Paket ADVANCED_NETWORKING
Version 3.10.4

Frank Meyer
E-Mail: frank@fli4l.de

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

25. Oktober 2015

Inhaltsverzeichnis

1 Dokumentation des Paketes ADVANCED_NETWORKING

1.1 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.

1.1.1 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'
```

1.1.2 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.

1.1.3 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.

1.1.4 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'
```

1.1.5 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 × 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 × 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 1.1 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 1.1: Werte für BRIDGE_DEV_x_DEV_x_PATHCOST in Abhängigkeit von der Bandbreite

1.1.6 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.

1.1.7 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.

1.1.8 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

1.1.9 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.

Abbildungsverzeichnis

Tabellenverzeichnis

Index