

# **Paket QOS**

## **Version 3.10.4**

Frank Meyer                      Das fli4l-Team  
E-Mail: [frank@fli4l.de](mailto:frank@fli4l.de)      E-Mail: [team@fli4l.de](mailto:team@fli4l.de)

25. Oktober 2015

# Inhaltsverzeichnis

|   |           |
|---|-----------|
| <b>1. Dokumentation des Paketes QOS</b> | <b>3</b>  |
| 1.1. QoS - Quality of Service . . . . . | 3         |
| 1.1.1. Konfiguration . . . . .          | 3         |
| 1.1.2. Anwendungsbeispiele . . . . .    | 11        |
| <b>A. Anhang zum Paket QOS</b>          | <b>19</b> |
| <b>Abbildungsverzeichnis</b>            | <b>20</b> |
| <b>Tabellenverzeichnis</b>              | <b>21</b> |
| <b>Index</b>                            | <b>22</b> |

# 1. Dokumentation des Paketes QOS

## 1.1. 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 bisschen 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.

### 1.1.1. 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 3).

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 stoppen. 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 Variablen `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` entsprechend 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 aktivieren. 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 asymmetrischen 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 Zielpport 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 in 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)

### 1.1.2. 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.

## 1. Dokumentation des Paketes QOS

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 1.1 dargestellte Szenario.

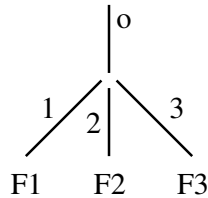


Abbildung 1.1.: Beispiel 1

Die Konfiguration sieht dann so aus:

Drei Rechner nach IP gefiltert die je 1/3 bekommen falls kein Gast anwesend ist:

```
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.

## 1. Dokumentation des Paketes QOS

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 1.1 dargestellte Szenario.

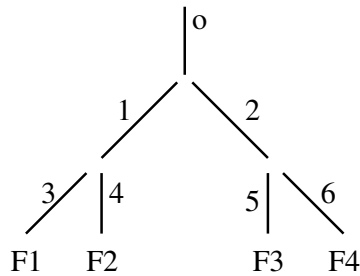


Abbildung 1.2.: Beispiel 2

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'
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.





In dem Bild sind jeweils die Bereiche eingekreist welche zusammengehören. 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 keinsten 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'
```

## 1. Dokumentation des Paketes QOS

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.

## **A. Anhang zum Paket QOS**

# Abbildungsverzeichnis

|                 |    |
|-----------------|----|
| 1.1. Beispiel 1 | 12 |
| 1.2. Beispiel 2 | 14 |
| 1.3. Beispiel 3 | 16 |

# **Tabellenverzeichnis**

# Index

OPT\_QOS, [3](#)

QOS\_CLASS\_N, [4](#)

QOS\_CLASS\_x\_DIRECTION, [6](#)

QOS\_CLASS\_x\_LABEL, [7](#)

QOS\_CLASS\_x\_MAXBANDWIDTH, [5](#)

QOS\_CLASS\_x\_MINBANDWIDTH, [5](#)

QOS\_CLASS\_x\_PARENT, [5](#)

QOS\_CLASS\_x\_PRIO, [6](#)

QOS\_FILTER\_N, [7](#)

QOS\_FILTER\_x\_CLASS, [7](#)

QOS\_FILTER\_x\_IP\_EXTERN, [8](#)

QOS\_FILTER\_x\_IP\_INTERN, [8](#)

QOS\_FILTER\_x\_OPTION, [9](#)

QOS\_FILTER\_x\_PORT, [8](#)

QOS\_FILTER\_x\_PORT\_TYPE, [9](#)

QOS\_INTERNET\_BAND\_DOWN, [3](#)

QOS\_INTERNET\_BAND\_UP, [4](#)

QOS\_INTERNET\_DEFAULT\_DOWN, [4](#)

QOS\_INTERNET\_DEFAULT\_UP, [4](#)

QOS\_INTERNET\_DEV\_N, [3](#)

QOS\_INTERNET\_DEV\_x, [3](#)