Quality of Service

Howto QOS unter Linux mit Iptables-Netfilter

Quality of Service

0 - Einführung

Ziel: Eine bestehende Datenleitung soll für die unterschiedlichen Dienste optimiert werden, sodass aus Nutzersicht alle Dienste gut funktionieren. Dazu müssen manche Datenpakete vorrangig abgefertigt werden, andere in der Geschwindigkeit reduziert werden.

Beispiel: IP-Telefonie ist auf einen schnellen Fluss relativ kleiner Datenmengen angewiesen, gleichzeitige grosse Up- oder Downloads können Gespräche stören oder unterbrechen.
Gelingt es den Telefonie-Daten eine Vorzugsbehandlung zu sichern, kann auch der grosse Datenverkehr gleichzeitig, ohne zu stören weiterlaufen.

Wichtigster Ansatzpunkt für Qos ist folgender Sachverhalt: TCP-Datenfluss ist eine geordnete Kommunikation bei der der Empfänger dem Sender den ordnungsgemässen Eingang der Datenpakete bestätigt und der Sender entsprechend im Fehlerfall die Sendung wiederholt bzw. die nächsten Daten schickt. Bei stark belasteten („verstopften“) Leitungen kommen häufig die Bestätigungsmeldungen beim Sender so spät an, dass er von einem Datenverlust ausgeht und bereits gesendete Pakete nochmal schickt, was wiederum die Belastung der Leitung zusätzlich erhöht und sich immer weiter aufschaukeln kann (Bestätigung kommt noch später, Daten werden nochmal gesendet usw. usw.). Die wirkungsvollste Massnahme ist daher immer die vorrangige Behandlung dieser Bestätigungspakete („syn/ack“).

Iptables/Netfilter beinhaltet dazu alle nötigen Werkzeuge, allerdings mit schwer verständlicher Syntax.

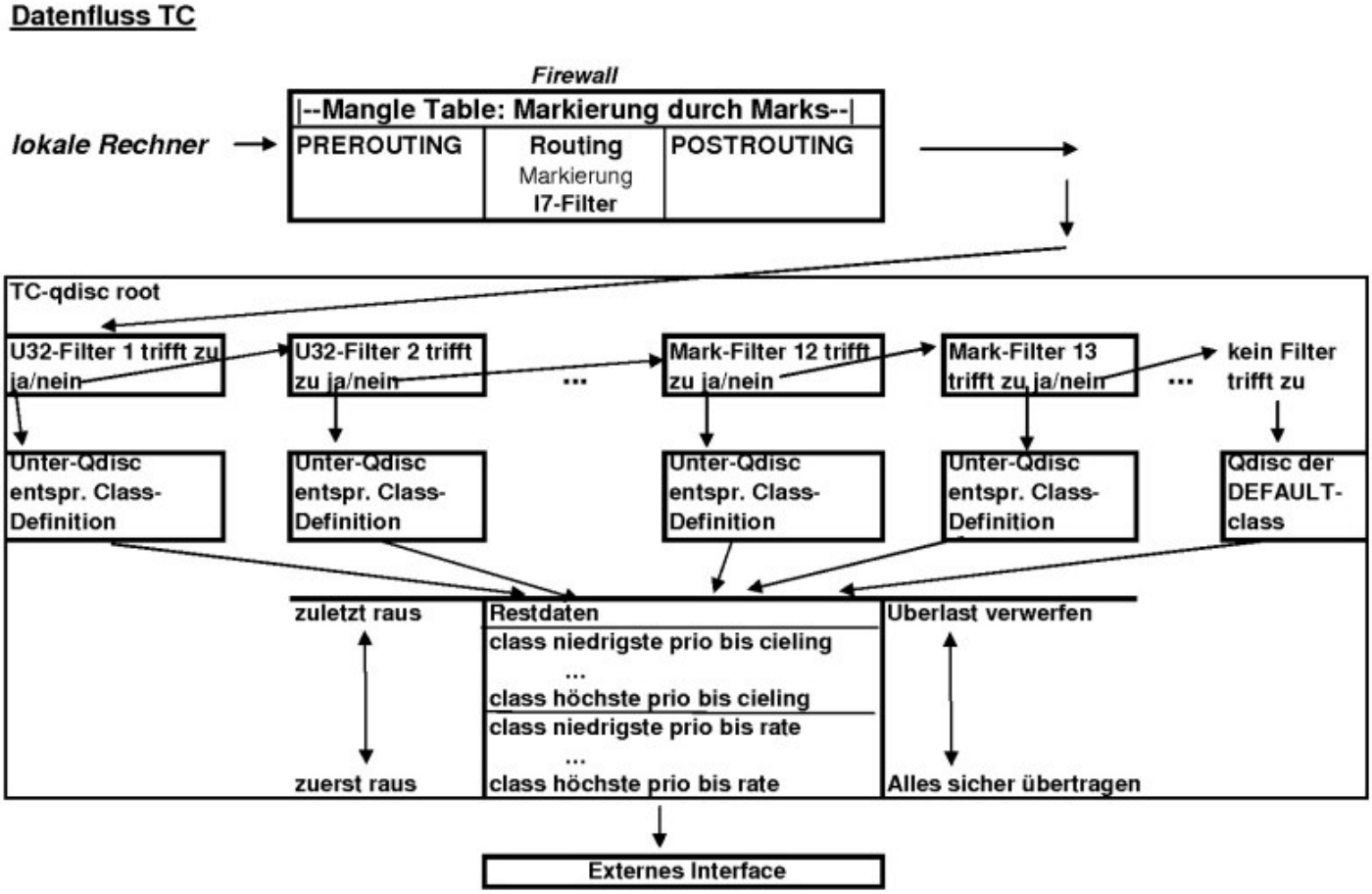
1) Es werden Puffer/Warteschlangen definiert.
2) Den Puffern werden Traffic-Klassen mit unterschiedlichen Bandbreiten und Prioritäten zugeordnet.
3) Filter sortieren die Daten in die einzelnen Traffic-Klassen.
4) Um zu erkennen, um welche Arten Daten es sich handelt gibt es vor allem 3 Methoden:
   a) u32-Paket-Analyse (siehe hierzu http://www.stearns.org/doc/iptables-u32.v0.1.7.html)
   b) Markierung der Daten in der Firewall über mangle-table je nach verwendeten Ports angesprochener IP oder was auch immer
   c) L7-Filter (Paketanalyse auf Layer 7, siehe http://l7-filter.sourceforge.net/)

QoS sollte nur am Upload (TC-Ausdruck: „root“) ansetzen, da beim Download (TC-Ausdruck: „ingress“) Datenpakete, die die Leitung bereits passiert haben, weggeworfen werden, entsprechend nochmal gesendet werden und damit für zusätzliche Last auf der Leitung sorgen.

Die maximale Datenrate sollte etwas kleiner als tatsächlich vorhanden gewählt werden, um zu verhindern, dass Daten sich auf dem Weg in anderen Puffern, z.B. Cisco-Routern, Switches stauen, da dort Qos nicht mehr wirken kann.

Quality of Service

1 – Datenfluss TC



2 - Beispiel in 7 Schritten

Jetzt wird's ernst (Beispiel für 2Mbit-Leitung mit 6 Klassen)

Voraussetzungen:
Iptables-Firewall, Netfilter Module und iproute2 tools installiert und Module geladen:
z.B. in der Firewall:

```
/sbin/modprobe ip_conntrack_ftp
/sbin/modprobe ip_nat_ftp
/sbin/modprobe ip_conntrack_netlink
Sauberer table mangle:
```

iptables -t mangle -F POSTROUTING
iptables -t mangle -F OUTPUT

Planung:

1) Welche Traffic-Klassen sollen unterschieden werden?
   1.1- Alles was schnell geht = technische Pakete
   1.2- Webseiten und Mails
   1.3- Telefon
   1.4- Online Spiele
   1.5- Nicht klassifizierte Daten
   1.6-File-Sharing
2) Wieviel Bandbreite für welche Klasse reservieren?
   a) Hauptklasse etwas kleiner: **`rate 1900kbit ceil 1950`**
   b) Unterklassen sollten Größenordung repräsentieren.
   überbuchen schadet nichts, wenn man im Betrieb auf „dropped"-Pakete kontrolliert

Schritt 1: Haupt-Warteschlange erstellen

```
/sbin/tc qdisc add dev Externes_Interface root handle 1:0 htb default 11

#das heisst:
#- Puffer hinzufügen
#- zu externem Interface für Upload=root
#- Puffer soll jetzt 1:0 heissen
#(Anmerkung: Die erste Zahl ist eine beliebige Ziffer. „1:0" und „1:"  sind äquivalent und bedeuten
#„es handelt sich um eine qdisk". Steht hinter dem Doppelpunkt anderes als 0 oder nichts handelt es
#sich um eine Klasse).
#- die Qdisk sei vom Typ htb=hierarchical token bucket (es gibt auch andere)
#- die Klasse für default = unklassifizierte Daten wollen wir unten 11  nennen.
```

Quality of Service

Schritt 2: Hauptklasse definieren

```
/sbin/tc class add dev Ext_Interface parent 1:0 classid 1:1  htb rate 1900kbit ceil 1950 prio 0

#das heisst:
#-füge Klasse hinzu
#-z.B. für dev eth1
#-sie gehört zu handle 1:0 = Haupt-Warteschlange
#-die Klassenid sei 1:1
#-die Qdisk sei vom Typ htb=hierarchical token bucket (es gibt auch andere)
#-“garantierte“ Bandbreite = 1900
#-maximale Bandbreite 1950
#- natürlich vor den Unterklassen abarbeiten
```

Schritt 3: Unter-Klassen definieren

```
# Traffic-Klasse 10 alles was schnell geht syn/ack, ping usw. + MARK 10
/sbin/tc class add dev Ext_Interface parent 1:1 classid 1:10 htb rate 500kbit ceil 1900kbit prio 1
#T11 Default Klasse - alle nicht klassifizierten Daten
/sbin/tc class add dev Ext_Interface parent 1:1 classid 1:11 htb rate 700kbit ceil 900kbit prio 5
#T12 Telefonierer
/sbin/tc class add dev Ext_Interface parent 1:1 classid 1:12 htb rate 1000kbit ceil 1500bit prio 3
#T13 Spieler
/sbin/tc class add dev Ext_Interface parent 1:1 classid 1:13 htb rate 1000kbit ceil 1500kbit prio 4
#T14 File Sharer
/sbin/tc class add dev Ext_Interface parent 1:1 classid 1:14 htb rate 300kbit ceil 800kbit prio 6
#T15 Test z.Z. http pop usw.
/sbin/tc class add dev Ext_Interface parent 1:1 classid 1:15 htb rate 500kbit ceil 1950kbit prio 2

#das heisst:
#-füge Klasse hinzu
#-z.B. für dev eth1
#-sie ist abgeleitet von handle 1:1 = Haupt-Klasse
#-die Klassenid sei 1:1X
#-die Qdisk sei vom Typ htb=hierarchical token bucket (es gibt auch andere)
#-rate = “garantierte“ Bandbreite
#-ceil = maximale Bandbreite
#- Prio gibt hier an, welche Daten zuerst abgearbeitet werden,also http zuerst, file-sharing
#zuletzt
```

Schritt 4: Unter-Warteschlangen erstellen

```
    # für T10
    /sbin/tc qdisc add dev Ext_Interface parent 1:10  handle 10: sfq perturb 10
    # für T11
    /sbin/tc qdisc add dev Externes_Interface parent 1:11  handle 11: sfq perturb 10
    # für T12
    /sbin/tc qdisc add dev Externes_Interface parent 1:12  handle 12: sfq perturb 10
    # für T13
    /sbin/tc qdisc add dev Externes_Interface parent 1:13  handle 13: sfq perturb 10
    # für T14
    /sbin/tc qdisc add dev Externes_Interface parent 1:14  handle 14: sfq perturb 10
    # für T15
    /sbin/tc qdisc add dev Externes_Interface parent 1:15  handle 15: sfq perturb 10
```

Quality of Service

```
#das heisst:
#-füge Puffer hinzu
#-für externes Interface z.B. für dev eth1
#-parent = 1:15, d.h. hier kommen die Daten von der Klasse mit dem handle 1:15 rein
# (=Klasse 5, http)
#-handle 15: der Name der Queue sei 15: bzw. 15:0
#-sfq perturb 10 = Stochastic Fair Queuing = a) Für jede Verbindung wird ein FIFO angelegt. Alle
Fifos werden nach Round-Robin abwechselnd bedient. Um die Bandbreite gleichmässig zu verteilen wird
die Round-Robin-Reihenfolge alle 10 Sec. durchgemischt.
```

Schritt 5: Filter definieren.

```
#Achtung Filter werden nach Prio abgearbeitet - da alle class 1:0
    #deshalb gilt z.B. bei Personen/IP-Markierung für User aus Traffic-Klasse 14=Filesharer
    # mit Mark 14:
    #VoIP volle Leistung Filter-Prio 1
    #auch TOS ICMP SYN-ACK Filter greifen,  da Prio = 2 und
    #der Mark 14 - Filter später  greift =  Prio 14.
    #U32-Filter= Prio bedeutsam, bei Mark-Filtern ist prio egal, da die Pakete
    #nur 1 Mark haben könnn
    #Marks < 10 für L7-Filter reserviert, 0-3 nicht benutzen
```

a) die wichtigsten U32-Filter

```
#(Zitat aus „cookbook“ ) :
    # Pakete mit TOS=Time of Service, Minimum Delay,  (ssh, NOT scp) in 1:10:
    /sbin/tc filter add dev Ext_Interface parent 1:0 protocol ip prio 2 u32 \
            match ip tos 0x10 0xff  flowid 1:10

     # ICMP (ip protocol 1) in the interactive class 1:10 so we
     # can do measurements & impress our friends:
     /sbin/tc filter add dev Ext_Interface parent 1:0 protocol ip prio 3 u32 \
             match ip protocol 1 0xff flowid 1:10

     # To speed up downloads while an upload is going on, put ACK packets in
     # the interactive class:

     /sbin/tc filter add dev Ext_Interface parent 1:0 protocol ip prio 4 u32 \
                match ip protocol 6 0xff \
                   match u8 0x05 0x0f at 0 \
                      match u16 0x0000 0xffc0 at 2 \
                         match u8 0x10 0xff at 33 \
                            flowid 1:10
#d.h.
#-Filter hinzufügen
#-für externes Interface
#-parent1:0 =für Daten, die  aus der Hauptwarteschlange kommen
#-nur Daten die der angegebenen u32-Klassifikation entsprechen
#siehe  http://www.stearns.org/doc/iptables-u32.v0.1.7.html
#prio 4 = diese Regel wird als 4. angewendet.
#Ist eine Regel zutreffend, gehen die Daten den angegebenen Weg.
#Weitere Regeln werden nicht geprüft.
```

```
#D.h. z.B. für Gnutella-Anwender: Deren Upload landet weiter unten in dem Filter mit prio 14
#und damit in der langsameren Traffic-Klasse 14. Ihr Download wird aber optimiert,
#da ihre syn/ack -Pakete nicht in der Masse der Upload-Daten hängen, sondern schon von dem Filter
#mit prio 4 in die schnellste Traffic-Klasse geschoben werden.
#flowid 1:10 = in die schnellste Traffic-Klasse 1:10 leiten

#VOIP (nach http://www.ruwenzori.net/code/wondershaper/wondershaper.jml
#Voip (+Half Live) über ports
    #4569 = IAX , 2427  =MGCP,5004 =rtp, 5060 =SIP, 5061 =SIP-tls,
    #27015+27030 = Half Live, valve source game engine
for port in  2427 4569 5004 5060 5061 27015 27030
 do
 /sbin/tc filter add dev $interface protocol ip parent 1:0 prio 1 u32 match ip dport $port 0xffff \
             match ip protocol 0x11 0xff flowid 1:12
 /sbin/tc filter add dev $interface protocol ip parent 1:0 prio 1 u32 match ip sport $port 0xffff \
             match ip protocol 0x11 0xff flowid 1:12
 done
#VOIP über tos
    /sbin/tc filter add dev $interface protocol ip parent 1:0 prio 1 u32 match ip tos 0x68 0xff\
                    match ip protocol 0x11 0xff  flowid 1:12
    /sbin/tc filter add dev $interface protocol ip parent 1:0 prio 1 u32 match ip tos 0xb8 0xff\
                    match ip protocol 0x11 0xff  flowid 1:12
#VOIP über Marks (siehe unten)
    #um ggf. zwischen udp und tcp unterscheiden zu koennen: Umweg über mark 12
    #/sbin/tc filter add dev $interface parent 1:0 prio 1 protocol ip handle 12 fw flowid 1:12
```

b) Mangle Filter

(Anmerkung: Da L7 die Marks 0-3 intern und 4-9 per default benutzt, hier nur Marks >=10)

```
# Traffic-Klasse 1
/sbin/tc filter add dev $interface parent 1:0 prio 20 protocol ip handle 11 fw flowid 1:11
#Traffic-Klasse 2
    #VOIP
    #um zwischen udp und tcp unterscheiden zu können: Umweg über mark 12
    #VOIP-udp-Pakete werden unten (siehe Schritt 6: Firewall) mit Mark 12 markiert
/sbin/tc filter add dev $interface parent 1:0 prio 12 protocol ip handle 12 fw flowid 1:12
# Traffic-Klasse 3
/sbin/tc filter add dev $interface parent 1:0 prio 13 protocol ip handle 13 fw flowid 1:13
# Traffic-Klasse 4
/sbin/tc filter add dev $interface parent 1:0 prio 14 protocol ip handle 14 fw flowid 1:14
# Traffic-Klasse 5
/sbin/tc filter add dev $interface parent 1:0 prio 15 protocol ip handle 15 fw flowid 1:15

#d.h:
#-Filter für externes Interface hinzufügen, parent 1:0=Daten kommen aus Haupt-Warteschlange
#-prio 15 = Regel wird als letzte geprüft
#-handle 15 = alle Pakete mit der Markierung 15 ausfiltern
#-flowid 15 = in Traffic-Klasse 15 = http schicken.
```

c) L7-Filter. L7-Filterung:

Ist (rechen-)aufwändig und macht nur bei DSL-Anbindung wirklich Sinn. Ich habe L7 anfangs zur

Identifikation von file-sharing eingesetzt, da Identifikation per Port ziemlich sinnlos ist.
Inzwischen gibt es aber von mir 2 Scripte, die viel einfacher, entweder eine einzelne Verbindung oder aber einen Benutzer komplett (das kann nur eine Notlösung sein), in die Traffic-Klasse filesharing einordnen,

Schritt 6: Firewall Einträge für Mangle schaffen - Beispiele

#Beispiel: böse Buben bekommen 14, Zocker 13//aber: Nutzer-bezogene Einträge sind nicht gut
#/sbin/iptables -A POSTROUTING -t mangle -s 10.10.180.130 -j MARK --set-mark 13

#Gnutella
```
for PORT in 6346:6347 1214 4661 4662
do
   /sbin/iptables -t mangle -A POSTROUTING -p tcp --dport $PORT -j MARK --set-mark 14
done
for PORT in 6346:6347 4665
do
   /sbin/iptables -t mangle -A POSTROUTING -p udp --dport $PORT -j MARK --set-mark 14
done
```

#world of warcraft
```
for PORT in 3724 6112 6881:6999
do
   /sbin/iptables -t mangle -A PREROUTING -p tcp --dport $PORT -j MARK --set-mark 13
done
```

#Bittorrent
```
/sbin/iptables -t mangle -A FORWARD -p tcp --dport 19945 -j MARK --set-mark 14
/sbin/iptables -t mangle -A FORWARD -p udp --dport 19945 -j MARK --set-mark 14
```

#HTTP(S)
```
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 80 -j MARK  --set-mark 15
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 443 -j MARK --set-mark 15
```
#die folgenden Einträge müssen bei mehreren externen Interfaces (Leitungsbündelung) das gleiche Mark wie http bekommen
#messenger
#msn: 80; 443;1853?; 7001
#yahoo: 3478
```
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 3478 -j MARK --set-mark 12
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 7001 -j MARK --set-mark 12
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 1853 -j MARK --set-mark 12
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 1863 -j MARK --set-mark 12
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 53 -j MARK --set-mark 12
/sbin/iptables -t mangle -A PREROUTING -p tcp --dport 1935 -j MARK --set-mark 12
```

#SIP (siehe auch oben die entsprechendes als U34-Regel)
```
/sbin/iptables -t mangle -A PREROUTING  -p udp --sport 5060:5069 -j MARK --set-mark 12   # SIP High
```

Quality of Service

/sbin/iptables -t mangle -A PREROUTING -p udp --dport 5060:5069 -j MARK --set-mark 12 # SIP High

/sbin/iptables -t mangle -A PREROUTING -p udp --sport 4569 -j MARK --set-mark 12 # IAX High
/sbin/iptables -t mangle -A PREROUTING -p udp --dport 4569 -j MARK --set-mark 12 # ""

/sbin/iptables -t mangle -A PREROUTING -p udp --sport 10000:11000 -j MARK --set-mark 12 # RTP High
/sbin/iptables -t mangle -A PREROUTING -p udp --dport 10000:11000 -j MARK --set-mark 12 # ""

#l7 Daten zuführen

```
#Dies nur wenn L7 eingesetzt wird

#Variante 1 nur Pakete ohne Mark
#/sbin/iptables -t mangle -A FORWARD -m mark --mark 11 -j ACCEPT
#/sbin/iptables -t mangle -A FORWARD -m mark --mark 12 -j ACCEPT
#/sbin/iptables -t mangle -A FORWARD -m mark --mark 13 -j ACCEPT
#zum Prüfen auf Funktion
#/sbin/iptables -t mangle -A FORWARD -m mark --mark 14 -j ULOG  --ulog-prefix "Mark 14 Fileshare:"
--ulog-nlgroup 3
#/sbin/iptables -t mangle -A FORWARD -m mark --mark 14 -j ACCEPT
#/sbin/iptables -t mangle -A FORWARD -m mark --mark 15 -j ACCEPT
#/sbin/iptables -t mangle -A FORWARD -j QUEUE

#Variante 2: Alle Pakete
#/sbin/iptables -t mangle -A FORWARD -j QUEUE
```

Wann Prerouting, Forward oder Postrouting?
Die Reihenfolge der Mark-Regeln ist erheblich, da frühere Regeln von späteren überschrieben werden können. Mark-Regeln werden von POSTROUTING überschreiben ebenso FORWARD und PREROUTING, FORWARD überschreibt PREROUTING. Bei L7 ist zu entscheiden: a) Soll L7 vorhandene Regeln überschreiben? b) Sollen L7-marks von später in FORWARD oder POSTROUTING überschrieben werden.

Schritt 7:
Funktion prüfen

```
tc -s qdisc
tc qdisc show dev ethx
tc class show dev ethx
tc filter show dev ethx
```

Quality of Service

3 - **Definitionsstruktur TC**

```
(Verwirrend sind die parent Angaben: erster parent ist root-qdisc, danach classes)

Gesamt-Datenpuffer - Handle:      1:0          root qdisc (handle=1,keine class =0)
                                   |
Hauptklasse - Classid:            1:1          (gehört zu qdisc 1, class-nr 1,
                                 /  |  \        parent qdisc =1:0, prio =0
                                /   |   \
                               /    |    \
                              /     |     \
Unterklassen - Classids:    1:10  1:11  1:12   (gehört zu qdisc 1, class-nr:10-12,
                             |      |     |     parent class 1:1
                             |      |     |     prio beliebig gestaffelt)
                             |      |     |
Unter-Datenpuffer-Handle: 10:0    11:   12:0    (qdisc, handle=10-12, keine class =0,
                                                 parents classes 1:10-1:12)
-------------------------------------------------------------------------------------
Filter                                         parent immer 1:0 =root qdisc
                                               prio regelt Reihenfolge der Filter
                                               flowid 1:10-1:12=Klassenzuordnung
                                               handle = xx = Pakete mit Mark xx filtern
```

4 - Literatur

http://lartc.org/howto/index.html
http://lartc.org/howto/lartc.cookbook.ultimate-tc.html
http://linux-ip.net/articles/Traffic-Control-HOWTO/index.html
http://www.ruwenzori.net/code/wondershaper/wondershaper.jml
speziell:
http://linux-ip.net/articles/Traffic-Control-HOWTO/software.html
zu u32:
http://www.stearns.org/doc/iptables-u32.v0.1.7.html
Turorial: http://www.kabelverhau.ch/elwms/de_shaping.php
http://www.gentooforum.de/artikel/14671/qos-mit-iptables-tc.html
http://www.little-idiot.de/linuxsolutionguide/firewall-qos.htm

5 - Kleine Portliste

```
# Filesharing
# 20 ftp-data
# 25 smtp
# 989 ftps-data
# 1080 gnunet
# 1194 openvpn
# 1214 fasttrack
# 1412 directconnect
# 2086 gnunet
# 2234 soulseek
# 3364 mldonkey
# 4444 directconnect
# 4662 edonkey
# 4666 edonkey
# 4672 edonkey
# 4949 openvpn
# 6346 gnutella
# 6347 gnutella 2
# 6699 opennapster
# 6881 bittorrent
# 6882 bittorrent
# 8436 freenet
# 9001 tor
# 9030 tor
# 9050 tor
# 9100 tor
# 9999 opennapster
# 44646 openvpn
# Source : http://mldonkey.sourceforge.net/WhatFirewallPortsToOpen

#VOIP
# 2427 mgcp
# 4569 iax
# 5004 rtp
# 5060 sip
# 5061 sip-tls
#Spiele
# 27015 valve source game engine
# 27030 valve source game engine
```