username für den SCP Datentransfer.

REMOTEPATHNAME Standardwert: `REMOTEPATHNAME='/boot'`

Ziel-Pfad für den SCP Datentransfer.

REMOTEPORT Standardwert: `REMOTEPORT='22'`

Zielport für den SCP Datentransfer.

SSHKEYFILE Standardwert: `SSHKEYFILE="`

Hier kann man eine SSH-Keydatei für den SCP-Remoteupdate angeben. Es kann somit ein Update ohne Angabe eines Passwortes erfolgen.

REMOTEREMOUNT Standardwert: `REMOTEREMOUNT='no'`

Mögliche Werte `'yes'` oder `'no'`. Wird hier `'yes'` gesetzt, wird ein eventuell Readonly eingehängtes Bootdevice "/boot"für das Remoteupdate Readwrite gemountet um das Remoteupdate möglich zu machen.

TFTPBOOTPATH Pfad an dem das Netboot-Image abgelegt wird.

TFTPBOOTIMAGE Name des Netboot-Images.

PXESUBDIR Unterverzeichnis für die PXE-Dateien relativ zu TFTPBOOTPATH.

SQUEEZE_SCRIPTS Aktiviert bzw. deaktiviert das Squeezen (Kommprimieren) von Skripten. Das Komprimieren eines Skripts mit Squeeze entfernt alle Kommentare und Zeileneinrückungen. Im Normalfall sollte hier immer der Standardwert `'yes'` benutzt werden.

MKFLI4L_DEBUG_OPTION Es können zum Debuggen zusätzliche Optionen an das mkfli4l-Programm (Seite 306) übergeben werden.

6. Anbindung von PCs im LAN

Für jeden Rechner im LAN ist einzustellen:

1. IP-Adresse (siehe IP-Adresse)
2. Name des Rechners plus Wunsch-Domain-Name (siehe Rechnername und Domain)
3. Standard-Gateway (siehe Gateway)
4. IP-Adresse des DNS-Servers (siehe DNS-Server)

6.1. IP-Adresse

Die IP-Adresse muss im gleichen Netz wie die IP-Adresse des fli4l-Routers (auf Ethernet-Seite) liegen, also z.B. 192.168.6.2, wenn der fli4l die Adresse 192.168.6.1 hat. Kein Rechner darf die gleiche IP-Adresse haben, weshalb man am besten (nur) die letzte Zahl ändert. Auch ist darauf zu achten, dass man hier die gleiche IP-Adresse angibt, wie man es für diesen Rechner in der Datei config/base.txt angegeben hat.

6.2. Rechnername und Domain

Der Name des Rechners ist dann z.B. “mein-pc”, die Domain “lan.fli4l”.

Wichtig: *Die im PC eingestellte Domain muss identisch mit der gewählten Domain im fli4l-Rechner sein, wenn man den fli4l-Router als DNS-Server verwenden will. Sonst kann es im Netz erhebliche Probleme geben.*

Grund: Windows-Rechner suchen regelmäßig nach Rechnern mit dem Namen ihrer Arbeitsgruppte: WORKGROUP.meine-domain.fli4l. Ist dies nicht die in fli4l eingestellte Domain (hier: meine-domain.fli4l), wird fli4l versuchen, diese Anfrage durch Weiterleiten ins Internet zu beantworten ...

Einzutragen ist die Domain in den TCP/IP Einstellungen des Rechners.

6.2.1. Windows 2000

Für Windows 2000 findet man das unter:
Start ⇒
Einstellungen ⇒
Systemsteuerung ⇒
Netzwerk- und DFÜ-Verbindungen ⇒
LAN-Verbindung ⇒
Eigenschaften ⇒
Internetprotokoll (TCP/IP) ⇒

Eigenschaften ⇒
Erweitert... ⇒
DNS ⇒
DNS-Suffix hinzufügen ⇒

"lan.fli4l" (bzw. die eingestellte domain) eingeben (ohne ""!) ⇒OK drücken.

6.2.2. NT 4.0

Start ⇒
Einstellungen ⇒
Systemsteuerung ⇒
Netzwerk ⇒
Protokolle ⇒
TCP/IP ⇒
Eigenschaften ⇒
DNS ⇒

- Hostname eintragen (eigener Rechnername)
- Domäne eintragen (wie in config/base.txt)
- IP-Adresse vom fli4l-Router hinzufügen
- DNS-Suffix hinzufügen (Domäne hinzufügen – siehe 2 Zeilen höher)

6.2.3. Win95/98

Start ⇒
Einstellungen ⇒
Systemsteuerung ⇒
Netzwerk ⇒
Konfiguration ⇒
TCP/IP (jenes, das an die Netzwerkkarte zum Router
angebunden ist) ⇒
Eigenschaften ⇒
DNS-Konfiguration:
DNS aktivieren und bei "Domäne:" dann "lan.fli4l" eingeben (ohne ""!).

6.2.4. Windows XP

Für Windows XP findet man das unter:
Start ⇒
Einstellungen ⇒
Systemsteuerung ⇒
Netzwerkverbindungen ⇒
LAN-Verbindung ⇒
Eigenschaften ⇒

Internetprotokoll (TCP/IP) ⇒
Eigenschaften ⇒
Erweitert... ⇒
DNS ⇒
DNS-Suffix für diese Verbindung ⇒

"lan.fli4l" (bzw. die eingestellte domain) eingeben (ohne ""!) ⇒OK drücken.

6.2.5. Windows 7

Für Windows 7 findet man das unter:
Windows Button (ex. Start) ⇒
Systemsteuerung ⇒
Netzwerk und Internet ⇒
Netzwerk- und Freigabecenter ⇒
LAN-Verbindung ⇒
Eigenschaften ⇒
Internetprotokoll Version 4 (TCP/IPv4) ⇒
Eigenschaften ⇒
Erweitert... ⇒
DNS ⇒
DNS-Suffix für diese Verbindung ⇒

"lan.fli4l" (bzw. die eingestellte domain) eingeben (ohne ""!) ⇒OK drücken.

6.2.6. Windows 8

Für Windows 8 findet man das unter:
Gleichzeitig Windows- und X-Taste drücken ⇒
Systemsteuerung ⇒
Netzwerk und Internet ⇒
Netzwerk- und Freigabecenter ⇒
Ihr Netzwerk wählen (Ehternet oder WLAN) ⇒
Eigenschaften ⇒
Internetprotokoll Version 4 (TCP/IPv4) ⇒
Eigenschaften ⇒
Erweitert... ⇒
DNS ⇒
DNS-Suffix für diese Verbindung ⇒

"lan.fli4l" (bzw. die eingestellte domain) eingeben (ohne ""!) ⇒OK drücken.

6.3. Gateway

Die Angabe des Standard-Gateways ist unbedingt erforderlich, denn ohne die Angabe der richtigen IP-Adresse an dieser Stelle funktioniert nichts. Es muß hier die IP-Adresse des fli4l-

Routers (auf Ethernet-Seite) angegeben werden, also z.B. 192.168.6.4 entsprechend der IP-Adresse, die hier in der Datei config/base.txt für den fli4l-Router angegeben wurde.

Es ist falsch, den fli4l-Router als Proxy in der Windows- oder Browser- Konfiguration einzutragen – außer man setzt ein Proxy auf dem fli4l-Router ein. Im Normalfall ist fli4l kein Proxy, daher bitte *nicht* fli4l als Proxy angeben!

6.4. DNS-Server

Als IP-Adresse des DNS-Servers gibt man nicht die Adresse des Provider-DNS-Servers an, sondern die des fli4l-Routers (Ethernet), da dieser nun selbst Anfragen beantworten kann bzw. diese bei Unkenntnis ins Internet weiterleitet.

Mit der Konstruktion von fli4l als DNS-Server werden viele von den Windows-PCs ausgeführten Anfragen nicht ins Internet weitergeroutet, sondern werden direkt vom fli4l-Router beantwortet.

6.5. Verschiedenes

Die Punkte 1 bis 4 brauchen bei konfiguriertem DHCP-Server nicht eingetragen zu werden, da dann der fli4l-Router die notwendigen Daten automatisch übermittelt.

Internetoptionen: Bei Verbindungen muß “keine Verbindung wählen” ausgewählt sein. Bei Einstellungen für lokales Netzwerk(LAN): es darf hier NICHTS angegeben werden (es sei dann es wird `OPT_Proxy` verwendet). Beides sind Standardeinstellungen, die im Normalfall nicht geändert werden müssen.

7. Client-/Server-Schnittstelle imon

7.1. imon-Server imond

imond ist ein netzwerkfähiges Server-Programm, welches bestimmte Anfragen beantwortet oder auch Kommandos zur Steuerung des Routers entgegennehmen kann.

Ausserdem steuert imond das Least-Cost-Routing. Dazu verwendet er eine Konfigurationsdatei /etc/imond.conf, welche beim Booten automatisch aus den `ISDN_CIRC_x_XXX`-Variablen der Datei config/isdn.txt und anderen über ein Shell-Script erzeugt wird.

imond läuft permanent als Daemon und horcht gleichzeitig auf TCP/IP-Port 5000 und Device /dev/isdninfo.

Folgende Kommandos sind über den TCP/IP-Port 5000 möglich:

Der TCP/IP-Port 5000 ist nur vom maskierten LAN aus erreichbar. Standardmäßig wird ein Zugriff von aussen über die Firewall-Konfiguration abgeblockt.

Imond unterstützt zwei Benutzerebenen: den User- und den Admin-Modus. Für beide Ebenen kann ein Passwort gesetzt werden mittels `IMOND_PASS` bzw. `IMOND_ADMIN_PASS`. Dadurch werden die imon-Clients von imond gezwungen, eine Password-Abfrage durchzuführen und anschließend das Password an imond zu übertragen. Solange dieses Password nicht übermittelt wurde, nimmt imond nur die beiden Kommandos “pass” und “quit” entgegen. Alle anderen werden mit einem Fehler zurückgewiesen.

Möchte man das weiter einschränken, z.B. den Zugriff nur von nur einem PC erlauben, muss die Firewall-Konfiguration angepasst werden.

Die Befehle

```
enable/disable/dialmode   dial/hangup   route   reboot/halt
```

können durch die Konfigurationsvariablen `IMOND_XXX` global ein- oder abgeschaltet werden (s. Kapitel “Konfiguration”).

Mit einem Unix/Linux-Rechner (oder einem Windows-Rechner in der DOS-Box) kann man das Ganze leicht ausprobieren:

Nach Eingabe von

```
telnet fli4l 5000        \# oder entsprechender Name des fli4l-Routers
```

kann man direkt die oben aufgeführten Kommandos eingeben und sich die Ausgabe anschauen.

Zum Beispiel bekommt man mit “help” die Hilfe angezeigt, mit “quit” wird die Verbindung zum imond abgebaut.

7.1.1. Least-Cost-Routing – Funktionsweise

imond konstruiert aus der Konfigurationsdatei /etc/imond.conf (welche wiederum beim Booten aus den Konfigurationsvariablen `ISDN_CIRC_x_TIMES` usw. erstellt wird), eine zeitabhängige

Admin-Befehle

addlink ci-index	Channel zum Circuit hinzufügen (Channel-Bundling)
adjust-time seconds	Ändert die Uhrzeit des Routers um die angegebenen Sekunden
delete filename pw	Löscht die Datei auf dem Router
hup-timeout #ci-index [value]	Anzeigen bzw. Setzen des HUP-Timeout für ISDN-Circuits
removelink ci-index	Zusätzlichen Channel wieder entfernen
reset-telmond-log-file	Löschen der telmond-Protokolldatei
reset-imond-log-file	Löschen der imond-Protokolldatei
receive filename #bytes pw	Eine Datei auf den Router übertragen. Dazu quittiert imond den Befehl mit einem ACK (0x06). Danach wird die Datei in 1024er-Blöcken übertragen, die imond auch jeweils mit einem ACK bestätigt. Als letztes übermittelt imond noch ein OK.
send filename pw	Wenn das Passwort stimmt und die Datei existiert, liefert imond ein OK #bytes. Anschliessend überträgt imond die Datei in 1024er Blöcken, die jeweils mit einem ACK (0x06) bestätigt werden müssen. Als letztes liefert imond noch ein OK.
support pw	Liefert den Status/Konfiguration vom Router
sync	Synchronisiert den Cache von gemounteten Laufwerken

Admin- oder User-Befehle

dial	Wählt den Provider an (Default-Route-Circuit)
dialmode [auto|manual|off]	Liefert bzw. setzt den Dialmode
disable	Hängt ein und setzt dialmode auf “off”
enable	Setzt dialmode auf “auto”
halt	Fährt den Router sauber herunter
hangup [#channel-id]	Hängt ein
poweroff	Fährt den Router herunter und schaltet ab
reboot	Reboot vom i4l-Router!
route [ci-index]	Setzen Default-Route auf Circuit X (0=automatisch)

User-Befehle

channels	Ausgabe Anzahl der verfügbaren ISDN-Kanäle
charge #channel-id	Ausgabe der Online-Kosten für einen Channel
chargetime #channel-id	Online-Zeit unter Berücksichtigung des Taktes
circuit [ci-index]	Ausgabe eines Circuit-Namens
circuits	Ausgabe Anzahl der Default-Route-Circuits
cpu	Liefert die Auslastung der CPU in Prozent
date	Ausgabe Datum/Uhrzeit
device ci-index	Liefert das Device des Circuits
driverid #channel-id	Ausgabe Driver-Id für Channel X
help	Ausgabe Hilfe
inout #channel-id	Ausgabe der Richtung (incoming/outgoing)
imond-log-file	Ausgabe imond-Protokolldatei
ip #channel-id	Ausgabe der IP
is-allowed command	Ausgabe, ob Befehl konfiguriert/gültig ist Mögliche Befehle: dial|dialmode|route|reboot| imond-log|telmond-log|mgetty-log
is-enabled	Ausgabe, ob dialmode auf off (0) oder auto (1)
links ci-index	Ausgabe Anzahl momentaner Channel 0, 1 oder 2, 0 heisst: Kein Channel-Bundling möglich
log-dir imond|telmond|mgetty	Liefert das Logverzeichnis
mgetty-log-file	Ausgabe mgetty-Protokolldatei
online-time #channel-id	Ausgabe Online-Zeit der akt. Verbindung in hh:mm:ss
pass [password]	Abfrage, ob Password nötig bzw. Password- Eingabe 1 Userpassword gesetzt 2 Adminpassword gesetzt 4 imond befindet sich im Admin-Modus
phone #channel-id	Ausgabe Telefonnummer/Name des “Gegners”
pppoe	Liefert die Anzahl der pppoe-Devices (also 0 oder 1)
quantity #channel-id	Liefert die übertragenen Datenmengen (in Byte)
quit	Beenden der Verbindung zu imond
rate #channel-id	Ausgabe Übertragungsraten (incoming/outgoing in B/sec)
status #channel-id	Ausgabe Status für Channel X
telmond-log-file	Ausgabe telmond-Protokolldatei
time #channel-id	Ausgabe Summe Online-Zeiten, Format hh:mm:ss
timetable [ci-index]	Ausgabe der Zeittabelle für LC-Routing
uptime	Ausgabe der Uptime des Routers in Sekunden
usage #channel-id	Ausgabe Art der Verbindung, mögliche Antworten: Fax, Voice, Net, Modem, Raw
version	Ausgabe der Protokoll- und Programm-Version

Tabelle (Time-Table). Diese umfasst eine komplette Kalenderwoche im 1-Stunden-Raster (168 Stunden = 168 Bytes). Die Tabelle setzt sich jedoch lediglich aus Circuits zusammen, für die eine Default-Route definert ist.

Mit dem imond-Kommando "timetable" kann man sich diese Tabelle anschauen.

Hier ein Beispiel:

Nehmen wir an, dass 3 Circuits definiert wurden, nämlich:

```
CIRCUIT_1_NAME='Addcom'
CIRCUIT_2_NAME='AOL'
CIRCUIT_3_NAME='Firma'
```

wobei lediglich die ersten beiden Circuits mit Default-Routen belegt sind, also die enstprechenden Variablen ISDN_CIRC_x_ROUTE den Wert '0.0.0.0' haben.

Wenn die dazugehörigen Variablen `ISDN_CIRC_x_TIMES` folgendermaßen aussehen:

```
ISDN_CIRC_1_TIMES='Mo-Fr:09-18:0.0388:N Mo-Fr:18-09:0.0248:Y
                   Sa-Su:00-24:0.0248:Y'

ISDN_CIRC_2_TIMES='Mo-Fr:09-18:0.019:Y Mo-Fr:18-09:0.049:N
                   Sa-Su:09-18:0.019:N Sa-Su:18-09:0.049:N'

ISDN_CIRC_3_TIMES='Mo-Fr:09-18:0.08:N Mo-Fr:18-09:0.03:N
                   Sa-Su:00-24:0.03:N'
```

dann wird daraus folgende Datei /etc/imond.conf:

```
#day   hour   device   defroute   phone         name         charge   ch-int
Mo-Fr  09-18  ippp0    no         010280192306  Addcom       0.0388    60
Mo-Fr  18-09  ippp0    yes        010280192306  Addcom       0.0248    60
Sa-Su  00-24  ippp0    yes        010280192306  Addcom       0.0248    60
Mo-Fr  09-18  ippp1    yes        019160        AOL  0.019    180
Mo-Fr  18-09  ippp1    no         019160        AOL  0.049    180
Sa-Su  09-18  ippp1    no         019160        AOL  0.019    180
Sa-Su  18-09  ippp1    no         019160        AOL  0.049    180
Mo-Fr  09-18  isdn2    no         0221xxxxxxx   Firma        0.08      90
Mo-Fr  18-09  isdn2    no         0221xxxxxxx   Firma        0.03      90
Sa-Su  00-24  isdn2    no         0221xxxxxxx   Firma        0.03      90
```

imond erstellt dann im Speicher folgende Time-Table – hier die Ausgabe über das imond-Kommando "timetable":

```
    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
-------------------------------------------------------------------------
Su  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
Mo  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
Tu  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
We  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
Th  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
Fr  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
Sa  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3

No.  Name                 DefRoute  Device  Ch/Min   ChInt
```

```
 1   Addcom                  no      ippp0   0.0388     60
 2   Addcom                  yes     ippp0   0.0248     60
 3   Addcom                  yes     ippp0   0.0248     60
 4   AOL             yes      ippp1   0.0190    180
 5   AOL             no       ippp1   0.0490    180
 6   AOL             no       ippp1   0.0190    180
 7   AOL             no       ippp1   0.0490    180
 8   Firma                   no      isdn2   0.0800     90
 9   Firma                   no      isdn2   0.0300     90
10   Firma                   no      isdn2   0.0300     90
```

Für den Circuit 1 (Addcom) sind also drei Zeitbereiche (1-3) eingetragen, für Circuit 2 (AOL) vier Zeitbereiche (4-7) und für den letzen drei Zeitbereiche (8-10).

In der Time-Table werden jeweils die Indices ausgegeben, welche für die jeweilige Stunde gültig sind. Hier tauchen lediglich die Indices 2-4 auf, da alle anderen keine LC-Default-Routen sind.

Sieht man in der Tabelle irgendwo Nullen, gibt es Lücken in den `ISDN_CIRC_X_TIMES`-Werten. Dann existiert zu diesen Zeiten keine Default-Route, Internet-Zugang abgeknipst!

Beim Programmstart ermittelt imond zunächst den Wochentag und die aktuelle Stunde. Anschließend wird dann über die Time-Table der Index ermittelt und damit dann auch der entsprechende Circuit. Auf diesen wird dann die Default-Route gesetzt.

Bei Zustandsänderungen der Channels, z.B. Wechsel von online nach offline – jedoch spätestens nach 1 Minute – geht das Spiel von neuem los: Zeit ermitteln, Lookup in Tabelle, Default-Route-Circuit ermitteln.

Ändert sich der aktuell verwendete Circuit, z.B. montags um 18:00 Uhr, wird die alte Default-Route gelöscht, eine vielleicht bestehende Verbindung beendet (sorry...) und anschließend die Default-Route auf den neuen Circuit gesetzt. Dies kann von imond bis zu 60 Sekunden später bemerkt werden, also wird spätestens um 18:00:59 umgeschaltet.

Bei Circuits, die keine Default-Route belegen, ändert sich überhaupt nichts. Hier wird der Inhalt von `ISDN_CIRC_x_TIMES` lediglich zur Berechnung der Telefonkosten verwendet. Diese können dann relevant sein, wenn man über den Client imonc das LC-Routing temporär ausschaltet und einen Circuit manuell wählt.

Man kann sich jedoch auch die Tabellen für andere Zeitbereich-Indices (im Beispiel von 1 bis 10) anschauen, auch die der "Non-LC-Default-Route-Circuits".

Kommando:

```
timetable index
```

Beispiel:

```
telnet fli4l 5000
timetable 5
quit
```

Die Ausgabe sieht dann so aus:

```
    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
-------------------------------------------------------------------------
Su  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Mo  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
```

```
Tu  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
We  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
Th  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
Fr  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
Sa  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

No.  Name                    DefRoute  Device  Ch/Min   ChInt
 5   AOL                 no      ippp1   0.0490    180
```

Alles klar?

Mit dem imond-Kommando “route” kann das LC-Routing ein- und ausgeschaltet werden. Bei Angabe eines positiven Circuit-Indices (1...N) wird die Default-Route auf den angegebenen Circuit gelegt. Ist der Index 0, wird das LC-Routing wieder aktiviert und der Circuit automatisch ausgewählt.

7.1.2. Zur Berechnung der Onlinekosten

Das ganze Modell zur Berechnung der Onlinekosten funktioniert nur korrekt, wenn der Zeittakt für einen Circuit (Variable `ISDN_CIRC_x_CHARGEINT`) über die ganze Woche konstant ist. Dies ist im Normalfall bei Internet-Providern die Regel. Wählt man sich jedoch über die Telekom (ich meine nicht T-Online!) z.B. in sein Firmennetz ein, gilt das als ganz normales Telefongespräch. Und da wechselt ab 18:00 der Takt von 90 Sekunden auf 4 Minuten (Stand Juni 00). Deshalb ist die Definition von

```
ISDN_CIRC_3_CHARGEINT='90'
ISDN_CIRC_3_TIMES='Mo-Fr:09-18:0.08:N Mo-Fr:18-09:0.03:N Sa-Su:00-24:0.03:N'
```

eigentlich nicht ganz korrekt. Es sind zwar abends umgerechnet auf die Minute 3 Pfennig (4 Minuten kosten 12 Telekom-Pfennige), jedoch ist der Takt falsch. Deshalb können bei der Kostenanzeige Differenzen zu den tatsächlichen Zahlen auftreten.

Hier ist ein Tipp, wie verschieden lange Taktzeiten dennoch richtig berücksichtigt werden können (auch wichtig für `ISDN_CIRC_x_CHARGEINT`): Man definiert einfach 2 Circuits, einen für tagsüber mit `ISDN_CIRC_1_CHARGEINT`=‘90’ und den anderen mit `ISDN_CIRC_2_CHARGEINT`=‘240’. Natürlich muss man dann auch noch `ISDN_CIRC_x_TIMES` entsprechend wählen, damit tagsüber Circuit 1 und abends Circuit 2 verwendet wird.

Wie gesagt: Bei Nutzung von Verbindungen zu Internet-Providern gibt es das Problem nicht, weil dort der Zeittakt immer konstant ist und lediglich die Kosten pro Minute wechseln (oder gibt es sowas doch??? Ich traue T-* alles zu :-).

7.2. Windows-Client imonc.exe

7.2.1. Einleitung

Das Gespann imond auf dem Router und imonc auf dem Client beherrschen zwei Benutzermodi: den User- und den Adminmodus. Im Adminmodus sind alle Steuerelemente aktiviert. Im Usermodus steuern die Variablen `IMOND_ENABLE` (Seite 74), `IMOND_DIAL` (Seite 74), `IMOND_ROUTE` (Seite 74) und `IMOND_REBOOT` (Seite 74) ob die jeweiligen Funktionen im Usermodus zur Verfügung stehen. Sind alle diese Variablen auf ‘no’ gesetzt, bedeutet dies für die Überblick-Seite,

dass alle Buttons bis auf den Exit- und den Admin-Mode-Button deaktiviert sind. Die Entscheidung, ob der User- oder Admin-Modus benutzt wird, wird anhand des übermittelten Passwortes getroffen. Über den Button Admin-Mode, der sich in der Statusleiste befindet, kann jederzeit unter Eingabe des Admin-Passwortes vom User- zum Admin-Modus gewechselt werden. Um wieder zurück zu wechseln, muss imonc beendet und neu gestartet werden.

Sobald imonc gestartet ist, wird ein zusätzliches Tray-Icon angezeigt, welches den Verbindgungsstatus der vorhandenen Kanäle anzeigt.

Die Farben bedeuten:

Rot : Offline

Gelb : Es wird gerade eine Verbindung aufgebaut

Hellgrün : Online und Traffic auf dem Kanal

Dunkelgrün : Online und so gut wie kein Traffic auf dem Kanal

Ein etwas vom Windows-Standard abweichendes Verhalten zeigt imonc, wenn der Minimieren-Button in der Titelleiste angeclickt wird. Daraufhin minimiert sich imonc in den Systemtray und es bleibt nur noch das Tray-Icon neben der Uhr übrig. Ein Doppelklick mit der linken Maustaste auf das Tray-Symbol holt das imonc-Fenster wieder in den Vordergrund. Mit der rechten Maustaste besteht auch die Möglichkeit über das Kontextmenü, die wichtigsten imonc-Kommandos direkt auszuwählen, ohne imonc wieder auf den Bildschirm zu holen.

Viele Eigenschaften (darunter auch alle Spaltenbreiten der StringGrids) speichert imonc in der Registry, damit imonc so an die eigenen Bedürfnisse angepasst werden kann. Imonc speichert die Informationen in dem Registry-Schlüssel HKCU\Software\fli4l.

Bestehen trotz sorgfältigen Lesens der Dokumentation noch Probleme in Bezug auf imonc oder auch des Routers selber, die man z.B. in der Newsgroup posten möchte, ist es sinnvoll, auf der Über-Seite des imonc den Punkt SystemInfo auszuwählen und dort den Punkt Support Infos. Daraufhin wird das Router-Passwort abgefragt (nicht das imond-Passwort!). Imonc erstellt dann eine Datei fli4lsup.txt, welche alle wichtigen Informationen bezüglich des Routers und imonc beinhaltet. Diese Datei kann auf explizite Nachfrage in die Newsgroup gepostet werden, so dass deutlich bessere Chancen auf rasche Hilfe bestehen.

Nähere Details betreffend der Entwicklung des Windows-Clients imonc findet man auf der Homepage vom Windows ImonC-Seiten http://www.imonc.de/. Hier kann man sehen, welche neuen Features und Bug-Fixes in der nächsten Version von imonc enthalten sein werden. Ausserdem gibt es dort den neusten imonc, wenn dieser nicht schon in der fli4l-Distribution enthalten ist.

7.2.2. Startparameter

ImonC benötigt den Namen oder die IP-Adresse des fli4l-Routers. Standardmäßig versucht das Programm, eine Verbindung mit dem Rechner “fli4l” herzustellen. Wenn dieser im DNS korrekt eingetragen ist, sollte es also direkt funktionieren. Ansonsten kann man in der Verknüpfung folgende Parameter übergeben:

- /Server:IP oder Hostname des Routers (Kurzform: /S:IP oder Hostname)
- /Password:Passwort (Kurzform: /P:Password)

- /log Die Logging-Option zum Protokollieren der Kommunikation zwischen imonc und imond. Ist diese Option eingeschaltet, wird beim Beenden von imonc eine Datei imonc.log geschrieben. Diese Datei beinhaltet die gesamte Kommunikation zwischen Router und Client und wird darum sehr groß. Deshalb sollte dieser Startparameter nur gesetzt werden, wenn Probleme bestehen.
- /iport:Portnummer Die Portnummer auf die imond lauscht. Default: 5000
- /tport:Portnummer Port auf dem telmond lauscht. Default: 5001
- /rc:"Command" Das hier angegebene Kommando wird ohne weitere Überprüfung an den Router übertragen und anschliessend imonc beendet. Sollen mehrere Kommandos gleichzeitig ausgeführt werden, müssen diese durch Semikolons getrennt werden. Damit es funktioniert, muss ein gesetztes imond-Passwort mit übergeben werden, da keine Abfrage des Passwortes erfolgt. Die möglichen Kommandos sind beim imond dokumentiert, siehe Kapitel 8.1. Zusätzlich zu den dort aufgeführten Befehlen gibt es noch den Befehl timesync. Dieser bewirkt, dass die Uhrzeit des Clients mit der des Routers synchronisiert wird. Der Befehl dialtimesync wird nicht mehr unterstützt da er sich als „dial; timesync“ schreiben lässt.
- /d:"fli4l-Directory" Hiermit kann das fli4l-Directory per Startparameter übergeben werden. Interessant wenn man mit mehreren fli4l-Versionen herumspielt
- /wait Wenn der Hostname nicht aufgelöst werden kann, beendet sich imonc nicht mehr – erneuter Verbindungsaufbau durch Doppelclick auf das TrayIcon
- /nostartcheck Dieser schaltet die Überprüfung ab, ob imonc bereits läuft. Nur sinnvoll, wenn mehrere, unterschiedliche fli4l-Router in einem Netz überwacht werden sollen. Bei weiteren Instanzen werden die eingebauten Syslog- und E-Mail-Funktionalitäten deaktiviert.

Usage (einzutragen in der Verknüpfung):

```
X:\...imonc.exe [/Server:Host] [/Password:Passwort] [/iport:Portnummer]
            [/log] [/tport:Portnummer] [/rc:"Command"]
```

Beispiel mit IP-Adresse:

```
C:\wintools\imonc /Server:192.168.6.4
```

oder mit Namen und Passwort:

```
C:\wintools\imonc /S:fli4l /P:geheim
```

oder mit Namen, Passwort und Routerkommando:

```
C:\wintools\imonc /S:fli4l /P:geheim /rc:"dialmode manual"
```

7.2.3. Seite Überblick

Der Windows-Client fragt einige imond-Informationen über die bestehenden Verbindungen ab und bereitet sie im Anzeigefenster auf. Neben generellen Statusinformationen wie Uptime des Router oder auch der Uhrzeit sowohl lokal wie auch vom Router selber, werden für jede bestehende Verbindung die folgenden Informationen angezeigt:

Status	Verbindungsaufbau/Online/Offline
Name	Telefonnummer des Gegners oder Circuit-Name
Richtung	Zeigt an, ob es sich um eine eingehende oder ausgehende Verbindung handelt
IP	Die IP, die man zugewiesen bekommen hat
IBytes	Empfangene Bytes
OBytes	Gesendete Bytes
Online-Zeit	Aktuelle Online-Zeit
Zeit	Summe aller Online-Zeiten
KZeit	Summe Online-Zeiten unter Berücksichtigung des Zeittaktes
Kosten	Berechnete Kosten

Die Daten werden standardmäßig alle 2 Sekunden aktualisiert. Im Kontextmenü dieser Übersicht besteht die Möglichkeit für jeden vorhandenen Kanal, mit dem der Router gerade online ist, sowohl die zugewiesene IP in die Zwischenablage zu kopieren, als auch den Kanal gezielt auflegen zu können. Letzteres ist für den Fall interessant, dass mehrere unterschiedliche Verbindungen bestehen, z.B. eine um im Internet zu surfen und eine andere zur Firma, und gezielt eine dieser Verbindungen getrennt werden soll.

Ist zusätzlich auf dem fli4l-Router der telmond-Prozess aktiv, kann imonc zusätzlich Informationen über eingehende Telefonanrufe (nämlich anrufende und angerufene MSN) anzeigen. Der letzte eingegangene Telefonanruf wird oberhalb der Buttons angezeigt. Ein Protokoll der eingegangenen Telefonanrufe erhält man durch Anzeige der Seite Anrufe.

Mit den sechs Buttons im imonc können folgende Kommandos angewählt werden:

Button	Beschriftung	Funktion
1	Verbinden/Trennen	Wählen/Einhängen
2	Add link/Rem link	Kanäle bündeln: ja/nein – dieses Feature steht nur im Admin-Mode zur Verfügung
3	Reboot	fli4l neu booten!
4	PowerOff	fli4l sauber runterfahren und anschliessend ausschalten
5	Halt	fli4l sauber runterfahren, um ihn anschliessend sicher ausschalten zu können
6	Beenden	Client beenden

Die ersten fünf Kommandos können in der Konfigurationsdatei des fli4l-Routers config/base.txt für den User-Modus einzeln ein- und ausgeschaltet werden. Im Admin-Modus sind immer alle aktiviert. Die Auswahl Dialmode steuert das Wahlverhalten des Routers:

Auto	Der Router baut automatisch eine Verbindung auf dem entsprechenden Circuit auf, wenn eine Anfrage aus dem lokalen Netz eintrifft.
Manuell	Der Benutzer muss selber die Verbindung aufbauen.
Aus	Es ist weder manuell noch automatisch möglich, eine Verbindung aufzubauen. Der Dial-Button ist dann deaktiviert.

Bleibt noch anzumerken, dass fli4l standardmäßig selbständig rauswählt, wenn man mit seinem Rechner in's Internet will. Man muss also eigentlich nie den Verbinden-Button drücken ...

Es besteht auch die Möglichkeit, den Default-Route-Circuit manuell zu wechseln, also das automatische LCR-Routing ein- und auszuschalten. Dafür ist in der Windows-Version von imonc die Auswahlliste "Default Route" vorgesehen. Ausserdem kann man die Hangup-TimeOut-Zeit jetzt auch über imonc direkt konfigurieren. Dazu dient der Button Config neben der Default Route. Dort werden alle konfigurierten Circuits des Routers angezeigt. Der Wert in der Spalte Hup-timeout kann für ISDN-Circuits direkt im StringGrid editiert werden (funktioniert bis dato noch nicht für DSL).

Einen Überblick über das LCR-Routing findet man auf der Seite Admin/TimeTable. Dort sieht man, welchen Circuit imond zu welcher Zeit automatisch auswählt.

7.2.4. Config-Dialog

Der Konfigurationsbereich ist über den Button Config in der Statuszeile erreichbar. Das aufgehende Fenster ist dann in die folgenden Bereiche unterteilt:

- Der Bereich Allgemein:
  - Aktualisierungsintervall: Hier wird eingestellt, wie oft die Seite Überblick aktualisiert werden soll.
  - Zeit beim Programmstart synchronisieren: Übernimmt beim Starten des Client die Zeit und das Datum des Routers als lokale Zeit. Diese Funktion kann auch manuell mit dem Button Synchronisieren auf der Überblicks-Seite aufgerufen werden.
  - Minimiert starten: Startet das Programm direkt minimiert, man sieht nur das Icon neben der Uhr.
  - Zusammen mit Windows starten: Hier kann man angeben, ob der Client direkt beim Starten von Windows mit gestartet werden soll. In dem Feld Parameter kann man die nötigen Start-Parameter angeben.
  - News von fli4l.de abholen: Sollen die News, die auf der fli4l-Homepage in der News-Sektion angezeigt werden, auch vom imonc geholt und angezeigt werden? Die Schlagzeilen werden dann in der Statusbar angezeigt. Ausserdem wird dann eine neue Seite News angezeigt, in der die kompletten Meldungen angezeigt werden.
  - Logdatei für Verbindungen: Den Dateinamen, den man hier angeben kann, wird dazu benutzt, die Verbindungs-Liste unter diesem Namen lokal auf dem Rechner abzuspeichern.
  - TimeOut für Router zum antworten: Wie lange soll auf eine Antwort der Routers gewartet werden, bevor angenommen wird, dass die Verbindung nicht mehr besteht.
  - Sprache: Hier kann die Sprache des imoncs ausgewählt werden.

  - Router Befehle bestätigen: Ist dieses Feature aktiviert, müssen alle Router-beeinflussenden Kommandos, wie zum Beispiel Reboot, Hangup ... generell bestätigt werden.
  - Auflegen auch bei Traffic: Soll kein Hinweis erfolgen, wenn die Verbindung beendet wird und noch Traffic auf der Leitung ist.
  - Automatisch Verbindung zum Router aufbauen: Soll, wenn die Verbindung zum Router unterbrochen wird (z.B. durch einen Neustart des Routers), automatisch probiert werden, die Verbindung wieder herzustellen.
  - Fenster in System Tray minimieren: Soll imonc beim Clicken auf den Beenden-Button in der Titelleiste sich in den System-Tray neben der Uhr minimieren anstatt zu beenden.

- Der Unterbereich Proxy: Hier kann ein Proxy für die http-Anfragen des imoncs definiert werden. Dieser wird dann zur Zeit für das Holen der News benutzt.
  - Proxy-Unterstützung für Http-Anfragen aktivieren: Soll ein Proxy benutzt werden
    * Adresse: Die Adresse des Proxy-Servers
    * Port: Die Portnummer des Proxy-Server (default: 8080)

- Der Unterbereich TrayIcon: Hier können die Farben des TrayIcons neben der Uhr an die eigene Bedürfnisse angepasst werden. Weiterhin kann ausgewählt werden, dass der aktuelle Dialmode als farblicher Hintergrund des TrayIcons dargestellt wird.

- Der Bereich Anrufe: Die Position des Call Notification-Fensters wird in der Registry gespeichert, so dass man sich das Fenster an die Position schieben kann, wo man es haben möchte. Es erscheint anschliessend immer wieder an dieser Stelle.
  - Aktualisierung: Hier kann ausgewählt werden, wie imonc über neue Anrufe informiert wird. Es gibt drei verschiedene Möglichkeiten. Diese erste besteht darin, regelmäßig den telmond-Dienst auf dem Router abzufragen. Eine weitere Möglichkeit besteht in der Auswertung der Syslog-Meldungen. Diese Variante ist der ersten vorzuziehen – dazu muss natürlich der Syslog-Client des imonc aktiviert sein. Wird imonc mit einem routenden eisfair eingesetzt, bietet sich noch die Möglichkeit das Capi2Text-Paket zur Anrufsignalisierung zu benutzen.
  - Führende Null wegen Telefonanlage löschen: Telefonanlage setzen manchmal eine zusätzliche Null vor die Rufnummer des Anrufer. Diese kann mit dieser Option unterdrückt werden.
  - Eigene Vorwahl: Hier kann die eigene Vorwahl hinterlegt werden. Wann dann ein Anruf mit gleichen Vorwahl eintrifft, wird die gesendete Vorwahl ausgeblendet.
  - Telefonbuch: Hier kann die Datei angegeben werden, in der das lokale Telefonbuch zur Auflösung von Telefonnummer gespeichert wird. Existiert die Datei nicht, wird sie vom Programm angelegt.
  - Logdatei: Der Dateinamen, den man hier angeben kann, wird dazu benutzt, die Calls-Liste unter diesem Namen lokal auf dem Rechner zu speichern. Dieser Menüpunkt ist nur sichtbar, wenn die Config-Variable `TELMOND_LOG` auf 'yes' gesetzt ist, dieses gilt auch für die eigentliche Anruf-Liste.

  - Externes Suchprogramm benutzen: In diesem Bereich kann ein Programm angegeben werden, dass aufgerufen wird, wenn eine Telefonnummer mittels des lokalen Telefonbuches nicht aufgelöst werden kann. Nähere Infos sollten den entsprechenden Programmen beiliegen. Bis jetzt gibt es eine Anbindung an die Telefonbuch-CD KlickTel sowie von Marcel Wappler eine Anbindung an die Palm-Datenbank.

- Der Unterbereich Call Notification: Hier kann das bestimmt werden, ob ein Hinweis auf eingehende Telefonanrufe angezeigt werden soll und wie dieser sich optisch präsentiert.
  - Call Notification aktivieren: Bestimmt, ob Anrufe signalisiert werden sollen.
  - Call Notification anzeigen: Soll bei eingehenden Anrufen ein Hinweisfenster mit den Infos: angerufene MSN, Rufnummer des Anrufers und Datum/Uhrzeit erscheinen? Dafür ist es nötig, dass in der Datei config/isdn.txt die Variable `OPT_TELMOND` auf 'yes' gesetzt wird.
    * Unterdrücken, wenn keine Nummer übertragen wurde: Soll Die Call Notification nicht angezeigt werden, wenn keine Rufnummer übertragen wurde.
    * Anzeigendauer: Diese Angabe beeinflußt die Dauer, wie lange das Call Notification-Fenster geöffnet bleiben soll. Die Angabe von "0" an dieser Stelle bewirkt, dass das Fenster nicht automatisch geschlossen wird.
    * Fontsize: Hiermit wird die Schriftgröße bestimmt. Dieses hat einen Einfluss auf die Größe des Fenster, da die notwendige Größe des Fenster anhand der tatsächlichen Größe der Mitteilung berechnet wird.
    * Farbe: Hiermit kann die Schriftfarbe ausgewählt werden. Ich selber benutzte die Farbe rot, damit ich es auch direkt wahrnehme.

- Der Unterbereich Phonebook: Die Seite Phonebook beinhaltet das Telefonbuch, welches zur Rufnummerauflösung der anrufenden Nummer als auch der eigenen MSN benutzt wird. Die Seite wird auch angezeigt, wenn die Konfigurationsvariable `TELMOND_LOG` auf 'no' gesetzt ist, da die Rufnummerauflösung auch für die Anzeige des letzten Anrufes auf der Summary-Seite benutzt wird. Alternativ kann statt dem Telefonbuch auf dem Router auch eine lokale Datei ausgewählt werden.

  Der Aufbau der Eintrag sieht wie folgt aus:

```
# Format:
# Telefonnummer=anzuzeigender Name[, Wavefilename]
# 0241123456789=Testuser
00=unbekannt
508402=Fax
0241606*=Elsa AG Aachen
```

  Dabei sind die ersten drei Zeilen Kommentare. Die vierte Zeile bewirkt, dass, wenn keine Rufnummer übermittelt wird, "unbekannt" angezeigt wird. In der fünften Zeile wird der MSN 508402 der Name "Fax" zugeordnet. Ansonsten ist das Format immer Telefonnummer=Name, der stattdessen angezeigt werden soll. In der sechsten Zeile ist noch die Möglichkeit demonstriert, eine Sammelrufnummer zu definieren. Damit wird erreicht, dass für alle Nebenstellen von 0241606 der Name angezeigt wird. Zu beachten hierbei ist, dass der erste Eintrag im Telefonbuch, welcher auf den Anruf passt, genommen

wird. Optional kann auch noch ein Wave-Datei angegeben werden, die abgespielt wird, wenn ein Telefonanruf von dieser Rufnummer eingeht.

Ab der Version 1.5.2 besteht jetzt auch die Möglichkeit auf der Seite Names das lokale Telefonbuch mit dem auf dem Router abgespeicherten (in /etc/phonebook) zu synchronisieren und umgekehrt. Dabei werden nicht nur einfach die Dateien ersetzt, sondern es werden die noch fehlende Einträge hinzugefügt. Gibt es eine Telefonnummer in beiden Telefonbüchern mit unterschiedlichen Namen, wird nachgefragt, welcher Eintrag genommen werden soll. Für die Synchronisierung des Telefonbuches auf dem Router ist noch anzumerken, dass dieses nur in der Ramdisk verändert wird, d.h. dass nach einem Reboot sämtliche Änderungen verloren gehen.

- Der Bereich Sound: Die Wave-Dateien, die hier angegeben werden, werden abgespielt, wenn das jeweilige Ereignis eingetreten ist.
  - E-Mail: Wenn der E-Mail-Checker auf einem angegebenen POP3-Server neue E-Mails vorfindet, wird die angegebene Wave-Datei abgespielt.
  - E-Mail-Error: Wenn ein Fehler beim Abrufen der E-Mails auftritt, wird diese Wave-Datei abgespielt.
  - Verbindung verloren: Wenn die Verbindung zum Router nicht mehr vorhanden ist (z.B. wenn der Router von einem anderen Client gerade neu gebootet wird), wird diese Wave-Datei abgespielt. Wenn die Option “Automatic Reconnect to router” nicht aktiviert ist, erscheint ausserdem eine MessageBox, die nachfragt, ob versucht werden soll, eine neue Verbindung zum Router aufzubauen.
  - Verbindungsmeldung: Wenn der Router eine Verbindung zum Internet aufgebaut hat, wird diese Wave-Datei abgespielt.
  - Verbingsabbau: Wenn der Router die Verbindung zum Internet wieder abgebaut hat, wird diese Wave-Datei abgespielt.
  - Anrufmeldung: Wenn die Call Notification aktiviert ist und ein neuer Anruf eingeht, wird die angegebene Wave-Datei abgespielt.
  - Fax Notification: Die hier angegebene Wave-Datei wird nach dem Empfang neuer Faxe abgespielt.
- Der Bereich E-Mails
  - Accounts: Dieser Bereich dient dazu, die vorhandenen POP3-Accounts zu konfigurieren.
  - E-Mail-Check aktivieren: Soll der E-Mail-Checker automatisch nach neuen E-Mails suchen
    - Check jede x Min: Hiermit wird angegeben, wie oft der E-Mail-Checker automatisch nach neuen E-Mails suchen soll. Achtung: ein zu kurzes Intervall kann dazu führen, dass der Router komplett online bleibt! Dies ist der Fall, wenn das Intervall kleiner ist als der Hangup-Timeout des verwendeten Circuits.
    - TimeOut x Sec: Wie lange soll auf einen einen POP3-Server gewartet werden, bis er antwortet. Der Wert “0” bedeutet, dass kein TimeOut gesetzt wird.

    * Auch wenn Router offline: Hiermit wird erreicht, dass der Router sich selbstständig einwählt, um nach E-Mails zu sehen. Nachdem alle POP3-Konton nach E-Mails überprüft worden sind, wird die Verbindung wieder getrennt. Um dieses Feature nutzen zu können, muss Dialmode auf 'auto' stehen. Achtung: Dadurch entstehen zusätzliche Kosten, wenn nicht gerade eine Flatrate benutzt wird!
    * Zu benutzender Circuit: Hiermit wird angegeben, welcher Circuit zur Einwahl beim E-Mail-Checken benutzt werden soll.
    * Anschliessend online bleiben: Soll direkt nach dem E-Mail-Check direkt die Verbindung getrennt werden oder eine Verbindungstrennung durch das Hanguptimeout realisiert werden.
    * E-Mail-Header laden: Sollen auch die E-Mail-Header geladen oder nur die Anzahl der vorhandenen E-Mails abgefragt werden? Das Laden der E-Mail-Header ist Voraussetzung, wenn man E-Mails direkt auf dem Server löschen möchte.
    * Notify only new E-Mails: Sollen nur neue E-Mails akustisch und mit dem Tray-Icon gemeldet werden
    * E-Mail-Client starten: Soll der angegebene E-Mail-Client automatisch gestartet werden, wenn neue E-Mails vorhanden sind.
    * E-Mail-Client: Hier wird der zu startende E-Mail-Client angegeben.
    * Param: Hier kann man zusätzliche Parameter angeben, die beim Start des E-Mail-Clients übergeben werden sollen. Wenn Outlook als E-Mail-Client benutzt wird (nicht Outlook Express), sollte /recyle als Parameter eingetragen werden, damit eine bereits geöffnete Instanz von Outlook beim Eintreffen von neuen E-Mails benutzt wird.

- Der Bereich Admin
  - root-Passwort: Hier sollte das Router-Passwort (in config/base.txt unter `PASSWORD` eingetragen) eingetragen werden, damit z.B. das Portforwarding lokal bearbeitet und wieder auf dem Router hinterlegt werden kann.
  - Dateien auf dem Router, die angezeigt werden sollen: Alle hier angegebenen Dateien, die sich auf dem Router befinden, können einfach per Maus-Click auf der Seite Admin/Dateien angezeigt werden. Somit kann man sich auf einfache Weise die Logfiles des Routers direkt im imonc anzeigen lassen.
  - Konfigdateien bearbeiten: Hier kann ausgewählt werden, ob die Konfigdateien alle im Editor geöffnet werden sollen (dies kann, wenn TXT-Dateien noch mit einem einfachen Editor verknüpft sind, dazu führen, dass sehr viele Instanzen des Editors geöffnet werden). Alternativ kann auch einfach nur das Verzeichnis geöffnet werden, so dass die Möglichkeit besteht, nur die Dateien auszuwählen, die bearbeitet werden sollen.
  - DynEisfaiLog: Wenn ein Account bei DynEisfair vorhanden ist, kann man hier seine Zugangsdaten eintragen und sich dann ein Log der Aktualisierung des Dienstes auf der Seite Admin/DynEisfairLog anschauen.

- Der Bereich LaunchList dient dazu, die Launchliste zu konfigurieren. Diese wird nach einem erfolgreichen Connect ausgeführt, wenn die Option "Activate Launchlist" aktiviert ist.

  - Programme: Alle hier eingetragenen Programme werden automatisch gestartet, sobald der Router eine Verbindung aufgebaut hat und die Launchliste aktiviert ist.
  - LaunchList aktivieren: Soll die Launchliste beim erfolgreichen Verbindungsaufbau ausgeführt werden?

- Der Bereich Traffic dient dazu, dass TrafficInfo-Fenster den eigenen Bedürfnissen anzupassen. Von einem User habe ich den Hinweis bekommen, dass es mit älteren DirectX-Versionen offenbar Darstellungsfehler gibt.
  - Separates Traffic-Info-Fenster anzeigen: Soll eine grafische Kanalauslastung in einem separaten Fenster angezeigt werden? In dem Kontextmenü des Fensters kann man festlegen, ob das Fenster das Attribut StayOnTop bekommen soll. Dieses bewirkt, dass sich das Fenster immer über allen anderen Fenstern plaziert. Auch dieser Wert wird in der Registry abgespeichert und steht somit auch nach einem erneuten Programmstart wieder zur Verfügung.
  - Titelleiste anzeigen: Soll die Titelleiste des Traffic-Info-Fensters angezeigt werden? In der Titelzeile wird angezeigt, mit welchem Circuit der Router gerade online ist.
    * CPU-Auslastung in Titelleiste: Soll auch die CPU-Auslastung in der Titelzeile angezeigt werden?
    * Online-Zeit in Titelleiste: Soll die Onlinezeit des Kanals auch in der Titelzeile angezeigt werden?
  - Semi-transparentes Fenster: Soll das Fenster transparent dargestellt werden? Diese Funktion steht nur unter Windows 2000 und Windows XP zur Verfügung.
  - Farben: Hier werden die Farben für das TrafficInfo-Fenster definiert. Zu Berücksichtigen ist dabei, dass der DSL-Kanal und der erste ISDN-Kanal die selben Farbwerte zugewiesen bekommen.
  - Limits: Hier können die max. Übertragungswerte für DSL eingestellt werden – Upload und Download.

- Der Bereich Syslog dient dazu, die Anzeige der Syslog-Meldungen zu konfigurieren.
  - Syslog-Client aktivieren: Sollen Syslog-Meldungen im imonc angezeigt werden? Diese Option sollte ausgeschaltet sein, wenn ein externer Syslog-Client, wie zum Beispiel Kiwi's Syslog Client, benutzt wird.
  - Alle Meldungen ab Stufe anzeigen: Ab welcher Prioritätsstufe sollen die Syslog-Meldungen angezeigt werden? Es ist sinnvoll am Anfang mit der Stufe Debug anzufangen, um damit festzustellen, welche Meldungen einen interessieren. Anschliessend kann hier dann die entsprechende Stufe eingetragen werden.
  - Syslog-Meldungen in einer Datei speichern: Sollen die angezeigten Syslog-Meldungen in einer Datei gespeichert werden? In der Groupbox können dann die Meldungen ausgewählt werden, die in der Datei geloggt werden sollen. Für den Dateinamen sind folgende Platzhalter eingefügt worden:

    %y – wird durch das aktuelle Jahr ersetzt

    %m – wird durch den aktuellen Monat ersetzt

    %d – wird durch den aktuellen Tag ersetzt

  - Portnamen anzeigen: Sollen statt den Portnummern deren Bedeutungen angezeigt werden?
  - Firewall-Meldungen auch im User-Modus anzeigen: Hiermit wird festgelegt, dass Firewall-Meldungen auch im User-Modus angezeigt werden sollen.

- Der Bereich Fax dient dazu, die Faxanzeige vom imonc zu konfigurieren. Damit dieser Punkt angezeigt wird, muss mgetty bzw. faxrcv auf dem Router installiert sein (zu finden als OPT-Pakete auf der fli4l-Homepage).
  - Logdatei für Faxe: Den Dateinamen, den man hier angeben kann, wird dazu benutzt, die Fax-Liste unter diesem Namen lokal auf dem Rechner abzuspeichern.
  - Lokales Verzeichnis: Um die Faxe anzeigen zu können, müssen sie lokal gespeichert werden. Dieses kann hier eingestellt werden.
  - Aktualisierung: Es gibt zwei verschiedene Möglichkeiten, wie imonc mitbekommt, dass ein neues Fax eingegangen ist. Entweder wertet imonc die entsprechenden Syslogmeldungen aus (dazu muss natürlich der Syslog-Client im imonc aktiviert sein) oder er schaut regelmäßig selber in der Logdatei nach. Die erste Variante ist zu bevorzugen. Falls die zweite Variante genutzt wird, kann man noch angeben, wieoft die Faxübersichtsseite aktualisiert werden soll. Dabei ist zu beachten, dass dieser Wert keine Angabe in Sekunden ist, sondern noch mit der Angabe von Allgemein/Aktualisierungsintervall multipliziert wird.

- Der Bereich Grids dient dazu die Grids (Tabellen) im imonc an die eigenen Bedürfnisse anzupassen. Einerseits kann für jedes Grid angegeben werden, welche Spalten angezeigt werden sollen, andererseits gibt es die Möglichkeit für die Grids im Bereich Anrufe, Verbindungen und Faxe anzugeben, von wann ab die Infos angezeigt werden sollen.

7.2.5. Seite Anrufe

Die Seite Calls wird nur angezeigt, wenn die Konfigurationsvariable `TELMOND_LOG` auf 'yes' eingestellt ist, denn sonst wird kein Anruf-Log geführt. Auf dieser Seite werden alle abgespeicherten Telefonanrufe angezeigt, die eingegangen sind, während der Router eingeschaltet war. Dabei kann umgeschaltet werden zwischen der Ansicht der lokal gespeicherten Anrufe oder nur der auf dem Router gespeicherten Anrufe. Wird bei der Anzeige der auf dem Router gespeicherten Anrufe der Zurücksetzen-Button gedrückt, wird das Logfile auf dem Router gelöscht.

In der Anruf-Übersicht kann mit der rechten Maustaste auf der Rufnummer oder der eigenen MSN diese ins Telefonbuch übernommen werden, um der Rufnummer bzw. MSN dort einen Namen zuzuweisen, der dann stattdessen angezeigt wird.

7.2.6. Seite Verbindungen

Neu ist ab der Version 1.4 die Anzeige der vom Router aufgebauten Verbindungen zum Internet. Diese befindet auf der Seite Connections. Somit hat man einen guten Überblick, wie sich der Router bei der automatischen Einwahl ins Internet verhält. Damit diese Seite angezeigt werden kann, muss in der Datei config/base.txt die Variable `IMOND_LOG` auf 'YES' gesetzt werden.

Genauso wie bei der Anruf-Übersicht kann auch hier zwischen den lokal gespeicherten und auf dem Router gespeicherten Verbindungen umgeschaltet werden. In der Ansicht der auf dem

Router gespeicherten Verbindungen bewirkt ein Drücken des Zurücksetzen-Buttons, dass das Logfile auf dem Router gelöscht wird.

Angezeigt werden pro Verbindung

- Provider
- Startdatum und -zeit
- Enddatum und -zeit
- Onlinezeit
- Abrechnungszeit
- entstandene Kosten
- empfangene Zeichen
- gesendete Zeichen

7.2.7. Seite Fax

Damit die Seite Faxe angezeigt wird, muss auf dem Router entweder das `OPT_MGETTY` von Michael Heimbach oder `OPT_MGETTY` von Felix Eckhofer installiert werden. Diese gibt es auf der fli4l-Homepage unter OPT-Pakete. Auf dieser Seite werden dann alle eingegangenen Faxe aufgelistet. Das Kontextmenü der Übersicht bietet mehrere Möglichkeiten, diese stehen allerdings nur im Admin-Modus zur Verfügung:

- Es kann ein Fax angezeigt werden. Dazu muss unter Admin/Remoteupdate der Pfad für das fli4l-Verzeichnis korrekt gesetzt werden, da die Faxe auf dem Router in gepackter Form vorliegen und somit gzip aus dem fli4l-Paket benötigt wird. Alternativ kann gzip.exe und win32gnu.dll auch ins imonc-Verzeichnis kopiert werden. Kann gzip.exe nicht an einer der beiden Stellen gefunden werden, wird stattdessen der Webserver des Routers probiert zu öffnen (direkt mit dem Aufruf des richtigen CGIs).
- Ein einzelnes Fax kann gelöscht werden. Dabei wird das Fax sowohl lokal als auch auf dem Router gelöscht (sowohl die eigentliche Faxdatei, als auch der Eintrag in den Logdateien).
- Sämtliche auf dem Router befindlichen Faxe löschen. Damit werden die Faxe und die Logdatei auf dem Router gelöscht. Die Faxe werden nicht aus der lokalen Logdatei gelöscht.

Genauso wie bei der Anruf-Übersicht kann auch hier zwischen den lokal gespeicherten und auf dem Router gespeicherten Faxen umgeschaltet werden.

7.2.8. Seite E-Mail

Diese Seite wird nur angezeigt, wenn im Config-Dialog mindestens ein aktiviertes POP3-E-Mail-Konto eingerichtet worden ist.

Die Seite E-Mail dürfte sich eigentlich selber erklären. Hiermit wird der mittlerweile eingebaute E-Mail-Checker beobachtet. Ist die Option “Check even if the router is offline” nicht

aktiviert, überprüft der E-Mail-Checker alle E-Mail-Konten nach E-Mails, sobald der Router online ist und anschließend im eingestellt Intervall. Ist die genannte Option aktiviert, überprüft der E-Mail-Checker im eingestellten Intervall. Ist der Router gerade online, wird die bestehende Verbindung benutzt. Ist er nicht online, wird eine Verbindung selbständig mit dem ausgewählten Circuit hergestellt, die, sobald alle E-Mail-Konten abgearbeitet sind, wieder getrennt wird. Damit man diese Option nutzen kann, muss Dialmode auf "auto" stehen.

Sind E-Mails auf dem POP3-Server vorhanden, wird entweder automatisch der eingestellte E-Mail-Client gestartet oder ein neues Symbol im Tray neben der Uhr angezeigt, welches als Hint die Anzahl der E-Mails auf jedem Server liefert. Ein Doppelclick startet dann den eingestellten E-Mail-Client. Ist ein Fehler bei einem der E-Mail-Konten aufgetreten, erscheint einerseits ein Hinweis in der E-Mail-History, andererseits wird das E-Mail-TrayIcon angezeigt, welches dadurch gekennzeichnet ist, dass die obere rechte Ecke rot gefärbt ist.

In der E-Mail-Übersicht kann man mit dem Kontextmenü Mails direkt auf dem Server löschen, ohne sie vorher komplett downloaden zu müssen. Dies geschieht, indem mit der rechten Maustaste das Kontextmenü angezeigt wird. Dabei sollte eine Zelle der entsprechenden Zeile markiert sein, wo die zu löschende E-Mail eingetragen ist. Im Kontextmenü wählt man den einzige Punkt Delete MailMessage aus.

7.2.9. Admin

Dieser Abschnitt steht nur zur Verfügung, wenn sich imonc im Admin-Modus befindet.

Der erste Punkt lieferte eine Übersicht über die verwendeten Circuits – sprich Internetprovider – die der Router automatisch per LCR auswählt. Ein Doppelclick auf einen Provider in der Providerübersicht zeigt an, für welche Zeiträume der Circuit in config/base.txt definiert worden ist.

Der zweite Punkt dort ist die Möglichkeit ein Fernupdate auf dem Router einzuspielen. Dabei kann ausgewählt werden, welche fünf Programmpakete (Kernel, Rootfilesystem, Opt-Datei, rc.cfg und syslinux.cfg) auf den Router kopiert werden sollen. Damit man das Update einspielen kann, muss man zuerst mal das fli4l-Verzeichnis angeben, damit imonc weiss, woher es die nötigen Dateien nehmen soll. Ausserdem muss angegeben werden, in welchem Unterverzeichnis die Konfigurationsdateien liegen (standardmäßig config), um die Opt-Datei, rc.cfg und syslinux.cfg jeweils neu zu erzeugen. Es ist ratsam, einen Reboot nach dem Einspielen des Updates durchführen, damit die Änderungen auch direkt wirksam werden. Wird während des Updates nach einem Passwort nachgefragt, ist das Passwort gemeint, welches in config/base.txt unter PASSWORD eingetragen ist.

Um die Beschränkung des Port-Forwarding zu umgehen, dass ein Port nur an genau einen Client-Rechner gebunden ist, besteht jetzt die Möglichkeit, die Konfiguration auf dem Router zu editieren. Damit die Änderungen aktiv werden, muss die Verbindung neu hergestellt werden. Da die Datei nur in der Ramdisk ersetzt wird, bleiben die Änderungen nur bis zum nächsten Neustart des Routers erhalten. Um die Änderungen dauerhaft zu speichern, muss ein neues Opt-File auf dem Router installiert werden mit einer geeignet angepassten base.txt aus dem Config-Verzeichnis.

Der vierte Punkt auf der Admin-Seite – Dateien – dient dazu, Konfigurations- und Logdateien des Routers einfach per Maus-Click anzuzeigen. Die Auswahlliste wird über den Punkt Config/Admin und dort "files on router to view" konfiguriert. Anschliessend kann einfach über die ComboBox auf dieser Seite ausgewählt werden, welche Datei angezeigt werden soll.

Der fünfte Punkt ist die Seite DynEisfairLog, sie erscheint nur wenn im Config-Dialog unter

Admin die Zugangsdaten des DynEisfair-Accounts eingetragen worden sind. Ist dies geschehen, wird auf dieser Seite Log des Dienstes angezeigt.

Als letzten Punkt gibt es noch die Seite Hosts. Hier werden alle in der Datei /etc/hosts eingetragenen Rechner angezeigt. Weiterhin wird probiert jeden dieser dort eingetragenen Rechner anzupingen und das Ergebnis davon wird ebenfalls angezeigt. Somit kann man schnell rausbekommen, welche dieser Rechner eingeschaltet sind.

7.2.10. Seiten Fehler, Syslog und Firewall

Die Seiten Fehler, Syslog und Firewall werden nur angezeigt, wenn in den entsprechenden Logs Einträge vorhanden sind. Die Einträge der Seiten Syslog und Firewall werden nur angezeigt, wenn man im Admin-Modus ist.

Auf der Seite Fehler werden sämtliche imonc/imond-spezifischen Fehler festgehalten. Wenn Probleme bestehen, kann unter Umständen ein Blick auf diese Seite die Ursache der Probleme anzeigen.

Auf der Seite Syslog werden die ankommende Syslog-Meldungen angezeigt, bis auf die Meldungen der Firewall. Diese werden auf der eigenen Seite Firewall dargestellt. Damit dies funktioniert, muss die Variable `OPT_SYSLOGD` in der Konfigurationsdatei config/base.txt auf 'yes' gesetzt werden. Ausserdem muss die Variable `SYSLOGD_DEST` auf die IP des Clients gesetzt werden (genau: `SYSLOGD_DEST`='@100.100.100.100 – wobei die IP natürlich an die IP des Clients angepasst werden muss). Angezeigt wird neben der eigentlichen Syslog-Meldung auch Datum, Uhrzeit, IP des Senders und die Prioritätsstufe.

Damit die Firewall-Meldungen bei den ganzen Syslog-Meldungen nicht untergehen, werden diese auf der separaten Seite Firewall angezeigt. Damit die Firewall- Meldungen angezeigt werden können, muss zusätzlich in der Datei config/base.txt die Konfigurationsvariable `OPT_KLOGD` auf 'yes' gesetzt werden.

7.2.11. Seite News

Auf dieser Seite werden, vorausgesetzt die Option ist im Config-Bereich des Imonc aktiviert, die News, welche auf der fli4l-Homepage angezeigt werden, auch direkt im Imonc angezeigt werden. Dazu wird mittels des http-Protokolls die URL http://www.fli4l.de/german/news.xml abgerufen. Neben den News werden mittlerweile auch die fünf aktuellsten Opt-Pakete angezeigt. Dazu wird die URL http://www.fli4l.de/german/imonc_opt_show.php abgefragt. Außerdem wird in der Statusleiste vom Imonc die Überschriften der News alternierend angezeigt.

7.3. Unix/Linux-Client imonc

Für Linux gibt es mittlerweile 2 Versionen: eine textbasierte (imonc) und eine mit graphischer Oberfläche (ximonc). Den Source zu ximonc findet man im Verzeichnis src. Die Dokumentation für ximonc wird erst in der 1.5-Final-Version zur Verfügung stehen. Ein erfahrener Linux-User sollte aber mit den Sources keine Probleme haben.

Beschränken wir uns daher hier zunächst auf die textbasierte Version von imonc: Dieses ist ein curses-basiertes Programm, hat also keine graphische Oberfläche. Der Source liegt im Verzeichnis unix.

Installation:

```
cd unix
make install
```

imonc wird dabei in /usr/local/bin installiert.
Aufruf:

```
imonc hostname
```

Dabei ist als hostname der Name oder die IP-Adresse des fli4l-Routers anzugeben, also z.B.

```
imonc fli4l
```

imonc zeigt folgende Informationen:

- Datum/Uhrzeit des fli4l-Routers
- Momentan eingestellte Route
- Default-Route-Circuits
- ISDN-Kanäle

  Status : Calling/Online/Offline

  Name : Telefonnummer des Gegners oder Circuit-Name

  Time : Online-Zeit

  Charge-Time : Online-Zeit unter Berücksichtigung des Zeittaktes

  Charge : Berechnete Kosten

Mögliche Kommandos sind:

Nr	Befehl	Bedeutung
0	quit	Programm beenden
1	enable	Aktivieren
2	disable	Deaktivieren
3	dial	Wählen
4	hangup	Einhängen
5	reboot	Neu booten
6	timetable	Zeittabelle ausgeben
7	dflt route	Neuen Default-Route-Circuit bestimmen
8	add channel	2. Kanal hinzuschalten
9	rem channel	2. Kanal deaktivieren

Zu den Kommandos im Einzelnen:

0 – quit Die Verbindung zum imond-Server wird abgebaut und das Programm beendet.

1 – enable Alle Cirucits werden auf dialmode “auto” gestellt. Das ist auch der Default-Zustand von fli4l nach dem Booten. Das heisst, dass fli4l bei einem Verbindungsaufbauwunsch eines Rechners im Netz automatisch rauswählt.

2 – disable Alle Circuits werden auf dialmode “off” gestellt. Damit ist fli4l so gut wie tot, bis er mit dem Enable-Kommando wieder geweckt wird.

3 – dial Manuelle Wahl auf dem Default-Route-Circuit. Ist eher für Testzwecke gedacht, da fli4l normalerweise automatisch wählt.

4 – hangup Manuelles Einhängen: damit kann man dem automatischen Einhängen von fli4l zuvorkommen.

5 – reboot fli4l wird neu gebootet. Ziemlich überflüssiges Kommando ...

6 – timetable Es wird die Zeittabelle für die Default-Route-Circuits ausgegeben. Beispiel: s.o.

7 – default route circuit Manuelles Wechseln des Default-Route-Circuits. Kann z.B. dann sinnvoll sein, wenn man das automatische LC-Routing von fli4l für eine Weile ausser Kraft setzen will, da einige Provider einen Zugriff auf das eigene Postfach nur über den eigenen Internet-Zugang erlauben.

8 – add channel Hier kann der 2. ISDN-Kanal manuell hinzugeschaltet werden. Voraussetzung: `ISDN_CIRC_x_BUNDLING` ist 'yes'.

9 – remove channel Abschalten des 2. ISDN-Kanals. Siehe auch "add channel".

Sonst gelten bei diesen Kommandos dieselben Bemerkungen wie für den Windows-imond-Client `imonc.exe`.

Noch zu bemerken ist: Ab fli4l-1.4 ist es nun auch möglich, auf dem fli4l-Router selbst einen "minimalisierten" imon-Client zu installieren, nämlich durch Setzen von `OPT_IMONC`='yes' im Paket `TOOLS`.

Damit kann man nun auch an der fli4l-Konsole bestimmte Einstellungen, z.B. Routing etc. mit imonc vornehmen. Achtung: Dieser Mini-imonc funktioniert nur auf dem fli4l-Router selbst! Auf einem Linux-/Unix- Client ist immer der "große Bruder" unix/imonc zu verwenden.

8. Entwickler-Dokumentation

8.1. Allgemeine Regeln

Damit ein neues Paket in die OPT-Datenbank auf der fli4l-Homepage aufgenommen wird, müssen einige Regeln beachtet werden. Pakete die sich nicht an diese Regeln halten, können ohne Vorwarnung aus der Datenbank entfernt werden.

1. KEINE Kopieraktionen von Seiten des Benutzers! fli4l bietet ein ausgefeiltes System, um die Daten der fli4l-Pakete in das Installationsarchiv einzupacken. Alle Dateien, die auf den Router sollen, liegen in `opt/`.

2. Pakete richtig packen und komprimieren: Die Pakete müssen so aufgebaut sein, dass sie sich mühelos ins fli4l-Verzeichnis entpacken lassen.

3. Die Pakete sollen sich VOLLSTÄNDIG über die Konfigurationsdatei konfigurieren lassen. Ein weiteres Bearbeiten der Konfigurationsdateien darf nicht vom Benutzer verlangt werden. Schwierige Entscheidungen dem Benutzer abnehmen oder in einen erweiterten Bereich verlagern (ans Ende der Konfigurationsdatei mit einem dicken Hinweis: ONLY MODIFY IF YOU KNOW WHAT YOU DO).

4. Noch ein Hinweis zur Konfigurationsdatei: Anhand des Namens einer Variablen muss sich eindeutig erkennen lassen, zu welchem OPT sie gehört. So gehören z. B. zum `OPT_HTTPD` die Variablen `OPT_HTTPD`, `HTTPD_USER_N`, usw.

5. Bitte, bitte, macht möglichst kleine Binaries (Programme)! Wenn ihr sie selbst im FBR übersetzt, dann denkt daran, unnötige Features zu deaktivieren.

6. Prüft euer Copyright! Wenn ihr Dateien als Vorlagen benutzt, achtet bitte darauf, das Copyright entsprechend zu ändern. Dies gilt besonders für die Config-, Check- und Opt-Textdateien. Ersetzt hier das Copyright durch euren eigenen Namen. Bei wortwörtlich kopierter Dokumentation muss natürlich das Copyright des Orginal-Autors erhalten bleiben!

7. Bitte als Archivtypen nur verbreitete, freie Formate benutzen. Dazu gehören:
   - ZIP (`.zip`)
   - GZIP (`.tgz` oder `.tar.gz`)

   Andere Formate wie RAR, ACE, Blackhole, LHA etc. bitte nicht verwenden. Auch Windows-Installer-Dateien (`.msi`) oder selbstextrahierende Archive und Installer (`.exe`) sind nicht zu benutzen.

8.2. Übersetzen von Programmen

Die für das Übersetzen von Programmen erforderliche Umgebung wird in dem separat erhältlichen Paket „src“ angeboten. Dort wird auch dokumentiert, wie sich eigene Programme für den fli4l übersetzen lassen.

8.3. Modulkonzept

fli4l wird seit der Version 2.0 in Module (Pakete) aufgeteilt, z. B.

- fli4l-3.10.4 <— Das Basis-Paket
- dns-dhcp
- dsl
- isdn
- sshd
- und viele weitere...

Mit dem Basis-Paket ist fli4l ein reiner Ethernet-Router. Für ISDN und/oder DSL ist das Paket isdn und/oder dsl in dem fli4l-Verzeichnis auszupacken. Entsprechendes gilt für die anderen Pakete.

8.3.1. mkfli4l

Aus den Paketen wird in Abhängigkeit von der konkreten Konfiguration eine Konfigurationsdatei namens `rc.cfg` und zwei Archive namens `rootfs.img` und `opt.img` erstellt, die alle Konfigurationsinformationen und alle benötigten Dateien enthalten. Diese Dateien werden mit Hilfe von `mkfli4l` erzeugt, welches die einzelnen Pakete einliest und auf Fehler in der Konfiguration prüft.

`mkfli4l` akzeptiert die in Tabelle 8.1 angegebenen Parameter. Fehlen sie, werden die in Klammern angegebenen default-Werte genommen. Eine vollständige Liste der Optionen (Tabelle 8.1) erhält man, wenn man

```
mkfli4l -h
```

aufruft.

8.3.2. Aufbau

Ein Paket kann mehrere OPTs enthalten, wenn es aber nur eins enthält, ist es allerdings zweckmäßig, das Paket genauso wie das OPT zu nennen. Im Folgenden ist `<PACKAGE>` durch den jeweiligen Paket-Namen zu ersetzen. Ein Paket besteht aus folgenden Teilen:

- Verwaltungsdateien
- Dokumentation
- Entwickler-Dokumentation

Tabelle 8.1.: Parameter für `mkfli4l`

Option	Bedeutung
-c, --config	Setzen des Verzeichnisses, in dem `mkfli4l` die config-Dateien der Pakete sucht (default: config)
-x, --check	Setzen des Verzeichnisses, in dem `mkfli4l` die zum Prüfen der Pakete benötigten Dateien sucht (`<package>.txt`, `<package>.exp` und `<package>.ext`; default: check)
-l, --log	Setzen der Logdatei; `mkfli4l` protokolliert Fehlermeldungen und Warnungen in dieser Datei (default: `img/mkfli4l.log`)
-p, --package	Angabe der Pakete, die geprüft werden sollen, diese Option kann mehrmals angegeben werden, wenn man mehrere Pakete im Zusammenhang prüfen will. Bei Verwendung von -p wird allerdings grundsätzlich zuerst die Datei `<check_dir>/base.exp` eingelesen, um die allgemeinen regulären Ausdrücke, die vom Basis-Paket bereitgestellt werden, zur Verfügung zu stellen. Diese Datei muss also existieren.
-i, --info	Gibt Auskunft über den Verlauf der Prüfung (welche Dateien werden gelesen, welche Prüfungen werden durchgeführt, welche besonderen Dinge traten während des Prüfprozesses auf)
-v, --verbose	Ausführlichere Variante von -i
-h, --help	Zeigt die Hilfe an
-d, --debug	Gestattet das Debuggen des Prüfprozesses. Dies ist als Hilfe für Paketentwickler gedacht, die etwas genauer wissen möchten, wie die Prüfung des Paketes abläuft.

Debugoption	Bedeutung
check	show check process
zip-list	show generation of zip list
zip-list-skipped	show skipped files
zip-list-regexp	show regular expressions for ziplist
opt-files	check all files in `opt/<package>.txt`
ext-trace	show trace of extended checks

- Client-Programme
- Quellcode
- Weitere Dateien

Die einzelnen Teile sind im Folgenden näher beschrieben.

8.3.3. Die Konfiguration der Pakete

In der Datei `config/<PACKAGE>.txt` werden vom Benutzer Änderungen an der Konfiguration des Pakets vorgenommen. Alle Variablen eines OPTs sollten einheitlich mit dem Namen des OPTs beginnen, also zum Beispiel:

```
#-----------------------------------------------------------------
# Optional package: TELNETD
#-----------------------------------------------------------------
OPT_TELNETD='no'        # install telnetd: yes or no
TELNETD_PORT='23'       # telnet port, see also FIREWALL_DENY_PORT_x
```

Ein OPT sollte in der Konfigurationsdatei durch einen Header (siehe oben) entsprechend abgegrenzt werden. Dies erhöht die Übersichtlichkeit, zumal ein Paket ja auch mehrere OPTs enthalten kann. Die dem OPT zugehörigen Variablen sollten — ebenfalls im Interesse der Übersichtlichkeit — nicht weiter eingerückt werden. Kommentare und Leerzeilen sind erlaubt, wobei Kommentare einheitlich in Spalte 33 beginnen sollen. Ist eine Variable inklusive ihrer Belegung länger als 32 Zeichen, ist der Kommentar eine Zeile versetzt ab Spalte 33 einzufügen. Längere Kommentare werden jeweils ab Spalte 33 beginnend auf mehrere Zeilen aufgeteilt. Diese Maßnahmen sollen die Lesbarkeit der Konfigurationsdatei erhöhen.

Alle Werte hinter dem Gleichheitszeichen müssen in Hochkommata[1] eingefasst werden, da es sonst beim Booten zu Problemen kommen kann.

Variablen, die aktiv sind (s. u.), werden in die `rc.cfg` übernommen, alles andere wird ignoriert. Einzige Ausnahme sind Variablen mit dem Namen `<PACKAGE>_DO_DEBUG`. Diese dienen zur Fehlersuche in Paketen und werden pauschal übernommen.

8.3.4. Die Liste der zu kopierenden Dateien

Die Datei `opt/<PACKAGE>.txt` enthält Anweisungen, die beschreiben

- welche Dateien zu welchem OPT gehören,
- wann sie in das zu generierende `opt.img` bzw. ins `rootfs.img` übernommen werden sollen,
- welche User-ID (uid), Group-ID (gid) und Rechte sie bekommen soll,
- welche Konvertierungen vor der Aufnahme ins Archiv erfolgen sollen.

[1] Es können sowohl einfache Hochkommata als auch doppelte Hochkommata verwendet werden. Man kann also `FOO='bar'` oder auch `FOO="bar"` schreiben. Die Verwendung von doppelten Hochkommata sollte allerdings die Ausnahme sein und man sollte sich vorher unbedingt darüber informieren, wie eine Unix-Shell mit einfachen und doppelten Hochkommata umgeht.

`mkfli4l` generiert darauf basierend die erforderlichen Archive.

Leere Zeilen und Zeilen, die mit „#“ anfangen, werden ignoriert. In einer der ersten Zeilen sollte die Version des Paket-Dateiformats wie folgt stehen:

```
<erste Spalte>      <zweite Spalte> <dritte Spalte>
opt_format_version  1                   -
```

Die restlichen Zeilen haben folgende Syntax:

```
<erste Spalte>  <zweite Spalte> <dritte Spalte> <folgende spalten>
Variable        Wert            Datei           Optionen
```

1. In der ersten Spalte steht der Name einer Variable, von deren Wert das Übernehmen der in der dritten Spalte stehenden Datei abhängt. Der Name einer Variable kann beliebig oft in der ersten Spalte auftauchen, falls mehrere Dateien von ihr abhängen. Jede Variable, die in der `opt/<PACKAGE>.txt`-Datei auftaucht, wird von `mkfli4l` markiert.

   Falls mehrere Variablen auf denselben Wert geprüft werden sollen, kann auch eine Liste von Variablen (durch Kommata getrennt) verwendet werden. In diesem Falle reicht es aus, wenn mindestens *eine* Variable den in der zweiten Spalte geforderten Wert enthält. Wichtig ist dabei, dass zwischen den einzelnen Variablen *keine* Leerzeichen stehen!

   Bei OPT-Variablen (also Variablen, die mit `OPT_` beginnen und typischerweise den Typ `YESNO` haben) kann das Präfix „`OPT_`“ weggelassen werden. Des Weiteren ist es unwichtig, ob Variablen in Groß- oder in Kleinbuchstaben (oder beliebig gemischt) notiert werden.

2. In der zweiten Spalte steht ein Wert. Stimmt die in der ersten Spalte stehende Variable mit diesem Wert überein und ist die Variable aktiv (s. u.), wird die Datei in der dritten Spalte übernommen. Steht eine %-Variable in der ersten Spalte, wird über alle Indizes iteriert und geprüft, ob die jeweilige Variable mit dem Wert übereinstimmt. Ist das der Fall, wird kopiert. Zusätzlich wird vermerkt, dass aufgrund des aktuellen Wertes der Variable eine Datei kopiert wurde.

   Es ist möglich, vor den Wert ein „!“ zu schreiben. In diesem Falle wird die Prüfung negiert, d. h. die Datei wird genau dann kopiert, wenn die Variable diesen Wert *nicht* enthält.

3. In der dritten Spalte steht der Name einer Datei. Die Pfadangabe erfolgt relativ zum `opt`-Verzeichnis. Die Datei muss existieren und lesbar sein, sonst gibt es beim Generieren des Bootmediums einen Fehler und die Generierung wird abgebrochen.

   Beginnt der Dateiname mit einem „`rootfs:`“-Präfix, wird die Datei in die Liste der ins RootFS aufzunehmenden Dateien übernommen. Der Präfix wird vorher entfernt.

   Liegt die Datei unterhalb der aktuellen Konfigurationsverzeichnisses, wird sie in die Liste der aus dem Konfigurationsverzeichnis zu übernehmenden Dateien aufgenommen, andernfalls wird die unter `opt/` liegende Datei genommen. Die Datei darf dann kein `rootfs:`-Präfix haben.

   Ist die zu kopierende Datei ein Kernel-Modul, kann man die konkrete Kernel-Version durch `${KERNEL_VERSION}` ersetzen. `mkfli4l` nimmt dann die Version aus der Konfiguration und setzt sie hier ein. Dadurch kann man einem Paket Module für verschiedene Kern-Versionen mitgeben, und es wird immer die für den Kern richtige Version auf den

Router kopiert. Für Kernel-Module kann der Pfad auch vollkommen entfallen, `mkfli4l` findet das Modul anhand von `modules.dep` und `modules.alias`, siehe den Abschnitt „Automatische Auflösung von Abhängigkeiten für Kernel-Module“ (Seite 312).

Tabelle 8.2.: Optionen für Dateien

Option	Bedeutung	Standardwert
type=	Der Typ des Eintrags: *local* Dateisystem-Objekt *file* Datei *dir* Verzeichnis *node* Gerät *symlink* (symbolische) Verknüpfung Wenn vorhanden, muss diese Option an erster Stelle stehen. Der Typ „local“ steht hierbei für den Typ eines im Dateisystem existierenden Objekts und entspricht somit „file“, „dir“, „node“ oder „symlink“ (je nachdem). Die anderen Typen mit Ausnahme von „file“ können Einträge im Archiv erzeugen, die nicht im lokalen Dateisystem vorliegen müssen. Das wird z. B. benutzt, um Gerätedateien im RootFS-Archiv anzulegen.	local
uid=	Der Eigentümer der Datei, entweder numerisch oder als Name aus passwd	root
gid=	Die Gruppe der Datei, entweder numerisch oder als Name aus group	root
mode=	Die Zugriffsrechte	Dateien und Geräte: `rw-r--r--` (644) Verzeichnisse: `rwxr-xr-x` (755) Verknüpfungen: `rwxrwxrwx` (777)
flags= (type=file)	Konvertierungen vor der Aufnahme ins Archiv: *utxt* Konvertierung ins Unix-Format *dtxt* Konvertierung ins DOS-Format *sh* Shell-Skript: Konvertierung ins Unix-Format, Entfernen überflüssiger Zeichen	
name=	Alternativer Name, unter dem der Eintrag ins Archiv aufgenommen wird	
devtype= (type=node)	Beschreibt den Typ des Geräts („c“ für zeichenorientierte und „b“ für blockorientierte Geräte). Muss an zweiter Stelle stehen.	
major= (type=node)	Beschreibt die so genannte „Major“-Nummer der Gerätedatei. Muss an dritter Stelle stehen.	
minor= (type=node)	Beschreibt die so genannte „Minor“-Nummer der Gerätedatei. Muss an vierter Stelle stehen.	
linktarget= (type=symlink)	Beschreibt das Ziel der symbolischen Verknüpfung. Muss an zweiter Stelle stehen.	

4. In den anderen Spalten können die in Tabelle 8.2 aufgeführten Optionen für den Eigentümer, die Gruppe, die Rechte der Dateien und Konvertierungen stehen.

Einige Beispiele:

- kopiere Datei, wenn `OPT_TELNETD='yes'`, setze uid/gid auf root und die Rechte auf 755 (`rwxr-xr-x`)

```
telnetd     yes    files/usr/sbin/in.telnetd mode=755
```

- kopiere Datei, setze uid/gid auf root, die Rechte auf 555 (`r-xr-xr-x`) und konvertiere die Datei ins Unix-Format bei gleichzeitigem Entfernen aller überflüssigen Zeichen

```
base    yes     etc/rc0.d/rc500.killall mode=555 flags=sh
```

- kopiere Datei, wenn `PCMCIA_PCIC='i82365'`, setze uid/gid auf root und die Rechte auf 644 (`rw-r--r--`)

```
pcmcia_pcic i82365 files/lib/modules/${KERNEL_VERSION}/pcmcia/i82365.ko
```

- kopiere Datei, wenn eine der `NET_DRV_%`-Variablen mit dem zweiten Feld übereinstimmt, setze uid/gid auf root und die Rechte auf 644 (`rw-r--r--`)

```
net_drv_%   3c503  3c503.ko
```

- kopiere Datei, wenn die Variable `POWERMANAGEMENT` *nicht* den Wert „none“ enthält:

```
powermanagement !none etc/rc.d/rc100.pm mode=555 flags=sh
```

- kopiere Datei, wenn irgendeine der OPT-Variablen `OPT_MYOPTA` oder `OPT_MYOPTB` den Wert „yes“ enthält:

```
myopta,myoptb yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
```

  Dieses Beispiel ist letztlich nur eine Kurzschreibweise für:

```
myopta yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
myoptb yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
```

  Und letzteres ist eine Kurzschreibweise für:

```
opt_myopta yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
opt_myoptb yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
```

- kopiere Datei `opt/files/usr/bin/beep.sh` ins RootFS-Archiv, aber benenne sie vorher in `bin/beep` um:

```
base yes rootfs:files/usr/bin/beep.sh mode=555 flags=sh name=bin/beep
```

Die Dateien werden nur kopiert, wenn die oben genannten Bedingungen erfüllt sind und das dazugehörige `OPT_PACKAGE='yes'` gesetzt ist. Welche OPT-Variable dazugehört, wird über die Datei `check/<PACKAGE>.txt` beschrieben.

Wenn im Paket eine Variable referenziert wird, die nicht vom Paket selbst definiert wird, kann es passieren, dass das entsprechende Paket nicht installiert ist. Das würde zu einer Fehlermeldung in `mkfli4l` führen, da es erwartet, dass alle von `opt/<PACKAGE>.txt` referenzierten Variablen definiert sind.

Um diese Situation korrekt handhaben zu können, wurde die „weak“-Deklaration eingeführt. Sie hat das folgende Format:

```
weak        <Variable>    -
```

Dadurch wird die Variable definiert, wenn sie nicht bereits vorhanden ist und ihr Wert wird auf „undefiniert“ gesetzt. Dabei ist jedoch zu beachten, dass hier das „`OPT_`“-Präfix *nicht* weggelassen werden darf (falls es existiert), weil sonst eine Variable *ohne* dieses Präfix definiert wird.

Ein Beispiel aus der `opt/rrdtool.txt`:

```
weak opt_openvpn -
[...]
openvpn    yes    files/usr/lib/collectd/openvpn.so
```

Ohne die `weak`-Definition würde `mkfli4l` bei der Nutzung des Pakets „rrdtool“ eine Fehlermeldung anzeigen, wenn das „openvpn“-Paket nicht ebenfalls vorliegt. Mit Hilfe der `weak`-Definition kommt auch in dem Fall, dass das „openvpn“-Paket nicht vorliegt, keine Fehlermeldung.

Konfigurations-spezifische Dateien

In manchen Situationen möchte man originale Dateien im Archiv durch konfigurationsspezifische Dateien wie z. B. Host-Keys, eigene Firewall-Skripte, ... ersetzen. `mkfli4l` unterstützt dieses Szenario, indem es prüft, ob eine zu kopierende Datei im Konfigurationsverzeichnis zu finden ist und übernimmt in diesem Falle diese Datei in die Liste der ins `opt.img` bzw. `rootfs.img` aufzunehmenden Dateien.

Eine weitere Möglichkeit, konfigurationsspezifische Dateien ans Archiv anzuhängen wird im Abschnitt Erweiterte Prüfungen der Konfiguration (Seite 331) beschrieben.

Automatische Auflösung von Abhängigkeiten für Kernel-Module

Kernel-Module bauen unter Umständen auf anderen Kernel-Modulen auf. Diese müssen vor ihnen geladen werden und daher gleichfalls in das Archiv aufgenommen werden. `mkfli4l` bestimmt diese Abhängigkeiten anhand von `modules.dep` und `modules.alias` (zweier beim Kernel-Bau generierter Dateien) und nimmt automatisch alle benötigten Module in die Archive auf. So führt z. B. folgender Eintrag

```
net_drv_%   ne2k-pci    ne2k-pci.ko
```

dazu, dass sowohl 8390.ko als auch crc32.ko ins Archiv aufgenommen werden, da ne2k_pci von beiden abhängt.

Die notwendigen Einträge in `modules.dep` und `modules.alias` werden in das RootFS mit aufgenommen und können von `modprobe` zum Laden der Treiber genutzt werden.

8.3.5. Die Prüfung der Konfiguration-Variablen

Mit Hilfe der Datei `check/<PACKAGE>.txt` können die Inhalte der Variablen auf Gültigkeit überprüft werden. Diese Überprüfung war in früheren Versionen fest im Programm `mkfli4l` eingebaut, wurde aber im Zuge der Modularisierung von fli4l in die Check-Dateien ausgelagert. In dieser Datei ist für jede Variable aus den Konfigurationsdateien eine Zeile vorhanden. Diese Zeilen bestehen aus vier bis fünf Spalten, welche folgende Funktionen haben:

1. Variable: Diese Spalte gibt den Namen der zu überprüfenden Variable aus der Konfigurationsdatei an. Wenn es sich dabei um eine so genannte *Array-Variable* handelt, die mehrmals mit verschiedenen Indizes auftauchen kann, wird an Stelle der Nummer ein Prozentzeichen (%) in den Variablenname eingefügt. Dieses wird immer als „`_%_`“ in der Mitte eines Namens bzw. „`_%`“ am Ende eines Namens verwendet. Der Name kann dabei mehrere Prozentzeichen enthalten, so dass man auch mehrdimensionale Arrays realisieren kann. Dann sollte zwischen den Prozentzeichen allerdings etwas stehen, muss aber nicht, was dann allerdings zu so seltsamen Namen wie „`FOO_%__%`“ führt.

   Oftmals hat man das Problem, dass bestimmte Variablen Optionen beschreiben, die man nur in bestimmten Situationen benötigt. Deshalb können Variablen als optional markiert werden. Optionale Variablen werden mit einem vorangestellten „+“ gekennzeichnet. Sie können dann da sein, müssen aber nicht. Arrays können auch mit einem „++“ Präfix versehen werden. Steht ein „+“ davor, kann das Array da sein oder ganz fehlen. Steht „++“ davor, können zusätzlich auch noch einzelne Elemente des Arrays fehlen.

2. `OPT_VARIABLE`: Diese Spalte teilt die Variable einem bestimmten OPT zu. Die Variable wird nur auf Gültigkeit überprüft, wenn die hier angegebene Variable auf „yes“ steht. Gibt es keine OPT-Variable, ist hier ein „-“ anzugeben. In diesem Fall muss die Variable in der Konfigurationsdatei definiert werden, es sei denn, es wird eine Standard-Belegung definiert (s. u.). Der Name der OPT-Variable kann beliebig sein, er sollte jedoch mit dem Präfix „`OPT_`“ beginnen.

   Falls eine Variable von keiner OPT-Variablen abhängt, gilt sie als *aktiv*. Falls sie von einer OPT-Variablen abhängig ist, ist sie genau dann aktiv, wenn

   - ihre OPT-Variable aktiv ist und
   - ihre OPT-Variable den Wert „yes“ enthält.

   Andernfalls ist die Variable inaktiv.

   Hinweis: Inaktive OPT-Variablen werden, wenn sie in der Konfiguration mit „yes“ belegt werden, auf den Wert „no“ zurückgesetzt; dies wird von `mkfli4l` auch mit einer entsprechenden Warnmeldung (bspw. „`OPT_Y='yes' ignored, because OPT_X='no'`“) kommentiert. Bei transitiven Abhängigkeitsketten (`OPT_Z` hängt von `OPT_Y` ab, das wiederum von `OPT_X` abhängt) funktioniert dies aber nur dann zuverlässig, wenn die Namen aller OPT-Variablen mit „`OPT_`“ beginnen.

3. `VARIABLE_N`: Steht in der ersten Spalte eine Variable mit einem % im Namen, wird hier die Variable angegeben, die die Häufigkeit des Auftretens der Variable beschreibt (die so genannte *N-Variable*). Ist die Variable mehrdimensional, wird die Häufigkeit des letzten Index beschrieben. Hängt die Variable von einem OPT ab, muss die N-Variable vom selben OPT oder von keinem OPT abhängig sein. Ist die Variable von keinem OPT abhängig, darf auch die N-Variable von keinem OPT abhängig sein. Gibt es keine N-Variable, ist hier ein „-“ anzugeben.

   Aus Kompatibilitätsgründen mit zukünftigen fli4l-Versionen *muss* die hier angegebene Variable identisch sein mit der Variable in `OPT_VARIABLE`, wobei das letzte „%“ durch ein „N“ ersetzt und alles dahinter entfernt wurde. Ein Array `HOST_%_IP4` bekommt also zwingend die N-Variable `HOST_N` zugewiesen und ein Array `PF_USR_CHAIN_%_RULE_%` also die N-Variable `PF_USR_CHAIN_%_RULE_N`, und diese N-Variable ist selbst wieder eine Array-Variable mit der zugehörigen N-Variable `PF_USR_CHAIN_N`. *Alle anderen Benennungen der N-Variable werden mit zukünftigen fli4l-Versionen inkompatibel sein!*

4. `VALUE`: Diese Spalte gibt an, welche Werte für diese Variable eingegeben werden können. Es sind dabei z. B. folgende Angaben möglich:

Name	Bedeutung
`NONE`	Es wird keine Überprüfung vorgenommen
`YESNO`	Die Variable muss „yes“ oder „no“ sein
`NOTEMPTY`	Die Variable darf nicht leer sein
`NOBLANK`	Die Variable darf kein Leerzeichen enthalten
`NUMERIC`	Die Variable muss numerisch sein
`IPADDR`	Die Variable muss eine IP-Adresse sein
`DIALMODE`	Die Variable muss „on“, „off“ oder „auto“ sein

   Werden die Werte mit einem „`WARN_`“-Präfix versehen, so führt ein illegaler Wert nicht zu einer Fehlermeldung und damit zu einem Abbruch von `mkfli4l`, sondern nur zur Ausgabe einer Warnung.

   Die möglichen Prüfungen werden durch reguläre Ausdrücke in `check/base.exp` definiert. Diese Datei kann erweitert werden und enhält neuerdings z. B. zusätzlich folgende Prüfungen: `HEX`, `NUMHEX`, `IP_ROUTE`, `DISK` und `PARTITION`.

   Die Anzahl der Ausdrücke kann jederzeit erweitert werden, hier ist Rückmeldung von den Paket-Entwicklern erforderlich.

   Zusätzlich können reguläre Ausdrücke auch direkt in den Check-Dateien angegeben werden, wobei man auch Bezug auf existierende Ausdrücke nehmen kann. Statt `YESNO` könnte man z. B. auch

   ```
   RE:yes|no
   ```

   schreiben. Sinnvoll ist es dann, wenn ein Test nur ein einziges Mal ausgeführt wird und relativ einfach ist. Für genauere Informationen siehe nächstes Kapitel.

5. Standard-Belegung: In dieser Spalte kann optional ein Standard-Wert für die Variable stehen, falls die Variable nicht in der Konfiguration steht.

   Hinweis: Dies funktioniert zur Zeit jedoch nicht für Array-Variablen. Auch darf die Variable nicht optional sein, es darf also kein „+“ vor dem Variablennamen stehen.

Beispiel:

```
OPT_TELNETD     -       -       YESNO     "no"
```

Fehlt `OPT_TELNETD` nun in der Konfigurationsdatei, wird „no“ angenommen und dieser Wert auch in die `rc.cfg` geschrieben.

Die Sache mit dem Prozentzeichen lässt sich am Besten mit einem Beispiel erklären. Nehmen wir an, in der `check/base.txt` steht:

```
NET_DRV_N           -                   -                   NUMERIC
NET_DRV_%           -                   NET_DRV_N           NONE
NET_DRV_%_OPTION    -                   NET_DRV_N           NONE
```

Das heißt, dass je nach Wert von `NET_DRV_N` die Variablen `NET_DRV_N`, `NET_DRV_1_OPTION`, `NET_DRV_2_OPTION`, `NET_DRV_3_OPTION`, usw. überprüft werden.

8.3.6. Eigene Definitionen zum Prüfen der Konfigurationsvariablen

Einführung regulärer Ausdrücke

In der Version 2.0 gab es nur die oben angeführten sieben Werte-Bereiche, auf die Variablen geprüft werden können: `NONE`, `NOTEMPTY`, `NUMERIC`, `IPADDR`, `YESNO`, `NOBLANK`, `DIALMODE`. Die Überprüfung war in `mkfli4l` fest eingebaut, nicht erweiterbar und beschränkte sich auf wesentliche „Datentypen“, die mit vertretbarem Aufwand geprüft werden können.

Mit der Version 2.1 wurde diese Prüfung neu implementiert. Ziel der neuen Implementierung ist eine flexiblere Prüfung der Variablen, die auch in der Lage ist, komplexere Ausdrücke zu prüfen. Deshalb werden reguläre Ausdrücke verwendet, die in einem oder mehreren separaten Dateien abgespeichert werden. Dadurch wird es zum einen möglich, Variablen zu prüfen, die im Augenblick noch nicht geprüft werden, und zum anderen können Entwickler optionaler Pakete eigene Ausdrücke definieren, um die Konfiguration ihrer Pakete prüfen zu lassen.

Eine Beschreibung regulärer Ausdrücke findet man via „man 7 regex“ oder z. B. hier: http://unixhelp.ed.ac.uk/CGI/man-cgi?regex+7.

Spezifikation regulärer Ausdrücke

Spezifizieren kann man die Ausdrücke auf zwei Wegen:

1. Paketspezifische exp-Datei `check/<PACKAGE>.exp`

   Diese Datei liegt im `check`-Verzeichnis und trägt den gleichen Namen wie das dazugehörige Paket, also z. B. `check/base.exp`. Sie enthält Definitionen für Ausdrücke, die in der Datei `check/<PACKAGE>.txt` referenziert werden können. So enthält `check/base.exp` im Augenblick Definitionen für die bekannten Prüfungen und `check/isdn.exp` eine Definition für die Variable `ISDN_CIRC_x_ROUTE` (das Fehlen dieser Überprüfung war der Auslöser dieser Änderungen).

   Die Syntax lautet wie folgt, wobei man auch hier bei Bedarf doppelte Hochkommata verwenden kann:

   ```
   <Name> = '<Regulärer Ausdruck>' : '<Fehlermeldung>'
   ```

oder am Beispiel aus `check/base.exp`:

```
NOTEMPTY = '.*[^ ]+.*'          : 'should not be empty'
YESNO    = 'yes|no'             : 'only yes or no are allowed'
NUMERIC  = '0|[1-9][0-9]*'      : 'should be numeric (decimal)'
OCTET    = '1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]'
         : 'should be a value between 0 and 255'
IPADDR   = '((RE:OCTET)\.){3}(RE:OCTET)' : 'invalid ipv4 address'
EIPADDR  = '()|(RE:IPADDR)'
         : 'should be empty or contain a valid ipv4 address'
NOBLANK  = '[^ ]+'              : 'should not contain spaces'
DIALMODE = 'auto|manual|off'    : 'only auto, manual or off are allowed'
NETWORKS = '(RE:NETWORK)([[:space:]]+(RE:NETWORK))*'
         : 'no valid network specification, should be one or more
            network address(es) followed by a netmask,
            for instance 192.168.6.0/24'
```

In den regulären Ausdrücken können auch Referenzen auf bereits existierende Definitionen enthalten sein. Diese werden dann einfach an der Stelle eingefügt. Dadurch ist es einfacher, reguläre Ausdrücke zu konstruieren. Eingefügt werden die Referenzen einfach durch '(RE:Referenz)'. (Siehe die Definition des Ausdrucks `NETWORKS` oben für ein entsprechendes Beispiel.)

Die Fehlermeldungen tendieren dazu, zu lang zu werden. Daher besteht die Möglichkeit, sie über mehrere Zeilen zu verteilen. Die folgenden Zeilen müssen dann immer mit einem Leerzeichen oder Tabulator beginnen. Beim Einlesen der `check/<PACKAGE>.exp`-Datei werden überflüssige Leerzeichen auf eins reduziert und Tabulatoren durch Leerzeichen ersetzt. Ein Eintrag in der `check/<PACKAGE>.exp` könnte dann so aussehen:

```
NUM_HEX         = '0x[[:xdigit:]]+'
                : 'should be a hexadecimal number
                   (a number starting with "0x")'
```

2. Reguläre Ausdrücke direkt in der Check-Datei `check/<PACKAGE>.txt`

   Manche Ausdrücke kommen nur einmal vor, dann lohnt es sich nicht, dafür einen regulären Ausdruck in einer `check/<PACKAGE>.exp`-Datei zu definieren. Dann kann man diesen Ausdruck einfach in die Check-Datei schreiben, z. B.:

   ```
   # Variable      OPT_VARIABLE    VARIABLE_N      VALUE
   MOUNT_BOOT      -               -               RE:ro|rw|no
   ```

   `MOUNT_BOOT` kann lediglich die Werte „ro“, „rw“ oder „no“ annehmen, alles andere wird abgelehnt.

   Will man Bezug auf existierende reguläre Ausdrücke nehmen, fügt man einfach eine Referenz via „(RE:...)“ ein. Beispiel:

   ```
   # Variable      OPT_VARIABLE    VARIABLE_N      VALUE
   LOGIP_LOGDIR    OPT_LOGIP       -               RE:(RE:ABS_PATH)|auto
   ```

Erweiterung existierender regulärer Ausdrücke

Fügt ein optionales Paket einen zusätzlichen Wert für eine Variable hinzu, die von einem regulären Ausdruck geprüft wird, muss der reguläre Ausdruck erweitert werden. Dies geschieht einfach durch Definition der neuen möglichen Werte durch einen regulären Ausdruck (wie oben beschrieben) und Ergänzung des bestehenden regulären Ausdrucks in einer eigenen `check/<PACKAGE>.exp`-Datei. Dass ein bestehender Ausdruck modifiziert werden soll, kennzeichnet ein führendes „+“. Der neue Ausdruck ergänzt den bestehenden Ausdruck, indem der neue Wert als Alternative an den bestehenden Wert angehängt wird. Verwendet ein anderer Ausdruck den ergänzten Ausdruck, gilt auch dort die Ergänzung. Die angegebene Fehlermeldung wird einfach an die vorhandene hinten angehängt.

Am Beispiel der Ethernet-Treiber könnte das wie folgt aussehen:

- Das Basis-Paket stellt eine Menge von Ethernet-Treibern bereit und prüft die Variable `NET_DRV_x` mit dem regulären Ausdruck `NET_DRV`, der wie folgt spezifiert ist:

```
NET_DRV         = '3c503|3c505|3c507|...'
                : 'invalid ethernet driver, please choose one'
                  ' of the drivers in config/base.txt'
```

- Das Paket „pcmcia“ stellt jetzt zusätzliche Gerätetreiber bereit, muss also `NET_DRV` ergänzen. Das sieht dann wie folgt aus:

```
PCMCIA_NET_DRV = 'pcnet_cs|xirc2ps_cs|3c574_cs|...' : ''
+NET_DRV       = '(RE:PCMCIA_NET_DRV)' : ''
```

Nun kann man zusätzlich auch noch PCMCIA-Treiber auswählen.

Regulären Ausdruck in Abhängigkeit von `YESNO`-Variablen erweitern

Wenn man `NET_DRV` wie oben um die PCMCIA-Treiber erweitert hat, aber das Paket „pcmcia“ deaktiviert hat, könnte man dennoch einen PCMCIA-Treiber in der `config/base.txt` auswählen, ohne dass eine Fehlermeldung beim Erstellen der Archive auftritt. Um das zu verhindern, kann man den regulären Ausdruck auch abhängig von einer `YESNO`-Variablen in der Konfiguration erweitern. Dazu wird der Name der Variablen, die bestimmt ob der Ausdruck erweitert wird, mit runden Klammern direkt hinter den Namen des Ausdrucks gehängt. Ist die Variable aktiv und hat den Wert „yes“, wird der Ausdruck erweitert, sonst nicht.

```
PCMCIA_NET_DRV       = 'pcnet_cs|xirc2ps_cs|3c574_cs|...' : ''
+NET_DRV(OPT_PCMCIA) = '(RE:PCMCIA_NET_DRV)' : ''
```

Wird jetzt `OPT_PCMCIA='no'` gesetzt, und in der `config/base.txt` wird z. B. der PCMCIA-Treiber `xirc2ps_cs` benutzt, gibt es beim Erstellen der Archive eine Fehlermeldung.

Hinweis: Dies funktioniert *nicht*, wenn die Variable nicht explizit in der Konfigurationsdatei gesetzt wird, sondern ihren Wert über eine Standard-Belegung in der `check/<PACKAGE>.txt` erhält. In diesem Fall muss man also in der Konfigurationsdatei die Variable explizit setzen und ggf. auf die Standard-Belegung verzichten.

Regulären Ausdruck in Abhängigkeit von anderen Variablen erweitern

Alternativ kann man auch beliebige Werte von Variablen als Bedingung verwenden, die Syntax sieht dann wie folgt aus:

```
+NET_DRV(KERNEL_VERSION=~'^3\.14\..*$') = ...
```

Wenn `KERNEL_VERSION` zu dem angegebenen regulären Ausdruck passt, also irgendein Kernel aus der 3.14er Versionsreihe genutzt wird, dann wird die Liste der erlaubten Netzwerktreiber um die angegebenen Treiber ergänzt.

Hinweis: Dies funktioniert *nicht*, wenn die Variable nicht explizit in der Konfigurationsdatei gesetzt wird, sondern ihren Wert über eine Standard-Belegung in der `check/<PACKAGE>.txt` erhält. In diesem Fall muss man also in der Konfigurationsdatei die Variable explizit setzen und ggf. auf die Standard-Belegung verzichten.

Fehlermeldungen

Findet die Prüfung einen Fehler, erscheint eine Fehlermeldung der folgenden Art:

```
Error: wrong value of variable HOSTNAME: '' (may not be empty)
Error: wrong value of variable MOUNT_OPT: 'rx' (user supplied regular expression)
```

Beim ersten Fehler wurde der Ausdruck in einer `check/<PACKAGE>.exp`-Datei definiert und ein Hinweis auf den Fehler wird mit ausgegeben. Im zweiten Falle wurde der Ausdruck direkt in einer `check/<PACKAGE>.txt`-Datei spezifiziert, deshalb gibt es keinen zusätzlichen Hinweis auf die Fehlerursache.

Definition regulärer Ausdrücke

Reguläre Ausdrücke sind wie folgt definiert:

Regulärer Ausdruck: Eine oder mehrere Alternativen, getrennt durch '|', z. B. „ro|rw|no“. Trifft eine der Alternativen zu, trifft der ganze Ausdruck zu (hier wären „ro“, „rw“ und „no“ gültige Ausdrücke).

Eine Alternative ist eine Verkettung mehrerer Teilstücke, die einfach aneinandergereiht werden.

Ein Teilstück ist ein „Atom“, gefolgt von einem einzelnen „*“, „+“, „?“ oder „{min, max}“. Die Bedeutung ist wie folgt:

- „a*“ — beliebig viele „a“s (erlaubt auch den Fall, das gar kein „a“ da ist)
- „a+“ — mindestens ein „a“
- „a?“ — kein oder ein „a“
- „a{2,5}“ — zwei bis fünf „a“s
- „a{5}“ — genau fünf „a“s
- „a{2,}“ — mindestens zwei „a“s
- „a{,5}“ — höchstens fünf „a“s

Ein „Atom“ ist ein

- regulärer Ausdruck eingeschlossen in Klammern, z. B. trifft „(a|b)+“ auf eine beliebige Zeichenkette zu, die mindestens ein „a“ oder „b“ enthält, sonst aber beliebig viele und in beliebiger Reihenfolge
- ein leeres Paar Klammern steht für einen „leeren“ Ausdruck
- ein Ausdruck mit eckigen Klammern „[]“ (siehe weiter unten)
- ein Punkt „.“, der auf irgendein einzelnes Zeichen zutrifft, z. B. trifft „.+“ auf eine beliebige Zeichenkette zu, die mindestens ein Zeichen enthält
- ein „^“ steht für den Zeilenanfang, z. B. trifft „^a.*“ auf eine Zeichenkette zu, die mit einem „a“ anfängt und in der beliebige Zeichen folgen, etwa „a“ oder „adkadhashdkash“
- ein „$“ steht für das Zeilenende
- ein „\“ gefolgt von einem der Sonderzeichen `^ . [ $ ( ) | * + ? { \` steht für genau das zweite Zeichen ohne seine spezielle Bedeutung
- ein normales Zeichen trifft auf genau das Zeichen zu, z. B. trifft „a“ genau auf „a“ zu.

Ein Ausdruck in rechteckigen Klammern bedeutet Folgendes:

- „x-y“ — trifft auf irgendein Zeichen zu, das zwischen „x“ und „y“ liegt, z. B. steht „[0-9]“ für alle Zeichen zwischen „0“ und „9“; „[a-zA-Z]“ steht für alle Buchstaben, egal ob groß oder klein
- „^ x-y“ — trifft auf irgendein Zeichen zu, das *nicht* im angegebenen Intervall liegt; so steht z. B. „[^ 0-9]“ für alle Zeichen, die *keine* Ziffern sind
- „[:*character-class*:]“ — trifft auf ein Zeichen der Zeichenklasse *character-class* zu. Relevante Standardzeichenklassen sind: `alnum`, `alpha`, `blank`, `digit`, `lower`, `print`, `punct`, `space`, `upper` und `xdigit`. So steht „[[:alpha:]]“ für alle Groß- und Kleinbuchstaben und ist somit identisch zu „[[:lower:] [:upper:]]“.

Beispiele für reguläre Ausdrücke

Sehen wir uns das mal an einigen Beispielen an!

`NUMERIC`: Ein numerischer Wert besteht aus mindestens einer, aber ansonsten beliebig vielen Ziffern. „Mindestens ein“ drückt man mit „+“ aus, eine Ziffer hatten wir schon als Beispiel. Zusammengesetzt ergibt das:

```
NUMERIC = '[0-9]+'
```

oder alternativ

```
NUMERIC = '[[:digit:]]+'
```

`NOBLANK`: Ein Wert, der keine Leerzeichen enthält, ist ein beliebiges Zeichen (außer dem Leerzeichen) und davon beliebig viele:

```
NOBLANK = '[^ ]*'
```

bzw. wenn der Wert zusätzlich auch nicht leer sein darf:

```
NOBLANK = '[^ ]+'
```

`IPADDR`: Sehen wir uns das Ganze nochmal am Beispiel der IPv4-Addresse an. Eine IPv4-Adresse besteht aus vier „Octets“, die durch einen Punkt („.“) voneinander getrennt sind. Ein Octet kann eine Zahl zwischen 0 und 255 sein. Definieren wir als erstes ein Octet. Es kann

eine Zahl zwischen 0 und 9 sein:	[0-9]
eine Zahl zwischen 10 und 99:	[1-9][0-9]
eine Zahl zwischen 100 und 199:	1[0-9][0-9]
eine Zahl zwischen 200 und 249:	2[0-4][0-9]
eine Zahl zwischen 250 und 255 sein:	25[0-5]

Das Ganze sind Alternativen, also fassen wir sie einfach mittels „|“ zu einem Ausdruck zusammen: „[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]“ und haben damit ein Octet. Daraus können wir nun eine IPv4-Adresse machen, vier Octets mit Punkten voneinander getrennt (der Punkt muss mittels eines *Backslashs* maskiert werden, da er sonst für ein beliebiges Zeichen steht). Basierend auf der Syntax der exp-Dateien sieht das Ganze dann wie folgt aus:

```
OCTET  = '[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]'
IPADDR = '((RE:OCTET)\.){3}(RE:OCTET)'
```

Unterstützung beim Entwurf regulärer Ausdrücke

Will man reguläre Ausdrücke entwerfen und testen, kann man dazu das „regexp“-Programm verwenden, das sich in dem Verzeichnis `unix` bzw. `windows` des Pakets „base“ befindet. Es akzeptiert die folgende Syntax:

```
usage: regexp [-c <check dir>] <regexp> <string>
```

Dabei bedeuten die Parameter Folgendes:

- `<check dir>` ist das Verzeichnis, das die Check-Dateien und damit auch die exp-Dateien enthält. Diese werden von „regexp“ eingelesen, damit man auf bereits definierte Ausdrücke zurückgreifen kann.
- `<regexp>` ist der reguläre Ausdruck (im Zweifelsfall immer in `'...'` oder `"..."` angeben, wobei doppelte Anführungsstriche nur nötig sind, wenn einfache Hochkommata in dem Ausdruck vorkommen sollen)
- `<string>` ist die zu prüfende Zeichenkette

Das könnte z. B. wie folgt aussehen:

```
./i586-linux-regexp -c ../check '[0-9]' 0
adding user defined regular expression='[0-9]' ('^([0-9])$')
checking '0' against regexp '[0-9]' ('^([0-9])$')
'[0-9]' matches '0'
```

```
./i586-linux-regexp -c ../check '[0-9]' a
adding user defined regular expression='[0-9]' ('^([0-9])$')
checking 'a' against regexp '[0-9]' ('^([0-9])$')
regex error 1 (No match) for value 'a' and regexp '[0-9]' ('^([0-9])$')

./i586-linux-regexp -c ../check IPADDR 192.168.0.1
using predefined regular expression from base.exp
adding IPADDR='((RE:OCTET)\.){3}(RE:OCTET)'
 ('^(((1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]))$')
'IPADDR' matches '192.168.0.1'

./i586-linux-regexp -c ../check IPADDR 192.168.0.256
using predefined regular expression from base.exp
adding IPADDR='((RE:OCTET)\.){3}(RE:OCTET)'
 ('^(((1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]))$')
regex error 1 (No match) for value '192.168.0.256' and regexp
 '((RE:OCTET)\.){3}(RE:OCTET)'
(unknown:-1) wrong value of variable cmd_var: '192.168.0.256' (invalid ipv4 address)
```

8.3.7. Erweiterte Prüfungen der Konfiguration

Manchmal ist es notwendig, komplexere Überprüfungen durchzuführen. Beispiele für solche komplexeren Dinge wären z. B. Abhängigkeiten zwischen Paketen oder Bedingungen, die nur erfüllt sein müssen, wenn Variablen bestimmte Werte annehmen. So muss z. B. bei Auswahl eines PCMCIA-ISDN-Adapters auch das Paket „pcmcia“ installiert werden.

Um diese Überprüfungen durchführen zu können, kann man in `check/<PACKAGE>.ext` (auch ext-Skript genannt) kleinere Tests schreiben. Die Sprache besteht aus folgenden Elementen:

1. Schlüsselwörter:
   - Kontrollfluss:
     - `if (`*`expr`*`) then `*`statement`*` else `*`statement`*` fi`
     - `foreach `*`var`*` in `*`set_var`*` do `*`statement`*` done`
     - `foreach `*`var`*` in `*`set_var_1 ... set_var_n`*` do `*`statement`*` done`
     - `foreach `*`var`*` in `*`var_n`*` do `*`statement`*` done`
   - Abhängigkeiten:
     - `provides `*`package`*` version `*`x.y.z`*
     - `depends on `*`package`*` version `*`x1.y1 x2.y2.z2 x3.y3 ...`*
   - Aktionen:
     - `warning "`*`warning`*`"`
     - `error "`*`error`*`"`
     - `fatal_error "`*`fatal error`*`"`
     - `set `*`var`*` = `*`value`*
     - `crypt (`*`variable`*`)`
     - `stat (`*`filename`*`, `*`res`*`)`

- `fgrep (filename, regex)`
- `split (string, set_variable, character)`

2. Datentypen: Zeichenketten, positive ganze Zahlen, Versionsnummern
3. Logische Operationen: `<`, `==`, `>`, `!=`, `!`, `&&`, `||`, `=~`, `copy_pending`, `samenet`, `subnet`

Datentypen

Zu den Datentypen ist zu sagen, dass Variablen auf Grund des zugehörigen regulären Ausdrucks fest einem Datentyp zugeordnet werden:

- Variablen, deren Typ mit „`NUM`“ beginnt, sind numerisch und enthalten positive ganze Zahlen
- Variablen, die eine N-Variable für irgendein Array sind, sind ebenfalls numerisch
- alle anderen Variablen werden wie Zeichenketten verarbeitet

Das bedeutet unter anderem, dass eine Variable vom Typ `ENUMERIC` *nicht* als Index beim Zugriff auf eine Array-Variable benutzt werden kann, auch wenn man sich vorher vergewissert hat, dass sie nicht leer ist. Der folgende Code funktioniert somit nicht:

```
# sei TEST eine Variable vom Typ ENUMERIC
if (test != "")
then
    # Fehler: You can't use a non-numeric ID in a numeric
    #         context. Check type of operand.
    set i=my_array[test]
    # Fehler: You can't use a non-numeric ID in a numeric
    #         context. Check type of operand.
    set j=test+2
fi
```

Eine Lösung für dieses Problem bietet `split` (Seite 330):

```
if (test != "")
then
    # alle Elemente von test_% sind numerisch
    split(test, test_%, ' ', numeric)
    # OK
    set i=my_array[test_%[1]]
    # OK
    set j=test_%[1]+2
fi
```

Zeichenketten und Variablenersetzung

An verschiedenen Stellen werden Zeichenketten benötigt, etwa wenn eine Warnung (Seite 325) ausgegeben werden soll. In einigen Fällen, die in dieser Dokumentation beschrieben werden, wird eine solche Zeichenkette dabei nach Variablen durchsucht; werden welche gefunden, werden diese durch ihren Inhalt oder andere Attribute *ersetzt*. Diese Ersetzung wird *Variablenersetzung* genannt.

Dies soll an einem Beispiel verdeutlicht werden. Es gelte die Konfiguration:

```
# config/base.txt
HOSTNAME='fli4l'
# config/dns_dhcp.txt
HOST_N='1' # Anzahl der Hosts
HOST_1_NAME='client'
HOST_1_IP4='192.168.1.1'
```

Dann werden die Zeichenketten wie folgt umgeschrieben, wenn die Variablenersetzung in dem jeweiligen Kontext aktiv ist:

```
"Mein Router heißt $HOSTNAME"
# --> "Mein Router heißt fli4l"
"HOSTNAME ist Teil des Pakets %{HOSTNAME}"
# --> "HOSTNAME ist Teil des Pakets base"
"@HOST_N ist $HOST_N"
# --> " # Anzahl der Hosts ist 1"
```

Wie man sehen kann, gibt es prinzipiell drei Möglichkeiten der Ersetzung:

- `$<Name>` bzw. `${<Name>}`: Ersetzt den Variablennamen durch den Inhalt der Variable. Dies ist die häufigste Form der Ersetzung. Der Name muss in `{...}` stehen, wenn direkt danach in der Zeichenkette ein Zeichen kommt, das gültiger Bestandteil eines Variablennamens sein kann, also ein Buchstabe, eine Ziffer oder ein Unterstrich. In allen anderen Fällen ist die Verwendung von geschweiften Klammern möglich, aber nicht zwingend.

- `%<Name>` bzw. `%{<Name>}`: Ersetzt den Variablennamen durch den Namen des Pakets, in dem die Variable definiert ist. Dies funktioniert *nicht* bei im Skript via `set` (Seite 325) zugewiesenen Variablen oder bei Laufvariablen einer `foreach`-Schleife (Seite 332), da solche Variablen kein Paket besitzen und für Laufvariablen diese Syntax eine andere Bedeutung erhält.

- `@<Name>` bzw. `@{<Name>}`: Ersetzt den Variablennamen durch den Kommentar, der in der Konfiguration hinter der Variablen steht. Auch dies ergibt keinen Sinn für im Skript definierte Variablen.

Will man ein „$“, „@“ oder „%“ im Text haben, schreibt man „$$“, „@@“ bzw. „%%“.
Hinweis: Elemente von Array-Variablen können auf diese Weise *nicht* in Zeichenketten integriert werden, weil es keine Möglichkeit gibt, einen Index anzugeben.
Generell unterliegen nur *Konstanten* der Variablenersetzung; Zeichenketten, die über eine Variable hereinkommen, bleiben unverändert. Ein Beispiel soll dies verdeutlichen - es sei die folgende Konfiguration gegeben:

```
HOSTNAME='fli4l'
TEST='${HOSTNAME}'
```

Dann führt der Code:

```
warning "${TEST}"
```

zur Ausgabe von:

```
Warning: ${HOSTNAME}
```

und *nicht* zur Ausgabe von:

```
Warning: fli4l
```

In den folgenden Abschnitten wird explizit darauf hingewiesen, unter welchen Umständen Zeichenketten der Variablenersetzung unterliegen.

Definition eines Dienstes mit einer dazugehörenden Versionsnummer: `provides`

Damit kann z. B. ein OPT deklarieren, dass es einen Drucker-Dienst oder einen Webserver-Dienst bereitstellt. Es kann jeweils nur ein einziges Paket geben, dass einen Dienst bereitstellt. Damit kann man verhindern, dass z. B. zwei Webserver parallel installiert werden, was naheliegenderweise nicht gehen würde, da sich die beiden Server um den Port 80 streiten würden. Zusätzlich wird die aktuelle Version des Dienstes angegeben, so dass Weiterentwicklungen Rechnung getragen werden kann. Die Versionsnummer besteht aus zwei- oder drei Zahlen, die durch Punkte voneinander getrennt sind, etwa „4.0“ oder „2.1.23“.

Typischerweise werden Dienste auf OPTs, nicht auf ganze Pakete abgebildet. So besitzt etwa das Paket „tools“ eine ganze Reihe von Programmen, die jeweils ihre eigene `provides`-Anweisung definieren, so sie denn via `OPT_...='yes'` aktiviert sind.

Die Syntax lautet:

```
provides <Name> version <Version>
```

Beispiel aus dem Paket „easycron“:

```
provides cron version 3.10.0
```

Die Versionsnummer sollte vom OPT-Entwickler in der dritten Komponente angehoben werden, wenn lediglich Funktionserweiterungen vorgenommen wurden und die Schnittstelle zum OPT kompatibel geblieben ist. Die Versionsnummer sollte in der ersten oder zweiten Komponente angehoben werden, wenn sich die Schnittstelle in irgendeiner Weise inkompatibel verändert hat (z. B. auf Grund von Variablenumbenennungen, Pfad-Änderungen, verschwundenen oder umbenannten Dienstprogrammen etc.).

Definition einer Abhängigkeit zu einem Dienst mit einer bestimmten Version: `depends`

Benötigt man zur Erbringung der eigenen Funktionalität einen anderen Dienst (z. B. einen Webserver), kann man hiermit diese Abhängigkeit zu einem Dienst mit einer bestimmten Version spezifizieren. Die Version kann zweistellig (z. B. „2.1“) oder dreistellig (z. B. „2.1.11“) angegeben werden, wobei die zweistellige Variante alle Versionen akzeptiert, die ebenfalls so beginnen, während die dreistellige Version nur genau diese angegebene Version akzeptiert. Des Weiteren kann eine Liste von solchen Versionsnummern angegeben werden, falls mehrere Versionen des Dienstes kompatibel mit dem Paket sind.

Die Syntax lautet:

```
depends on <Name> version <Version>+
```

Ein Beispiel: Paket „server“ enthalte:

```
provides server version 1.0.1
```

Sei ein Paket „client“ gegeben. Darin seien folgende `depends`-Anweisungen beispielhaft enthalten:[2]

```
depends on server version 1.0       # OK, '1.0' passt zu '1.0.1'
depends on server version 1.0.1     # OK, '1.0.1' passt zu '1.0.1'
depends on server version 1.0.2     # Fehler, '1.0.2' passt nicht zu '1.0.1'
depends on server version 1.1       # Fehler, '1.1' passt nicht zu '1.0.1'
depends on server version 1.0 1.1   # OK, '1.0' passt zu '1.0.1'
depends on server version 1.0.2 1.1 # Fehler, weder '1.0.2' noch '1.1' passen
                                    # zu '1.0.1'
```

Kommunikation mit dem Nutzer: `warning`, `error`, `fatal_error`

Mit Hilfe dieser drei Funktionen kann man Nutzer warnen, einen Fehler signalisieren oder die Prüfung sofort abbrechen. Die Syntax sieht wie folgt aus:

- `warning "text"`
- `error "text"`
- `fatal_error "text"`

Alle an diese Funktionen übergebenen Zeichenketten-Konstanten unterliegen der Variablenersetzung (Seite 322).

Zuweisungen

Benötigt man aus irgendeinem Grund eine temporäre Variable, kann man diese einfach mit „`set var [= value]`“ anlegen. *Die Variable darf kein Konfigurationsvariable sein!*[3] Lässt man den „= value“ Teil weg, wird die Variable einfach auf „yes“ gesetzt, so dass man sie hinterher einfach in einer `if`-Anweisung testen kann. Wird ein Zuweisungsteil angegeben, kann hinter dem Gleichheitszeichen alles stehen: normale Variablen, indizierte Variablen, Zahlen, Zeichenketten, Versionsnummern.

Zu beachten ist, dass durch diese Zuweisung gleichzeitig der *Typ* der temporären Variablen festgelegt wird. Wird eine Zahl zugewiesen, „merkt“ `mkfli4l` sich, dass diese Variable eine Zahl enthält, und erlaubt später das Rechnen damit. Versucht man, mit einer anders getypten Variable zu rechnen, wird dies fehlschlagen. Beispiel:

```
set i=1   # OK, i ist eine numerische Variable
set j=i+1 # OK, j ist eine numerische Variable und enthält den Wert 2
set i="1" # OK, i ist nun eine Zeichenketten-Variable
set j=i+1 # Fehler "You can't use a non-numeric ID in a numeric
          #         context. Check type of operand."
          # --> mit Zeichenketten kann man nicht rechnen!
```

Man kann auch temporäre Arrays (siehe unten) anlegen. Beispiel:

[2] Natürlich nur jeweils eine zur selben Zeit!
[3] Dies ist eine bewusste Entscheidung: Durch check-Skripte lässt sich die Benutzerkonfiguration *nicht* verändern.

```
set prim_%[1]=2
set prim_%[2]=3
set prim_%[3]=5
warning "${prim_n}"
```

Dabei wird die Anzahl der Elemente in dem Array in der Variable `prim_n` von `mkfli4l` verwaltet. Der obige Code führt somit zu folgender Ausgabe:

```
Warning: 3
```

Wenn auf der rechten Seite einer Zuweisung eine Zeichenketten-Konstante steht, unterliegt sie zum Zeitpunkt der Zuweisung der Variablenersetzung (Seite 322). Dies wird im folgenden Beispiel demonstriert. Der Code:

```
set s="a"
set v1="$s" # v1="a"
set s="b"
set v2="$s" # v2="b"
if (v1 == v2)
then
  warning "gleich"
else
  warning "ungleich"
fi
```

produziert die Ausgabe „ungleich“, weil die Variablen `v1` und `v2` bereits während der Zuweisung den aktuellen Inhalt der Variablen `s` ersetzen.

Hinweis: Eine in einem Skript gesetzte Variable ist bei der Abarbeitung weiterer Skripte sichtbar – es existiert zur Zeit kein Lokalitätsprinzip für derart eingeführte Variablen. Da die Reihenfolge, in der die Skripte verschiedener Pakete abgearbeitet wird, nicht definiert ist, sollte man sich nie darauf verlassen, dass Variablen irgendwelche Werte besitzen bzw. von einem anderen Paket übernommen haben.

Arrays

Will man auf einzelne Elemente einer %-Variablen (eines Arrays) zugreifen, muss man den Original-Namen der Variable, wie er in der `check/<PACKAGE>.txt`-Datei steht, verwenden, und dabei für jedes „%“-Zeichen einen Index mit Hilfe von „[*Index*]“ anhängen.

Beispiel: Will man auf die Elemente der Variable `PF_USR_CHAIN_%_RULE_%` zugreifen, benötigt man zwei Indizes, weil die Variable zwei „%“-Zeichen besitzt. Alle Elemente ausgeben kann man z. B. mit Hilfe des folgenden Codes (die `foreach`-Schleife wird weiter unten (Seite 332) erläutert):

```
foreach i in pf_usr_chain_n
do
    # nur ein Index nötig, da nur ein '%' im Variablennamen
    set j_n=pf_usr_chain_%_rule_n[i]
    # Achtung: ein
    # foreach j in pf_usr_chain_%_rule_n[i]
    # ist leider nicht möglich, deshalb der Umweg über j_n!
    foreach j in j_n
```

```
    do
        # zwei Indizes nötig, da zwei '%' im Variablennamen
        set rule=pf_usr_chain_%_rule_%[i][j]
        warning "Rule $i/$j: ${rule}"
    done
done
```

Mit der folgenden Beispiel-Konfiguration

```
PF_USR_CHAIN_N='2'
PF_USR_CHAIN_1_NAME='usr-chain_a'
PF_USR_CHAIN_1_RULE_N='2'
PF_USR_CHAIN_1_RULE_1='ACCEPT'
PF_USR_CHAIN_1_RULE_2='REJECT'
PF_USR_CHAIN_2_NAME='usr-chain_b'
PF_USR_CHAIN_2_RULE_N='1'
PF_USR_CHAIN_2_RULE_1='DROP'
```

gibt es dann die folgenden Ausgaben:

```
Warning: Rule 1/1: ACCEPT
Warning: Rule 1/2: REJECT
Warning: Rule 2/1: DROP
```

Alternativ kann man direkt über alle Werte des Arrays iterieren, kennt dann allerdings nicht die exakten Indizes der Einträge (was auch nicht immer erforderlich ist):

```
foreach rule in pf_usr_chain_%_rule_%
do
    warning "Rule %{rule}='${rule}'"
done
```

Das produziert mit der Beispiel-Konfiguration von oben die folgenden Ausgaben:

```
Warning: Rule PF_USR_CHAIN_1_RULE_1='ACCEPT'
Warning: Rule PF_USR_CHAIN_1_RULE_2='REJECT'
Warning: Rule PF_USR_CHAIN_2_RULE_1='DROP'
```

An dem zweiten Beispiel sieht man auch schön die Bedeutung der `%<Name>`-Syntax: Innerhalb der Zeichenkette wird `%rule` durch den *Namen* der betrachteten Variable ersetzt (also z. B. `PF_USR_CHAIN_1_RULE_1`), während `$rule` durch dessen *Inhalt* (also z. B. `ACCEPT`) ersetzt wird.

Verschlüsseln eines Passwortes: `crypt`

Einige Variablen enthalten Passwörter, die nicht im Klartext in der `rc.cfg` stehen sollen. Diese Variablen können mittels `crypt` verschlüsselt werden und werden damit in das Format überführt, dass auch auf dem Router benötigt wird. Verwendet wird das wie folgt:

```
crypt (<Variable>)
```

Die `crypt`-Funktion ist die *einzige* Stelle, an der eine Konfigurationsvariable verändert werden kann.

Abfragen von Eigenschaften einer Datei: `stat`

`stat` ermöglicht es, Eigenschaften einer Datei abzufragen. Zur Verfügung gestellt wird im Augenblick lediglich die Größe einer Datei. Wenn man auf Dateien unterhalb des aktuellen Konfigurationsverzeichnisses testen will, kann man die interne Variable `config_dir` benutzen. Die Syntax lautet:

```
stat (<Dateiname>, <Schlüssel>)
```

Der Aufruf sieht wie folgt aus (wobei die verwendeten Parameter nur Beispiele sind):

```
foreach i in openvpn_%_secret
do
   stat("${config_dir}/etc/openvpn/$i.secret", keyfile)
   if (keyfile_res != "OK")
   then
      error "OpenVPN: missing secretfile <config>/etc/openvpn/$i.secret"
   fi
done
```

In dem Beispiel wird geprüft, ob eine Datei im aktuellen Konfigurationsverzeichnis vorhanden ist. Wenn also `OPENVPN_1_SECRET='test'` in der Konfigurationsdatei gesetzt wird, prüft die Schleife im ersten Durchlauf, ob im aktuellen Konfigurationsverzeichnis die Datei `etc/openvpn/test.secret` vorhanden ist.

Nach dem Aufruf sind zwei Variablen definiert:

- `<Schlüssel>_res`: Resultat des Systemaufrufs stat() („OK“, wenn Systemruf erfolgreich, sonst Fehlermeldung des Systemaufrufs)
- `<Schlüssel>_size`: Größe der Datei

Das könnte dann z. B. so aussehen:

```
stat ("unix/Makefile", test)
if ("$test_res" == "OK")
then
        warning "test_size = $test_size"
else
        error "Error '$test_res' while trying to get size of 'unix/Makefile'"
fi
```

Ein als Zeichenketten-Konstante übergebener Dateiname unterliegt der Variablenersetzung (Seite 322).

Durchsuchen von Dateien: `fgrep`

Wenn Sie in einer Datei per „grep“[4] suchen wollen, steht Ihnen das `fgrep`-Kommando zur Verfügung. Die Syntax lautet:

```
fgrep (<Dateiname>, <RegEx>)
```

[4] „grep“ ist ein auf Unix-Betriebsystemen verbreitetes Kommando zum Filtern von Textströmen.

Wenn die Datei <Dateiname> nicht existiert wird mkfli4l mit einem fatalen Fehler beendet! Wenn Sie also nicht sicher sind, ob die Datei immer vorhanden ist, testen Sie die Existenz von <Dateiname> vorher mit stat ab. Nach dem Aufruf von fgrep steht Ihnen das Suchresultat in dem Array FGREP_MATCH_% zur Verfügung, wobei der Index x wie üblich von eins bis FGREP_MATCH_N reicht. FGREP_MATCH_1 verweist dabei auf den gesamten Bereich der Zeile, auf den der reguläre Ausdruck gepasst hat, während FGREP_MATCH_2 bis FGREP_MATCH_N den jeweils *n-1*-ten geklammerten Teil beinhalten.

Ein erstes einfaches Beispiel soll die Verwendung demonstrieren. Die Datei opt/etc/shells enthält die Zeile:

```
/bin/sh
```

Der folgende Code

```
fgrep("opt/etc/shells", "^/(.)(.*)/")
foreach v in FGREP_MATCH_%
do
  warning "%v='$v'"
done
```

produziert die folgende Ausgabe:

```
Warning: FGREP_MATCH_1='/bin/'
Warning: FGREP_MATCH_2='b'
Warning: FGREP_MATCH_3='in'
```

Der reguläre Ausdruck hat (nur) auf „/bin/“ gepasst, deshalb steht auch (nur) dieser Teil der Zeile in der Variable FGREP_MATCH_1. Der erste geklammerte Teil im Ausdruck passt auf das erste Zeichen hinter dem ersten „/“, deshalb steht auch nur „b“ in FGREP_MATCH_2. Der zweite geklammerte Teil umfasst den Rest hinter den „b“ bis zum letzten „/“, somit steht „in“ in der Variable FGREP_MATCH_3.

Das folgende zweite Beispiel demonstriert eine praxisnahe Verwendung von fgrep an einem Beispiel aus der check/base.ext. Hier wird getestet, ob alle in der PF_FORWARD_x angegebenen tmpl:-Referenzen vorhanden sind:

```
foreach n in pf_forward_n
do
  set rule=pf_forward_%[n]
  if (rule =~ "tmpl:([^[:space:]]+)")
  then
    foreach m in match_%
    do
      stat("$config_dir/etc/fwrules.tmpl/$m", tmplfile)
      if(tmplfile_res == "OK")
      then
        add_to_opt "etc/fwrules.tmpl/$m"
      else
        stat("opt/etc/fwrules.tmpl/$m", tmplfile)
        if(tmplfile_res == "OK")
        then
          add_to_opt "etc/fwrules.tmpl/$m"
```

```
        else
          fgrep("opt/etc/fwrules.tmpl/templates", "^$m[[:space:]]+")
          if (fgrep_match_n == 0)
          then
            error "Can't find tmpl:$m for PF_FORWARD_${n}='$rule'!"
          fi
        fi
      fi
    done
  fi
done
```

Sowohl ein als Zeichenketten-Konstante übergebener Dateiname als auch als Zeichenketten-Konstante übergebener regulärer Ausdruck unterliegen der Variablenersetzung (Seite 322).

Auseinandernehmen von Parametern: `split`

Oftmals werden Variablen mit mehreren Parametern belegt, die dann in Startup-Skripten erst wieder auseinandergenommen werden. Will man diese bereits vorher auseinandernehmen und Tests auf ihnen durchführen, nimmt man `split`. Die Syntax lautet:

```
split (<Zeichenkette>, <Array>, <Trennzeichen>)
```

Die Zeichenkette kann durch eine Variable oder direkt als Konstante angegeben werden. `mkfli4l` zerlegt ihn an den Stellen, an denen das Trennzeichen auftaucht, und erzeugt pro Teil ein Element des Arrays. Über diese Elemente kann man dann hinterher iterieren und Prüfungen vornehmen. Steht zwischen zwei Trennzeichen nichts, wird ein Array-Element mit einer leeren Zeichenkette als Wert erzeugt. Ausnahme ist „ “: Hier werden alle Leerzeichen konsumiert und keine leeren Variablen erzeugt.

Sollen die bei der Zerlegung entstandenen Elemente in einem numerischen Kontext verwendet werden (z. B. als Indizes), muss das beim Aufruf von `split` spezifiert werden. Das geschieht durch das zusätzliche Attribut „numeric“. Der Aufruf sieht dann wie folgt aus:

```
split (<Zeichenkette>, <Array>, <Trennzeichen>, numeric)
```

Ein Beispiel:

```
set bar="1.2.3.4"
split (bar, tmp_%, '.', numeric)
foreach i in tmp_%
do
        warning "%i = $i"
done
```

Die produzierte Ausgabe ist:

```
Warning: TMP_1 = 1
Warning: TMP_2 = 2
Warning: TMP_3 = 3
Warning: TMP_4 = 4
```

Hinweis: Wenn die „numeric“-Variante verwendet wird, dann prüft `mkfli4l` zum Zeitpunkt der Zerlegung *nicht*, ob die Teil-Zeichenketten auch wirklich numerisch sind! Bei einer späteren Verwendung in einem numerischen Kontext (etwa beim Addieren) löst `mkfli4l` jedoch einen fatalen Fehler aus, wenn eine solche Variable doch nicht numerisch ist. Beispiel:

```
set bar="a.b.c.d"
split (bar, tmp_%, '.', numeric)
# Fehler: invalid number 'a'
set i=tmp_%[1]+1
```

Eine an `split` im ersten Parameter übergebene Zeichenketten-Konstante unterliegt der Variablenersetzung (Seite 322).

Hinzufügen von Dateien zum Archiv: `add_to_opt`

Mit der Funktion `add_to_opt` können zusätzliche Dateien ans Opt- oder RootFS-Archiv angehängt werden. Es können dabei *alle* Dateien unterhalb von `opt/` oder aus dem Konfigurationsverzeichnis ausgewählt werden. Eine Beschränkung nur auf die Dateien, die mit einem bestimmten Paket geliefert werden, gibt es nicht. Liegt eine Datei sowohl unter `opt/` als auch im Konfigurationsverzeichnis im gleichen Pfad, bevorzugt `add_to_opt` die Dateien aus dem Konfigurationsverzeichnis. Die Funktion `add_to_opt` wird in der Regel dann eingesetzt, wenn komplexe logische Regeln darüber entscheiden, ob und welche Dateien in das Archiv aufgenommen werden müssen.

Die Syntax sieht wie folgt aus:

```
add_to_opt <Datei> [<Flags>]
```

Die Flags sind optional. Es gelten die in Tabelle 8.2 aufgeführten Standard-Werte, falls keine Flags angegeben sind.

Es folgt ein Beispiel aus dem Paket „sshd“:

```
if (opt_sshd)
then
   foreach pkf in sshd_public_keyfile_%
   do
     stat("$config_dir/etc/ssh/$pkf", publickeyfile)
     if(publickeyfile_res == "OK")
     then
         add_to_opt "etc/ssh/$pkf" "mode=400 flags=utxt"
     else
         error "sshd: missing public keyfile %pkf=$pkf"
     fi
   done
fi
```

Mit `stat` (Seite 328) wird zunächst geprüft, ob die Datei im Konfigurationsverzeichnis existiert. Ist die Datei vorhanden, wird sie ans Archiv angehängt, andernfalls bricht `mkfli4l` mit einer entsprechenden Fehlermeldung ab.

Hinweis: Auch bei `add_to_opt` prüft (Seite 312) `mkfli4l` zuerst, ob die zu kopierende Datei im Konfigurationsverzeichnis zu finden ist.

Sowohl ein als Zeichenketten-Konstante übergebener Dateiname als auch als Zeichenketten-Konstante übergebene Flags unterliegen der Variablenersetzung (Seite 322).

Kontrollfluss

```
if (expr)
then
        statement
else
        statement
fi
```

Eine klassische Fallunterscheidung, wie man sie kennt. Ist die Bedingung wahr, wird der `then`-Teil ausgeführt, ist die Bedingung falsch, wird der `else`-Teil ausgeführt.

Will man Tests über Array-Variablen durchführen, muss man jede einzelne Variable testen. Dazu gibt es die `foreach`-Schleife in zwei Varianten.

1. Iterieren über Array-Variablen:

   ```
   foreach <Laufvariable> in <Array-Variable>
   do
           <Anweisung>
   done

   foreach <Laufvariable> in <Array-Variable-1> <Array-Variable-2> ...
   do
           <Anweisung>
   done
   ```

   Diese Schleife iteriert über alle angegebenen Array-Variablen, jeweils angefangen beim ersten Element bis hin zum letzten; die Anzahl der Elemente im Array wird dabei der dem Array zugeordneten N-Variable entnommen. Die Laufvariable nimmt dabei die jeweiligen Werte der Array-Variablen an. Zu beachten ist dabei, dass bei optionalen Array-Variablen, die in der Konfiguration nicht vorhanden sind, ein leeres Element generiert wird. Unter Umständen muss das im Skript berücksichtigt werden, was man z. B. wie folgt tun kann:

   ```
   foreach i in template_var_opt_%
   do
       if (i != "")
       then
           warning "%i is present (%i='$i')"
       else
           warning "%i is undefined (empty)"
       fi
   done
   ```

   Wie man auch am Beispiel erkennen kann, lässt sich der *Name* der jeweiligen Array-Variablen durch die `%<Laufvariable>`-Konstruktion ermitteln.

   Die Anweisung in der Schleife kann eine der oben beschriebenen Kontrollelemente oder Funktionen (`if`, `foreach`, `provides`, `depends`, ...) sein.

   Will man auf genau ein Element eines Arrays zugreifen, kann man dieses mittels der Syntax `<Array>[<Index>]` ansprechen. Der Index kann dabei eine normale Variable, eine Zahlenkonstante oder wiederum ein indiziertes Array sein.

2. Iterieren über N-Variablen:

```
foreach <Laufvariable> in <N-Variable>
do
        <Anweisung>
done
```

Diese Schleife läuft von 1 bis zum Wert, der in der N-Variable steht. Man kann die Laufvariable dazu benutzen, um Array-Variablen zu indizieren. Will man also nicht nur über eine Array-Variable iterieren, sondern über mehrere gleichzeitig, die alle durch *dieselben* N-Variable kontrolliert werden, nimmt man diese Variante der Schleife und verwendet die Laufvariable zum Indizieren mehrerer Array-Variablen. Beispiel:

```
foreach i in host_n
do
    set name=host_%_name[i]
    set ip4=host_%_ip4[i]
    warning "$i: name=$name ip4=$ip4"
done
```

Das ergibt bei entsprechend gefüllten `HOST_%_NAME`- und `HOST_%_IP4`-Arrays beispielsweise:

```
Warning: 1: name=berry ip4=192.168.11.226
Warning: 2: name=fence ip4=192.168.11.254
Warning: 3: name=sandbox ip4=192.168.12.254
```

Ausdrücke

Ausdrücke verknüpfen Werte und Operatoren zu einem neuen Wert. Ein Wert kann dabei eine gewöhnliche Variable, ein Array-Element oder eine Konstante (Zahl, Zeichenkette oder Versionsnummer) sein. Alle Zeichenketten-Konstanten, die in Ausdrücken auftreten, unterliegen der Variablenersetzung (Seite 322).

Operatoren erlauben so gut wie alles, was man von einer Programmiersprache gewöhnt ist. Ein Test auf die Gleichheit zweier Variablen könnte also so aussehen:

```
var1 == var2
"$var1" == "$var2"
```

Zu beachten ist dabei, dass der Vergleich in Abhängigkeit vom Typ der Variable erfolgt, der in `check/<PACKAGE>.txt` festgelegt wurde. Ist eine der beiden Variablen numerisch (Seite 322), erfolgt der Vergleich auf numerischer Basis, d. h. die Zeichenketten werden in Zahlen umgewandelt und dann verglichen. Sonst erfolgt der Vergleich auf Zeichenketten-Basis; ein Vergleich `"05" == "5"` ergibt „falsch“, ein Vergleich `"18" < "9"` ergibt „wahr“ auf Grund der lexikographischen Ordnung auf Zeichenketten: die Ziffer „1“ liegt vor der Ziffer „9“ im zugrunde liegenden ASCII-Zeichensatz.

Für den Vergleich von Versionen wird das Hilfskonstrukt `numeric(version)` eingeführt, welches den numerischen Wert für eine Versionsnummer für Vergleichszwecke bestimmt. Dabei gilt:

```
numeric(version) := major * 10000 + minor * 1000 + sub
```

wobei „major“ die erste Komponente der Versionsnummer darstellt, „minor“ die zweite und „sub“ die dritte; fehlt „sub“, entfällt der Term in der obigen Summe einfach (oder anders ausgedrückt, für „sub“ wird null angenommen).

Eine vollständige Auflistung aller Ausdrücke ist in Tabelle 8.3 zu finden. Dabei steht „val“ für einen beliebig getypten Wert, „number“ für einen numerischen Wert und „string“ für eine Zeichenkette.

Tabelle 8.3.: Logische Ausdrücke

Ausdruck	wahr wenn
id	id == „yes“
val == val	identisch getypte Werte sind gleich
val != val	identisch getypte Werte sind ungleich
val == number	numerischer Wert von val == number
val != number	numerischer Wert von val != number
val < number	numerischer Wert von val < number
val > number	numerischer Wert von val > number
val == version	numeric(val) == numeric(version)
val < version	numeric(val) < numeric(version)
val > version	numeric(val) > numeric(version)
val =~ string	regulärer Ausdruck in string auf val passt
(expr)	Ausdruck in Klammern ist wahr
expr && expr	beide Ausdrücke sind wahr
expr || expr	mind. einer der beiden Audrücke ist wahr
copy_pending(id)	siehe Beschreibung
samenet (string1, string2)	string1 das gleiche netz wie string2 beschreibt
subnet (string1, string2)	string1 ein Subnetz von string2 beschreibt

Match-Operator

Mit dem Match-Operator `=~` kann man prüfen, ob ein regulärer Ausdruck auf den Wert einer Variable passt. Weiterhin kann man den Operator auch nutzen, um Teilausdrücke aus einer Variablen zu extrahieren. Nach erfolgreichem Anwenden eines regulären Ausdrucks auf eine Variable enthält das Array `MATCH_%` die gefundenen Teile. Das könnte z. B. wie folgt aussehen:

```
set foo="foobar12"
if ( foo =~ "(foo)(bar)([0-9]*)" )
then
        foreach i in match_%
        do
                warning "match %i: $i"
        done
fi
```

Ein `mkfli4l`-Aufruf führt dann zu folgender Ausgabe:

```
Warning: match MATCH_1: foo
Warning: match MATCH_2: bar
Warning: match MATCH_3: 12
```

Bei Verwendung von `=~` kann Bezug auf alle existierenden regulären Ausdrücke genommen werden. Will man z. B. prüfen, ob ein PCMCIA-Ethernet-Treiber ausgewählt wurde, ohne dass `OPT_PCMCIA` auf „yes“ gesetzt wurde, könnte das wie folgt aussehen:

```
if (!opt_pcmcia)
then
    foreach i in net_drv_%
    do
       if (i =~ "^(RE:PCMCIA_NET_DRV)$")
       then
           error "If you want to use ..."
       fi
    done
fi
```

Wie in dem Beispiel demonstriert wird, ist es wichtig, den regulären Ausdruck mit Hilfe von `^` und `$` zu *verankern*, wenn man den Ausdruck auf die *gesamte* Variable anwenden will. Ansonsten liefert der Match-Ausdruck schon „wahr“, wenn nur ein *Teil* der Variable vom regulären Ausdruck abgedeckt wird, was in diesem Fall sicherlich nicht erwünscht ist.

Prüfen, ob in Abhängigkeit vom Wert einer Variable eine Datei kopiert wurde: `copy_pending`

Mit den im Check-Prozess gewonnenen Informationen prüft die Funktion `copy_pending`, ob in Abhängigkeit vom Wert einer Variable eine Datei kopiert wurde oder nicht. Das kann man verwenden, um z. B. zu testen, ob der vom Nutzer angegebene Treiber auch wirklich existiert und kopiert wurde. `copy_pending` akzeptiert den zu prüfenden Namen in Form einer Variablen oder einer Zeichenkette.[5] `copy_pending` prüft dazu, ob

- die Variable aktiv ist (wenn sie von einem OPT abhängt, muss dieses auf „yes“ gesetzt sein),
- die Variable in einer `opt/<PACKAGE>.txt`-Datei referenziert wurde, und
- ob in Abhängigkeit vom aktuellen Wert eine Datei kopiert wurde.

Dabei liefert `copy_pending` „wahr“ zurück, wenn im letzten Schritt festgestellt wurde, dass *keine* Datei kopiert wurde, das Kopieren also somit noch aussteht (also „pending“ ist).

Ein kleines Beispiel für die Anwendung all dieser Funktionen findet man in `check/base.ext`:

```
foreach i in net_drv_%
do
    if (copy_pending("%i"))
    then
        error "No network driver found for %i='$i', check config/base.txt"
    fi
done
```

[5] Wie eingangs beschrieben unterliegt die Zeichenkette der Variablenersetzung, so dass man z. B. mittels einer `foreach`-Schleife (Seite 332) und der `%<Name>`-Ersetzung (Seite 322) alle Elemente eines Arrays prüfen kann.

Hier werden alle Elemente des Arrays `NET_DRV_%` angemeckert, für die keine Kopieraktion vorgenommen wurde, für die also in der `opt/base.txt` kein entsprechender Eintrag existiert.

Vergleich von Netzwerkadressen: `samenet` und `subnet`

Zum Prüfen von Routen benötigt man ab und zu einen Test, ob zwei Netzwerke identisch sind oder eines ein Subnetz eines anderen ist. Dazu gibt es die beiden Funktionen `samenet` und `subnet`. Dabei liefert

```
samenet (netz1, netz2)
```

„wahr“, wenn beide Netze identisch sind, und

```
subnet (netz1, netz2)
```

gibt „wahr“ zurück, wenn „netz1“ ein Subnetz von „netz2“ ist.

Erweitern der Kernel-Kommandozeile

Ist ein OPT gezwungen, dem Kernel andere Boot-Parameter zu übergeben, so musste früher die Variable `KERNEL_BOOT_OPTION` geprüft werden, ob der nötige Parameter enthalten war, und ggf. eine Warnung oder eine Fehlermeldung ausgegeben werden. Mit der internen Variable `KERNEL_BOOT_OPTION_EXT` kann man nötige, aber fehlende Optionen direkt im ext-Skript ergänzen. Ein Beispiel aus der `check/base.ext`:

```
if (powermanagement =~ "apm.*|none")
then
    if ( ! kernel_boot_option =~ "acpi=off")
    then
        set kernel_boot_option_ext="${kernel_boot_option_ext} acpi=off"
    fi
fi
```

Damit wird „acpi=off“ an den Kernel übergeben, falls keine Energieverwaltung oder welche vom Typ „APM“ gewünscht ist.

8.3.8. Unterstützung verschiedener Kernelversionslinien

Verschiedene Kernelversionslinien unterscheiden sich häufig in einigen Details:

- es stehen andere Treiber zur Verfügung, einige sind weggefallen, andere hinzugekommen
- die Module heißen teilweise einfach anders
- die Modul-Abhängigkeiten sehen anders aus
- die Module liegen woanders

Diese Unterschiede werden zum großen Teil durch `mkfli4l` automatisch behandelt. Um die zur Verfügung stehenden Module zu beschreiben, kann man zum einen die zur Prüfung verwendeten Prüfungen in Abhängigkit von der Version erweitern (bedingte reguläre Ausdrücke (Seite 318)), und zum anderen erlaubt `mkfli4l` *versionsabhängige* `opt/<PACKAGE>.txt`-Dateien. Dies heißen dann `opt/<PACKAGE>_<Kernel-Version>.txt`, wobei die Komponenten der Kernel-Version durch Unterstriche voneinander getrennt werden. Ein Beispiel: Das Paket „base“ enthält die folgenden Dateien im `opt`-Verzeichnis:

- `base.txt`
- `base_3_14.txt`
- `base_3_17.txt`

Die erste Datei (`base.txt`) wird *immer* verarbeitet. Die anderen beiden Dateien werden nur verarbeitet, wenn die Kernelversion „3.14(.*)“ bzw. „3.17(.*)“ lautet. Wie man sieht, können Versionskomponenten im Dateinamen weggelassen werden, wenn man eine ganze Gruppe von Kerneln „erschlagen“ möchte. Unter Annahme von `KERNEL_VERSION='3.14.26'` werden für ein Paket `<PACKAGE>` die folgenden Dateien (sofern vorhanden) eingelesen und verarbeitet:

- `<PACKAGE>.txt`
- `<PACKAGE>_3.txt`
- `<PACKAGE>_3_14.txt`
- `<PACKAGE>_3_14_26.txt`

8.3.9. Dokumentation

Die Dokumentation wird in den Dateien

- `doc/<SPRACHE>/opt/<PACKAGE>.txt`
- `doc/<SPRACHE>/opt/<PACKAGE>.html`

abgelegt. Die HTML-Dateien können auch aufgeteilt werden, d. h. für jedes enthaltene OPT eine. Dann muss trotzdem eine `<PACKAGE>.html` angelegt werden, die auf die anderen Dateien verweist. Änderungen sollten in folgenden Dateien dokumentiert werden:

- `changes/<PACKAGE>.txt`

Die gesamte Text-Dokumentation darf keine Tabulatoren enthalten und muss nach spätestens 79 Zeichen einen harten Zeilenumbruch haben. Die stellt sicher, dass die Dokumentation auch mit einem Editor ohne automatischen Zeilenumbruch richtig gelesen werden kann.

Wer mag kann auch eine Dokumentation im LaTeX-Format erstellen und daraus dann HTML- und PDF-Fassungen erzeugen. Als Beispiel kann die Dokumentation von fli4l dienen. Einen Rahmen für die Dokumentation und die minimal benötigten LaTeX-Macros kann man im Paket „template“ finden. Eine kurze Beschreibung ist in den folgenden Unterabschnitten zu finden.

Die fli4l-Dokumentation steht zur Zeit in den folgenden Sprachen zur Verfügung: deutsch, englisch (`<SPRACHE>` = „english“) und französisch (`<SPRACHE>` = „french“). Es steht einem Paket-Entwickler jedoch frei, sein Paket in beliebigen Sprachen zu dokumentieren. Im Sinne der Verständlichkeit wird jedoch empfohlen, eine Dokumentation in deutsch und/oder englisch (idealerweise in beiden Sprachen) anzufertigen.

Voraussetzungen für die Erstellung einer LaTeX-Dokumentation

Zum Erstellen der Dokumentation aus LaTeX-Quellen gibt es folgende Anforderungen an die Umgebung:

- Linux/OS X-Umgebung: Zur einfachen Erzeugung gibt es ein Makefile, mit dem alle weiteren Aufrufe automatisiert sind (Cygwin müsste auch funktionieren, wird aber nicht vom fli4l-Team getestet)
- LaTeX2HTML für die HTML-Version
- natürlich LaTeX (empfohlen wird „TeX Live“ für Linux/OS X und „MiKTeX“ für Microsoft Windows) mit dem „pdftex“-Programm und folgenden TeX-Paketen:
  - aktuelles KOMA-Skript (mindestens Version 2)
  - alle notwendigen Pakete für pdftex
  - ausgepacktes Dokumentationspaket für fli4l, welches die benötigten Makefiles und TeX-Stile bereitstellt

Dateinamen

Die Dateien der Dokumentation werden nach folgendem Schema benannt:

`<PACKAGE>_main.tex`: Diese Datei enthält den Hauptteil der Dokumentation. `<PACKAGE>` steht hier für den Namen des Pakets, das beschrieben werden soll (in Kleinbuchstaben).

`<PACKAGE>_appendix.tex`: Sollen zu diesem Paket noch weitere Anmerkungen im Anhang hinzugefügt werden, so werden diese hier abgelegt.

Diese Dateien werden im Verzeichnis `fli4l/<PACKAGE>/doc/<SPRACHE>/tex/<PACKAGE>` abgelegt. Für das Paket „sshd“ sieht das z. B. wie folgt aus:

```
$ ls fli4l/doc/deutsch/tex/sshd/
Makefile sshd_appendix.tex  sshd_main.tex  sshd.tex
```

Das Makefile ist für die Generierung der Dokumentation verantwortlich, die `sshd.tex`-Datei stellt einen Rahmen für die eigentliche Dokumentation und den Anhang bereit, der sich in den anderen beiden Dateien befindet. Ansehen kann man sich das am Beispiel der Dokumentation des „template“-Pakets.

LaTeX-Grundlagen

LaTeX arbeitet ähnlich wie HTML „Tag-orientiert“, nur dass die Tags hier „Kommandos“ heißen und folgendes Format aufweisen: `\kommando` bzw. `\begin{umgebung}` ... `\end{umgebung}`

Nach Möglichkeit sollte man mit Hilfe von Kommandos eher die *Bedeutung* des jeweiligen Textes auszeichnen und weniger dessen *Darstellung*. Es ist also vorteilhaft, z. B.

```
\warning{Bitte␣nicht␣...␣tun}
```

statt

```
\emph{Bitte␣nicht␣...␣tun}
```

zu verwenden.

Jedes Kommando bzw. jede Umgebung kann noch weitere Parameter aufnehmen, die mit `\kommando{parameter1}{paramter2}{paramterN}` geschrieben werden.

Manche Kommandos haben optionale Parameter, die in eckigen (statt geschweiften) Klammern stehen: `\kommando[optionalerParameter]{parameter1}` ... Dabei kommt im Normalfall nur ein optionaler Parameter vor, in seltenen Fällen aber auch mehrere.

Einzelne Absätze werden im Dokument durch Leerzeilen getrennt. Innerhalb dieser Absätze nimmt LaTeX selbst den Zeilenumbruch und die Worttrennung vor.

Folgende Buchstaben haben eine spezielle Bedeutung in LaTeX und müssen, sollten sie in normalem Text vorkommen, mit einem vorangestellten `\` maskiert werden: # $ & _ % { }. „~“ und „^“ müssen wie folgt geschrieben werden: `\verb?~? \verb?^?`

Die wichtigsten LaTeX-Kommandos werden in der Dokumentation des „template“-Pakets verwendet und erklärt.

8.3.10. Dateiformate

Alle Textdateien (sowohl Dokumentation als auch Skripte, die später auf dem Router liegen) müssen im DOS-Dateiformat, also mit CR/LF statt nur LF am Zeilenende in das Paket gelegt werden. Dadurch wird erreicht, dass Windows-Nutzer die Dokumentation auch mit „notepad“ lesen können und durch eine Änderung eines Skripts unter Windows das Ganze später auf dem Router trotzdem lauffähig bleibt. Die Skripte werden beim Bauen der Archive in das auf dem Router benötigte Format konvertiert (siehe die Beschreibung der Flags in Tabelle 8.2).

8.3.11. Entwickler-Dokumentation

Sollte ein Programm aus dem Paket eine neue Schnittstelle definieren, die andere Programme nutzen können, so ist die Dokumentation dieser Schnittstelle in einer separaten Dokumentation unter `doc/dev/<PACKAGE>.txt` abzulegen.

8.3.12. Client-Programme

Sollte ein Paket zusätzliche Client-Programme mitliefern, so sind diese im Verzeichnis `windows/` für Windows-Clients und im Verzeichnis `unix/` für Unix- und Linux-Clients abzulegen.

8.3.13. Quellcode

Angepasste Programme und Quellcodes können im Verzeichnis `src/<PACKAGE>/` beigelegt werden. Sollen die Programme genauso wie die restlichen fli4l-Programme gebaut werden, bitte einen Blick in die Dokumentation des „src“-Pakets (Seite 259) werfen.

8.3.14. Weitere Dateien

Alle Dateien, die nachher auf dem Router liegen, werden unter `opt/etc/` und `opt/files/` abgelegt. Dabei liegen unter:

- `opt/etc/boot.d/` und `opt/etc/rc.d/` Skripte, die beim Starten des Systems ausgeführt werden sollen
- `opt/etc/rc0.d/` Skripte, die beim Herunterfahren des Systems ausgeführt werden

- `opt/etc/ppp/` Skripte, die beim Einwählen und Auflegen ausgeführt werden
- `opt/files/` die ausführbaren Programme und sonstige Dateien entsprechend ihrer Positionen im Dateisystem (d. h. die Datei `opt/files/bin/busybox` wird später auf dem Router im Verzeichnis `/bin` liegen)

Die Skripte in `opt/etc/boot.d/`, `opt/etc/rc.d/` und `opt/etc/rc0.d/` werden wie folgt benannt:

```
rc<nummer>.<name>
```

Die Nummer entscheidet über die Reihenfolge der Ausführung, der Name gibt einen Hinweis darauf, welches Programm/Paket von diesem Skript behandelt wird.

8.4. Allgemeine Skript-Erstellung auf fli4l

Hier folgt jetzt *keine* allgemeine Einführung in Shell-Skripte, das kann jeder im Internet selber nachlesen, es wird nur auf die spezielle Gegebenheiten bei fli4l eingegangen. Informationen dazu gibt es in den diversen Unix-/Linux-Hilfeseiten. Folgende Links können als Einstiegspunkte zu diesem Thema dienen:

- Einführung in Shell-Skripte:
  - http://cip.physik.uni-freiburg.de/main/howtos/sh.php
- Hilfeseiten online:
  - http://linux.die.net/
  - http://heapsort.de/man2web
  - http://man.he.net/
  - http://www.linuxcommand.org/superman_pages.php

8.4.1. Aufbau

In der Unix-Welt ist es nötig, ein Skript mit dem Namen des Interpreters zu beginnen, daher steht in der ersten Zeile:

```
#!/bin/sh
```

Damit man später leichter erkennen kann, was ein Skript macht und wer es geschrieben hat, sollte jetzt ein kurzer Header folgen, in etwa so:

```
#-----------------------------------------------------------------
# /etc/rc.d/rc500.dummy - start my cool dummy server
#
# Creation:     19.07.2001  Toller Hecht <toller-hecht@example.net>
# Last Update:  11.11.2001  Süße Maus <suesse-maus@example.net>
#-----------------------------------------------------------------
```

Nun kann das eigentliche Skript folgen...

8.4.2. Umgang mit Konfigurationsvariablen

Pakete werden über die Datei `config/<PACKAGE>.txt` konfiguriert. Die darin enthaltenen und aktiven Variablen (Seite 313) werden beim Erzeugen des Boot-Mediums in die Datei `rc.cfg` übernommen. Beim Booten des Routers wird diese Datei eingelesen, bevor irgend ein rc-Skript (Skripte unter `/etc/rc.d/`) gestartet wird. Diese Skripte können dadurch auf alle Konfigurationsvariablen einfach durch `$<Variablenname>` zugreifen.

Benötigt man Werte von Konfigurationsvariablen auch noch nach dem Booten, dann kann man sie aus der `/etc/rc.cfg` extrahieren, in welche während des Bootens die Konfiguration des Boot-Mediums geschrieben wurde. Möchte man beispielsweise den Wert der Variable `OPT_DNS` in einem Skript auslesen, so kann man dies folgendermaßen tun:

```
eval $(grep "^OPT_DNS=" /etc/rc.cfg)
```

Das funktioniert auch mit mehreren Variablen effizient (d. h. mit nur einem Aufruf des `grep`-Programms):

```
eval $(grep "^\(HOSTNAME\|DOMAIN_NAME\|OPT_DNS\|DNS_LISTEN_N\)=" /etc/rc.cfg)
```

8.4.3. Persistente Speicherung von Daten

Gelegentlich benötigt ein Paket die Möglichkeit, Daten persistent abzulegen, die also einen Neustart des Routers überleben. Dazu existiert die Funktion `map2persistent`, die von einem Skript in `/etc/rc.d/` aufgerufen werden kann. Sie erwartet eine Variable, die einen Pfad enthält, und ein Unterverzeichnis. Die Idee ist, dass die Variable entweder einen tatsächlichen Pfad beschreibt – dann wird dieser Pfad auch genommen, denn der Nutzer hat ihn so gewünscht, oder die Zeichenkette „auto“ – dann wird unterhalb eines Verzeichnisses auf einem persistenten Medium ein entsprechendes Unterverzeichnis gemäß dem zweiten Parameter erzeugt. Die Funktion liefert das Resultat in eben der Variable zurück, deren Name im ersten Parameter übergeben wurde.

Ein Beispiel soll dies verdeutlichen. Sei `VBOX_SPOOLPATH` eine Variable, die einen Pfad oder die Zeichenkette „auto“ enthält. Dann führt der Aufruf

```
begin_script VBOX "Configuring vbox ..."
[...]
map2persistent VBOX_SPOOLPATH /spool
[...]
end_script
```

dazu, dass die Variable `VBOX_SPOOLPATH` entweder gar nicht verändert wird (falls sie einen Pfad enthält), oder dass sie durch den Pfad `/var/lib/persistent/vbox/spool` ersetzt wird (falls sie die Zeichenkette „auto“ enthält). Dabei verweist[6] `/var/lib/persistent` auf ein Verzeichnis auf einem beschreibbaren und nicht flüchtigen Speichermedium, und `<SCRIPT>` stellt das aufrufende Skript in Kleinbuchstaben dar (dieser Name wird vom ersten Argument des `begin_script`-Aufrufs (Seite 342) abgeleitet). Falls kein geeignetes Medium existieren sollte (was durchaus sein kann), ist `/var/lib/persistent` ein Verzeichnis in der RAM-Disk.

[6] mit Hilfe eines so genannten „bind“-Mounts

Zu beachten ist, dass der Pfad, der von `map2persistent` zurückgegeben wird, *nicht* automatisch erzeugt wird – das muss der Aufrufer selbst erledigen (etwa durch einen Aufruf von `mkdir -p <Pfad>`).

In der Datei `/var/run/persistent.conf` kann nachgeschaut werden, ob die persistente Speicherung von Daten möglich ist. Beispiel:

```
. /var/run/persistent.conf
case $SAVETYPE in
persistent)
    echo "Persistente Speicherung möglich!"
    ;;
transient)
    echo "Persistente Speicherung NICHT möglich!"
    ;;
esac
```

8.4.4. Fehlersuche

Bei Start-Skripten ist es oft sinnvoll, diese bei Bedarf im Debug-Modus der Shell laufen zu lassen, um festzustellen, wo „der Wurm drin ist“. Dazu wird am Anfang und am Ende folgendes eingefügt:

```
begin_script <OPT-Name> "start message"
<script code>
end_script
```

Im normalen Betrieb erscheint jetzt beim Start des Skriptes der angegebene Text und am Ende der gleiche Text mit einem vorangestellten „finished“.

Will man die Skripte debuggen, muss man zwei Dinge tun:

1. Man muss `DEBUG_STARTUP` (Seite 31) auf „yes“ setzen.
2. Man muss das Debuggen für dieses OPT aktivieren. Das tut man in der Regel durch den Eintrag

   ```
   <OPT-Name>_DO_DEBUG='yes'
   ```

   in der Konfigurationsdatei.[7] Jetzt wird während der Ausführung am Bildschirm genau dargestellt, was passiert.

Weitere beim Debuggen hilfreiche Variablen

DEBUG_ENABLE_CORE Diese Variable gestattet das Erzeugen von „Core-Dumps“ (Speicherauszügen). Stürzt ein Programm aufgrund eines Fehlers ab, wird ein Abbild des aktuellen Zustandes im Dateisystem abgelegt, der hinterher zur Analyse des Problems verwendet werden kann. Die Core-Dumps werden unter `/var/log/dumps/` abgelegt.

DEBUG_IP Wird diese Variable gesetzt, werden alle Aufrufe des Programms `ip` protokolliert.

[7] Manchmal werden mehrere Start-Skripte verwendet, die dann auch verschiedene Namen für ihre Debug-Variablen haben. Hier hilft ein kurzer Blick in die Skripte.

DEBUG_IPUP Wird diese Variable auf „yes“ gesetzt, werden während der Ausführung der `ip-up`/`ip-down`-Skripte die ausgeführten Anweisungen mitgezeichnet und im System-Protokoll gespeichert.

LOG_BOOT_SEQ Wird diese Variable auf „yes“ gesetzt, protokolliert der `bootlogd` während des Bootens alle auf der Konsole getätigten Ausgaben in der Datei `/var/tmp/boot.log`. Diese Variable hat standardmäßig den Wert „yes“.

DEBUG_KEEP_BOOTLOGD Normalerweise wird der `bootlogd` am Ende des Bootvorganges beendet. Ein Setzen dieser Variable unterbindet das und erlaubt ein Protokollieren der Konsolenausgaben über den Bootvorgang hinaus.

DEBUG_MDEV Ein Setzen dieser Variable generiert ein Protokoll des `mdev`-Daemons, der für das Anlegen der Geräte-Dateien unter `/dev` zuständig ist.

8.4.5. Hinweise

- Es ist *immer* besser, geschweifte Klammern „{...}“ an Stelle von runden Klammern „(...)“ zu benutzen. Allerdings muss dabei darauf geachtet werden, dass nach der öffnenden Klammer ein Leerzeichen oder eine neue Zeile vor dem nächsten Befehl kommt und vor der schließenden Klammer ein Semikolon oder auch eine neue Zeile kommt. Beispielsweise ist

  ```
  { echo "cpu"; echo "quit"; } | ...
  ```

  gleichbedeutend mit:

  ```
  {
          echo "cpu"
          echo "quit"
  } | ...
  ```

- Ein Skript kann mit „exit“ vorzeitig beendet werden. Dies ist aber bei den Start-Skripten (`opt/etc/boot.d/...`, `opt/etc/rc.d/...`), den Stopp-Skripten (`opt/etc/rc0.d/...`) und den `ip-up`/`ip-down`-Skripten (`opt/etc/ppp/...`) geradezu tödlich, da auch nachfolgende Skripte nicht mehr ausgeführt werden. Im Zweifelsfall immer weglassen.

- KISS – Keep it small and simple. Du willst Perl als Skript-Sprache verwenden? Dir reichen die Skripting-Möglichkeiten von fli4l nicht? Überdenke deine Einstellung! Ist dein OPT wirklich nötig? fli4l ist immer noch „nur“ ein Router, ein Router sollte eigentlich keine Serverdienste anbieten.

- Die Fehlermeldung „: not found“ heißt meistens, dass das Skript noch im DOS-Format vorliegt. Weitere Fehlerquelle: Das Skript ist nicht ausführbar. In beiden Fällen sollte die `opt/<PACKAGE>.txt`-Datei daraufhin geprüft werden, ob sie die korrekten Optionen (in Bezug auf „mode“, „gid“, „uid“ und Flags) enthält. Wenn das Skript erst bem Booten erzeugt wird, sollte ein „chmod +x <Skriptname>“ ausgeführt werden.

- Für temporäre Dateien sollte der Pfad `/tmp` genutzt werden. Es ist aber unbedingt darauf zu achten, dass hier nur wenig Platz ist, weil dies in der RootFS-RAM-Disk liegt! Wenn mehr Platz benötigt wird, muss man sich eine eigene RAM-Disk erstellen und mounten. Details dazu verrät der Abschnitt „RAM-Disks“ in dieser Dokumentation.
- Damit temporäre Dateien eindeutige Namen erhalten, sollte man grundsätzlich die aktuelle Prozess-ID, die in der Shell-Variable „$“ gespeichert ist, an den Dateiname anhängen. `/tmp/<OPT-Name>.$$` stellt somit einen guten Dateinamen dar, `/tmp/<OPT-Name>` eher weniger, wobei `<OPT-Name>` natürlich nicht so stehen bleiben soll, sondern geeignet ersetzt werden muss.

8.5. Arbeit mit dem Paketfilter

8.5.1. Hinzufügen von eigenen Ketten und Regeln

Zur Manipulation des Paketfilters steht eine Reihe von Routinen zur Verfügung, mit deren Hilfe man Ketten (engl. „Chains“) und Regeln hinzufügen und wieder löschen kann. Eine Kette ist eine benannte und geordnete Liste von Regeln. Es gibt einen Satz vordefinierter Ketten (`PREROUTING`, `INPUT`, `FORWARD`, `OUTPUT`, `POSTROUTING`); mit Hilfe dieser Funktionen können weitere Ketten nach Bedarf erstellt werden.

`add_chain/add_nat_chain <chain>:` Fügt eine Kette zur „filter“- oder „nat“-Tabelle hinzu.

`flush_chain/flush_nat_chain <chain>:` Entfernt alle Regeln aus einer Kette der „filter“- oder „nat“-Tabelle.

`del_chain/del_nat_chain <chain>:` Entfernt eine Kette aus der „filter“- oder „nat“-Tabelle. Ketten müssen leer sein, bevor sie gelöscht werden können, und es darf auch keine Referenz mehr auf sie geben. Eine solche Referenz ist z. B. eine `JUMP`-Aktion, deren Ziel eben diese Kette ist.

`add_rule/ins_rule/del_rule:` Fügt Regeln am Ende einer Kette (`add_rule`) bzw. an beliebiger Stelle in einer Kette (`ins_rule`) ein bzw. löscht Regeln aus einer Kette (`del_rule`). Ein Aufruf sieht wie folgt aus:

```
add_rule <table> <chain> <rule> <comment>
ins_rule <table> <chain> <rule> <position> <comment>
del_rule <table> <chain> <rule> <comment>
```

wobei die Parameter folgende Bedeutung haben:

table Die Tabelle, in der sich die Kette befindet

chain Die Kette, in welche die Regel eingefügt werden soll

rule Die Regel, die eingefügt werden soll; das Format entspricht dem in der Konfigurationsdatei verwendeten

position Die Position, an der die Regel eingefügt werden soll (nur bei `ins_rule`)

comment Ein Kommentar, der zusammen mit der Regel angezeigt werden soll, wenn sich jemand den Paketfilter ansieht.

8.5.2. Einordnen in bestehende Regeln

fli4l konfiguriert den Paketfilter mit einem gewissen Standardregelsatz. Will man eigene Regeln einfügen, wird man diese in der Regel nach dem Standardregelsatz einfügen wollen. Ebenfalls wird man wissen wollen, was denn die vom Nutzer gewünschte Aktion beim Verwerfen eines Paketes ist. Diese Informationen bekommt man für die `FORWARD`- und `INPUT`-Ketten durch Aufruf zweier Funktionen, `get_defaults` und `get_count`. Nach Aufruf von

```
get_defaults <chain>
```

erhält man die folgenden Ergebnisse:

`drop`: Diese Variable enthält die Kette, in die verzweigt wird, wenn ein Paket verworfen wird.

`reject`: Diese Variable enthält die Kette, in die verzweigt wird, wenn ein Paket abgelehnt wird.

Nach Aufruf von

```
get_count <chain>
```

erhält man in der Variable `res` die Anzahl der Regeln in der Kette `<chain>`. Diese Position ist insofern wichtig, als man *nicht* einfach `add_rule` verwenden kann, um eine Regel am Ende der vordefinierten „filter“-Ketten `INPUT`, `FORWARD` und `OUTPUT` einzufügen. Dies liegt daran, dass diese Ketten mit einer Standardregel abgeschlossen werden, welche alle verbliebenen Pakete behandelt, je nach Belegung der `PF_<Kette>_POLICY`-Variablen. Ein Einfügen *hinter* dieser letzten Regel hat also keine Auswirkungen. Die Funktion `get_count` erlaubt es nun hingegen, die Stelle direkt *vor* dieser letzten Regel zu ermitteln und die Position dann an die `ins_rule`-Funktion im Parameter `<position>` zu übergeben, um die Regel wie gewünscht am Ende der jeweiligen Kette, aber vor der letzten Auffang-Regel einzubauen.

Ein Beispiel aus dem Skript `opt/etc/rc.d/rc390.dns_dhcp` des Pakets „dns_dhcp“ soll dies verdeutlichen:

```
case $OPT_DHCPRELAY in
    yes)
        begin_script DHCRELAY "starting dhcprelay ..."

        idx=1
        interfaces=""
        while [ $idx -le $DHCPRELAY_IF_N ]
        do
            eval iface='$DHCPRELAY_IF_'$idx

            get_count INPUT
            ins_rule filter INPUT "prot:udp  if:$iface:any 68 67 ACCEPT" \
                $res "dhcprelay access"

            interfaces=$interfaces' -i '$iface
            idx=`expr $idx + 1`
        done
        dhcrelay $interfaces $DHCPRELAY_SERVER

        end_script
    ;;
esac
```

Hier sieht man inmitten der Schleife einen Aufruf von `get_count`, gefolgt von einem Aufruf der `ins_rule`-Funktion, der unter anderem die `res`-Variable als `position`-Parameter übergeben wird.

8.5.3. Erweiterung der Paketfilter-Tests

fli4l verwendet in den Paketfilterregeln die Syntax `match:params`, um zusätzliche Bedingungen an die Pakete zu stellen (siehe `mac:`, `limit:`, `length:`, `prot:`, ...). Will man zusätzliche Tests hinzufügen, wird das folgendermaßen gemacht:

1. Anlegen einer Datei `opt/etc/rc.d/fwrules-<name>.ext`. In dieser Datei steht in etwa folgendes:

```
# extension is available
foo_p=yes

# the actual extension, adding matches to match_opt
do_foo()
{
    param=$1
    get_negation $param
    match_opt="$match_opt -m foo $neg_opt --fooval $param"
}
```

2. Testen der Erweiterung:

```
$ cd opt/etc/rc.d
$ sh test-rules.sh 'foo:bar ACCEPT'
add_rule filter FORWARD 'foo:bar ACCEPT'
iptables -t filter -A FORWARD -m foo --fooval bar -s 0.0.0.0/0 \
    -d 0.0.0.0/0 -m comment --comment foo:bar ACCEPT -j ACCEPT
```

3. Aufnahme der Erweiterung und aller noch benötigten Dateien (`iptables`-Komponenten) ins Archiv über einen der bekannten Mechanismen.

4. Zulassen der Erweiterung in der Konfiguration durch Erweitern von `FW_GENERIC_MATCH` in einer exp-Datei, z. B. wie folgt:

```
+FW_GENERIC_MATCH(OPT_FOO) = 'foo:bar' : ''
```

8.6. CGI-Erstellung für das *httpd*-Paket

8.6.1. Allgemeines zum Webserver

Der Webserver, der bei fli4l verwendet wird, ist der `mini_httpd` von ACME Labs. Die Quellen können unter http://www.acme.com/software/mini_httpd/ heruntergeladen werden. Allerdings wurden für fli4l ein paar Änderungen vorgenommen. Die Anpassungen befinden sich im *src*-Paket im Verzeichnis `src/fbr/buildroot/package/mini_httpd`.

8.6.2. Skriptnamen

Der Skriptname sollte möglichst vielsagend sein, damit er von anderen Skripten leichter zu unterscheiden ist und es keine Namensüberschneidungen bei verschiedenen OPTs gibt.

Damit die Skripte ausführbar gemacht werden und DOS-Zeilenumbrüche in Unix-Zeilenumbrüche umgewandelt werden, muss in der `opt/<PAKET>.txt` ein entsprechender Eintrag gemacht werden, siehe Tabelle 8.2 (Seite 310).

8.6.3. Menü-Einträge

Um einen Eintrag im Menü vorzunehmen, muss eine Eintragung in der Datei `/etc/httpd/menu` vorgenommen werden. Dieser Mechanismus erlaubt es OPTs, auch im laufendem Betrieb Änderungen am Menü vorzunehmen. Dies sollte nur mit dem Skript `httpd-menu.sh` gemacht werden, da dieses darauf achtet, dass das Dateiformat dieser Datei immer konsistent ist. Um neue Menüpunkte einzufügen, wird es folgendermaßen aufgerufen:

```
httpd-menu.sh add [-p <priority>] <link> <name> [section] [realm]
```

So wird ein Eintrag mit dem Namen `<name>` in den Abschnitt `[section]` eingetragen. Wenn `[section]` weggelassen wird, wird es standardmäßig in den Abschnitt „OPT-Pakete“ eingetragen. `<link>` gibt das Ziel des neuen Links an. `<priority>` spezifiziert die Priorität eines Menüeintrags in seinem Abschnitt. Wird sie nicht angegeben, wird die Standardpriorität 500 benutzt. Die Priorität sollte eine dreistellige Nummer sein. Je kleiner die Priorität, desto weiter oben steht der Link in dem Abschnitt. Soll ein Eintrag möglichst weit nach unten, so ist z. B. die Priorität 900 zu wählen. Bei gleicher Priorität werden die Einträge nach dem Ziel des Links sortiert. Bei `[realm]` wird der Bereich angegeben, für den ein angemeldeter Benutzer mindestens die Berechtigung *view* haben muss, damit der Menüpunkt angezeigt wird. Wird `[realm]` nicht angegeben, wird der Menüpunkt immer angezeigt. Siehe hierzu auch den Abschnitt „Benutzerrechte“ (Seite 352).

Beispiel:

```
httpd-menu.sh add "neuedatei.cgi" "Hier klicken" "Tools" "tools"
```

Dieses Beispiel erzeugt im Abschnitt „Tools“ einen Link mit dem Titel „Hier klicken“ und dem Link-Ziel „neuedatei.cgi“ und legt den Abschnitt, falls dieser nicht vorhanden ist, an.

Das Skript kann auch Einträge aus dem Menü wieder entfernen:

```
httpd-menu.sh rem <link>
```

Mit diesem Aufruf wird der Eintrag mit dem Link `<link>` wieder entfernt.

Wichtig: *Wenn mehrere Menüeinträge auf die gleiche Datei verweisen, werden alle diese Einträge aus dem Menü entfernt.*

Da Abschnitte auch Prioritäten haben können, können diese auch manuell angelegt werden. Wird ein Abschnitt automatisch beim Hinzufügen eines Eintrages angelegt, erhält er automatisch die Priorität 500. Der Syntax zum Anlegen von Abschnitten lautet:

```
httpd-menu.sh addsec <priority> <name>
```

Auch hier sollte `<priority>` nur dreistellige numerische Werte annehmen.

Um sinnvolle Prioritäten in Erfahrung zu bringen, lohnt es sich, auf einem laufenden fli4l in die Datei `/etc/httpd/menu` zu schauen, die Prioritäten stehen in der zweiten Spalte.

Der Vollständigkeit halber wird hier kurz auf das Dateiformat der Menüdatei eingegangen. Wem die Funktion von `httpd-menu.sh` reicht, der kann diesen Abschnitt überspringen. Die Datei `/etc/httpd/menu` hat den folgenden Aufbau: Sie ist in vier Spalten eingeteilt. In der ersten Spalte steht ein Kennbuchstabe, der Überschriften und Einträge unterscheidet. In der zweiten Spalte steht die Sortierungspriorität. Die dritte Spalte enthält bei Einträgen das Ziel des Links und bei Überschriften einen Bindestrich, da dieses Feld bei Überschriften keine Bedeutung hat. Im Rest der Zeile steht der Text, der nachher im Menü erscheint.

Überschriften benutzen den Kennbuchstaben „t“, dann wird ein neuer Menüabschnitt begonnen. Normale Menüeinträge benutzen den Kennbuchstaben „e“. Ein Beispiel:

```
t 300 - Mein tolles OPT
e 200 meinopt.cgi Mach etwas Tolles
e 500 meinopt.cgi?mehr=ja Mach mehr Tolles
```

Beim Bearbeiten dieser Datei muss man darauf achten, dass das `httpd-menu.sh`-Skript die Datei immer sortiert abspeichert. Die einzelnen Abschnitte sind sortiert und die Einträge pro Abschnitt sind in diesem Abschnitt sortiert. Der genaue Sortieralgorithmus kann aus `httpd-menu.sh` übernommen werden – besser wäre allerdings, dieses Skript um mögliche neue Funktionen zu erweitern, damit alle Menü-Bearbeitungen an zentraler Stelle passieren.

8.6.4. Aufbau eines CGI-Skriptes

Die Kopfzeilen

Alle Skripte des Webservers sind einfache Shell-Skripte (Interpreter wie z. B. Perl, PHP, etc. sind viel zu groß für fli4l). Sie sollten mit dem obligatorischen Skript-Header anfangen (Verweis auf den Interpreter, Name, Sinn des Skriptes, Autor, Lizenz).

Das Hilfsskript “cgi-helper“

Gleich nach den Kopfzeilen sollte dann schon das Hilfsskript `cgi-helper` mit folgendem Aufruf eingebunden werden:

```
. /srv/www/include/cgi-helper
```

Wichtig ist ein Leerzeichen zwischen Punkt und Schrägstrich!

Dieses Skript stellt diverse Hilfsfunktionen bereit, die das Erstellen von CGIs für fli4l wesentlich vereinfachen sollen. Außerdem werden mit dem Einbinden des `cgi-helper` auch noch Standardaufgaben ausgeführt, wie beispielsweise das Parsen von Variablen, die mit Formularen oder über die URL übergebenen wurden, oder das Laden von Sprach- und CSS-Dateien.

Tabelle 8.4 gibt einen Überblick über die Funktionen des `cgi-helper`-Skriptes.

Der Inhalt eines CGI-Skriptes

Um ein einheitliches Design und vor allem die Kompatibilität mit zukünftigen fli4l-Versionen zu gewährleisten, ist es sehr zu empfehlen, die Funktionen des Hilfsskriptes `cgi-helper` zu benutzen, auch wenn man in einem CGI theoretisch alle Ausgaben selbst generieren kann.

Eine einfaches CGI-Skript könnte wie folgt aussehen:

Tabelle 8.4.: Funktionen des `cgi-helper`-Skriptes

Name	Funktion
`check_rights`	Überprüfung der Benutzerrechte
`http_header`	Ausgabe eines Standard-HTTP-Headers oder eines speziellen HTTP-Headers, beispielsweise zum Download von Dateien
`show_html_header`	Ausgabe des kompletten Seitenheaders (inkl. HTTP-Header, Kopfzeile und Menü)
`show_html_footer`	Ausgabe des Abschlusses der HTML-Seite
`show_tab_header`	Ausgabe eines Inhalt-Fensters mit Reitern
`show_tab_footer`	Ausgabe des Abschlusses des Inhaltsfensters
`show_error`	Ausgabe einer Box für Fehlermeldungen (Hintergrundfarbe: rot)
`show_warn`	Ausgabe einer Box für Warnmeldungen (Hintergrundfarbe: gelb)
`show_info`	Ausgabe einer Box für Informationen oder Erfolgsmeldungen (Hintergrundfarbe: grün)

```
#!/bin/sh
# --------------------
# Header (c) Autor Datum
# --------------------
# get main helper functions
. /srv/www/include/cgi-helper

show_html_header "Mein erstes CGI"
echo '   <h2>Willkommen</h2>'
echo '   <h3>Dies ist ein Beispiel-CGI-Skript</h3>'
show_html_footer
```

Die Funktion `show_html_header`

Die Funktion `show_html_header` erwartet eine Zeichenkette als Parameter. Diese Zeichenkette stellt den Titel der zu generierenden Seite dar. Die Funktion generiert automatisch das Menü und bindet ebenso automatisch zum Skript gehörende CSS- und Sprachdateien ein. Voraussetzung dafür ist, dass diese sich in den Verzeichnissen `/srv/www/css` bzw. `/srv/www/lang` befinden und denselben Namen (aber natürlich eine andere Endung) wie das Skript haben. Ein Beispiel:

```
/srv/www/admin/OpenVPN.cgi
/srv/www/css/OpenVPN.css
/srv/www/lang/OpenVPN.de
```

Sowohl das Benutzen von Sprachdateien als auch von CSS-Dateien ist optional. Immer eingebunden werden `css/main.css` und `lang/main.<lang>`, wobei `<lang>` der gewählten Sprache entspricht.

Der Funktion `show_html_header` können aber neben dem Titel noch weitere Parameter übergeben werden. Ein Aufruf mit allen möglichen Parametern könnte so aussehen:

```
show_html_header "Titel" "refresh=$time;url=$url;cssfile=$cssfile;showmenu=no"
```

Alle zusätzlichen Parameter müssen, wie im Beispiel gezeigt, mit Anführungszeichen umschlossen und durch ein Semikolon getrennt werden. Eine Überprüfung der Syntax erfolgt *nicht*! Es ist also notwendig, sehr genau auf die Parameterübergabe zu achten.

Hier eine kurze Übersicht über die Funktion der Parameter:

- `refresh=`*time*: Zeit in Sekunden in der die Seite vom Browser neu geladen werden soll
- `url=`*url*: die URL, die bei einem Refresh neu geladen wird
- `cssfile=`*cssfile*: Name einer CSS-Datei, wenn diese vom Namen des CGIs abweicht
- `showmenu=no`: unterdrückt die Anzeige des Menüs und des Headers

Sonstige Richtlinien zum Inhalt:

- Fasst euch kurz :-)
- Schreibt sauberes HTML (SelfHTML[8] ist da ein guter Ansatzpunkt).
- Verzichtet auf hochmodernen Schnick-Schnack (JavaScript ist i. O., wenn es nicht stört, sondern den Benutzer unterstützt, das Ganze muss auch ohne JavaScript funktionieren).

Die Funktion `show_html_footer`

Die Funktion `show_html_footer` schließt den Block im CGI-Skript ab, der durch die Funktion `show_html_header` eingeleitet wurde.

Die Funktion `show_tab_header"`

Damit der Inhalt eurer mit Hilfe des CGIs erzeugten Webseite auch hübsch geordnet aussieht, könnt ihr die `cgi-helper`-Funktion `show_tab_header` nutzen. Sie erzeugt dann anklickbare Reiter („Tabs“ genannt), so dass ihr eure Seite in mehrere logisch voneinander getrennte Bereiche unterteilen könnt.

Der `show_tab_header`-Funktion werden Parameter immer in Paaren übergeben. Der erste Wert entspricht dem Titel eines Reiters, der zweite dem Link. Wird als Link die Zeichenkette „no“ übergeben, wird lediglich der Titel ausgegeben und der Reiter ist nicht anklickbar (und blau).

Im folgenden Beispiel wird ein „Fenster“ mit dem Titel „Ein tolles Fenster“ erzeugt. Im Fenster steht „foo bar“:

```
show_tab_header "Ein tolles Fenster" "no"
echo "foo"
echo "bar"
show_tab_footer
```

Im nächsten Beispiel werden zwei anklickbare Reiter generiert, die dem Skript die Variable `action` mit verschiedenen Werten übergibt.

```
show_tab_header "1. Auswahltab" "$myname?action=machdies" \
                "2. Auswahltab" "$myname?action=machjenes"
echo "foo"
echo "bar"
show_tab_footer
```

[8] siehe http://de.selfhtml.org/

Nun kann das Skript den Inhalt der Variablen `FORM_action` (siehe weiter unten zur Variablenauswertung) auswerten und je nachdem unterschiedliche Inhalte bereitstellen. Damit der angeklickte Reiter auch ausgewählt erscheint und nicht mehr angeklickt werden kann, müsste der Funktion statt dem Link wie schon erwähnt ein „no“ übergeben werden. Das geht aber auch einfacher, wenn man sich an die Konvention in folgendem Beispiel hält:

```
_opt_machdies="1. Auswahltab"
_opt_machjenes="2. Auswahltab"
show_tab_header "$_opt_machdies" "$myname?action=opt_machdies" \
                "$_opt_machjenes" "$myname?action=opt_machjenes"
case $FORM_action in
    opt_machdies) echo "foo" ;;
    opt_machjenes) echo "bar" ;;
esac
show_tab_footer
```

Wird also für den Titel eine Variable verwendet, deren Namen dem Inhalt der Variablen `action` mit führendem Unterstrich (`_`) entspricht, wird der entsprechende Reiter ausgewählt dargestellt.

Die Funktion `show_tab_footer`

Die Funktion `show_tab_footer` schließt den Block im CGI-Skript ab, der durch die Funktion `show_tab_header` eingeleitet wurde.

Mehrsprachfähigkeit

Das Hilfsskript `cgi-helper` enthält weiterhin Funktionen, um CGI-Skripte mehrsprachfähig zu machen. Dazu müssen „nur“ für alle Textausgaben Variablen mit führenden Unterstrichen (`_`) verwendet und diese Variablen in den entsprechenden Sprachdateien definiert werden.
Beispiel:
`lang/opt.de` enthalte:

```
_opt_machdies="Eine Ausgabe"
```

`lang/opt.en` enthalte:

```
_opt_machdies="An Output"
```

`admin/opt.cgi` enthalte:

```
...
echo $_opt_machdies
...
```

Formular-Auswertung

Um Formulare zu verarbeiten, muss man noch einige Dinge wissen. Egal ob die Formular-Methode `GET` oder `POST` verwendet wird, die Parameter finden sich nach dem Einbinden des `cgi-helper`-Skripts (welches wiederum das Hilfsprogramm proccgi aufruft) in den Variablen

`FORM_<Parameter>` wieder. Wenn das Formularfeld also den Namen „eingabe“ hatte, kann im CGI-Skript mit `$FORM_eingabe` darauf zugegriffen werden.

Weitere Informationen zum Programm `proccgi` finden sich unter http://www.fpx.de/fp/Software/ProcCGI.html.

Benutzerrechte: Die Funktion `check_rights`

Um zu prüfen, ob der Benutzer ausreichende Rechte zur Nutzung eines CGI-Skripts besitzt, muss am Anfang des CGI-Skripts die Funktion `check_rights` wie folgt aufgerufen werden:

```
check_rights <Bereich> <Aktion>
```

Das CGI-Skript wird dann nur ausgeführt, wenn der aktuell angemeldete Benutzer

- alle Rechte hat (`HTTPD_RIGHTS_x='all'`), oder
- alle Rechte für den angegebenen Bereich hat (`HTTPD_RIGHTS_x='<Bereich>:all'`), oder
- das Recht hat, die angegebene Aktion im angegebenen Bereich auszuführen (`HTTPD_RIGHTS_x='<Bereich>:<Aktion>'`).

Die Funktion `show_error`

Diese Funktion gibt eine Fehlermeldung in einer roten Box aus. Sie erwartet zwei Parameter: einen Titel sowie eine Meldung. Beispiel:

```
show_error "Error: No key" "No key was specified!"
```

Die Funktion `show_warn`

Diese Funktion gibt eine Warnmeldung in einer gelben Box aus. Sie erwartet zwei Parameter: einen Titel sowie eine Meldung. Beispiel:

```
show_info "Warnung" "Derzeit besteht keine Verbindung!"
```

Die Funktion `show_info`

Diese Funktion gibt eine Informations- oder Erfolgsmeldung in einer grünen Box aus. Sie erwartet zwei Parameter: einen Titel sowie eine Meldung. Beispiel:

```
show_info "Info" "Aktion wurde erfolgreich ausgeführt!"
```

Das Hilfsskript “cgi-helper-ip4“

Gleich nach den Hilfsskript "cgi-helper"kann dann das Hilfsskript cgi-helper-ip4 mit folgendem Aufruf eingebunden werden:

```
. /srv/www/include/cgi-helper-ip4
```

Wichtig ist ein Leerzeichen zwischen Punkt und Schrägstrich!

Dieses Skript stellt Hilfsfunktionen bereit, um Prüfungen von IPv4-Adressen vornehmen zu können.

Die Funktion `ip4_isvalidaddr`

Diese Funktion überprüft, ob eine gültige IPv4-Adresse übergeben wurde. Beispiel:

```
if ip4_isvalidaddr ${FORM_inputip}
then
    ...
fi
```

Die Funktion `ipv4_normalize`

Diese Funktion entfernt aus der übergebenen IPv4-Adresse führende Nullen. Beispiel:

```
ip4_normalize ${FORM_inputip}
IP=$res
if [ -n "$IP" ]
then
    ...
fi
```

Die Funktion `ipv4_isindhcprange`

Diese Funktion prüft, ob die übergebene IPv4-Adresse sich im Bereich der übergebenen Start- und Zieladresse befindet. Beispiel:

```
if ip4_isindhcprange $FORM_inputip $ip_start $ip_end
then
    ...
fi
```

8.6.5. Sonstiges

Dies und das (ja, das ist auch noch wichtig!):

- Der `mini_httpd` schützt Unterverzeichnisse nicht mit einem Passwort. Es muss für jedes Verzeichnis eine eigene `.htaccess`-Datei oder ein Link auf eine andere `.htaccess`-Datei angelegt werden.
- KISS - Keep it simple, stupid!
- Diese Angaben können sich jederzeit ohne Vorankündigung ändern!

8.6.6. Fehlersuche

Um die Fehlersuche innerhalb eines CGI-Skripts zu erleichtern, kann man vor der Einbindung des `cgi-helper`-Skripts den Debugging-Modus aktivieren. Dazu muss die Variable `set_debug` auf den Wert „yes“ gesetzt werden. Dies führt zur Erstellung der Datei `debug.log`, die über die URL `http://<fli4l-Host>/admin/debug.log` heruntergeladen werden kann. Diese enthält alle Aufrufe des CGIs. Die `set_debug`-Variable ist nicht global, muss also in jedem Problem-CGI erneut gesetzt werden. Beispiel:

```
set_debug="yes"
. /srv/www/include/cgi-helper
```

Alternativ kann auch die jeweilige CGI-URL um den Parameter `debug=yes` ergänzt werden, etwa `http://<fli4l-Host>/admin/meinopt.cgi?debug=yes`.

Des Weiteren eignet sich cURL[9] hervorragend zur Fehlersuche, insbesondere wenn die HTTP-Kopfzeilen nicht korrekt zusammengesetzt werden oder der Browser nur weiße Seiten anzeigt. Auch ist das Cachingverhalten moderner Webbrowser bei der Fehlersuche hinderlich.

Beispiel: Mit dem folgenden Aufruf wird der HTTP-Header („*d*ump“, `-D`) sowie die normale Ausgabe des CGIs `admin/mein.cgi` ausgegeben. Es soll der Benutzername („*u*ser“, `-u`) „admin“ verwendet werden.

```
curl -D - http://fli4l/admin/mein.cgi -u admin
```

8.7. Hochfahren, Herunterfahren, Einwählen und Auflegen unter fli4l

8.7.1. Bootkonzept

fli4l 2.0 sollte eine saubere Installation auf eine Festplatte oder ein CompactFlash(TM)-Medium bieten, aber auch eine Installation auf ein Zip-Medium oder die Erstellung einer bootfähigen CD-ROM sollte möglich sein. Zusätzlich sollte die Festplattenversion sich nicht grundlegend von einer Installation auf Diskette[10] unterscheiden.

Diese Anforderungen wurden realisiert, indem die Dateien des `opt.img`-Archivs aus der bisherigen RAM-Disk auf eine anderes Medium verlagert werden können. Dies kann eine Partition auf einer Festplatte bzw. einem CF-Medium sein. Dieses zweite Volume wird unter `/opt` eingehängt, und dort liegende Programme werden nur noch über symbolische Links in das RootFS integriert. Das entstehende Layout im RootFS-Dateisystem entspricht dann dem im `opt`-Verzeichnis der ausgepackten fli4l-Distribution mit einer Ausnahme – das `files`-Präfix entfällt. Die Datei `opt/etc/rc` ist dann also direkt unter `/etc/rc` zu finden, `opt/files/bin/busybox` unter `/bin/busybox`. Dass diese Dateien unter Umständen nur symbolische Verknüpfungen auf ein im Nur-Lese-Modus eingehängtes Volume sind, kann man ignorieren, solange man die Dateien nicht modifizieren möchte. Will man dies tun, muss man die Dateien vorher mit `mk_writable` (s. u.) schreibbar machen.

8.7.2. Start- und Stopp-Skripte

Skripte, die beim Hochfahren des Systems ausgeführt werden sollen, befinden sich in den Verzeichnissen `opt/etc/boot.d/` und `opt/etc/rc.d/` und werden auch in dieser Reihenfolge ausgeführt. Des Weiteren befinden sich in `opt/etc/rc0.d/` Skripte, die beim Herunterfahren des Systems ausgefühlt werden.

Wichtig: *Da zum Ausführen dieser Skripte kein separater Prozess erzeugt wird, dürfen sie nicht mit „exit“ abgeschlossen werden. Ein solcher Befehl führt zum vorzeitigen Abbruch des Bootvorgangs!*

[9] siehe http://de.wikipedia.org/wiki/CURL

[10] Ursprünglich konnte fli4l auch von einer Diskette betrieben werden. Da fli4l inzwischen dafür zu groß geworden ist, wird dies nicht mehr unterstützt.

Start-Skripte in `opt/etc/boot.d/`

Skripte, die in diesem Verzeichnis liegen, werden als erstes ausgeführt. Ihre Aufgabe ist es, das Boot-Volume einzuhängen, die auf dem Boot-Medium liegende Konfigurationsdatei `rc.cfg` einzulesen und das Opt-Archiv zu entpacken. Je nach Boot-Typ (Seite 25) sind diese Skripte mehr oder weniger kompliziert und tun die folgenden Dinge:

- Laden von Hardware-Treibern (optional)
- Boot-Volume einhängen (optional)
- Konfigurationsdatei `rc.cfg` vom Boot-Volume einlesen und in die Datei `/etc/rc.cfg` schreiben
- Einhängen des Opt-Volumes (optional)
- Extrahieren des Opt-Archivs (optional)

Damit die Skripte überhaupt eine Chance haben, etwas über die fli4l-Konfiguration zu erfahren, wird die Konfigurationsdatei auch ins RootFS-Archiv unter `/etc/rc.cfg` eingebunden; die Konfigurationsvariablen in dieser Datei stehen den Start-Skripten in `opt/etc/boot.d/` direkt zur Verfügung. Nach dem Einhängen des Boot-Volumes wird die `/etc/rc.cfg` durch die Konfigurationsdatei vom Boot-Volume ersetzt, so dass den Start-Skripten in `opt/etc/rc.d/` (s. u.) die aktuelle Konfiguration vom Boot-Volume zur Verfügung steht. [11]

Start-Skripte in `opt/etc/rc.d/`

Befehle, die immer beim Starten des Routers ausgeführt werden müssen, können in Skripten im Verzeichnis `opt/etc/rc.d/` abgelegt werden. Hierbei gelten folgende Konventionen:

1. Es gilt folgende Namenskonvention:

   ```
   rc<dreistellige Zahl>.<Name des OPTs>
   ```

   Die Skripte werden in aufsteigender Reihenfolge der Zahlen gestartet. Ist mehreren Skripten dieselbe Zahl zugeordnet, entscheidet die alphabetische Sortierung nach dem Punkt. Falls der Start eines Pakets erst nach einem anderen erfolgen darf, wird das durch die Zahl festgelegt.

   Hier eine ungefähre Richtlinie, welche Nummern für welche Aufgaben verwendet werden sollten:

2. In diesen Skripten *müssen* alle Funktionen, die das RootFS verändern, hinterlegt werden, etwa das Anlegen eines Verzeichnisses `/var/log/lpd`.

[11] Normalerweise sind diese beiden Dateien identisch. Eine Abweichung entsteht nur, wenn die Konfigurationsdatei auf dem Boot-Volume händisch editiert wird, etwa um die Konfiguration nachträglich abzuändern, ohne die fli4l-Archive neu zu bauen.

Nummer	Aufgabe
000-099	Grundsystem (Hardware, Zeitzone, Dateisystem)
100-199	Kernel-Module (Treiber)
200-299	externe Verbindungen (PPPoE, ISDN4Linux, PPtP)
300-399	Netzwerk (Routing, Interfaces, Paketfilter)
400-499	Server (DHCP, HTTPD, Proxy, etc.)
500-900	beliebig
900-997	Alles, was eine Einwahl hervorrufen könnte
998-999	reserviert (bitte nicht benutzen!)

3. In diesen Skripten dürfen *keine* Schreibzugriffe auf Dateien erfolgen, die Teil des Opt-Archivs sein können, da diese Dateien auf einem im Nur-Lese-Modus eingehängten Volume liegen können. Muss man eine solche Datei modifizieren, muss man sie vorher mit Hilfe der Funktion `mk_writable` (s. u.) beschreibbar machen. Durch den Aufruf der Funktion wird die Datei (wenn nötig) als beschreibbare Kopie im RootFS abgelegt. Ist die Datei bereits beschreibbar, bewirkt der `mk_writable`-Aufruf nichts.

   Wichtig: *`mk_writable` muss direkt auf Dateien im RootFS angewandt werden, nicht über den Umweg des `opt`-Verzeichnisses. Will man also `/usr/local/bin/foo` modifizieren, ruft man `mk_writable` mit dem Argument `/usr/local/bin/foo` auf.*

4. Diese Skripte müssen vor der Ausführung der eigentlichen Befehle prüfen, ob das dazugehörige OPT auch aktiv ist. Das ist normalerweise durch eine einfache Fallunterscheidung erledigt:

   ```
   if [ "$OPT_<OPT-Name>" = "yes" ]
   then
       ...
       # Hier OPT starten!
       ...
   fi
   ```

5. Um die Fehlersuche zu erleichtern, sollten die Skripte mit `begin_script` und `end_script` geklammert werden:

   ```
   if [ "$OPT_<OPT-Name>" = "yes" ]
   then
       begin_script FOO "configuring foo ..."
       ...
       end_script
   fi
   ```

   Die Fehlersuche einzelner Start-Skripte kann man dann einfach via `FOO_DO_DEBUG='yes'` aktivieren.

6. Den Skripten stehen alle Konfigurationsvariablen direkt zur Verfügung. Im Abschnitt „Umgang mit Konfigurationsvariablen“ (Seite 341) wird erklärt, wie man von anderen Skripten aus auf die Konfigurationsvariablen zugreifen kann.

7. Der Pfad `/opt` darf auch nicht als Speicherplatz für Datenbestände der OPTs benutzt werden. Falls zusätzlicher Speicherplatz benötigt wird, sollte dem Benutzer über eine Konfigurationsvariable die Möglichkeit gegeben werden, einen geeigneten Pfad auszuwählen. Je nach Art der zu speichernden Daten (persistente oder transiente Daten) sind verschiedene Standard-Belegungen sinnvoll. So bieten sich für transiente Daten etwa Pfade unterhalb von `/var/run/` an; für persistente Daten sollte die Funktion map2persistent (Seite 341) in Kombination mit einer geeigneten Konfigurationsvariable verwendet werden.

Stopp-Skripte in `opt/etc/rc0.d/`

Jeder Rechner muss mal heruntergefahren oder neu gestartet werden. Nun kann es vorkommen, dass man Vorgänge ausführen muss, bevor der Rechner heruntergefahren oder neu gestartet wird. Zum Herunterfahren und Neustarten sind die Befehle „halt“ bzw. „reboot“ zuständig. Diese Befehle werden auch aufgerufen, wenn man im IMONC oder in der Web-GUI auf die entsprechenden Schaltflächen klickt.

Alle Stopp-Skripte liegen im Verzeichnis `opt/etc/rc0.d/`. Ihre Dateinamen werden analog zu den Start-Skripten gebildet. Sie werden ebenfalls in *auf*steigender Reihenfolge der Zahlen ausgeführt.

8.7.3. Hilfsfunktionen

In `/etc/boot.d/base-helper` werden verschiedene Funktionen bereitgestellt, die von den Start- und anderen Skripten verwendet werden können. Das betrifft Dinge wie Unterstützung zur Fehlersuche, Laden von Kernel-Modulen oder Ausgabe von Meldungen. Die einzelnen Funktionen werden im Folgenden aufgelistet und kurz beschrieben.

Skript-Steuerung

`begin_script <Symbol> <Meldung>`: Gibt eine Meldung aus und aktiviert die Fehlersuche im Skript mittels `set -x`, falls `<Symbol>_DO_DEBUG` auf „yes“ steht.

`end_script`: Gibt eine Abschluss-Meldung aus und deaktiviert die Fehlersuche, falls sie mit `begin_script` aktiviert wurde. Für jeden `begin_script`-Aufruf muss es einen zugehörigen `end_script`-Aufruf geben (und umgekehrt).

Laden von Kernel-Modulen

`do_modprobe [-q] <Modul> <Parameter>*`: Lädt ein Kernel-Modul inkl. eventueller Parameter bei gleichzeitiger Auflösung der Modulabhängigkeiten. Der Parameter „-q“ verhindert, dass im Fehlerfall eine Meldung auf der Konsole ausgegeben und ins Boot-Protokoll geschrieben wird. Die Funktion liefert im Erfolgsfall den Rückgabewert null zurück und im Fehlerfall einen Wert ungleich null. Damit lässt sich Code schreiben, der ein Fehlschlagen des Ladens eines Kernel-Moduls geeignet behandelt:

```
if do_modprobe -q acpi-cpufreq
then
    # kein CPU-Frequenzsteuerung via ACPI
    log_error "CPU-Frequenzsteuerung via ACPI nicht verfügbar!"
```

```
    # [...]
else
    log_info "CPU-Frequenzsteuerung via ACPI aktiviert."
    # [...]
fi
```

do_modrobe_if_exists [-q] <Modulpfad> <Modul> <Parameter>*:
Prüft, ob das Modul /lib/modules/<Kernel-Version>/<Modulpfad>/<Modul> existiert und ruft bei Vorhandensein die Funktion do_modprobe auf.

Wichtig: *Das Modul muss tatsächlich unter dem angegebenen Modulnamen existieren, der Modulname darf kein Alias sein. Bei einem Alias wird direkt* ***do_modprobe*** *aufgerufen.*

Meldungen und Fehlerbehandlung

log_info <Meldung>: Schreibt eine Meldung auf die Konsole und nach /bootmsg.txt. Wird keine Meldung als Parameter übergeben, liest log_info von der Standard-Eingabe. Die Funktion liefert als Rückgabewert immer null zurück.

log_warn <Meldung>: Schreibt eine Warnmeldung auf die Konsole und nach /bootmsg.txt, wobei vor die Meldung die Zeichenkette WARN: gesetzt wird. Wird keine Meldung als Parameter übergeben, liest log_warn von der Standard-Eingabe. Die Funktion liefert als Rückgabewert immer null zurück.

log_error <Meldung>: Schreibt eine Fehlermeldung auf die Konsole und nach /bootmsg.txt, wobei vor die Meldung die Zeichenkette ERR: gesetzt wird. Wird keine Meldung als Parameter übergeben, liest log_error von der Standard-Eingabe. Die Funktion liefert als Rückgabewert immer einen Wert ungleich null zurück.

set_error <Meldung>: Gibt die Fehlermeldung aus und setzt eine interne Fehlervariable, das später via is_error geprüft werden kann.

is_error: Setzt die interne Fehlervariable zurück und liefert wahr zurück, falls sie vorher durch set_error gesetzt wurde.

Netzwerk-Funktionen

translate_ip_net <Wert> <Variablenname> [<Ergebnisvariable>]: Ersetzt symbolische Referenzen in Parametern. Momentan finden folgende Übersetzungen statt:

***.*.*.*, none, default, pppoe** werden nicht übersetzt

any wird durch 0.0.0.0/0 ersetzt

dynamic wird durch die IP-Adresse des Routers ersetzt, über welche die Verbindung zum Internet besteht

IP_NET_x wird durch das in der Konfiguration stehende Netzwerk ersetzt

IP_NET_x_IPADDR wird durch die in der Konfiguration stehende IP-Adresse ersetzt

IP_ROUTE_x wird durch das in der Konfiguration stehende geroutete Netzwerk ersetzt

@<Hostname> wird durch die in der Konfiguration für den angegeben Host spezifizierte IP-Adresse ersetzt

Das Ergebnis der Übersetzung wird in der Variable gespeichert, deren Name im dritten Parameter übergeben wird; fehlt dieser Parameter, wird das Ergebnis in der Variable `res` gespeichert. Der Variablenname, der im zweiten Parameter übergeben wird, wird nur für Fehlermeldungen benutzt, falls die Übersetzung fehlschlägt; hier kann also vom Aufrufer die Quelle des zu übersetzenden Wertes angegeben werden. Im Fehlerfall wird dann eine Meldung wie

```
Unable to translate value '<Wert>' contained in <Variablenname>.
```

ausgegeben.

Der Rückgabewert ist null, falls die Übersetzung erfolgreich war, und ungleich null, falls ein Fehler aufgetreten ist.

Diverses

mk_writable <Datei>: Stellt sicher, dass die übergebene Datei beschreibbar ist. Befindet sich die Datei auf einem im Nur-Lese-Modus eingehängten Volume und ist lediglich über eine symbolische Verknüpfung ins Dateisystem eingebunden, wird eine lokale Kopie angelegt, die dann beschreibbar ist.

unique <Liste>: Entfernt Duplikate aus der übergebenen Liste. Das Resultat wird in der Variable `list` zurückgegeben.

8.7.4. ttyI-Geräte

Für die ttyI-Geräte (`/dev/ttyI0` ... `/dev/ttyI15`), über welche die „Modem-Emulation“ der ISDN-Karte genutzt werden kann, existiert ein Zähler, um Konflikte zwischen verschiedenen Paketen, die diese Geräte nutzen, zu vermeiden. Hierzu wird beim Start des Routers die Datei `/var/run/next_ttyI` angelegt, die von den verschiedenen OPTs abgefragt und fortgezählt werden kann. Mit dem folgenden Beispielskript kann dieser Wert abgefragt, um eins erhöht und wieder für das nächste OPT exportiert werden.

```
ttydev_error=
ttydev=$(cat /var/run/next_ttyI)
if [ $ttydev -le 16 ]
then                                    # ttyI device available? yes
    ttydev=$((ttydev + 1))              # ttyI device + 1
    echo $ttydev >/var/run/next_ttyI    # save it
else                                    # ttyI device available? no
    log_error "No ttyI device for <Name deines OPTs> available!"
    ttydev_error=true                   # set error for later use
fi

if [ -z "$ttydev_error" ]               # start OPT only if next tty device
then                                    # was available to minimize error
    ...                                 # messages and minimize the
                                        # risk of uncomplete boot
fi
```

8.7.5. Skripte beim Einwählen und Auflegen

Allgemeines

Nach dem Herstellen bzw. Trennen einer Wählverbindung werden die Skripte in `/etc/ppp/` abgearbeitet. Hier können OPTs Aktionen hinterlegen, die nach dem Herstellen bzw. Auflegen der Verbindung nötig sind. Benannt werden die Dateien wie folgt:

```
ip-up<dreistellige Zahl>.<OPT-Name>
ip-down<dreistellige Zahl>.<OPT-Name>
```

Dabei werden die `ip-up`-Skripte nach dem *Aufbau* und die `ip-down`-Skripte nach dem *Abbau* der Verbindung ausgeführt.

Wichtig: *In den `ip-down`-Skripten dürfen keine Aktionen ausgeführt werden, die zu einer erneuten Einwahl führen, da dadurch nur ein Dauer-Online-Zustand erreicht wird, was für Nicht-Flatrate-Benutzer ein teures Unterfangen ist.*

Wichtig: *Da für die einzelnen Skripte kein eigener Prozess erzeugt wird, dürfen auch diese Skripte* nicht *mit „exit“ abgeschlossen werden!*

Hinweis: Wenn ein Skript prüfen will, ob überhaupt die `ip-up`-Skripte ausgeführt werden, kann es ab `rc400` die Variable `ip_up_events` prüfen. Steht diese auf „yes“, gibt es Wählverbindungen, und die `ip-up`-Skripte werden ausgeführt. Steht diese auf „no“, sind keine Wählverbindungen konfiguriert, und es werden keine `ip-up`-Skripte ausgeführt. Von dieser Regel gibt es eine Ausnahme: Wenn ein reiner Ethernet-Router ohne Wählverbindungen konfiguriert wurde, aber eine Default-Route (`0.0.0.0/0`) existiert, so werden am Ende des Boot-Vorgangs die `ip-up`-Skripte genau einmal ausgeführt. (Analog werden vor dem Herunterfahren einmalig die `ip-down`-Skripte ausgeführt.)

Variablen

Durch das spezielle Aufrufkonzept stehen die folgenden Variablen den `ip-up`- und `ip-down`-Skripten zur Verfügung:

`real_interface`	die aktuelle Schnittstelle, also z. B. `ppp0`, `ippp0`, ...
`interface`	das IMOND-Interface, also `pppoe`, `ippp0`, ...
`tty`	verbundenes Terminal, möglicherweise leer!
`speed`	die Verbindunggeschwindigkeit, bei ISDN z. B. 64000
`local`	die eigene IP-Adresse
`remote`	die IP-Adresse des Point-To-Point-Partners
`is_default_route`	gibt an, ob das aktuelle `ip-up`/`ip-down` für die Schnittstelle durchgeführt wird, über welche die Default-Route geht (kann „yes“ oder „no“ sein)

Default-Route

Seit Version 2.1.0 werden die `ip-up`/`ip-down`-Skripte nicht nur für die Schnittstelle ausgeführt, über welche die Default-Route geht, sondern für alle Verbindungen, welche die `ip-up`- und

ip-down-Skripte aufrufen. Um das alte Verhalten zu simulieren, muss in ip-up- und ip-down-Skripten die folgende Abfrage eingefügt werden:

```
# is a default-route-interface going up?
if [ "$is_default_route" = "yes" ]
then
    # die eigentlichen Aktionen
fi
```

Natürlich darf das neue Verhalten auch für spezielle Aktionen ausgenutzt werden.

8.8. Paket „template“

Um einiges von dem hier Beschriebenen etwas zu veranschaulichen, liegt der fli4l-Distribution das Paket „template“ bei. Dieses erklärt an kleinen Beispielen, wie:

- eine Konfigurationsdatei auszusehen hat (config/template.txt)
- eine Check-Datei geschrieben wird (check/template.txt)
- die erweiterten Prüfmöglichekiten verwendet werden (check/template.ext)
- Konfigurationsvariablen für spätere Verwendung abgelegt werden können (opt/etc/rc.d/rc999.template)
- abgelegte Konfigurationsvariablen wieder ausgelesen werden (opt/files/usr/bin/template_show_config)

8.9. Aufbau des Boot-Datenträgers

Seit Version 1.5 wird das Programm syslinux zum Booten verwendet. Dieses hat den Vorteil, dass ein DOS-kompatibles Dateisystem auf dem Datenträger zur Verfügung steht.

Der Boot-Datenträger enthält folgende Dateien:

ldlinux.sys	der Urlader („Boot loader“) syslinux
syslinux.cfg	Konfigurationsdatei für syslinux
kernel	Linux-Kernel
rootfs.img	RootFS: enthält zum Booten nötige Programme
opt.img	Optionale Dateien: Treiber und Pakete
rc.cfg	Konfigurationsdatei mit den benutzten Variablen aus den Dateien des Konfigurationsverzeichnisses
boot.msg	Texte für das syslinux-Bootmenü
boot_s.msg	Texte für das syslinux-Bootmenü
boot_z.msg	Texte für das syslinux-Bootmenü
hd.cfg	Konfigurationsdatei zur Zuordnung der Partitionen

Durch das Skript mkfli4l.sh (bzw. mkfli4l.bat) werden zunächst die Dateien opt.img, syslinux.cfg und rc.cfg sowie das rootfs.img erzeugt. Die dafür nötigen Dateien ermittelt das Programm mkfli4l (im unix- bzw. windows-Unterverzeichnis). In den beiden Archiven sind

die benötigten Kernel- und andere Pakete enthalten. Die Datei `rc.cfg` befindet sich sowohl im Opt-Archiv als auch auf dem Boot-Datenträger.[12]

Anschließend werden die Dateien `kernel`, `rootfs.img`, `opt.img` und `rc.cfg` zusammen mit den `syslinux`-Dateien auf den Datenträger kopiert.

Beim Booten von fli4l wird über das Skript `/etc/rc` die `rc.cfg` ausgewertet und das komprimierte `opt.img`-Archiv in die RootFS-RAM-Disk integriert (je nach Installationstyp werden dabei die Dateien direkt in die RootFS-RAM-Disk entpackt oder über symbolische Verknüpfungen eingebunden). Danach werden die Skripte in `/etc/rc.d/` in alphanumerischer Reihenfolge ausgeführt und somit die Treiber geladen und die Dienste gestartet.

8.10. Konfigurationsdateien

Hier werden die Dateien kurz aufgeführt, die vom fli4l-Router on-the-fly beim Booten erzeugt werden.

1. Konfiguration Provider
   - `etc/ppp/pap-secrets`
   - `etc/ppp/chap-secrets`
2. Konfiguration DNS
   - `etc/resolv.conf`
   - `etc/dnsmasq.conf`
   - `etc/dnsmasq_dhcp.conf`
   - `etc/resolv.dnsmasq`
3. Hosts-Datei
   - `etc/hosts`
4. imond-Konfiguration
   - `etc/imond.conf`

8.10.1. Konfiguration Provider

Für den ausgesuchten Provider wird in `etc/ppp/pap-secrets` die User-ID und das Passwort angepasst.

Beispiel für Provider Planet-Interkom:

```
# Secrets for authentication using PAP
# client        server  secret                  IP addresses
"anonymer"      *       "surfer"                *
```

Dabei ist im Beispiel „anonymer“ die USER-ID. Als Remote-Server wird prinzipiell jeder erlaubt (deshalb „*“). „surfer“ ist das Passwort für den Provider Planet-Interkom.

[12] Die Fassung im Opt-Archiv ist während der frühen Boot-Phase nötig, weil zu diesem Zeitpunkt das Boot-Volume noch nicht eingehängt ist.

8.10.2. Konfiguration DNS

Man kann den fli4l-Router als DNS-Server einsetzen. Warum dies sinnvoll und bei Windows-Rechnern im LAN sogar zwingend notwendig ist, wird in der Dokumentation des „base“-Pakets erläutert.

Die Resolver-Datei `etc/resolv.conf` enthält den Domainnamen und den zu verwendenten Nameserver. Sie hat folgenden Inhalt (wobei „domain.de“ nur ein Platzhalter für den Wert der Konfigurationsvariable `DOMAIN_NAME` ist):

```
search domain.de
nameserver 127.0.0.1
```

Der DNS-Server dnsmasq wird über die Datei `etc/dnsmasq.conf` konfiguriert. Sie wird beim Booten vom Skript `rc001.base` sowie `rc370.dnsmasq` automatisch erzeugt und könnte wie folgt aussehen:

```
user=dns
group=dns
resolv-file=/etc/resolv.dnsmasq
no-poll
no-negcache
bogus-priv
log-queries
domain-suffix=lan.fli4l
local=/lan.fli4l/
domain-needed
expand-hosts
filterwin2k
conf-file=/etc/dnsmasq_dhcp.conf
```

8.10.3. Hosts-Datei

Diese Datei enthält eine Zuordnung von Host-Namen zu IP-Adressen. Diese Zuordnung ist jedoch nur lokal auf dem fli4l verwendbar, für andere Rechner im LAN ist sie nicht sichtbar. Diese Datei ist eigentlich überflüssig, wenn zusätzlich ein lokaler DNS-Server gestartet wird.

8.10.4. imond-Konfiguration

Die Datei `etc/imond.conf` wird unter anderem aus den Konfigurationsvariablen `CIRC_x_NAME`, `CIRC_x_ROUTE`, `CIRC_x_CHARGEINT` und `CIRC_x_TIMES` konstruiert. Sie kann aus bis zu 32 Zeilen (ohne die Kommentarzeilen) bestehen. Jede Zeile besteht aus acht Spalten:

1. Bereich Wochentag bis Wochentag
2. Bereich Stunde bis Stunde
3. Device (`ippp`*X* oder `isdn`*X*)
4. Circuit mit Default-Route: „yes“/„no“
5. Telefonnummer

6. Name des Circuits
7. Telefonkosten pro Minute in Euro
8. Zeittakt („Charge interval“) in Sekunden

Hier ein Beispiel:

```
#day  hour  device  defroute  phone        name        charge  ch-int
Mo-Fr 18-09 ippp0   yes       010280192306 Addcom      0.0248  60
Sa-Su 00-24 ippp0   yes       010280192306 Addcom      0.0248  60
Mo-Fr 09-18 ippp1   yes       019160       Compuserve  0.019   180
Mo-Fr 09-18 isdn2   no        0221xxxxxxx  Firma       0.08    90
Mo-Fr 18-09 isdn2   no        0221xxxxxxx  Firma       0.03    90
Sa-Su 00-24 isdn2   no        0221xxxxxxx  Firma       0.03    90
```

Weitere Erklärungen zum Least-Cost-Routing findet man in der Dokumentation des „base“-Pakets.

8.10.5. Die `/etc/.profile`-Datei

Die Datei `/etc/.profile` enthält benutzerdefinierte Einstellungen für die Shell. Um die Standard-Einstellungen zu überschreiben, ist es nötig, unterhalb seines Konfigurationsverzeichnisses eine Datei `etc/.profile` zu erstellen. Dort können dann Einstellungen zum Prompt oder Abkürzungen (so genannte „Aliase“) eingetragen werden.

Wichtig: *Diese Datei darf kein* `exit` *enthalten!*

Beispiel:

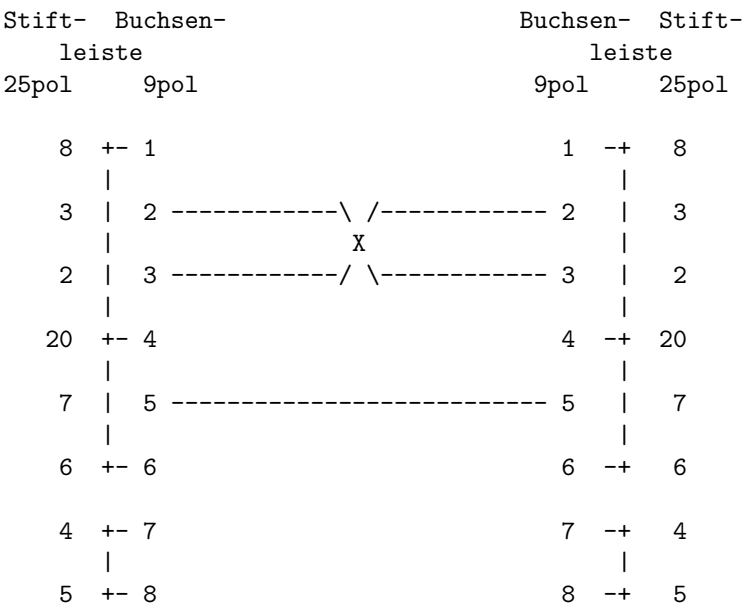
```
alias ll='ls -al'
```

A. Anhang zum Basispaket

A.1. Nullmodemkabel

Für die Verwendung des optionalen Programmpakets PPP (Seite 200) benötigt man ein Nullmodemkabel.

Dieses muß mindestens 3 Adern haben. Hier die Anschluß-Belegung:



Bei den Steckern müssen die im Schaltbild gezeigte Brücken eingelötet werden.

A.2. Serielle Console

fli4l kann ohne Monitor und Tastatur eingesetzt werden. Ein Nachteil davon ist, dass eventuelle Fehlermeldungen nicht bemerkt werden, weil sich nicht alle Meldungen über die syslog-Schnittstelle umleiten lassen.

Eine Möglichkeit ist die Umlenkung der Console-Meldungen auf seinen PC oder auf ein klassisches Terminal, nämlich über die serielle Schnittstelle. Die Konfiguration erfolgt ueber die Variablen SER_CONSOLE (Seite 30), SER_CONSOLE_IF (Seite 31) und SER_CONSOLE_RATE (Seite 31).

Rechner mit älteren Mainboards/Karten unterstützen keine höheren Geschwindigkeiten als 38400 Bd. Deshalb sollte man es zunächst mit höchstens 38400 Bd probieren, bevor man sich an höhere Geschwindigkeiten heranwagt. Da lediglich Text-Ausgaben über die Console gehen, sind höhere Geschwindigkeiten eigentlich auch gar nicht notwendig.

Sämtliche Meldungen, die normalerweise auf der Console ausgegeben werden, werden nun auf die serielle Schnittstelle gelenkt – auch die Meldungen des Bootvorgangs!

Als Kabel zum Terminal oder PC mit Terminalemulation kommt ein Nullmodemkabel (Seite 365) zum Einsatz. Ich rate aber davon ab, ein Standard-Nullmodemkabel zu verwenden, weil dort normalerweise auch die Handshake-Leitungen verdrahtet sind. Ist das Terminal bzw. der PC abgeschaltet (oder die Terminalemulation nicht geladen), kann es bei Verwendung eines Standard-Nullmodemkabels zu einem Hangup von fli4l kommen!

Deshalb ist hier eine spezielle Verdrahtung notwendig, um fli4l auch mit abgeschaltetem Terminal betreiben zu können. Es wird dafür ein 3-adriges Kabel benötigt, wobei einige Kontakte an den Steckern gebrückt werden müssen. Siehe dazu bei Nullmodemkabel (Seite 365).

A.3. Programme

Um Platz auf dem Bootmedium zu sparen, wird unter anderem das Paket “BusyBox” verwendet. Das Programm ist ein einzelnes Exceutable, welches die Standard-Unix-Programme

```
[, [[, arping, ash, base64, basename, bbconfig, blkid, bunzip2, bzcat, bzip2,
cat, chgrp, chmod, chown, chroot, cmp, cp, cttyhack, cut, date, dd, df,
dirname, dmesg, dnsdomainname, echo, egrep, expr, false, fdflush, fdisk, find,
findfs, grep, gunzip, gzip, halt, hdparm, head, hostname, inetd, init, insmod,
ip, ipaddr, iplink, iproute, iprule, iptunnel, kill, killall, klogd, less, ln,
loadkmap, logger, ls, lsmod, lzcat, makedevs, md5sum, mdev, mkdir, mknod,
mkswap, modprobe, mount, mv, nameif, nice, nslookup, ping, ping6, poweroff,
ps, pscan, pwd, reboot, reset, rm, rmmod, sed, seq, sh, sleep, sort, swapoff,
swapon, sync, sysctl, syslogd, tail, tar, test, top, tr, true, tty, umount,
uname, unlzma, unxz, unzip, uptime, usleep, vi, watch, xargs, xzcat, zcat
```

nachbildet. Zumeist sind es jedoch “minimalistische” Implementationen, welche nicht den vollen Funktionsumfang abdecken, aber vollauf den bescheidenen Anforderungen von fli4l genügen.

BusyBox steht unter GPL und ist als Source komplett erhältlich unter
http://www.busybox.net/

A.4. Andere i4l-Tools

Es gibt für isdn4linux viele weitere Tools, die auch fli4l bereichern würden. Das Problem ist leider der Platz! Bestimmt wäre isdnlog als Tool zum Berechnen der Online-Gebühren wesentlich geeigneter, jedoch ist isdnlog einfach zu fett!

imond braucht weniger als 10% des Platzes, übernimmt dabei Monitoring, Controlling und LC-Routing, wenn auch nicht alles ganz perfekt ist.

A.5. Fehlersuche

Hilfreich bei der Fehlersuche sind natürlich einmal die Console-Outputs. Diese rauschen aber oft einfach so durch, dass man gar nicht mehr mitlesen kann. Hinweis: Mit SHIFT-BILD-RAUF kann man zurück-, mit SHIFT-BILD-RUNTER wieder vorblättern.

Falls im Betrieb des Routers Fehlermeldungen “try-to-free-pages” auftreten ist zuwenig für Programme nutzbares RAM übrig. Als Abhilfe stehen dann folgende Optionen zur Verfügung:

- mehr RAM einbauen
- weniger Opt-Pakete einsetzen
- eine Festplatteninstallation nach Typ B (Seite 17) durchführen

Auch proc-Dateien können bei der Fehlersuche helfen. z.B. gibt der Befehl

```
cat /proc/interrupts
```

die von den Treibern verwendeten Interrupts aus – nicht die tatsächlich von der Hardware belegten!

Weitere interessante Dateien unter /proc sind devices, dma, ioports, kmsg, meminfo, modules, uptime, version und pci (falls der Router einen PCI-Bus hat).

Meist liegt ein Verbindungsproblem bei ipppd vor, insbesondere bei der Authentifizierung. Dann helfen oft die Variablen

```
OPT_SYSLOGD='yes'

OPT_KLOGD='yes'
```

in config/base.txt und

```
ISDN_CIRC_x_DEBUG='yes'
```

in config/isdn.txt weiter.

A.6. Literaturhinweise

- Computer Networks, Andy Tanenbaum
- TCP/IP Netzanbindung von PCs, Craig Hunt
- TCP/IP, Kevin Washburn, Jim Evans, Verlag: Addison-Wesley, ISBN: 3-8273-1145-4
- TCP/IP Netzanbindung von PCs, ISBN 3-930673-28-2
- TCP/IP Netzwerk Administration, ISBN 3-89721-110-6
- Linux-Anwenderhandbuch, ISBN 3-929764-06-7
- TCP/IP im Detail: http://www.nickles.de/c/s/ip-adressen-112-1.htm
- Generell das online Linuxanwenderhandbuch von Lunetix unter: http://www.linux-ag.com/LHB/
- Einführung in die Linux-Firewall: http://www.little-idiot.de/firewall/

A.7. Präfixe

Bei den Einheiten richten die Präfixe in dieser Doku sich nach IEC 60027-2.
Siehe: http://physics.nist.gov/cuu/Units/binary.html.

A.8. Gewähr und Haftung

Natürlich kann für das gesamte fli4l-Paket oder für Teile davon keine Gewähr dafür übernommen werden, dass es überhaupt funktioniert oder dass irgendeine Dokumentation in diesem Verzeichnis oder einem der Unterverzeichnisse korrekt ist.

Auch ist jede Haftung wegen evtl. entstandender Schäden oder Kosten ausgeschlossen!

A.9. Danke

In diesem Abschnitt der Dokumentation soll all dennen durch namentlich Nennung gedankt werden die zur Entwicklung von fli4l beitragen bzw. beigetragen haben.

A.9.1. Projektgründung

Meyer, Frank E-Mail: frank(at)fli4l(dot)de

Frank startete am 04.05.2000 das Projekt fli4l!
Siehe: http://www.fli4l.de/home/eigenschaften/historie/

A.9.2. Entwickler- und Testteam

Das fli4l-Team bilden (in alphabetischer Reihenfolge):

Eckhofer, Felix (*Dokumentation, Howtos*)
E-Mail: felix(at)fli4l(dot)de
Franke, Roland (*OW, FBR*)
E-Mail: fli4l(at)franke-prem(dot)de
Hilbrecht, Claas (*VPN, Kernel*)
E-Mail: claas(at)jucs-kramkiste(dot)de
Klein, Sebastian (*Kernel, Wlan*)
E-Mail: -
Knipping, Michael (*Accounting*)
E-Mail: fli4l(at)knibo(dot)de
Krister, Stefan (*Opt-Cop, lcd4linux*)
E-Mail: stefan(dot)krister(at)chreativ(dot)chaos(dot)de
Miksch, Gernot (*LCD*)
E-Mail: gernot_miksch(at)gmx(dot)de
Schiefer, Peter (*fli4l-CD, Opt-Cop, Webseite, Releasemanagement*)
E-Mail: peter(at)fli4l(dot)de
Schulz, Christoph (*FBR, IPv6, Kernel*)
E-Mail: fli4l(at)kristov(dot)de
Siebmanns, Harvey (*Dokumentation*)
E-Mail: -

Spieß, Carsten (*Dsltool, Hwsupp, Rrdtool, Webgui*)
E-Mail: -
Vosselman, Arwin (*LZS-Kompression, Dokumentation*)
E-Mail: arwin(at)xs4all(dot)nl
Wallmeier, Nico (*Windows-Imonc*)
E-Mail: nico(at)fli4l(dot)de
Walter, Gerd (*UMTS*)
E-Mail: fli4l(at)hgwb(dot)de
Walter, Oliver (*QoS*)
E-Mail: owb(at)gmx(dot)de
Weiler, Manuela (*CD-Versand, Kassenwart*)
E-Mail: -
Weiler, Marcel (*Qualitätsmanagement*)
E-Mail: -

Das fli4l-Test- und Übersetzerteam bilden (in alphabetischer Reihenfolge):

Bußmann, Lars
Charrier, Bernard
Fischer, Joerg
Frauenhoff, Peter
Schliesing, Manfred
Schmitts, Jupp
Wolters, Forian

A.9.3. Entwickler- und Testteam (nicht mehr aktive)

Arndt, Kai-Christian (*USB*)
Behrends, Arno (*Support*)
Bork, Thomas (*lpdsrv*)
Bauer, Jürgen (*LCD-Package, fliwiz*)
Blokland, Kees (*Englische Übersetzung*)
Cerny, Carsten (*Webseite, fliwiz*)
Dawid, Oliver (*dhcp, uClibc*)
Ebner, Hannes (*QoS*)
Grabner, Hans-Joerg (*imonc*)
Grammel, Matthias (*Englische Übersetzung*)
Gruetzmacher, Tobias (*Mini-httpd, imond, proxy*)
Hahn, Joerg (*IPSEC*)
Hanselmann, Michael (*Mac OS X/Darwin*)
Hoh, Jörg (*Newsletter, NIC-DB, Veranstaltungen*)
Hornung, Nicole (*Verein*)
Horsmann, Karsten (*Mini-httpd, WLAN*)
Janus, Frank (*LCD*)
Kaiser, Gerrit (*Logo*)
Karner, Christian (*PPTP-Package*)
Klein, Marcus (*Problemfeedback*)
Lammert, Gerrit (*HTML-Dokumentation*)
Lanz, Ulf (*LCD*)
Lichtenfeld, Nils (*QoS*)
Neis, Georg (*fli4l-CD, Dokumentation*)
Peiser, Steffen (*FAQ*)
Peus, Christoph (*uClibc*)
Pohlmann, Thorsten (*Mini-httpd*)
Raschel, Tom (*IPX*)
Reinard, Louis (*CompactFlash*)
Resch, Robert (*PCMCIA, WLAN*)
Schäfer, Harald (*HDD-Support*)
Strigler, Stefan (*GTK-Imonc, Opt-DB, NG*)
Wolter, Jean (*Paketfilter, uClibc*)
Zierer, Florian (*Wunschliste*)

A.9.4. Sponsoren

Auch ist mittlerweile fli4l als Wort-/Bildmarke eingetragen. Folgende fli4l-Anwender (neben einigen, die nicht genannt werden wollten) haben geholfen das dafür nötige Geld zusammen zu bekommen:

Bebensee, Norbert
Becker, Heiko
Behrends, Arno
Böhm, Stefan
Brederlow, Ralf
Groot, Vincent de
Hahn, Olaf
Hogrefe, Paul
Holpert, Christian
Hornung, Nicole
Kuhn, Robert
Lehnen, Jens
Ludwig, Klaus-Ruediger
Mac Nelly, Christa
Mahnke, Hans-Jürgen
Menck, Owen
Mende, Stefan
Mücke, Michael
Roessler, Ingo
Schiele, Michael
Schneider, Juergen
Schönleber, Suitbert
Sennewald, Matthias
Sternberg, Christoph
Vollmar, Thomas
Walter, Oliver
Wiebel, Christian
Woelk, Fabian

Seit einiger Zeit hat fli4l nun auch seine eigenen Sponsoren, die mit Ihrer (Hardware-)Spende die Weiterentwicklung von fli4l unterstützen. Dabei handelt es sich um Adapter, CompactFlash und Ethernetkarten.

Hardwarespender (in alphabetischer Reihenfolge):

Baglatzis, Stephanos
Bauer, Jürgen
Dross, Heiko
Kappenhagen, Wenzel
Kipka, Joachim
Klopfer, Tom
Peiser, Steffen
Reichelt, Detlef
Reinard, Louis
Stärkel, Christopher

Weitere Sponsoren sind auf der fli4-Homepage gelistet:
http://www.fli4l.de/sonstiges/sponsoren/

A.10. Feedback

Kritik, Feedback und Zusammenarbeit ist jederzeit willkommen.

Die primäre Anlaufstelle dafür sind die fli4l-Newsgroups. Wer Probleme bei der Einrichtung eines fli4l-Routers hat, sollte sich erst FAQ, Howtos und NG-Archiv anschauen, bevor er sich an die Newsgroups wendet. Informationen über die verschiedenen Gruppen und die Netiquette findet man auf der fli4l-Webseite:

http://www.fli4l.de/hilfe/newsgruppen/
http://www.fli4l.de/hilfe/faq/
http://www.fli4l.de/hilfe/howtos/

Gerade weil für fli4l-Router meist ältere Hardware eingesetzt wird, kann es immer wieder mal zu Problemen kommen. Informationen können anderen fli4l-Usern bei Problemen mit der Hardware weiterhelfen, denn es gibt immer wieder Probleme mit den PC-Karten bzgl. I/O-Adressen, Interrupts und so weiter.

Auf der fli4l-Webseite wurde eine Netzwerkkarten-Datenbank eingerichtet, in die man z.B. die passenden Treiber für eine bestimmte Karte eintragen kann.

http://www.fli4l.de/hilfe/nic-db/

Viel Spaß mit fli4l!

B. Anhänge der optionalen Pakete

B.1. CHRONY - Benachrichtigung anderer Applikationen über Timewarps

Stellt chrony fest, dass die Uhr sehr weit von der tatsächlichen Uhrzeit abweicht, korrigiert chrony die Zeit in einem grossen Schritt und führt Scripts aus, um andere Anwednungen von diesem Zeitsprung zu informieren. Um z.B. den Imond von einem Zeitsprung zu informieren, macht chrony folgendes:

1. Scripte ins Archiv aufnehmen

   Chrony fügt dem Archiv zwei Files hinzu:

   ```
   start_imond  yes  etc/chrony.d/timewarp.sh mode=555 flags=sh
   start_imond  yes  etc/chrony.d/timewarp100.imond mode=555 flags=sh
   ```

   timewarp.sh führt alle Scripts im gleichen Verzeichnis aus, die dem Namen timewarp<3 ziffern>.<name> entsprechen.

2. Script zur Verfügung stellen

   chrony nimmt folgendes Script mit ins Archiv auf:

   ```
   # inform imond about time warp
   imond-stat "adjust-time $timewarp 1"
   ```

   Damit wird der imond über den Zeitsprung informiert und kann seine interne Zeitbasis anpassen.

B.2. DSL - PPPD und Active Filter

Für fli4l setzen wir den im Link angegebenen Ausdruck ein:

```
'outbound and not icmp[0] != 8 and not tcp[13] & 4 != 0'
```

und erreichen damit, dass grundsätzlich nur vom lokalen Netz ins Internet gesendete Pakete die Verbindung offen halten, mit ein paar Ausnahmen:

- *TCP-RST*: Antworten auf abgelehnte Verbindungswünsche von außen setzen den Timeout nicht zurück,
- *ICMP*: gesendete ICMP-Nachrichten setzen den Timeout ebenfalls nicht zurück, es sei denn, es wird ein Echo-Request gesendet.

Dieser Ausdruck wird vom PPPD in einen vom Kernel verwendbaren Paket-Filter umgesetzt. Dieser sieht in diesem Beispiel wie folgt aus:

```
#
# Expression: outbound and not icmp[0] != 8 and not tcp[13] & 4 != 0
#
(000) ldb      [0]
(001) jeq      #0x0             jt 17   jf 2
(002) ldh      [2]
(003) jeq      #0x21            jt 4    jf 18
(004) ldb      [13]
(005) jeq      #0x1             jt 6    jf 11
(006) ldh      [10]
(007) jset     #0x1fff          jt 18   jf 8
(008) ldxb     4*([4]&0xf)
(009) ldb      [x + 4]
(010) jeq      #0x8             jt 18   jf 17
(011) jeq      #0x6             jt 12   jf 18
(012) ldh      [10]
(013) jset     #0x1fff          jt 18   jf 14
(014) ldxb     4*([4]&0xf)
(015) ldb      [x + 17]
(016) jset     #0x4             jt 17   jf 18
(017) ret      #0
(018) ret      #4
```

B.3. DYNDNS

B.3.1. Hinzufügen von neuen Providern

Das Hinzufügen von neuen Providern ist eigentlich sehr leicht, da die Update-Scripts komplett von den Provider-Daten getrennt sind. Für einen neuen Provider müssen folgende Dateien angepasst werden:

Datei opt/etc/dyndns/provider.NAME

Dies ist die Datei, in der definiert wird, wie ein Update bei diesem speziellen Provider funktioniert. Meistens besteht sie nur aus einer Liste von Variablen, da es aber ein ganz normales Shell-Script ist, können hier auch komplexere Operationen durchgeführt werden, das sollte aber selten nötig sein. In dieser Datei können folgende Variablen benutzt werden:

$ip Die IP des Interfaces, das den dynamischen Hostnamen erhalten soll.

$host Der komplette Hostname, wie ihn der Benutzer in seiner Konfiguration angegeben hat.

$subdom Alle Komponenten des Hostnamen bis zum vorletzten Punkt (**name**.provider.dom)

$domain Die letzten beiden Komponenten des Hostnamen (name.**provider.dom**)

$provider Der symbolische Name des Providers, wie ihn der Benutzer in seiner Konfgurationsdatei angegeben hat.

$user Der Benutzername für diesen Dienst.

$pass Das dazugehörige Passwort.

Diese Variablen können zur klareren Abgrenzung gegenüber anderem Text mit geschweiften Klammern geschrieben werden, aus `$ip` wird z.B. `${ip}`. Bei Verwendung von Anführungszeichen ist zu beachten, das innerhalb von einfachen Anführungszeichen die oben genannten Variablen *nicht* expandiert werden, bei doppelten Anführungszeichen schon. Als Faustregel kann man also sagen: Immer einfache Anführungszeichen benutzen, aber sobald man Variablen benutzt, doppelte Anführungszeichen benutzen.

Die folgenden Variablen müssen in dieser Datei definiert werden, damit das Paket weiß, wie ein Update bei dem entsprechenden Provider funktioniert:

provider_update_type Dies bestimmt die Art der Anfrage, die an den Server des Providers geschickt wird. Momentan werden unterstützt:

- **http** Es wird automatisiert eine bestimmte Webseite des Providers abgerufen und so der DynDNS-Eintrag aktualisiert.
- **netcat** Es wird einfach ein bestimmter Text an den Server des Providers geschickt, der das Update auslöst.
- **gnudip** Ein relativ einfaches aber sicheres Updateverfahren, welches über zwei HTTP-Anfragen ausgeführt wird.

provider_host Der Hostname des Providers, der bei einem Update kontaktiert wird.

provider_port Der Port auf dem Host des Providers, der angesprochen werden soll. Der Standard-Port für HTTP ist 80.

Je nach Update-Typ müssen weitere Variablen angegeben werden:

HTTP provider_url Hier wird die relative URL (ohne Hostname, aber mit / am Anfang zum Script des Providers abgelegt. Für Beispiele bitte die Dateien der anderen Provider ansehen.

provider_auth (optional) Benötigt der Provider eine Anmeldung per Basic Authentication, so sind hier die entsprechenden Daten anzugeben. Das Format ist `"USER:PASSWORD"`.

Netcat provider_data Dies ist der Text, der an den Server des Providers geschickt wird. Siehe z.B. `provider.DYNEISFAIR`.

GNUDip provider_script Der Pfad zum GNUDip-Script auf dem Server, dies ist meist etwas wie z.B. `'/cgi-bin/gdipupdt.cgi'`.

Datei opt/dyndns.txt

Hier müssen eine oder mehr Zeilen für den neuen Provider eingefügt werden. Meistens reicht eine Zeile in der Art:

```
dyndns_%_provider   NAME   etc/dyndns/provider.NAME
```

Wird für den Provider HTTP und Basic Authentication benutzt, so braucht man noch das base64-Tool:

```
dyndns_%_provider   NAME   files/usr/local/bin/base64
```

Sollten noch andere Tools benötigt werden, bitte mir vorher eine Mail schicken, damit geprüft werden kann, ob das für das OPT_DYNDNS geeignet ist.

Datei check/dyndns.exp

In dieser Datei muss an der langen Zeile, die mit `DYNPROVIDER =` beginnt, der Providername mit einem senkrechten Strich von den anderen abgetrennt, hinten angefügt werden.

Datei doc/<Sprache>/tex/dyndns/dyndns_main.tex

In der Dokumentation einen neuen Abschnitt eintragen. Auch hier sind die Provider alphabetisch nach dem Kurznamen, den der Benutzer in der Config-Datei eingibt, sortiert. Das prov-Makro ist am Anfang der Datei dokumentiert, genug Beispiele sollten vorhanden sein.

B.3.2. Dank

Als allererstes möchten wir dem danken, der dieses Paket ins Leben gerufen hat und lange Zeit dieses Paket betreut hat: Thomas Müller (E-Mail: opt_dyndns@s2h.cx) hat hier hervorragende Arbeit geleistet, ohne ihn wäre das Paket in der heutigen Form nicht möglich gewesen.

Desweiteren möchten wir auch Marcel Döring (E-Mail: m@rcel.to) danken, der das Paket einige Zeit lang gepflegt hat.

Bei der Entwicklung des Paketes haben sehr viele Leute geholfen und Ideen beigetragen. Mein Dank gilt allen diesen fleißigen Helfern.

Außerdem danken wir Frank Meyer und dem Rest des fli4l-Tems für ihre unermüdliche Arbeit, um den besten Router der Welt zu basteln (Bitte nicht ganz Ernst nehmen ;-).

Weiterhin möchten wir folgenden Leuten danken, die sich mit Tips, neuen Providern, Fehlerberichten, etc. an dem Paket beteiligt haben:

- Paul Bischof für den Provider AFRAID.
- Jens Fischer schrieb das Paket opt_dtdns, welches mich erst auf die Idee brachte, ein Paket für DynDNS.org zu schreiben.
- Till Jäger schrieb das Paket opt_cjb, welches in in opt_dyndns übernommen habe.
- Tobias Gruetzmacher hat auf http://portfolio16.de/index.de Informationen zu weiteren DynDNS-Anbietern zusammengetragen, die hier verwendet werden.
- Die Anbieter dynamischer DNS, die auf ihren Webseiten zum Teil sehr gute, zum Teil weniger gute Beschreibungen des zu verwendenden Protokolls veröffentlicht haben.
- Die Programmierer diverser Update-Programme für DynDNS Anbieter, aus deren Code schamlos geklaut wurde. ;-)
- Heiko Ambos von dynaccess.de hat mich bei der Entwicklung der Unterstützung für diesen Anbieter unterstützt.

- Dennis Neuhäuser, der die Idee hatte, die Antworten der Dienste per Webserver verfügbar zu machen statt sie auf der Konsole auszugeben und auch gleich eine erste Implementation dafür geschickt hat.
- Lars Winkler der freundlicherweise die Änderungen, um das Paket unter 2.0pre2 zum Laufen zu bringen zur Verfügung gestellt hat.
- Markus Kraft und Tobias Gruetzmacher haben die Grundlage für die Anpassung an fli4l 2.0 gelegt.
- Georg Bärwald für die Daten zu Selfhost.de
- Mark C. Storck für die Daten zu Storck.org
- Arne Biermann für den Hinweis auf den Anbieter hn.org
- Detlef Paschke für die Daten zu dyn.ee und dyndns.dk
- Martin Kisser für seine Idee zum Vermeiden von Updates, wenn die IP sich nicht geändert hat.
- Björn Hoffmann für die Daten von DnsArt.com
- Christian Busch für die Daten von no-ip.com.
- Ralf Gill für das Update der Daten von selfhost.de.
- Michael (HeinB) für eine weitere Möglichkeit sich mit fli4l selbst in den Fuss zu schiessen. ;-)
- Marcus Mönnig, dito.

B.3.3. Lizenz

Der Text der GNU General Public License ist auch im Internet unter http://www.gnu.org/licenses/gpl.txt veröffentlicht. Eine inoffizielle deutsche Übersetzung findet sich unter http://www.gnu.de/documents/gpl.de.html Diese Übersetzung soll jedoch nur zu einem besseren Verständnis der GPL verhelfen, rechtsverbindlich ist die englischsprachige Version.

B.4. EASYCRON - Crontab in der Boot-Phase ergänzen

Hinweis: *Die folgenden Ausführungen richten sich nur an Entwickler von Opt-Paketen für den fli4l Router.*

Ab Version 2.1.7 stellt das rc-Script von Easycron die Funktion `add_crontab_entry()` zur Verfügung. Durch Aufruf dieser Funktion können andere rc-Scripts ab Startposition rc101 bis Startposition 949 Einträge an die Crontab anhängen. Am Ende der Boot-Phase mit dem Starten des cron Daemon sind die zusätzlichen Einträge aktiv.

`add_crontab_entry time command [custom]`

Mit `time` wird die Ausführungszeit in cron Notation übergeben, siehe die Manpage zu crontab(5) (http://linux.die.net/man/5/crontab). `command` enthält den auszuführenden Befehl. Der dritte Parameter `custom` ist optional. Mit ihm können Umgebungsvariablen passend zum Befehl gesetzt werden. Soll mehr als eine Variable gesetzt werden, sind die Zuweisungen durch \\ zu trennen. Bitte **nicht** die Umgebungsvariable `PATH` verändern, da sonst nachfolgende crontab Einträge nicht mehr korrekt abgarbeitet werden können.

```
#
# example I: normal use, 2 parameters
#
     crontime="0 5 1 * *"
     croncmd="rotate_i_log.sh"

     add_crontab_entry "$crontime" "$croncmd"

#
# end of example I
#

#
# example II: extended use, 3 parameters and
#                           multiple environment values
#
     croncustom="source=/var/log/current \\ dest=/mnt/data/log"
     croncmd='cp $source $dest-`date +\%Y\%m\%d`; > $source'
     crontime="59 23 * * *"

     add_crontab_entry "$crontime" "$croncmd" "$croncustom"

#
# end of example II
#
```

Die Richtigkeit der Einträge muß das aufrufende Script sicherstellen.

B.5. HD - Fehler im Zusammenhang mit Festplatten/CompactFlashs

Problem:

- der Router erkennt die Festplatte überhaupt nicht.

Mögliche Ursachen:

- über `OPT_HDDRV` müssen eventuell zusätzliche Treiber für den HD-Controller definiert werden
- Platte ist falsch im BIOS eingetragen
- der Controller ist defekt oder abgeschaltet
- es wird bei der Installation die falsche Platte angegeben
- der Controller wird nicht von fli4l unterstützt. Manche Controller benötigen spezielle Treiber, die in fli4l nicht enthalten sind

Problem:

- die Installation bricht ab
- nach einem Remote-Update des opt-Archives bootet der Router nicht mehr
- es gibt Fehlermeldungen beim Partitionieren oder Formatieren der Festplatte

Mögliche Ursachen:

- bei IDE-Festplatten könnte es an zu langen oder ungeeigneten IDE-Kabeln liegen
- bei älteren Festplatten ist die Einstellung der Transferrate/PIO-Mode im Bios oder auf dem Controler evtl. zu schnell für die Platte.
- ungeeigneter Chipsatz

Bemerkungen:

- bei Problemen mit den DMA-Einstellungen kann man versuchen im Paket base die Einstellung `LIBATA_NODMA='no'` zu setzen. (Der Standardwert ist hier 'yes'). Dies aktiviert DMA-Zugriffe an ATA Geräten.

Problem:

- nach der Installation bootet fli4l nicht von Festplatte

Mögliche Ursache:

- wenn der Bootvorgang von einem CF-Modul fehlschlägt sollte man prüfen ob das CF-Modul im Bios mit LBA oder LARGE erkannt wurde. Die richtige Einstellung für Module unter 512MB ist NORMAL oder CHS.

- es wird ein Adaptec 2940 Controller mit altem BIOS eingesetzt und das erweiterte Mapping für Festplatten über 1GB ist aktiv. Als Abhilfe kann man das BIOS des SCSI-Controllers aktualisieren oder das Mapping umschalten. **Beim Umschalten des Mappings gehen alle Daten auf der Platte verloren!**

Problem:

- Windows sagt während des Erstellens einer CF-Card: „Medium im Laufwerk (X:) besitzt kein FAT. [Abbruch]“.

Mögliche Ursache:

- Die Compactflash wurde zu früh / ohne Abmeldung aus dem Reader entfernt. Windows hatte den letzten Schreibvorgang noch nicht abgeschlossen, das Dateisystem ist nun beschädigt. Erstelle die CompactFlash nochmals direkt am fli4l mittels HD-Install.

B.6. HTTPD

B.6.1. Zusätzliche Einstellungen

Diese Einstellungen stehen normalerweise nicht in der Konfigurationsdatei, müssen also hinzugefügt werden, wenn sie benötigt werden.

HTTPD_USER Mit dieser Option ist es möglich, den Webserver mit den Rechten eines anderen Benutzers als „root” laufen zu lassen. Dies ist besonders sinnvoll, wenn der Webserver benutzt wird, um andere Seiten als das Admin-Interface bereitzustellen. Achtung: Es kann sein, dass einige Scripts, die Zugriff auf Konfigurationsdateien brauchen, dann nicht mehr laufen. Die Standard-Scripts dieses Pakets laufen unter jedem Benutzer.

B.6.2. Allgemeine Bemerkungen

Wenn man TELMOND installiert hat, werden auf der Status- und der Calls- Seite die Telefonnummern der Anrufer angezeigt. Eine Namenszuordnung lässt sich in der Datei opt/etc/phonebook vornehmen. Diese Datei hat das gleiche Format wie die Telefonnummerndatei vom IMONC. Es können also Telefonbücher zwischen IMONC und Router ausgetauscht werden. Das Format jeder Zeile ist dabei “Telefonnummer=Name[,WAV-Datei]” (ohne die Anführungszeichen). Die WAV-Datei wird aber nur vom IMONC benutzt und vom Webserver ignoriert.

Das komplette Webinterface ist seit der Version 2.1.12 auf ein Framefreies Design mit CSS umgestellt worden. Alte Browser könnten damit Probleme haben. Allerdings hat das den Vorteil, dass man das Aussehen der Oberfläche fast beliebig verändern kann, einfach indem man die CSS-Dateien (im wesentlichen /opt/srv/www/css/main.css) anpasst.

Das Webserver-Paket wurde von Thorsten Pohlmann (E-Mail: pohlmann@tetronik.com) erstellt und wird zur Zeit von Tobias Gruetzmacher (E-Mail: fli4l@portfolio16.de) gepflegt. Das neue Design (seit der Version 2.1.12) wurde von Helmut Hummel (E-Mail: hh@fli4l.de) realisiert.

B.7. HWSUPP - Geräteabhängige Einstellungen

B.7.1. Verfügbare LED-Devices

Je nach `HWSUPP_TYPE` sind verschiedene LED-Devices verfügbar. Bei nicht aufgeführter Hardware sind die PC-Tastatur LED's wie bei generic-pc verfügbar.

Zusätzlich LED-Devices können z.B. auf WLAN-Karten verfügbar sein. Die gültigen Namen der LED-Devices ermittelt man mittels Eingabe von `ls /sys/class/leds/` z.B. per ssh auf der Router-Console.

sim

LED Simulation, erzeugt Eintrag im `syslog`:

- `simu::1`
- ...
- `simu::8`

generic-pc

PC-Tastatur LED's:

- `keyboard::scroll`
- `keyboard::caps`
- `keyboard::num`

generic-acpi

PC-Tastatur-LED's, wie generic-pc

pcengines-alix

- `alix::1`
- `alix::2`
- `alix::3`

pcengines-apu

- `apu::1`
- `apu::2`
- `apu::3`

pcengines-wrap

- `wrap::1`
- `wrap::2`
- `wrap::3`

soekris-net4801

- `net48xx::error`

soekris-net5501

- `net5501::error`

B.7.2. Verfügbare Button-Devices

Je nach `HWSUPP_TYPE` sind verschiedene GPIO-Devices für Taste vorbelegt.

pcengines-alix

- `gpio::24`

pcengines-apu

- `gpio::252`

pcengines-wrap

- `gpio::40`

soekris-net5501

- `gpio::25`
  Der Taster ist am soekris Gehäuse mit 'Reset' beschriftet.
  Achtung: der Taster muss im BIOS freigeschaltet werden.

B.7.3. Hinweise zu spezieller Hardware

pcengines-alix

Beim Alix führt ein fehlerhafter Treiber für den lm90 Temperatursensor nach einiger Zeit zum Ausfall der Temperaturanzeige.

Als Workaraound wird der lm90 Treiber entladen und wieder neu geladen. Dies geschieht automatisch per cron-Job. Dazu muss das Paket easycron geladen werden (`OPT_EASYCRON='yes'`).

B.8. HWSUPP - Konfigurations-Beispiele

B.8.1. generic-pc

```
OPT_HWSUPP='yes'
HWSUPP_TYPE='generic-pc'

HWSUPP_WATCHDOG='no'
HWSUPP_CPUFREQ='no'

HWSUPP_LED_N='3'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='keyboard::num'
HWSUPP_LED_2='online'
HWSUPP_LED_2_DEVICE='keyboard::caps'
HWSUPP_LED_3='wlan'
HWSUPP_LED_3_DEVICE='keyboard::scroll'
HWSUPP_LED_3_WLAN='wlan0'

HWSUPP_BUTTON_N='0'
```

B.8.2. pcengines-apu

```
OPT_HWSUPP='yes'
HWSUPP_TYPE='pcengines-apu'

HWSUPP_WATCHDOG='yes'
HWSUPP_CPUFREQ='yes'
HWSUPP_CPUFREQ_GOVERNOR='ondemand'

HWSUPP_LED_N='3'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='apu::1'
HWSUPP_LED_2='wlan'
HWSUPP_LED_2_DEVICE='apu::2'
HWSUPP_LED_2_WLAN='wlan0'
HWSUPP_LED_3='online'
HWSUPP_LED_3_DEVICE='apu::3'

HWSUPP_BUTTON_N='1'
HWSUPP_BUTTON_1='wlan'
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_1_PARAM='wlan0'
```

B.8.3. pcengines-apu mit GPIO's

```
OPT_HWSUPP='yes'
```

```
HWSUPP_TYPE='pcengines-apu'

HWSUPP_WATCHDOG='yes'
HWSUPP_CPUFREQ='yes'
HWSUPP_CPUFREQ_GOVERNOR='ondemand'

HWSUPP_LED_N='5'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='apu::1'
HWSUPP_LED_2='wlan'
HWSUPP_LED_2_DEVICE='apu::2'
HWSUPP_LED_2_WLAN='wlan0'
HWSUPP_LED_3='online'
HWSUPP_LED_3_DEVICE='apu::3'
HWSUPP_LED_4='trigger'
HWSUPP_LED_4_PARAM='phy0rx'
HWSUPP_LED_4_DEVICE='gpio::237'
HWSUPP_LED_5='trigger'
HWSUPP_LED_5_PARAM='phy0tx'
HWSUPP_LED_5_DEVICE='gpio::245'

HWSUPP_BUTTON_N='2'
HWSUPP_BUTTON_1='wlan'
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_1_PARAM='wlan0'
HWSUPP_BUTTON_2='online'
HWSUPP_BUTTON_2_DEVICE='gpio::236'
```

B.9. HWSUPP - Blinkfolge

Die folgenden Blinkfolgen werden während des Bootvorgangs angezeigt:

Während der Abarbeitung von `rc002.*` bis `rc250.*` wird die erste Folge angezeigt (1 * Blinken - Pause),
von `rc250.*` bis `rc500.*` die zweite (2 * Blinken - Pause),
von `rc500.*` bis `rc750.*` die 3. und
von `rc750.*` bis zum Ende des Bootvorgangs die 4. Folge (Dauerblinken).

B.10. HWSUPP - Hinweise für Paket-Entwickler

Im folgenden ist beschrieben was ein Paket-Entwickler zu tun hat, um Button- oder LED-Funktionalität zu einem Paket hinzuzufügen[1].

B.10.1. LED-Erweiterungen

LED-Typ

In der Datei `check/myopt.exp` wird die Liste der erlaubten LED-Typen die in `HWSUPP_LED_x` eingetragen werden können erweitert.

Beispiel:

```
+HWSUPP_LED_TYPE(OPT_MYOPT) = 'myopt'
                            : ', myopt'
```

Parameterprüfung

In der Datei `check/myopt.ext` werden die Parameter die für den neuen LED-Typen in `HWSUPP_LED_x_PARAM` eingetragen werden können geprüft.

Beispiel:

```
if (opt_hwsupp)
then
    depends on hwsupp version 4.0

    foreach i in hwsupp_led_n
    do
        set action=hwsupp_led_%[i]
        set param=hwsupp_led_%_param[i]
        if (action == "myopt")
        then
            if (!(param =~ "(RE:MYOPT_LED_PARAM)"))
            then
                error "When HWSUPP_LED_\${i}='myopt', ...
                       must be entered in HWSUPP_LED_\${i}_PARAM"
            fi
        fi
    done
fi
```

LED schalten

Um eine LED zu schalten ist in einem eigenen Skript (z.B. `/usr/bin/<opt>_setled`) das Kommando `/usr/bin/hwsupp_setled <LED> <Status>/` aufzurufen.

Die LED-Nummer kann aus `/var/run/hwsupp.conf>` ausgelesen werden.

Als Status ist `off`, `on` oder `blink` zu übergeben.

Beispiel:

[1] Wenn man im WLAN-Paket nach `##HWSUPP##` sucht so findet man die anzupassenden Stellen.

```
if [ -f /var/run/hwsupp.conf ]
then
    . /var/run/hwsupp.conf
    [ 0$hwsupp_led_n -eq 0 ] || for i in `seq 1 $hwsupp_led_n`
    do
        eval action=\$hwsupp_led_${i}
        eval param=\$hwsupp_led_${i}_param
        if [ "$action" = "<opt>" ]
        then
            if [ <myexpression> ]
            then
                /usr/bin/hwsupp_setled $i on
            else
                /usr/bin/hwsupp_setled $i off
            fi
        fi
    done
fi
```

Den aktuellen Zustand einer LED kann man mit `/usr/bin/hwsupp_getled <LED>/` abfragen. Es wird je nach Status `off`, `on` oder `blink` ausgegeben.

B.10.2. Button-Erweiterungen

B.10.3. Button-Aktion

In der Datei `check/myopt.exp` wird die Liste der erlaubten Button-Typen die in `HWSUPP_BUTTON_x` eingetragen werden können erweitert.

Beispiel:

```
+HWSUPP_BUTTON_TYPE(OPT_MYOPT) = 'myopt'
                               : ', myopt'
```

Parameterprüfung

In der Datei `check/myopt.ext` werden die Parameter, die für den neuen Button-Typen in `HWSUPP_BUTTON_x_PARAM` eingetragen werden können, geprüft.

Beispiel:

```
if (opt_hwsupp)
then
    depends on hwsupp version 4.0

    foreach i in hwsupp_button_n
    do
        set action=hwsupp_buttonn_%[i]
        set param=hwsupp_button_%_param[i]
        if (action == "myopt")
        then
```

```
            add_to_opt "files/usr/bin/myopt_keyprog" "mode=555 flags=sh"
            if (!(param =~ "(RE:MYOPT_BUTTON_PARAM)"))
            then
                error "When HWSUPP_BUTTON_\${i}='myopt', ...
                       must be entered in  HWSUPP_BUTTON_\${i}_PARAM"
            fi
        fi
    done
fi
```

Button-Funktion

Wenn eine Taste gedrückt wird, wird die Datei `/usr/bin/myopt_keyprog` ausgeführt.
Als Parameter wird er Inhalt von `HWSUPP_BUTTON_x_PARAM` übergeben
Beispiel:

```
##TODO## example
```

B.11. IPV6 - Anbindung ans IPv6-Internet mit Hilfe eines SixXS-Tunnels

In diesem Abschnitt wird beschrieben, wie das IPv6-Paket genutzt werden kann, um mit Hilfe eines Tunnels des Anbieters SixXS (https://www.sixxs.net/) das eigene Heimnetzwerk mit dem IPv6-Internet zu verbinden.

B.11.1. Account erstellen

Zuerst muss ein SixXS-Account unter “Signup for new users” beantragt werden. Hat man diese Hürde genommen, besitzt man einen Benutzernamen der Form YYYYY-SIXXS sowie ein zugehöriges Passwort. Diese Daten benötigt man später für die Tunnelkonfiguration.

B.11.2. Tunnel konfigurieren

Vorbereitungen

Doch vorher muss man den Tunnel beantragen. Dies geschieht nach der Anmeldung über den Menüpunkt “Request tunnel”. Hier ist es wichtig, beim Tunneltyp den zweiten Eintrag, “Dynamic IPv4 Endpoint using Heartbeat protocol” auszuwählen, weil diese Konfiguration vom fli4l direkt unterstützt wird. Die dritte Variante, “Static IPv4 Endpoint”, ist ebenfalls möglich, wenn man eine fest zugeordnete IPv4-Adresse sein Eigen nennt, die sich nie ändert. Die Tunnelvariante “Dynamic NAT-traversing IPv4 Endpoint using AYIYA” wird derzeit vom IPv6-Paket nicht unterstützt.

Sobald man in den anderen Feldern Angaben zum Ort des Routers gemacht und via “Next step” zur zweiten Seite gewechselt hat, stehen hier ein oder mehrere PoPs (Points of Presence) zur Auswahl, die später für den Tunnelaufbau wichtig sind. Man sollte denjenigen nehmen, der am dichtesten ist, um das Tunneln von IPv6-Paketen möglichst effizient zu gestalten.

Sind alle Angaben gemacht und via “Place request for new Tunnel” abgeschickt worden, kommt irgendwann darauf eine E-Mail mit den nötigen Tunneldaten an. Dazu gehören:

1. die Identifikationsnummer des Tunnels (T...)
2. der Name des zugeordneten PoPs
3. die IPv4-Adresse des zugeordneten PoPs (“SixXS IPv4”)
4. die lokale IPv6-Adresse des Tunnels inklusive Subnetzmaske (bei SixXS typischerweise /64), also die Adresse des Routers (“Your IPv6”)
5. die entfernte IPv6-Adresse des Tunnels inklusive Subnetzmaske (die mit der Subnetzmaske der lokalen IPv6-Adresse identisch ist), d.h. die Adresse des PoPs (“SixXS IPv6”)

Konfiguration

Jetzt kann der Tunnel konfiguriert werden! Als erstes wird die Variable `IPV6_TUNNEL_N` auf “1” gesetzt, weil genau ein Tunnel aufgebaut werden soll:

```
IPV6_TUNNEL_N='1'
```

Die SixXS-Angaben werden wie folgt in der IPv6-Konfiguration vermerkt:

1. Die Identifikationsnummer des Tunnels wird in `IPV6_TUNNEL_1_TUNNELID` eingetragen.
2. entfällt (der Name des PoPs ist uninteressant)
3. Die IPv4-Adresse des PoPs wird in der Variable `IPV6_TUNNEL_1_REMOTEV4` vermerkt.
4. Die lokale IPv6-Adresse des Tunnels landet *inklusive* Subnetzmaske in der Variable `IPV6_TUNNEL_1_LOCALV6`.
5. Die entfernte IPv6-Adresse des Tunnels landet *ohne* Subnetzmaske in der Variable `IPV6_TUNNEL_1_REMOTEV6`.

Zusätzlich müssen Benutzername und Passwort in der Tunnelkonfiguration in den Variablen `IPV6_TUNNEL_1_USERID` und `IPV6_TUNNEL_1_PASSWORD` angegeben werden. Schließlich muss in der Variable `IPV6_TUNNEL_1_TYPE` vermerkt werden, dass der konfigurierte Tunnel ein SixXS-Tunnel ist:

```
IPV6_TUNNEL_1_TYPE='sixxs'
```

Hat man von SixXS den PoP “deham01” mit der IPv4-Adresse `212.224.0.188` sowie die Tunnelendpunkte `2001:db8:900:551::1/64` (entfernt) und `2001:db8:900:551::2/64` (lokal) erhalten, und lauten die Tunnel-ID “T1234”, der Benutzername “USER1-SIXXS” und das Passwort “sixxs” (bitte dieses Passwort *nicht* verwenden!), dann sieht die resultierende Konfiguration so aus:

```
IPV6_TUNNEL_N='1'
IPV6_TUNNEL_1_LOCALV4='dynamic' # oder feste lokale IPv4-Adresse
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
IPV6_TUNNEL_1_LOCALV6='2001:db8:900:551::2/64'
IPV6_TUNNEL_1_REMOTEV6='2001:db8:900:551::1'
IPV6_TUNNEL_1_TYPE='sixxs'
IPV6_TUNNEL_1_USERID='USER1-SIXXS'
IPV6_TUNNEL_1_PASSWORD='sixxs'
IPV6_TUNNEL_1_TUNNELID='T1234'
```

Test

Hat man dies geschaffft, dann kann man nach Aktualisierung der Konfiguration den fli4l-Router neu starten. Nach dem Einloggen auf dem Router (direkt oder z.B. per SSH) sollte man den Tunnelendpunkt bereits anpingen können. Das Ganze sieht dann mit den oben genannten Beispiel-Daten folgendermaßen aus:

```
garm 3.6.0-revXXXXX # ping -c 4 2001:db8:900:551:0:0:0:1
PING 2001:db8:900:551::1 (2001:db8:900:551::1): 56 data bytes
64 bytes from 2001:db8:900:551::1: seq=0 ttl=64 time=67.646 ms
64 bytes from 2001:db8:900:551::1: seq=1 ttl=64 time=72.001 ms
64 bytes from 2001:db8:900:551::1: seq=2 ttl=64 time=70.082 ms
64 bytes from 2001:db8:900:551::1: seq=3 ttl=64 time=67.996 ms

--- 2001:db8:900:551::1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 67.646/69.431/72.001 ms
```

Wichtig ist in der vorletzten Zeile der Teil "0% packet loss", d.h. dass für alle PING-Pakete Antwortpakete empfangen wurden. Kommt keine Antwort vom anderen Ende des Tunnels, sieht das Ergebnis anders aus:

```
garm 3.6.0-revXXXXX # ping -c 4 2001:db8:900:551:0:0:0:1
PING 2001:db8:900:551::1 (2001:db8:900:551::1): 56 data bytes

--- 2001:db8:900:551::1 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

Hier fällt auf, dass für kein einziges PING-Paket eine Antwort empfangen wurde ("100% packet loss"). Das bedeutet, dass entweder die Konfiguration nicht korrekt ist, oder dass der Tunnel seitens SixXS noch nicht vollständig eingerichtet worden ist. Im zweiten Fall sollte man erst einige Zeit abwarten, weil die Konfiguration auf Seiten des PoPs durchaus ein paar Stunden dauern kann. Hat man die Konfiguration doppelt und dreifach geprüft und keinen Fehler entdeckt, und ist einige Zeit verstrichen, ohne dass der Tunnel funktioniert, sollte man sich per E-Mail an SixXS wenden und das Problem möglichst detailliert beschreiben.

B.11.3. Subnetz konfigurieren

Vorbereitungen

Funktioniert der Tunnel, hat man den ersten großen Punkt geschafft. Fertig ist man aber noch nicht. Denn nun hat der Router zwar die Möglichkeit, Pakete ins IPv6-Internet zu schicken und von dort zu empfangen, aber die Hosts im lokalen Netz noch nicht. Dafür muss erst ein IPv6-Subnetz konfiguriert werden, innerhalb dessen die Hosts eingebunden werden.

Hier fällt ein kleiner, aber wesentlicher Unterschied zur Konfiguration eines IPv4-Netzwerks auf: Wegen der Adressknappheit ist in der Regel nur ein Host direkt mit dem Internet verbunden. Die anderen Hosts im lokalen Netz erhalten nur netzinterne, d.h. nicht nach außen geroutete Adressen aus den Bereichen `192.168.*.*`, `172.16.*.*` bis `172.31.*.*` sowie `10.*.*.*`, je nach Größe des Subnetzes.[2] Bei IPv6 gibt es Adressen in Hülle und Fülle, somit entfällt

[2] siehe RFC 1918 (http://tools.ietf.org/html/rfc1918) für Details

die Notwendigkeit, netzinterne Adressen zu verwenden. Wegen des globalen Charakters lokaler Subnetze muss jedoch sichergestellt werden, dass die Adressen lokaler Hosts nicht mit anderen Adressen im Internet kollidieren. Deshalb muss ein Subnetz vom IPv6-Anbieter zugeteilt werden, um solche Kollisionen zu vermeiden.

Bei SixXS geschieht dies mit dem Menüpunkt "Request subnet". Hier muss man hauptsächlich den zu verwendenden Tunnel angeben, was leicht ist, weil bisher nur einer konfiguriert worden ist. Nach dem Abschicken des Formulars via "Place request for new subnet" erhält man nach einiger Zeit eine E-Mail mit den folgenden Informationen:

1. Die IPv6-Adresse des Subnetzes inklusive Subnetzmaske ("Subnet IPv6")
2. Die IPv6-Adresse des Routers im Tunnel, wohin das Subnetz seitens SixXS geroutet wird ("Routed to")
3. Die IPv4-Adresse des Routers ("Your IPv4")[3]

Diese Daten reichen aus, beim fli4l jetzt ein eigenes IPv6-Subnetz zu konfigurieren. Allerdings muss man eines noch wissen: Das zugewiesene Subnetz ist in der Regel sehr groß. SixXS teilt für gewöhnlich /48er-Subnetze zu, d.h. innerhalb der 128 Bit langen IPv6-Adresse belegt der Anteil, der auf das Netzpräfix fällt, 48 Bit, und der Anteil, der für die Adressierung der Hosts zur Verfügung steht, beträgt 128 - 48 = 80 Bit. Ein solch großes Subnetz hat jedoch zwei große Nachteile. Der erste Nachteil ist die schiere Größe: Man kann in dem Netz $2^{80} \approx 1209$ Trilliarden Hosts unterkriegen. Das zu verwalten, ohne eine weitere Struktur auf dem Hosts-Anteil der Adresse zu nutzen, erscheint nicht ratsam. Der zweite Nachteil wiegt schwerer: Innerhalb eines solch großen Subnetzes funktioniert die so genannte *IPv6-Autokonfiguration* nicht mehr. Das ist ein Vorgang, bei dem ein IPv6-Host über bestimmte Protokolle das Subnetz-Präfix erhält und sich seine IPv6-Adresse mit Hilfe der MAC-Adresse seines Netzwerk-Adapters zurechtbastelt. Die MAC-Adresse besteht aus sechs Bytes, und mit Hilfe des Standards EUI-64 kann man sie auf acht Bytes strecken. Das entspricht 64 Bit, und dann ist Schluss. Für 80 Bit stehen einfach nicht genügend Informationen seitens des Hosts zur Verfügung.

Lange Rede, kurzer Sinn: Das Subnetz muss kleiner gemacht werden, und zwar muss, damit später die Autokonfiguration auch funktionieren kann, daraus ein /64er-Subnetz werden. Das ist ganz einfach: Die Subnetzmaske wird einfach zu /64 abgeändert. Wurde also von SixXS z.B. das Subnetz `2001:db8:123::/48` zugewiesen, dann ist das Subnetz, für das der fli4l konfiguriert werden soll, einfach `2001:db8:123::/64`. Das bedeutet im Detail, dass das /48-Subnetz in $2^{(64-48)} = 2^{16} = 65536$ Sub-Subnetze eingeteilt wird, von denen das erste mit der Nummer Null vom fli4l verwendet werden soll. Dazu muss man sich erinnern, dass die Kurzform `2001:db8:123::` eigentlich für die Adresse `2001:db8:123:0:0:0:0:0` steht, wobei die ersten drei Zahlen die vom IPv6-Anbieter global eindeutig vergebenen Teile des Subnetzes sind, die vierte Zahl das eigens ausgesuchte Sub-Subnetz "Null" darstellt,[4] und die letzten vier Zahlen für den Host-Anteil reserviert sind. Das ergibt dann zwar immer noch ein riesiges (Sub-)Subnetz, in dem bis zu $2^{64} \approx 18,4$ Trillionen Hosts untergebracht werden können. Dank der IPv6-Autokonfiguration kommt man aber mit den tatsächlichen Adressen nicht in Berührung. Und das ist gut so...

[3] falls der Router die IPv4-Adresse dynamisch erhält, steht hier "heartbeat"
[4] Man kann natürlich sich für ein anderes Sub-Subnetz entscheiden!

Konfiguration

Zurück zur Konfiguration! Als erstes wird die Variable `IPV6_NET_N` auf "1" gesetzt, weil genau ein lokales IPv6-Subnetz eingerichtet werden soll. Die IPv6-Adresse des /64-Subnetzes landet inklusive Subnetzmaske in der Variable `IPV6_NET_1`. Doch das ist nicht ganz richtig: Vielmehr steht hier die IPv6-Adresse *des Routers innerhalb dieses Subnetzes*, aber *ohne* das Subnetz-Präfix, das dem Tunnel zugeordnet ist. Das wird nämlich an anderer Stelle konfiguriert, und zwar in der Tunnelkonfiguration. Dort muss nun die Variable `IPV6_TUNNEL_1_PREFIX` auf das angoforderte Subnetz-Präfix gesetzt werden.

Hat man nun das /48er-IPv6-Subnetz `2001:db8:123::/48` von SixXS erhalten, daraus das Subnetz mit der Nummer '456' als zu verwendendes /64er-Sub-Subnetz ausgewählt und schließlich bestimmt, dass der Router innerhalb dieses Subnetzes die Adresse "1" erhalten soll, dann erhalten wir die folgende Konfiguration:

```
IPV6_NET_N='1'
IPV6_NET_1='0:0:0:456::1/64'              # IPv6-Adresse des Routers (ohne
                                          # Subnetz-Präfix) + Subnetzmaske
IPV6_TUNNEL_1_PREFIX='2001:db8:123::/48' # /48-Subnetz-Präfix
```

Zu beachten ist, dass die ersten drei Nullen in `IPV6_NET_1` quasi den Platz für das dem Tunnel zugeordnete /48-Subnetz-Präfix freihalten. Zusammen mit dem /48-Subnetz-Präfix, der vom Tunnelanbieter zugewiesen wird, ergeben sich das /64-Subnetz `2001:db8:123:456::/64` und die IPv6-Routeradresse `2001:db8:123:456::1`.

Jetzt fehlt noch der Name der Netzwerkschnittstelle, an die dieses Subnetz gebunden werden soll. Jedes Subnetz wird genau einer Netzwerkschnittstelle zugeordnet. Falls sich nur eine konfigurierte Netzwerkkarte im Router befindet, so lautet der Name der Netzwerkschnittstelle typischerweise "eth0" für kabelgebundene oder "wlan0" für drahtloses Adapter. Schauen Sie im Zweifelsfall einfach die Einstellung für `IP_NET_1_DEV` ("IP" ohne "6") und kopieren Sie den Inhalt einfach herüber:

```
IPV6_NET_1_DEV='eth0' # Netzwerkschnittstelle für dieses IPv6-Subnetz
```

Schließlich müssen wir nur noch alle Hebel der IPv6-Autokonfiguration ziehen:

```
IPV6_NET_1_ADVERTISE='yes'     # /64-Subnetz-Präfix und Default-Route per RA
IPV6_NET_1_ADVERTISE_DNS='yes' # DNS-Server per RA (erfordert
                               # DNS_SUPPORT_IPV6='yes'!)
IPV6_NET_1_DHCP='yes'          # Domänen-Name und DNS-Server per DHCPv6
                               # (letzteres erfordert DNS_SUPPORT_IPV6='yes')
```

Die beiden letzten Einstellungen sind nicht zwingend notwendig für ein funktionierendes IPv6-Subnetz, sind aber ganz hilfreich. Sie dienen der Verbreitung zusätzlicher Informationen im IPv6-Subnetz, nämlich der IPv6-Adresse des DNS-Servers sowie der verwendeten Domäne. Den DNS-Server kann man sogar auf zweierlei Weise veröffentlichen. Weil verschiedene Systeme hier unterschiedliche Vorlieben an den Tag legen, ist es von Vorteil, beide Verfahren (RDNSS via Router Advertisements sowie DHCPv6) zu aktivieren.

Test

Die gesamte IPv6-Konfiguration dieses Beispiels (`DNS_SUPPORT_IPV6='yes'` wird dabei angenommen!) lautet:

```
IPV6_NET_N='1'
IPV6_NET_1='0:0:0:456::1/64'   # IPv6-Adresse des Routers (ohne
                               # Subnetz-Präfix) + Subnetzmaske
IPV6_NET_1_DEV='eth0'          # Netzwerkschnittstelle für dieses IPv6-Subnetz
IPV6_NET_1_ADVERTISE='yes'     # /64-Subnetz-Präfix und Default-Route per RA
IPV6_NET_1_ADVERTISE_DNS='yes' # DNS-Server per RA
IPV6_NET_1_DHCP='yes'          # Domänen-Name und DNS-Server per DHCPv6

IPV6_TUNNEL_N='1'
IPV6_TUNNEL_1_PREFIX='2001:db8:123::/48' # /48-Subnetzmaske
IPV6_TUNNEL_1_LOCALV4='dynamic' # oder feste lokale IPv4-Adresse
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
IPV6_TUNNEL_1_LOCALV6='2001:db8:900:551::2/64'
IPV6_TUNNEL_1_REMOTEV6='2001:db8:900:551::1'
IPV6_TUNNEL_1_TYPE='sixxs'
IPV6_TUNNEL_1_USERID='USER1-SIXXS'
IPV6_TUNNEL_1_PASSWORD='sixxs'
IPV6_TUNNEL_1_TUNNELID='T1234'
```

Ein normal konfigurierter Windows 7-Host sollte mit einem solch konfigurierten fli4l-Router automatisch seine IPv6-Adresse sowie Default-Route, DNS-Server und Domäne konfigurieren und somit den Rechner IPv6-tauglich machen. Das kann man z.B. mit einem enfachen PING vom Windows-Rechner ins IPv6-Internet realisieren. Im folgenden Beispiel versuchen wir, vom Windows-Host aus den fli4l.de-Webserver zu erreichen (wir benutzen dabei direkt die IPv6-Adresse, um nicht von der Korrektheit der DNS-Funktionalität auszugehen):

```
C:\>ping 2001:bf0:c000:a::2:132

Ping wird ausgeführt für 2001:bf0:c000:a::2:132 mit 32 Bytes Daten:

Antwort von 2001:bf0:c000:a::2:132: Zeit=104ms
Antwort von 2001:bf0:c000:a::2:132: Zeit=102ms
Antwort von 2001:bf0:c000:a::2:132: Zeit=106ms
Antwort von 2001:bf0:c000:a::2:132: Zeit=106ms

Ping-Statistik für 2001:bf0:c000:a::2:132:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0 (0% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 102ms, Maximum = 106ms, Mittelwert = 104ms
```

Schließlich kann man das Werkzeug “traceroute” (unter Windows: “tracert”) verwenden, um zu untersuchen, ob ein Paket korrekt geroutet wird. Ein Beispiel aus dem lokalen Netz des Autors ist untenstehend abgebildet. Daran kann man gut erkennen, dass ein Paket erst zum fli4l-Router (erste Zeile), dann zum anderen Ende des Tunnels (zweite Zeile) und schließlich in das weltweite IPv6-Internet (ab der dritten Zeile) gelangt:

```
C:\>tracert 2001:bf0:c000:a::2:132
```

```
Routenverfolgung zu virtualhost.in-berlin.de [2001:bf0:c000:a::2:132]
  über maximal 30 Abschnitte:

  1    <1 ms    <1 ms    <1 ms  garm.example.org [2001:db8:13da:1::1]
  2    70 ms    79 ms    71 ms  gw-1362.ham-01.de.sixxs.net [2001:db8:900:551::1]
  3    67 ms    71 ms    76 ms  2001:db8:800:1003::209:55
  4    68 ms     *       70 ms  2001:db8:1:0:87:86:71:240
  5    69 ms     *       71 ms  2001:db8:1:0:87:86:77:67
  6    72 ms     *       71 ms  2001:db8:1:0:86:87:77:81
  7    71 ms     *       71 ms  2001:db8:1:0:87:86:77:83
  8    90 ms     *       81 ms  2001:db8:1:0:87:86:77:62
  9    84 ms     *       88 ms  2001:db8:1:0:87:86:77:71
 10    99 ms    83 ms    83 ms  2001:db8:1:0:87:86:77:249
 11    94 ms    87 ms    87 ms  20gigabitethernet4-3.core1.fra1.he.net
  [2001:7f8::1b1b:0:1]
 12    96 ms    99 ms    99 ms  10gigabitethernet1-4.core1.ams1.he.net
  [2001:470:0:47::1]
 13   105 ms   108 ms   107 ms  2001:7f8:8:5:0:73e6:0:1
 14   106 ms   107 ms   104 ms  virtualhost.in-berlin.de [2001:bf0:c000:a::2:132]

Ablaufverfolgung beendet.
```

B.12. ISDN

B.12.1. Technische Details zu Einwahl und Routing bei ISDN

Dieses Kapitel ist nur für Leute interessant, die ein wenig verstehen wollen, was intern passiert, die spezielle Konfigurationswünsche haben oder die nach der Lösung für Probleme suchen. Andere sollten dieses Kapitel bitte *nicht* lesen.

Nach dem Herstellen einer Verbindung zum Provider konfiguriert der ipppd-Daemon, der diese Verbindung hergestellt hat, das Interface neu, um die ausgehandelten IP-Adressen zu setzen. Dabei werden vom Linux-Kern automatisch auch Routen gesetzt, die der Remote-IP und der Netzmaske entsprechen und vorhandene, spezielle Routen werden gelöscht. Wird keine Netzmaske vorgegeben, leitet der ipppd aus der Remote-IP die Netzmaske ab (er benutzt dazu die Überholte Einteilung in Class A,B und C Subnetze). Das Verschwinden der Routen und die automatisch gesetzten neuen Routen haben immer wieder für Probleme gesorgt:

- Firmennetze waren nicht mehr erreichbar, weil die Routen verschwunden waren oder von den gesetzten neuen Routen überlagert wurden
- Interfaces wählten sich scheinbar ohne Grund ein, da ein Paket statt über die default Route über die vom Kern generierte Route auf ein anderes Interface ging
- ...

Daher wird nunmehr versucht, diese unerwünschten Routen zu verhindern.
Dazu werden folgende Dinge geändert:

- remote ip wird auf 0.0.0.0 gesetzt, wenn nichts anderes spezifiert ist. Dadurch verschwinden die Routen, die beim Konfigurieren des Interfaces vom Kern eingerichtet werden.

- zusätzlich angegebene Routen über den Circuit werden in einer Datei zwischengespeichert
- wird eine Netzwerkmaske für den Circuit angegeben, wird diese an den ipppd weitergereicht, damit der sie nach Aushandeln der IP für die Konfiguration des Interfaces (und damit für die Generierung von Routen) nutzt.
- nach dem Einwählen werden die zwischengespeicherten Routen des Circuits ausgelesen und wieder gesetzt (sie wurden vom Kern beim Neukonfigurieren des Interfaces durch ipppd gelöscht)
- nach dem Auflegen wird das Interface wieder neu konfiguriert und die Routen werden neu gesetzt um die Ausgangssituation wieder herzustellen.

Die Konfiguration der Circuits sieht dann wie folgt aus:

- item default route

  ```
  ISDN_CIRC_%_ROUTE='0.0.0.0'
  ```

  Ist der Circuit ein lcr circuit und gerade "aktiv", wird eine default route auf diesen Circuit (bzw. das dazugehörige Interface) gesetzt. Nach dem Einwählen erscheint eine Host-Route zum Provider, die nach dem Auflegen wieder verschwindet.

- spezielle Routen

  ```
  ISDN_CIRC_%_ROUTE='network/netmaskbits'
  ```

  Es werden die angegeben Routen auf den Circuit (bzw. das dazugehörige Interface) eingerichtet. Nach dem Einwählen werden die von Kern gelöschten Routen wieder hergestellt und es gibt eine Host-Route zum Einwahlknoten. Nach dem Auflegen wird der Originalzustand wieder hergestellt.

- remote ip

  ```
  ISDN_CIRC_%_REMOTE='ip address/netmaskbits'
  ISDN_CIRC_%_ROUTE='network/netmaskbits'
  ```

  Beim Konfigurieren des Interfaces erscheinen Routen in das Zielnetz (entsprechend ipadress AND netmask). Wird die spezifizierte IP nach dem Einwählen beibehalten (dass heißt, es wird keine andere ip während des Verbindungsaufbaus ausgehandelt), bleibt die Route bestehen.

  Wird allerdings beim Einwählen eine andere IP ausgehandelt, ändert sich die Route entsprechend (new ip AND netmask).

  Für die zusätzlichen Routen gilt das oben gesagte.

Das löst hoffentlich vorläufig *alle* Probleme, die mit speziellen Routen auftraten. Die Form der Korrektur mag sich in Zukunft noch ändern, an dem Prinzip ändert sich hoffentlich nichts mehr.

B.12.2. Fehlermeldungen des ISDN-Subsystems (englisch, i4l-Dokumentation)

Im folgenden ein Auszug aus der Isdn4Linux Dokumentation (man 7 isdn_cause).

Cause messages are 2-byte information elements, describing the state transitions of an ISDN line. Each cause message describes its origination (location) in one byte, while the cause code is described in the other byte. Internally, when EDSS1 is used, the first byte contains the location while the second byte contains the cause code. When using 1TR6, the first byte contains the cause code while the location is coded in the second byte. In the Linux ISDN subsystem, the cause messages visible to the user are unified to avoid confusion. All user visible cause messages are displayed as hexadecimal strings. These strings always have the location coded in the first byte, regardless if using 1TR6 or EDSS1. When using EDSS1, these strings are preceeded by the character 'E'.

LOCATION The following location codes are defined when using EDSS1:

00 Message generated by user.
01 Message generated by private network serving the local user.
02 Message generated by public network serving the local user.
03 Message generated by transit network.
04 Message generated by public network serving the remote user.
05 Message generated by private network serving the remote user.
07 Message generated by international network.
0A Message generated by network beyond inter-working point.

CAUSE The following cause codes are defined when using EDSS1:

01 Unallocated (unassigned) number.
02 No route to specified transit network.
03 No route to destination.
06 Channel unacceptable.
07 Call awarded and being delivered in an established channel.
10 Normal call clearing.
11 User busy.
12 No user responding.
13 No answer from user (user alerted).
15 Call rejected.
16 Number changed.
1A Non-selected user clearing.
1B Destination out of order.
1C Invalid number format.
1D Facility rejected.
1E Response to status enquiry.
1F Normal, unspecified.
22 No circuit or channel available.
26 Network out of order.
29 Temporary failure.
2A Switching equipment congestion.
2B Access information discarded.
2C Requested circuit or channel not available.
2F Resources unavailable, unspecified.
31 Quality of service unavailable.
32 Requested facility not subscribed.

39 Bearer capability not authorised.
3A Bearer capability not presently available.
3F Service or option not available, unspecified.
41 Bearer capability not implemented.
42 Channel type not implemented.
45 Requested facility not implemented.
46 Only restricted digital information bearer.
4F Service or option not implemented, unspecified.
51 Invalid call reference value.
52 Identified channel does not exist.
53 A suspended call exists, but this call identity does not.
54 Call identity in use.
55 No call suspended.
56 Call having the requested call identity.
58 Incompatible destination.
5B Invalid transit network selection.
5F Invalid message, unspecified.
60 Mandatory information element is missing.
61 Message type non-existent or not implemented.
62 Message not compatible with call state or message or message type non existent or not implemented.
63 Information element non-existent or not implemented.
64 Invalid information element content.
65 Message not compatible.
66 Recovery on timer expiry.
6F Protocol error, unspecified.
7F Inter working, unspecified.

B.13. UMTS

B.13.1. Unterstützte Hardware

```
Dieses Paket unterstützt folgende UMTS-Hardware:
Für den Betrieb sind unter anderem auch weitere Pakete erforderlich.
Für USB-Adapter muß das USB-Paket aktiviert werden.
OPT_USB='yes'

Hardware:               getestet        zusätzliche Pakete

Novatel Adapter:
Merlin U530             ja              PCMCIA, TOOLS (serial)
Merlin U630             nein            PCMCIA, TOOLS (serial)

MC950D                  ja              USB


OPTION Adapter:
3G Datacard             nein            PCMCIA, USB
GT 3G Quad              ja              PCMCIA, USB
GT Fusion               nein            PCMCIA, USB
```

```
GT MAX HSUPA GX0301     ja              PCMCIA, USB
bei den vier Cardbusadaptern ist PCMCIA_PCIC='yenta_socket' zu setzen

Icon 225 (GIO225)       ja              USB

Huawei Adapter:
E220, E230, E270        ja              USB
E510                    ja              USB
E800                    ja              USB
K3520                   ja              USB

ZTE Adapter:
MF110                   ja              USB
MF190                   ja              USB
```

B.13.2. Modemschnittstelle nicht aktiviert

Bei einigen OPTION UMTS Sticks kann es vorkommen, das die Modemschnittstelle nicht aktiviert ist, welche aber für den pppd benötigt wird.

```
Beispiel anhand des GIO225 Adapter

Kontrolle mittels:
grep "" /sys/bus/usb/devices/*/tty/*/hsotype

Die Ausgabe sollte etwa so aussehen:
/sys/bus/usb/devices/2-1:1.0/tty/ttyHS0/hsotype:Control
/sys/bus/usb/devices/2-1:1.0/tty/ttyHS1/hsotype:Application
/sys/bus/usb/devices/2-1:1.1/tty/ttyHS2/hsotype:Diagnostic

Hier fehlt die Ausgabe hsotype:Modem.

Jetzt kann man mit folgenden Befehl die Interface Konfiguration kontrollieren:
chat -e -t 1 '' "AT_OIFC?" OK >/dev/ttyHS0 </dev/ttyHS0

Die Ausgabe sollte folgendermassen aussehen:
AT_OIFC?
_OIFC: 3,1,1,0

OK

Sollte dort folgendes stehen:
AT_OIFC?
_OIFC: 2,1,1,0

OK
```

```
kann man die Modemschnittstelle mit folgenden Befehl aktivieren:
chat -e -t 1 '' "AT_OIFC=3,1,1,0" OK >/dev/ttyHS0 </dev/ttyHS0

Danach den Adapter noch einmal abziehen und neu anstecken.
Jetzt sollte mittels:
grep "" /sys/bus/usb/devices/*/tty/*/hsotype

auch ein Modemeintrag vorhanden sein.
/sys/bus/usb/devices/2-1:1.0/tty/ttyHS0/hsotype:Control
/sys/bus/usb/devices/2-1:1.0/tty/ttyHS1/hsotype:Application
/sys/bus/usb/devices/2-1:1.1/tty/ttyHS2/hsotype:Diagnostic
/sys/bus/usb/devices/2-1:1.2/tty/ttyHS3/hsotype:Modem
```

B.14. Unterschiede Version 3.10.4 und 3.6.2

Package ADVANCED_NETWORKING

Neue Variablen

BCRELAY_N (Seite 81)
BCRELAY_x_IF_N (Seite 81)
BCRELAY_x_IF_x (Seite 81)
ETHTOOL_DEV_N (Seite 92)
ETHTOOL_DEV_x (Seite 92)
ETHTOOL_DEV_x_OPTION_N (Seite 92)
ETHTOOL_DEV_x_OPTION_x_NAME (Seite 92)
ETHTOOL_DEV_x_OPTION_x_VALUE (Seite 92)
OPT_BCRELAY (Seite 81)
OPT_ETHTOOL (Seite 92)
OPT_IPSET (Seite ??)

Gelöschte Variablen

Package BASE

Neue Variablen

ARCH (Seite ??)
COMP_TYPE_ROOTFS (Seite 28)
DEBUG_IPTABLES (Seite 32)
FLI4L_BUILDDATE (Seite ??)
FLI4L_BUILDDIR (Seite ??)
FLI4L_BUILDTIME (Seite ??)
FLI4L_VERSION (Seite ??)
LIBATA_DMA (Seite 26)
PF_DNS_EXCEPTIONS (Seite ??)
PF_INPUT_ICMP_ECHO_REQ_SIZE (Seite 56)
PF_OUTPUT_ACCEPT_DEF (Seite 58)
PF_OUTPUT_CT_ACCEPT_DEF (Seite 71)
PF_OUTPUT_CT_N (Seite 71)
PF_OUTPUT_CT_x (Seite 71)
PF_OUTPUT_CT_x_COMMENT (Seite 71)
PF_OUTPUT_LOG (Seite 58)
PF_OUTPUT_LOG_LIMIT (Seite 58)

Gelöschte Variablen

COMPRESS_KERNEL
COMPRESS_OPT
COMPRESS_ROOTFS
DENY_ICMP
DMZ_LOG
DMZ_NAT
DMZ_ORANGE_RED_N
DMZ_ORANGE_RED_x
DMZ_ORANGE_ROUTER_N
DMZ_ORANGE_ROUTER_x
DMZ_RED_DEV
FORWARD_DENY_PORT_N
FORWARD_DENY_PORT_x
FORWARD_HOST_N
FORWARD_HOST_WHITE
FORWARD_HOST_x
IMOND_USE_ORIG

PF_OUTPUT_N (Seite 58)
PF_OUTPUT_POLICY (Seite 57)
PF_OUTPUT_REJ_LIMIT (Seite 58)
PF_OUTPUT_UDP_REJ_LIMIT (Seite 58)
PF_OUTPUT_x (Seite 58)
PF_OUTPUT_x_COMMENT (Seite 58)
PF_PREROUTING_CT_ACCEPT_DEF (Seite 71)
PF_PREROUTING_CT_N (Seite 71)
PF_PREROUTING_CT_x (Seite 71)
PF_PREROUTING_CT_x_COMMENT (Seite 71)
SYSLOGD_ROTATE_AT_SHUTDOWN (Seite 78)

INPUT_ACCEPT_PORT_N
INPUT_ACCEPT_PORT_x
INPUT_POLICY
IP_NET_x_TYPE
MASQ_NETWORK
OPT_DMZ
OPT_EVSS
OPT_MOUNTFLOPPY
PACKETFILTER_LOG
PACKETFILTER_LOG_LEVEL
PF_NEW_CONFIG
PRESERVE
ROUTE_NETWORK
TRUSTED_NETS

Package DHCP_CLIENT

Neue Variablen

DHCP_CLIENT_x_WAIT (Seite 98)

Gelöschte Variablen

Package DNS_DHCP

Neue Variablen

DNS_AUTHORITATIVE (Seite 102)
DNS_AUTHORITATIVE_IPADDR (Seite 103)
DNS_AUTHORITATIVE_NS (Seite 103)
DNS_BIND_INTERFACES (Seite 99)
DNS_FORWARD_LOCAL (Seite **??**)
DNS_LISTEN_N (Seite 100)
DNS_LISTEN_x (Seite 100)
DNS_LOCAL_HOST_CACHE_TTL (Seite 102)
DNS_ZONE_DELEGATION_N (Seite 103)
DNS_ZONE_DELEGATION_x_DOMAIN_N (Seite **??**)
DNS_ZONE_DELEGATION_x_DOMAIN_x (Seite **??**)
DNS_ZONE_DELEGATION_x_NETWORK_N (Seite **??**)
DNS_ZONE_DELEGATION_x_NETWORK_x (Seite **??**)
DNS_ZONE_DELEGATION_x_UPSTREAM_-
SERVER_N (Seite **??**)
DNS_ZONE_DELEGATION_x_UPSTREAM_-
SERVER_x_IP (Seite **??**)
DNS_ZONE_DELEGATION_x_UPSTREAM_-
SERVER_x_QUERYSOURCEIP (Seite **??**)
DNS_ZONE_NETWORK_N (Seite 103)
DNS_ZONE_NETWORK_x (Seite 103)
HOST_x_IP6_NET (Seite **??**)
OPT_YADIFA (Seite **??**)
YADIFA_ALLOW_QUERY_N (Seite 110)
YADIFA_ALLOW_QUERY_x (Seite **??**)
YADIFA_LISTEN_N (Seite 110)
YADIFA_LISTEN_x (Seite **??**)
YADIFA_SLAVE_ZONE_N (Seite 110)
YADIFA_SLAVE_ZONE_x (Seite 110)
YADIFA_SLAVE_ZONE_x_ALLOW_QUERY_N
(Seite 110)
YADIFA_SLAVE_ZONE_x_ALLOW_QUERY_x
(Seite 110)
YADIFA_SLAVE_ZONE_x_MASTER (Seite 110)
YADIFA_SLAVE_ZONE_x_USE_DNSMASQ_ZONE_-
DELEGATION (Seite **??**)

Gelöschte Variablen

DNS_LISTENIP_N
DNS_LISTENIP_x
DNS_SPECIAL_N
DNS_SPECIAL_x_DNSIP
DNS_SPECIAL_x_DOMAIN
DNS_SPECIAL_x_NETWORK

YADIFA_USE_DNSMASQ_ZONE_DELEGATION
(Seite **??**)

Package DSL

Neue Variablen

FRITZDSL_FILTER_EXPR (Seite 114)
PPPOE_FILTER_EXPR (Seite 114)
PPTP_FILTER_EXPR (Seite 114)

Gelöschte Variablen

OPT_PFC
OPT_PPPOE_CIRC
PPPOE_CIRC_N
PPPOE_CIRC_x_CHARGEINT
PPPOE_CIRC_x_DEBUG
PPPOE_CIRC_x_ETH
PPPOE_CIRC_x_FILTER
PPPOE_CIRC_x_HUP_TIMEOUT
PPPOE_CIRC_x_MRU
PPPOE_CIRC_x_MTU
PPPOE_CIRC_x_NAME
PPPOE_CIRC_x_PASS
PPPOE_CIRC_x_TIMES
PPPOE_CIRC_x_TYPE
PPPOE_CIRC_x_USEPEERDNS
PPPOE_CIRC_x_USER

Package DYNDNS

Neue Variablen

DYNDNS_LOGINTIME (Seite 124)
DYNDNS_x_LOGIN (Seite 124)
OPT_STUN (Seite **??**)
STUN_SERVER_N (Seite **??**)
STUN_SERVER_x (Seite **??**)

Gelöschte Variablen

Package HD

Neue Variablen

Gelöschte Variablen

HDIT_DATA
HDIT_POWEROFF
HDIT_SIZES
OPT_HDINSTALL_TEST

Package HTTPD

Neue Variablen

HTTPD_ARPING_IGNORE_N (Seite 132)
HTTPD_ARPING_IGNORE_x (Seite 132)

Gelöschte Variablen

Package IPV6

Neue Variablen

IPV6_NET_x_ADVERTISE_PREF_LIFETIME (Seite **??**)
IPV6_NET_x_ADVERTISE_VALID_LIFETIME (Seite **??**)
PF6_DNS_EXCEPTIONS (Seite **??**)
PF6_INPUT_ICMP_ECHO_REQ_LIMIT (Seite **??**)
PF6_INPUT_ICMP_ECHO_REQ_SIZE (Seite 150)
PF6_LOG_LEVEL (Seite 149)
PF6_OUTPUT_ACCEPT_DEF (Seite 152)
PF6_OUTPUT_CT_ACCEPT_DEF (Seite **??**)
PF6_OUTPUT_CT_N (Seite **??**)
PF6_OUTPUT_CT_x (Seite **??**)
PF6_OUTPUT_CT_x_COMMENT (Seite **??**)
PF6_OUTPUT_LOG (Seite 152)
PF6_OUTPUT_LOG_LIMIT (Seite 152)
PF6_OUTPUT_N (Seite 153)
PF6_OUTPUT_POLICY (Seite 152)
PF6_OUTPUT_REJ_LIMIT (Seite 152)
PF6_OUTPUT_UDP_REJ_LIMIT (Seite 153)
PF6_OUTPUT_x (Seite 153)
PF6_OUTPUT_x_COMMENT (Seite 153)
PF6_POSTROUTING_N (Seite 154)
PF6_POSTROUTING_x (Seite 154)
PF6_POSTROUTING_x_COMMENT (Seite 154)
PF6_PREROUTING_CT_ACCEPT_DEF (Seite **??**)
PF6_PREROUTING_CT_N (Seite **??**)
PF6_PREROUTING_CT_x (Seite **??**)
PF6_PREROUTING_CT_x_COMMENT (Seite **??**)
PF6_PREROUTING_N (Seite 154)
PF6_PREROUTING_x (Seite 154)
PF6_PREROUTING_x_COMMENT (Seite 154)

Gelöschte Variablen

Package ISDN

Neue Variablen

ISDN_FILTER_EXPR (Seite 160)
OPT_RCAPID (Seite 172)
RCAPID_PORT (Seite 172)
TELMOND_CAPI_CTRL_N (Seite 171)
TELMOND_CAPI_CTRL_x (Seite 172)

Gelöschte Variablen

Package OPENVPN

Neue Variablen

OPENVPN_x_IPV6 (Seite **??**)
OPENVPN_x_LOCAL_VPN_IPV6 (Seite 180)
OPENVPN_x_PF6_FORWARD_N (Seite 190)
OPENVPN_x_PF6_FORWARD_x (Seite 191)
OPENVPN_x_PF6_INPUT_N (Seite 190)
OPENVPN_x_PF6_INPUT_x (Seite 190)
OPENVPN_x_REMOTE_VPN_IPV6 (Seite 180)
OPENVPN_x_RENEG_SEC (Seite **??**)

Gelöschte Variablen

OPENVPN_DEFAULT_PF_DMZ_TYPE
OPENVPN_VERSION
OPENVPN_x_PF_DMZ_TYPE

Package PCMCIA

Neue Variablen

Gelöschte Variablen

PCMCIA_CARDMGR_OPTS
PCMCIA_CORE_OPTS
PCMCIA_PCIC_EXTERN

Package PROXY

Neue Variablen

Gelöschte Variablen

IGMPPROXY_ALT_N (Seite 213)
IGMPPROXY_ALT_NET_x (Seite 213)
IGMPPROXY_DEBUG (Seite 212)
IGMPPROXY_DEBUG2 (Seite 212)
IGMPPROXY_DOWNLOAD_DEV (Seite **??**)
IGMPPROXY_QUICKLEAVE_ON (Seite **??**)
IGMPPROXY_UPLOAD_DEV (Seite 213)
IGMPPROXY_WLIST_N (Seite 213)
IGMPPROXY_WLIST_NET_x (Seite **??**)
OPT_IGMPPROXY (Seite 207)
OPT_KAMAILIO (Seite **??**)
OPT_RTPPROXY (Seite **??**)
OPT_SIPROXD (Seite **??**)
OPT_STUNNEL (Seite 215)
STUNNEL_DEBUG (Seite 215)
STUNNEL_N (Seite 215)
STUNNEL_x_ACCEPT (Seite 216)
STUNNEL_x_ACCEPT_IPV4 (Seite 216)
STUNNEL_x_ACCEPT_IPV6 (Seite 217)
STUNNEL_x_CERT_CA_FILE (Seite 218)
STUNNEL_x_CERT_FILE (Seite 218)
STUNNEL_x_CERT_VERIFY (Seite 218)
STUNNEL_x_CLIENT (Seite 215)
STUNNEL_x_CONNECT (Seite 217)
STUNNEL_x_DELAY_DNS (Seite 217)
STUNNEL_x_NAME (Seite 215)
STUNNEL_x_OUTGOING_IP (Seite 217)

Package QOS

Neue Variablen

Gelöschte Variablen

QOS_CLASS_x_LABEL (Seite 224)

Package SSHD

Neue Variablen

Gelöschte Variablen

OPT_SCP

Package TOOLS

Neue Variablen

FTP_PF_ENABLE_ACTIVE (Seite 241)
OPT_ATH_INFO (Seite 245)
OPT_DIG (Seite 241)
OPT_I2CTOOLS (Seite 245)
OPT_IWLEEPROM (Seite 245)
OPT_NGREP (Seite 242)
OPT_OPENSSL (Seite 246)
OPT_REAVER (Seite 246)
OPT_SOCAT (Seite 243)

Gelöschte Variablen

OPT_ARP
OPT_BCRELAY
OPT_ETHTOOL
OPT_NETSTAT
OPT_SERIAL
OPT_TRACEROUTE
OPT_TRACEROUTE6
WGET_SSL

Package USB

Neue Variablen

Gelöschte Variablen

USB_LOWLEVEL

B.15. Unterschiede Version 3.10.4 und 3.10.3

Abbildungsverzeichnis

3.1. Struktur des Paketfilters 45
3.2. Verzeichnisstruktur fli4l 53

4.1. VPN-Konfigurationsbeispiel — Tunnel zwischen zwei Routern 174
4.2. fli4l config Directory mit OpenVPN *.secret Dateien 178
4.3. Verbindungsübersicht 192
4.4. Detailansicht einer Verbindung (Keymanagement) 193
4.5. Fli4l in der Standardkonfiguration 209
4.6. Fli4l in der IPTV-Konfiguration 209
4.7. Beispiel 1 230
4.8. Beispiel 2 232
4.9. Beispiel 3 234
4.10. Verzeichnisstruktur fli4l 237

5.1. Einstellungen 275
5.2. Einstellungen für Remoteupdate 276
5.3. Einstellungen für HD-pre-install 277

Tabellenverzeichnis

3.1. Übersicht über die (Zusatz-)Pakete . . . 19
3.2. Automatische Einstellung der maximalen Verbindungsanzahl . . . 29
3.3. Tabelle der möglichen Netzwerkkartentreiber; Legende: v=virt, n=nonfree, vn=virt-nonfree . . . 39
3.4. Tabelle der möglichen WLAN-Kartentreiber; Legende: v=virt, n=nonfree, vn=virt-nonfree . . . 41
3.5. Aktionen in Paketfilterregeln . . . 46
3.6. Quell- und Zieleinschränkungen in Paketfilterregeln . . . 47
3.7. Zustandseinschränkungen in Paketfilterregeln . . . 49
3.8. Im Lieferumfang von fli4l enthaltene Schablonen . . . 52
3.9. Verfügbare Conntrack-Helfer im Paketfilter . . . 70
3.10. Format der Imond-Logdatei . . . 74

4.1. Werte für BRIDGE_DEV_x_DEV_x_PATHCOST in Abhängigkeit von der Bandbreite . . . 91
4.2. Arten der pppoe-Paketerzeugung . . . 116
4.3. Bandbreite und CPU-Auslastung bei pppoe . . . 116
4.4. Fritz-Karten . . . 117
4.5. PPTP-Modemtypen . . . 118
4.6. Beispiel für die Konfiguration des Setup-Mediums . . . 128
4.7. Beispiel für eine Installation nach Typ A oder B . . . 128
4.9. Aktionen der OpenVPN-Webgui . . . 193
4.10. Unterschiedliche MTU Parameter der unterschiedlichen OpenVPN Versionen. . . . 195
4.11. Unterschiedliche MTU Parameter der fli4l-Router Versionen. . . . 195
4.12. OpenVPN Konfiguration mit 2 fli4l-Routern . . . 196
4.13. OpenVPN Konfiguration mit 2 fli4l-Routern deren Netzwerk über eine Funkverbindung gebridgt wird. . . . 196
4.14. OpenVPN Konfiguration mit 2 fli4l-Routern deren Netzwerk über eine Funkverbindung gebridgt wird. Die Konfiguration der Bridge in advanced_networking. 197
4.15. OpenVPN Konfiguration mit 2 fli4l-Routern deren Netzwerk über eine Funkverbindung gebridgt wird. Die Konfiguration der Bridge in der Basiskonfiguration (base.txt). . . . 197
4.16. OpenVPN Konfiguration mit einem Windowsrechner über GPRS. . . . 198
4.17. OpenVPN Absicherung eines WLAN. . . . 199

8.1. Parameter für `mkfli4l` . . . 307
8.2. Optionen für Dateien . . . 310
8.3. Logische Ausdrücke . . . 334
8.4. Funktionen des `cgi-helper`-Skriptes . . . 349

Index

base.txt, 19
BCRELAY_N, 81
BCRELAY_x_IF_N, 81
BCRELAY_x_IF_x, 81
BEEP, 30
Beispiel-Datei(base.txt), 19
BONDING_DEV_N, 82
BONDING_DEV_x_ARP_INTERVAL, 85
BONDING_DEV_x_ARP_IP_TARGET_N, 85
BONDING_DEV_x_ARP_IP_TARGET_x, 86
BONDING_DEV_x_DEV_N, 84
BONDING_DEV_x_DEV_x, 84
BONDING_DEV_x_DEVNAME, 82
BONDING_DEV_x_DOWNDELAY, 85
BONDING_DEV_x_LACP_RATE, 85
BONDING_DEV_x_MAC, 84
BONDING_DEV_x_MIIMON, 84
BONDING_DEV_x_MODE, 82
BONDING_DEV_x_PRIMARY, 85
BONDING_DEV_x_UPDELAY, 84
BONDING_DEV_x_USE_CARRIER, 84
BOOT_TYPE, 25
BOOTMENU_TIME, 26
BRIDGE_DEV_BOOTDELAY, 88
BRIDGE_DEV_N, 88
BRIDGE_DEV_x_AGING, 89
BRIDGE_DEV_x_DEV_N, 88
BRIDGE_DEV_x_DEV_x_DEV, 88
BRIDGE_DEV_x_DEV_x_PATHCOST, 90
BRIDGE_DEV_x_DEV_x_PORT_PRIORITY, 90
BRIDGE_DEV_x_DEVNAME, 88
BRIDGE_DEV_x_FORWARD_DELAY, 89
BRIDGE_DEV_x_GARBAGE_COLLECTION_INTERVAL, 89
BRIDGE_DEV_x_HELLO, 90
BRIDGE_DEV_x_MAX_MESSAGE_AGE, 90
BRIDGE_DEV_x_NAME, 88
BRIDGE_DEV_x_PRIORITY, 89
BRIDGE_DEV_x_STP, 89
BUILDDIR, 278

CHRONY_BIOS_TIME, 96
CHRONY_LOG, 96
CHRONY_TIMESERVER_N, 95
CHRONY_TIMESERVER_x, 96
CHRONY_TIMESERVICE, 95
COMP_TYPE_OPT, 28
COMP_TYPE_ROOTFS, 27
COMPRESS_KERNEL, 398
COMPRESS_OPT, 398
COMPRESS_ROOTFS, 398
CONSOLE_BLANK_TIME, 30

DEBUG_ENABLE_CORE, 31, 342
DEBUG_IP, 32, 342
DEBUG_IPTABLES, 31
DEBUG_IPUP, 342
DEBUG_KEEP_BOOTLOGD, 343
DEBUG_MDEV, 31, 343
DEBUG_MODULES, 31
DEBUG_STARTUP, 31
DENY_ICMP, 398
DEV_MTU_N, 87
DEV_MTU_x, 87
DHCP_CLIENT_DEBUG, 98
DHCP_CLIENT_N, 97
DHCP_CLIENT_TYPE, 97
DHCP_CLIENT_x_HOSTNAME, 97

DHCP_CLIENT_x_IF, 97
DHCP_CLIENT_x_ROUTE, 97
DHCP_CLIENT_x_STARTDELAY, 97
DHCP_CLIENT_x_USEPEERDNS, 97
DHCP_CLIENT_x_WAIT, 97
DHCP_DENY_MAC_N, 108
DHCP_DENY_MAC_x, 108
DHCP_EXTRA_RANGE_N, 107
DHCP_EXTRA_RANGE_x_DEVICE, 107
DHCP_EXTRA_RANGE_x_DNS_SERVER, 107
DHCP_EXTRA_RANGE_x_END, 107
DHCP_EXTRA_RANGE_x_GATEWAY, 107
DHCP_EXTRA_RANGE_x_MTU, 107
DHCP_EXTRA_RANGE_x_NETMASK, 107
DHCP_EXTRA_RANGE_x_NTP_SERVER, 107
DHCP_EXTRA_RANGE_x_START, 107
DHCP_LEASES_DIR, 106
DHCP_LEASES_VOLATILE, 106
DHCP_LS_TIME_DYN, 105
DHCP_LS_TIME_FIX, 106
DHCP_MAX_LS_TIME_DYN, 106
DHCP_MAX_LS_TIME_FIX, 106
DHCP_RANGE_N, 106
DHCP_RANGE_x_DNS_DOMAIN, 106
DHCP_RANGE_x_DNS_SERVER1, 106
DHCP_RANGE_x_DNS_SERVER2, 106
DHCP_RANGE_x_END, 106
DHCP_RANGE_x_GATEWAY, 107
DHCP_RANGE_x_MTU, 107
DHCP_RANGE_x_NET, 106
DHCP_RANGE_x_NTP_SERVER, 107
DHCP_RANGE_x_OPTION_N, 107
DHCP_RANGE_x_OPTION_x, 107
DHCP_RANGE_x_PXE_FILENAME, 108
DHCP_RANGE_x_PXE_OPTIONS, 108
DHCP_RANGE_x_PXE_SERVERIP, 108
DHCP_RANGE_x_PXE_SERVERNAME, 108
DHCP_RANGE_x_START, 106
DHCP_TYPE, 105
DHCP_VERBOSE, 105
DHCP_WINSSERVER_1, 106
DHCP_WINSSERVER_2, 106
DHCPRELAY_IF_N, 108
DHCPRELAY_IF_x, 108
DHCPRELAY_SERVER, 108
DIALMODE, 75
DMZ_LOG, 398
DMZ_NAT, 398
DMZ_ORANGE_RED_N, 398
DMZ_ORANGE_RED_x, 398
DMZ_ORANGE_ROUTER_N, 398
DMZ_ORANGE_ROUTER_x, 398
DMZ_RED_DEV, 398
DNS_AUTHORITATIVE, 102
DNS_AUTHORITATIVE_IPADDR, 103
DNS_AUTHORITATIVE_NS, 102
DNS_BIND_INTERFACES, 99
DNS_BOGUS_PRIV, 101
DNS_FILTERWIN2K, 101
DNS_FORBIDDEN_N, 100
DNS_FORBIDDEN_x, 100
DNS_FORWARD_LOCAL, 101
DNS_FORWARDERS, 71
DNS_LISTEN_N, 100
DNS_LISTEN_x, 100
DNS_LISTENIP_N, 399
DNS_LISTENIP_x, 399
DNS_LOCAL_HOST_CACHE_TTL, 102
DNS_MX_SERVER, 100
DNS_REBINDOK_N, 105
DNS_REBINDOK_x_DOMAIN, 105
DNS_REDIRECT_N, 101
DNS_REDIRECT_x, 101
DNS_REDIRECT_x_IP, 101
DNS_SPECIAL_N, 399
DNS_SPECIAL_x_DNSIP, 399
DNS_SPECIAL_x_DOMAIN, 399
DNS_SPECIAL_x_NETWORK, 399
DNS_SUPPORT_IPV6, 102
DNS_VERBOSE, 100
DNS_ZONE_DELEGATION_N, 103
DNS_ZONE_DELEGATION_x, 103

DNS_ZONE_DELEGATION_x_-
DOMAIN, 103
DNS_ZONE_DELEGATION_x_-
NETWORK, 103
DNS_ZONE_DELEGATION_x_-
UPSTREAM_SERVER_x, 103
DNS_ZONE_DELEGATION_x_-
UPSTREAM_SERVER_x_IP,
103
DNS_ZONE_DELEGATION_x_-
UPSTREAM_SERVER_x_-
quERYSOURCEIP, 103
DNS_ZONE_NETWORK_N, 103
DNS_ZONE_NETWORK_x, 103
DOMAIN_NAME, 71
DYNDNS_ALLOW_SSL, 124
DYNDNS_DEBUG_PROVIDER, 124
DYNDNS_LOGINTIME, 124
DYNDNS_LOOKUP_NAMES, 124
DYNDNS_N, 123
DYNDNS_SAVE_OUTPUT, 122
DYNDNS_x_CIRCUIT, 123
DYNDNS_x_EXT_IP, 124
DYNDNS_x_HOSTNAME, 123
DYNDNS_x_LOGIN, 124
DYNDNS_x_PASSWORD, 123
DYNDNS_x_PROVIDER, 123
DYNDNS_x_RENEW, 123
DYNDNS_x_UPDATEHOST, 123
DYNDNS_x_USER, 123

EASYCRON_MAIL, 125
EASYCRON_N, 125
EASYCRON_x_COMMAND, 125
EASYCRON_x_CUSTOM, 125
EASYCRON_x_TIME, 125
ETHTOOL_DEV_N, 92
ETHTOOL_DEV_x, 92
ETHTOOL_DEV_x_OPTION_N, 92
ETHTOOL_DEV_x_OPTION_x_-
NAME, 92
ETHTOOL_DEV_x_OPTION_x_-
VALUE, 92
EXTMOUNT_N, 129
EXTMOUNT_x_FILESYSTEM, 129
EXTMOUNT_x_MOUNTPOINT, 130
EXTMOUNT_x_OPTIONS, 130
EXTMOUNT_x_VOLUMEID, 129

FILESONLY, 278
FLI4L_UUID, 28
FORWARD_DENY_PORT_N, 398
FORWARD_DENY_PORT_x, 398
FORWARD_HOST_N, 398
FORWARD_HOST_WHITE, 398
FORWARD_HOST_x, 398
FRITZDSL_CHARGEINT, 112
FRITZDSL_DEBUG, 112
FRITZDSL_FILTER, 114
FRITZDSL_FILTER_EXPR, 114
FRITZDSL_HUP_TIMEOUT, 112
FRITZDSL_MRU, 114
FRITZDSL_MTU, 114
FRITZDSL_NAME, 111
FRITZDSL_NF_MSS, 114
FRITZDSL_PASS, 112
FRITZDSL_PROVIDER, 117
FRITZDSL_TIMES, 112
FRITZDSL_TYPE, 117
FRITZDSL_USEPEERDNS, 111
FRITZDSL_USER, 112
ftp, 70
FTP_PF_ENABLE_ACTIVE, 241

h323, 70
HDDRV_N, 131
HDDRV_x, 131
HDDRV_x_OPTION, 131
HDIT_DATA, 400
HDIT_POWEROFF, 400
HDIT_SIZES, 400
HDSLEEP_TIMEOUT, 130
HOST_EXTRA_N, 99
HOST_EXTRA_x_IP4, 99
HOST_EXTRA_x_IP6, 99
HOST_EXTRA_x_NAME, 99
HOST_N, 98
HOST_x_ALIAS_N, 98
HOST_x_ALIAS_x, 98
HOST_x_DHCPTYP, 98
HOST_x_DOMAIN, 98
HOST_x_IP4, 98
HOST_x_IP6, 98
HOST_x_MAC, 98

HOST_x_MAC2, 98
HOST_x_NAME, 98
HOST_x_PXE_FILENAME, 108
HOST_x_PXE_OPTIONS, 108
HOST_x_PXE_SERVERIP, 108
HOST_x_PXE_SERVERNAME, 108
HOSTNAME, 25
HOSTNAME_ALIAS_N, 72
HOSTNAME_ALIAS_x, 72
HOSTNAME_IP, 72
HOSTNAME_IP6, 143
HTTPD_ARPING, 132
HTTPD_ARPING_IGNORE_N, 132
HTTPD_ARPING_IGNORE_x, 132
HTTPD_GUI_LANG, 132
HTTPD_GUI_SKIN, 132
HTTPD_LISTENIP, 132
HTTPD_PORT, 132
HTTPD_PORTFW, 132
HTTPD_USER, 380
HTTPD_USER_N, 133
HTTPD_USER_x_PASSWORD, 133
HTTPD_USER_x_RIGHTS, 133
HTTPD_USER_x_USERNAME, 133
HW_DETECT_AT_BOOTTIME, 245
HWSUPP_BOOT_LED, 139
HWSUPP_BUTTON_N, 139
HWSUPP_BUTTON_x, 139
HWSUPP_BUTTON_x_DEVICE, 140
HWSUPP_BUTTON_x_PARAM, 140
HWSUPP_CPUFREQ, 137
HWSUPP_CPUFREQ_GOVERNOR, 137
HWSUPP_DRIVER_N, 141
HWSUPP_DRIVER_x, 141
HWSUPP_I2C_N, 141
HWSUPP_I2C_x_ADDRESS, 141
HWSUPP_I2C_x_BUS, 141
HWSUPP_I2C_x_DEVICE, 141
HWSUPP_LED_N, 137
HWSUPP_LED_PARAM, 138
HWSUPP_LED_x, 138
HWSUPP_LED_x_DEVICE, 138
HWSUPP_TYPE, 136
HWSUPP_WATCHDOG, 137

IGMPPROXY_ALT_N, 213
IGMPPROXY_ALT_NET_x, 213
IGMPPROXY_DEBUG, 212
IGMPPROXY_DEBUG2, 212
IGMPPROXY_DOWNLOAD_DEV, 213
IGMPPROXY_QUICKLEAVE_ON, 212
IGMPPROXY_UPLOAD_DEV, 213
IGMPPROXY_WHLIST_NET_x, 213
IGMPPROXY_WLIST_N, 213
IMOND_ADMIN_PASS, 73
IMOND_BEEP, 73
IMOND_DIAL, 74
IMOND_ENABLE, 74
IMOND_LED, 73
IMOND_LOG, 73
IMOND_LOGDIR, 74
IMOND_PASS, 72
IMOND_PORT, 72
IMOND_REBOOT, 74
IMOND_ROUTE, 74
IMOND_USE_ORIG, 398
INPUT_ACCEPT_PORT_N, 399
INPUT_ACCEPT_PORT_x, 399
INPUT_POLICY, 399
IP_CONNTRACK_MAX, 28
IP_DYN_ADDR, 75
IP_NET_N, 41
IP_NET_x, 41
IP_NET_x_COMMENT, 43
IP_NET_x_DEV, 42
IP_NET_x_MAC, 42
IP_NET_x_NAME, 43
IP_NET_x_TYPE, 43, 399
IP_ROUTE_N, 43
IP_ROUTE_x, 43
IPV6_NET_N, 143
IPV6_NET_x, 143
IPV6_NET_x_ADVERTISE, 144
IPV6_NET_x_ADVERTISE_DNS, 145
IPV6_NET_x_DEV, 144
IPV6_NET_x_DHCP, 145
IPV6_NET_x_NAME, 145
IPV6_NET_x_TUNNEL, 144
IPV6_ROUTE_N, 148
IPV6_ROUTE_x, 148
IPV6_TUNNEL_N, 145
IPV6_TUNNEL_x_DEFAULT, 146
IPV6_TUNNEL_x_DEV, 147

IPV6_TUNNEL_x_LOCALV4, 146
IPV6_TUNNEL_x_LOCALV6, 147
IPV6_TUNNEL_x_MTU, 147
IPV6_TUNNEL_x_PASSWORD, 148
IPV6_TUNNEL_x_PREFIX, 146
IPV6_TUNNEL_x_REMOTEV4, 147
IPV6_TUNNEL_x_REMOTEV6, 147
IPV6_TUNNEL_x_TIMEOUT, 148
IPV6_TUNNEL_x_TUNNELID, 148
IPV6_TUNNEL_x_TYPE, 146
IPV6_TUNNEL_x_USERID, 148
irc, 70
ISDN_CIRC_N, 161
ISDN_CIRC_x_AUTH, 167
ISDN_CIRC_x_BANDWIDTH, 162
ISDN_CIRC_x_BUNDLING, 162
ISDN_CIRC_x_CALLBACK, 166
ISDN_CIRC_x_CBDELAY, 166
ISDN_CIRC_x_CBNUMBER, 166
ISDN_CIRC_x_CHARGEINT, 167
ISDN_CIRC_x_CLAMP_MSS, 163
ISDN_CIRC_x_DEBUG, 167
ISDN_CIRC_x_DIALIN, 165
ISDN_CIRC_x_DIALOUT, 165
ISDN_CIRC_x_EAZ, 166
ISDN_CIRC_x_FRAMECOMP, 164
ISDN_CIRC_x_HEADERCOMP, 164
ISDN_CIRC_x_HUP_TIMEOUT, 167
ISDN_CIRC_x_LOCAL, 163
ISDN_CIRC_x_MRU, 163
ISDN_CIRC_x_MTU, 163
ISDN_CIRC_x_NAME, 161
ISDN_CIRC_x_PASS, 164
ISDN_CIRC_x_REMOTE, 163
ISDN_CIRC_x_REMOTENAME, 164
ISDN_CIRC_x_ROUTE_N, 165
ISDN_CIRC_x_ROUTE_X, 165
ISDN_CIRC_x_SLAVE_EAZ, 167
ISDN_CIRC_x_TIMES, 168
ISDN_CIRC_x_TYPE, 162
ISDN_CIRC_x_USEPEERDNS, 161
ISDN_CIRC_x_USER, 164
ISDN_DEBUG_LEVEL, 159
ISDN_FILTER, 160
ISDN_FILTER_EXPR, 160
ISDN_IO, 156
ISDN_IO0, 156
ISDN_IO1, 156
ISDN_IP, 156
ISDN_LZS_COMP, 160
ISDN_LZS_DEBUG, 160
ISDN_LZS_TWEAK, 160
ISDN_MEM, 156
ISDN_PORT, 156
ISDN_TYPE, 156
ISDN_VERBOSE_LEVEL, 159

KERNEL_BOOT_OPTION, 27
KERNEL_VERSION, 27
KEYBOARD_LOCALE, 33

LIBATA_DMA, 26
LOCALE, 30
LOG_BOOT_SEQ, 343
LOGIP_LOGDIR, 78

MASQ_NETWORK, 399
Masquerading, 69
MKFLI4L_DEBUG_OPTION, 279
MOUNT_BOOT, 26

NET_DRV_N, 33
NET_DRV_x, 33
NET_DRV_x_OPTION, 34

OAC_ALL_INVISIBLE, 134
OAC_BLOCK_UNKNOWN_IF_x, 135
OAC_GROUP_N, 134
OAC_GROUP_x_BOOTBLOCK, 135
OAC_GROUP_x_CLIENT_N, 135
OAC_GROUP_x_CLIENT_x, 135
OAC_GROUP_x_INVISIBLE, 135
OAC_GROUP_x_NAME, 135
OAC_INPUT, 134
OAC_LIMITS, 134
OAC_MODE, 134
OAC_WANDEVICE, 134
OPENVPN_DEFAULT_ALLOW_-
ICMPPING, 183
OPENVPN_DEFAULT_CIPHER, 182
OPENVPN_DEFAULT_COMPRESS,
182
OPENVPN_DEFAULT_CREATE_-
SECRET, 182
OPENVPN_DEFAULT_DIGEST, 182

OPENVPN_DEFAULT_FLOAT, 183
OPENVPN_DEFAULT_FRAGMENT, 186
OPENVPN_DEFAULT_KEYSIZE, 183
OPENVPN_DEFAULT_LINK_MTU, 186
OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE, 185
OPENVPN_DEFAULT_MSSFIX, 186
OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS, 185
OPENVPN_DEFAULT_OPEN_OVPNPORT, 183
OPENVPN_DEFAULT_PF_DMZ_TYPE, 401
OPENVPN_DEFAULT_PF_FORWARD_LOG, 183
OPENVPN_DEFAULT_PF_FORWARD_POLICY, 184
OPENVPN_DEFAULT_PF_INPUT_LOG, 183
OPENVPN_DEFAULT_PF_INPUT_POLICY, 183
OPENVPN_DEFAULT_PING, 184
OPENVPN_DEFAULT_PING_RESTART, 184
OPENVPN_DEFAULT_PROTOCOL, 185
OPENVPN_DEFAULT_RENEG_SEC, 184
OPENVPN_DEFAULT_RESOLV_RETRY, 184
OPENVPN_DEFAULT_RESTART, 184
OPENVPN_DEFAULT_SHAPER, 186
OPENVPN_DEFAULT_START, 185
OPENVPN_DEFAULT_TUN_MTU, 186
OPENVPN_DEFAULT_TUN_MTU_EXTRA, 186
OPENVPN_DEFAULT_VERBOSE, 185
OPENVPN_EXPERT, 186
OPENVPN_N, 175
OPENVPN_VERSION, 401
OPENVPN_WEBGUI, 191
OPENVPN_x_ACTIV, 187
OPENVPN_x_ALLOW_ICMPPING, 189
OPENVPN_x_BRIDGE, 178
OPENVPN_x_BRIDGE_COST, 178
OPENVPN_x_BRIDGE_PRIORITY, 179
OPENVPN_x_CHECK_CONFIG, 187
OPENVPN_x_CIPHER, 187
OPENVPN_x_COMPRESS, 187
OPENVPN_x_CREATE_SECRET, 187
OPENVPN_x_DIGEST, 188
OPENVPN_x_DNSIP, 181
OPENVPN_x_DOMAIN, 181
OPENVPN_x_FLOAT, 188
OPENVPN_x_FRAGMENT, 191
OPENVPN_x_IPV6, 180
OPENVPN_x_ISDN_CIRC_NAME, 188
OPENVPN_x_KEYSIZE, 188
OPENVPN_x_LINK_MTU, 191
OPENVPN_x_LOCAL_HOST, 176
OPENVPN_x_LOCAL_PORT, 176
OPENVPN_x_LOCAL_VPN_IP, 179
OPENVPN_x_LOCAL_VPN_IPV6, 180
OPENVPN_x_MANAGEMENT_LOG_CACHE, 188
OPENVPN_x_MSSFIX, 191
OPENVPN_x_MUTE_REPLAY_WARNINGS, 189
OPENVPN_x_NAME, 187
OPENVPN_x_OPEN_OVPNPORT, 189
OPENVPN_x_PF6_FORWARD_N, 190
OPENVPN_x_PF6_FORWARD_x, 190
OPENVPN_x_PF6_INPUT_N, 190
OPENVPN_x_PF6_INPUT_x, 190
OPENVPN_x_PF_DMZ_TYPE, 401
OPENVPN_x_PF_FORWARD_LOG, 189
OPENVPN_x_PF_FORWARD_N, 190
OPENVPN_x_PF_FORWARD_POLICY, 190
OPENVPN_x_PF_FORWARD_x, 190
OPENVPN_x_PF_INPUT_LOG, 189
OPENVPN_x_PF_INPUT_N, 189
OPENVPN_x_PF_INPUT_POLICY, 189
OPENVPN_x_PF_INPUT_x, 189
OPENVPN_x_PF_POSTROUTING_N, 190

OPENVPN_x_PF_POSTROUTING_x, 190
OPENVPN_x_PF_PREROUTING_N, 190
OPENVPN_x_PF_PREROUTING_x, 190
OPENVPN_x_PING, 188
OPENVPN_x_PING_RESTART, 188
OPENVPN_x_PROTOCOL, 188
OPENVPN_x_REMOTE_HOST, 175
OPENVPN_x_REMOTE_HOST_N, 175
OPENVPN_x_REMOTE_HOST_x, 176
OPENVPN_x_REMOTE_PORT, 176
OPENVPN_x_REMOTE_VPN_IP, 179
OPENVPN_x_REMOTE_VPN_IPV6, 180
OPENVPN_x_RESOLV_RETRY, 188
OPENVPN_x_RESTART, 189
OPENVPN_x_ROUTE_N, 180
OPENVPN_x_ROUTE_x, 180
OPENVPN_x_ROUTE_x_DNSIP, 182
OPENVPN_x_ROUTE_x_DOMAIN, 181
OPENVPN_x_SECRET, 177
OPENVPN_x_SHAPER, 191
OPENVPN_x_START, 188
OPENVPN_x_TUN_MTU, 191
OPENVPN_x_TUN_MTU_EXTRA, 191
OPENVPN_x_TYPE, 177
OPENVPN_x_VERBOSE, 188
OPT_ARP, 403
OPT_ATH_INFO, 245
OPT_BCRELAY, 81, 403
OPT_BONDING_DEV, 82
OPT_BRIDGE_DEV, 88
OPT_CHRONY, 95
OPT_DHCP, 105
OPT_DHCP_CLIENT, 97
OPT_DHCPRELAY, 108
OPT_DIG, 241
OPT_DMZ, 399
OPT_DNS, 99
OPT_DYNDNS, 122
OPT_E3, 245
OPT_EASYCRON, 125
OPT_EBTABLES, 91
OPT_ETHTOOL, 92, 403
OPT_EVSS, 399
OPT_EXTMOUNT, 129
OPT_FRITZDSL, 116
OPT_FTP, 241
OPT_HDDRV, 131
OPT_HDINSTALL, 126
OPT_HDINSTALL_TEST, 400
OPT_HDSLEEP, 130
OPT_HOSTS, 98
OPT_HTTPD, 131
OPT_HW_DETECT, 245
OPT_HWSUPP, 136
OPT_I2CTOOLS, 245
OPT_IFTOP, 241
OPT_IGMPPROXY, 207, 212
OPT_IMONC, 242
OPT_IPERF, 242
OPT_IPV6, 143
OPT_ISDN, 155
OPT_ISDN_COMP, 160
OPT_IWLEEPROM, 245
OPT_KLOGD, 78
OPT_LOGIP, 78
OPT_LSPCI, 245
OPT_MAKEKBL, 33
OPT_MOUNT, 129
OPT_MOUNTFLOPPY, 399
OPT_MTOOLS, 245
OPT_NETCAT, 242
OPT_NETSTAT, 403
OPT_NGREP, 242
OPT_NTTCP, 242
OPT_OAC, 134
OPT_OPENSSL, 246
OPT_OPENVPN, 175
OPT_PCMCIA, 199
OPT_PFC, 400
OPT_PLINK_CLIENT, 240
OPT_PNP, 79
OPT_POESTATUS, 119
OPT_PPP, 200
OPT_PPPOE, 115
OPT_PPPOE_CIRC, 400
OPT_PPTP, 118
OPT_PRIVOXY, 202
OPT_QOS, 221
OPT_RCAPID, 172

OPT_REAVER, 246
OPT_RECOVER, 130
OPT_RTMON, 243
OPT_SCP, 402
OPT_SERIAL, 403
OPT_SFTPSERVER, 241
OPT_SHRED, 246
OPT_SIPPROXY, 207
OPT_SOCAT, 243
OPT_SS5, 206
OPT_SSH_CLIENT, 240
OPT_SSHD, 236
OPT_STRACE, 246
OPT_STUNNEL, 215
OPT_SYSLOGD, 76
OPT_TCPDUMP, 243
OPT_TELMOND, 169
OPT_TFTP, 109
OPT_TOR, 204
OPT_TRACEROUTE, 403
OPT_TRACEROUTE6, 403
OPT_TRANSPROXY, 206
OPT_UMTS, 246
OPT_USB, 249
OPT_VALGRIND, 246
OPT_VLAN_DEV, 86
OPT_VPN_CARD, 141
OPT_WGET, 244
OPT_WLAN, 252
OPT_Y2K, 78
OPT_YADIFA, 109
OPT_YADIFA_SLAVE_ZONE_USE_DNSMASQ_ZONE_DELEGATION, 110
OPT_YADIFA_USE_DNSMASQ_ZONE_DELEGATION, 109
OPT_YTREE, 246

PACKETFILTER_LOG, 399
PACKETFILTER_LOG_LEVEL, 399
PASSWORD, 25
PCMCIA_CARDMGR_OPTS, 402
PCMCIA_CORE_OPTS, 402
PCMCIA_MISC_N, 199
PCMCIA_MISC_x, 199
PCMCIA_PCIC, 199
PCMCIA_PCIC_EXTERN, 402
PCMCIA_PCIC_OPTS, 199
PF6_FORWARD_ACCEPT_DEF, 151
PF6_FORWARD_LOG, 151
PF6_FORWARD_LOG_LIMIT, 151
PF6_FORWARD_N, 151
PF6_FORWARD_POLICY, 150
PF6_FORWARD_REJ_LIMIT, 151
PF6_FORWARD_UDP_REJ_LIMIT, 151
PF6_FORWARD_x, 151
PF6_FORWARD_x_COMMENT, 152
PF6_INPUT_ACCEPT_DEF, 149
PF6_INPUT_ICMP_ECHO_REQ_LIMIT, 150
PF6_INPUT_ICMP_ECHO_REQ_SIZE, 150
PF6_INPUT_LOG, 149
PF6_INPUT_LOG_LIMIT, 149
PF6_INPUT_N, 150
PF6_INPUT_POLICY, 149
PF6_INPUT_REJ_LIMIT, 149
PF6_INPUT_UDP_REJ_LIMIT, 149
PF6_INPUT_x, 150
PF6_INPUT_x_COMMENT, 150
PF6_LOG_LEVEL, 149
PF6_OUTPUT_ACCEPT_DEF, 152
PF6_OUTPUT_LOG, 152
PF6_OUTPUT_LOG_LIMIT, 152
PF6_OUTPUT_N, 153
PF6_OUTPUT_POLICY, 152
PF6_OUTPUT_REJ_LIMIT, 152
PF6_OUTPUT_UDP_REJ_LIMIT, 152
PF6_OUTPUT_x, 153
PF6_OUTPUT_x_COMMENT, 153
PF6_POSTROUTING_N, 154
PF6_POSTROUTING_x, 154
PF6_POSTROUTING_x_COMMENT, 154
PF6_PREROUTING_N, 154
PF6_PREROUTING_x, 154
PF6_PREROUTING_x_COMMENT, 154
PF6_USR_CHAIN_N, 153
PF6_USR_CHAIN_x_NAME, 153
PF6_USR_CHAIN_x_RULE_N, 153
PF6_USR_CHAIN_x_RULE_x, 154

PF6_USR_CHAIN_x_RULE_x_-
COMMENT, 154
PF_FORWARD_ACCEPT_DEF, 56
PF_FORWARD_LOG, 57
PF_FORWARD_LOG_LIMIT, 57
PF_FORWARD_N, 57
PF_FORWARD_POLICY, 56
PF_FORWARD_REJ_LIMIT, 57
PF_FORWARD_UDP_REJ_LIMIT, 57
PF_FORWARD_x, 57
PF_FORWARD_x_COMMENT, 57
PF_INPUT_ACCEPT_DEF, 55
PF_INPUT_ICMP_ECHO_REQ_-
LIMIT, 56
PF_INPUT_ICMP_ECHO_REQ_SIZE,
56
PF_INPUT_LOG, 55
PF_INPUT_LOG_LIMIT, 55
PF_INPUT_N, 56
PF_INPUT_POLICY, 54
PF_INPUT_REJ_LIMIT, 56
PF_INPUT_UDP_REJ_LIMIT, 56
PF_INPUT_x, 56
PF_INPUT_x_COMMENT, 56
PF_LOG_LEVEL, 54
PF_NEW_CONFIG, 44, 399
PF_OUTPUT_ACCEPT_DEF, 58
PF_OUTPUT_CT_ACCEPT_DEF, 71
PF_OUTPUT_CT_N, 71
PF_OUTPUT_CT_x, 71
PF_OUTPUT_CT_x_COMMENT, 71
PF_OUTPUT_LOG, 58
PF_OUTPUT_LOG_LIMIT, 58
PF_OUTPUT_N, 58
PF_OUTPUT_POLICY, 57
PF_OUTPUT_REJ_LIMIT, 58
PF_OUTPUT_UDP_REJ_LIMIT, 58
PF_OUTPUT_x, 58
PF_OUTPUT_x_COMMENT, 58
PF_POSTROUTING_N, 60
PF_POSTROUTING_x, 60
PF_POSTROUTING_x_COMMENT, 60
PF_PREROUTING_CT_ACCEPT_-
DEF, 70
PF_PREROUTING_CT_N, 71
PF_PREROUTING_CT_x, 71
PF_PREROUTING_CT_x_-
COMMENT, 71
PF_PREROUTING_N, 60
PF_PREROUTING_x, 60
PF_PREROUTING_x_COMMENT, 60
PF_USR_CHAIN_N, 59
PF_USR_CHAIN_x_NAME, 59
PF_USR_CHAIN_x_RULE_N, 59
PF_USR_CHAIN_x_RULE_x, 59
PF_USR_CHAIN_x_RULE_x_-
COMMENT, 59
POWERMANAGEMENT, 28
PPP_DEV, 200
PPP_IPADDR, 200
PPP_NETMASK, 200
PPP_NETWORK, 200
PPP_PEER, 200
PPP_SPEED, 200
PPPOE_CHARGEINT, 112
PPPOE_CIRC_N, 400
PPPOE_CIRC_x_CHARGEINT, 400
PPPOE_CIRC_x_DEBUG, 400
PPPOE_CIRC_x_ETH, 400
PPPOE_CIRC_x_FILTER, 400
PPPOE_CIRC_x_HUP_TIMEOUT, 400
PPPOE_CIRC_x_MRU, 400
PPPOE_CIRC_x_MTU, 400
PPPOE_CIRC_x_NAME, 400
PPPOE_CIRC_x_PASS, 400
PPPOE_CIRC_x_TIMES, 400
PPPOE_CIRC_x_TYPE, 400
PPPOE_CIRC_x_USEPEERDNS, 400
PPPOE_CIRC_x_USER, 400
PPPOE_DEBUG, 112
PPPOE_ETH, 115
PPPOE_FILTER, 114
PPPOE_FILTER_EXPR, 114
PPPOE_HUP_TIMEOUT, 112, 116
PPPOE_MRU, 114
PPPOE_MTU, 114
PPPOE_NAME, 111
PPPOE_NF_MSS, 114
PPPOE_PASS, 112
PPPOE_TIMES, 112
PPPOE_TYPE, 115
PPPOE_USEPEERDNS, 111
PPPOE_USER, 112

pptp, 70
PPTP_CHARGEINT, 112
PPTP_CLIENT_LOGLEVEL, 118
PPTP_CLIENT_REORDER_TO, 118
PPTP_DEBUG, 112
PPTP_ETH, 118
PPTP_FILTER, 114
PPTP_FILTER_EXPR, 114
PPTP_HUP_TIMEOUT, 112
PPTP_MODEM_TYPE, 118
PPTP_NAME, 111
PPTP_PASS, 112
PPTP_TIMES, 112
PPTP_USEPEERDNS, 111
PPTP_USER, 112
PRESERVE, 399
PRIVOXY_MENU, 202
PRIVOXY_N, 202
PRIVOXY_x_ACTIONDIR, 202
PRIVOXY_x_ALLOW_N, 202
PRIVOXY_x_ALLOW_x, 202
PRIVOXY_x_CONFIG, 203
PRIVOXY_x_HTTP_PROXY, 203
PRIVOXY_x_LISTEN, 202
PRIVOXY_x_LOGDIR, 203
PRIVOXY_x_LOGLEVEL, 204
PRIVOXY_x_SOCKS_PROXY, 203
PRIVOXY_x_TOGGLE, 203
PXESUBDIR, 279

QOS_CLASS_N, 222
QOS_CLASS_x_DIRECTION, 223
QOS_CLASS_x_LABEL, 224
QOS_CLASS_x_MAXBANDWIDTH, 223
QOS_CLASS_x_MINBANDWIDTH, 223
QOS_CLASS_x_PARENT, 222
QOS_CLASS_x_PRIO, 224
QOS_FILTER_N, 224
QOS_FILTER_x_CLASS, 225
QOS_FILTER_x_IP_EXTERN, 226
QOS_FILTER_x_IP_INTERN, 225
QOS_FILTER_x_OPTION, 227
QOS_FILTER_x_PORT, 226
QOS_FILTER_x_PORT_TYPE, 226
QOS_INTERNET_BAND_DOWN, 221
QOS_INTERNET_BAND_UP, 221
QOS_INTERNET_DEFAULT_DOWN, 221
QOS_INTERNET_DEFAULT_UP, 222
QOS_INTERNET_DEV_N, 221
QOS_INTERNET_DEV_x, 221

RCAPID_PORT, 172
REMOTEHOSTNAME, 278
REMOTEPATHNAME, 278
REMOTEPORT, 279
REMOTEREMOUNT, 279
REMOTEUPDATE, 278
REMOTEUSERNAME, 278
ROUTE_NETWORK, 399

sane, 70
SER_CONSOLE, 30
SER_CONSOLE_IF, 31
SER_CONSOLE_RATE, 31
sip, 70
snmp, 70
SQUEEZE_SCRIPTS, 279
SS5_ALLOW_N, 206
SS5_ALLOW_x, 206
SS5_LISTEN_N, 206
SS5_LISTEN_x, 206
SSH_CLIENT_PRIVATE_KEYFILE_N, 240
SSH_CLIENT_PRIVATE_KEYFILE_x, 240
SSHD_ALLOWPASSWORDLOGIN, 236
SSHD_CREATEHOSTKEYS, 236
SSHD_PORT, 238
SSHD_PUBLIC_KEY_N, 239
SSHD_PUBLIC_KEY_x, 239
SSHD_PUBLIC_KEYFILE_N, 239
SSHD_PUBLIC_KEYFILE_x, 239
SSHKEYFILE, 279
START_IMOND, 72
STUNNEL_DEBUG, 215
STUNNEL_N, 215
STUNNEL_x_ACCEPT, 216
STUNNEL_x_ACCEPT_IPV4, 216
STUNNEL_x_ACCEPT_IPV6, 217
STUNNEL_x_CERT_CA_FILE, 218
STUNNEL_x_CERT_FILE, 217
STUNNEL_x_CERT_VERIFY, 218

STUNNEL_x_CLIENT, 215
STUNNEL_x_CONNECT, 217
STUNNEL_x_DELAY_DNS, 217
STUNNEL_x_NAME, 215
STUNNEL_x_OUTGOING_IP, 217
SYSLOGD_DEST_N, 76
SYSLOGD_DEST_x, 76
SYSLOGD_RECEIVER, 76
SYSLOGD_ROTATE, 77
SYSLOGD_ROTATE_AT_-
SHUTDOWN, 78
SYSLOGD_ROTATE_DIR, 78
SYSLOGD_ROTATE_MAX, 78

TELMOND_CAPI_CTRL_N, 171
TELMOND_CAPI_CTRL_x, 172
TELMOND_CMD_N, 170
TELMOND_CMD_x, 170
TELMOND_LOG, 170
TELMOND_LOGDIR, 170
TELMOND_MSN_N, 170
TELMOND_MSN_x, 170
TELMOND_PORT, 169
tftp, 70
TFTP_PATH, 109
TFTPBOOTIMAGE, 279
TFTPBOOTPATH, 279
TIME_INFO, 27
TOR_ALLOW_N, 205
TOR_ALLOW_x, 205
TOR_CONTROL_PASSWORD, 205
TOR_CONTROL_PORT, 205
TOR_DATA_DIR, 205
TOR_HTTP_PROXY, 205
TOR_HTTP_PROXY_AUTH, 205
TOR_HTTPS_PROXY, 205
TOR_HTTPS_PROXY_AUTH, 205
TOR_LISTEN_N, 205
TOR_LISTEN_x, 205
TOR_LOGFILE, 206
TOR_LOGLEVEL, 205
TRANSPROXY_ALLOW_N, 207
TRANSPROXY_ALLOW_x, 207
TRANSPROXY_LISTEN_N, 207
TRANSPROXY_LISTEN_x, 207
TRANSPROXY_TARGET_IP, 207
TRANSPROXY_TARGET_PORT, 207
TRUSTED_NETS, 399

UMTS_ADAPTER, 248
UMTS_APN, 247
UMTS_CHARGEINT, 247
UMTS_CTRL, 249
UMTS_DEBUG, 246
UMTS_DEV, 249
UMTS_DIALOUT, 247
UMTS_DRV, 248
UMTS_FILTER, 248
UMTS_GPRS_UMTS, 247
UMTS_HUP_TIMEOUT, 247
UMTS_IDDEVICE, 248
UMTS_IDDEVICE2, 248
UMTS_IDVENDOR, 248
UMTS_IDVENDOR2, 248
UMTS_NAME, 247
UMTS_PASSWD, 247
UMTS_PIN, 246
UMTS_SWITCH, 248
UMTS_TIMES, 247
UMTS_USEPEERDNS, 248
UMTS_USER, 247
USB_EXTRA_DRIVER_N, 250
USB_EXTRA_DRIVER_x, 250
USB_EXTRA_DRIVER_x_PARAM,
250
USB_LOWLEVEL, 403
USB_MODEM_WAITSECONDS, 250

VERBOSE, 278
VLAN_DEV_N, 86
VLAN_DEV_x_DEV, 86
VLAN_DEV_x_VID, 86
VPN_CARD_TYPE, 141

WGET_SSL, 403
WLAN_N, 252
WLAN_REGDOMAIN, 252
WLAN_WEBGUI, 252
WLAN_x_ACL_MAC_N, 255
WLAN_x_ACL_MAC_x, 255
WLAN_x_ACL_POLICY, 255
WLAN_x_AP, 255
WLAN_x_BRIDGE, 256
WLAN_x_CHANNEL, 253
WLAN_x_DIVERSITY, 255

WLAN_x_DIVERSITY_RX, 256
WLAN_x_DIVERSITY_TX, 256
WLAN_x_ENC_ACTIVE, 254
WLAN_x_ENC_MODE, 254
WLAN_x_ENC_N, 253
WLAN_x_ENC_x, 254
WLAN_x_ESSID, 252
WLAN_x_MAC, 252
WLAN_x_MAC_OVERRIDE, 252
WLAN_x_MODE, 252
WLAN_x_NOESSID, 253
WLAN_x_PSKFILE, 256
WLAN_x_RATE, 253
WLAN_x_RTS, 253
WLAN_x_WPA_DEBUG, 255
WLAN_x_WPA_ENCRYPTION, 255
WLAN_x_WPA_KEY_MGMT, 254
WLAN_x_WPA_PSK, 254
WLAN_x_WPA_TYPE, 255
WLAN_x_WPS, 256

Y2K_DAYS, 79
YADIFA_ALLOW_QUERY_N, 110
YADIFA_ALLOW_QUERY_x, 110
YADIFA_LISTEN_N, 110
YADIFA_SLAVE_ZONE_N, 110
YADIFA_SLAVE_ZONE_x, 110
YADIFA_SLAVE_ZONE_x_ALLOW_-
QUERY_N, 110
YADIFA_SLAVE_ZONE_x_ALLOW_-
QUERY_x, 110
YADIFA_SLAVE_ZONE_x_MASTER,
110