



# **Traffic Control & QoS mit Linux 2.4**

Heiko Reese

# Traffic Control & QoS mit Linux

- ? Was?
- ? Warum?
- ? Wie?
- ? Wie genau?
- ? So geht das (Beispiele)!
- ? Und was gibt's da sonst noch?

# Was?

- ? Linux bringt fast alles mit für denjenigen, der seinen Traffic
  - ? begrenzen will
  - ? einteilen will
  - ? an spezielle Anforderungen bezüglich Bandbreite oder Prioritäten anpassen will
- ? es existieren noch weitere alternative Möglichkeiten, Traffic zu beeinflussen (z.B. diffserv, userspace-shaping deamons)

# Warum?

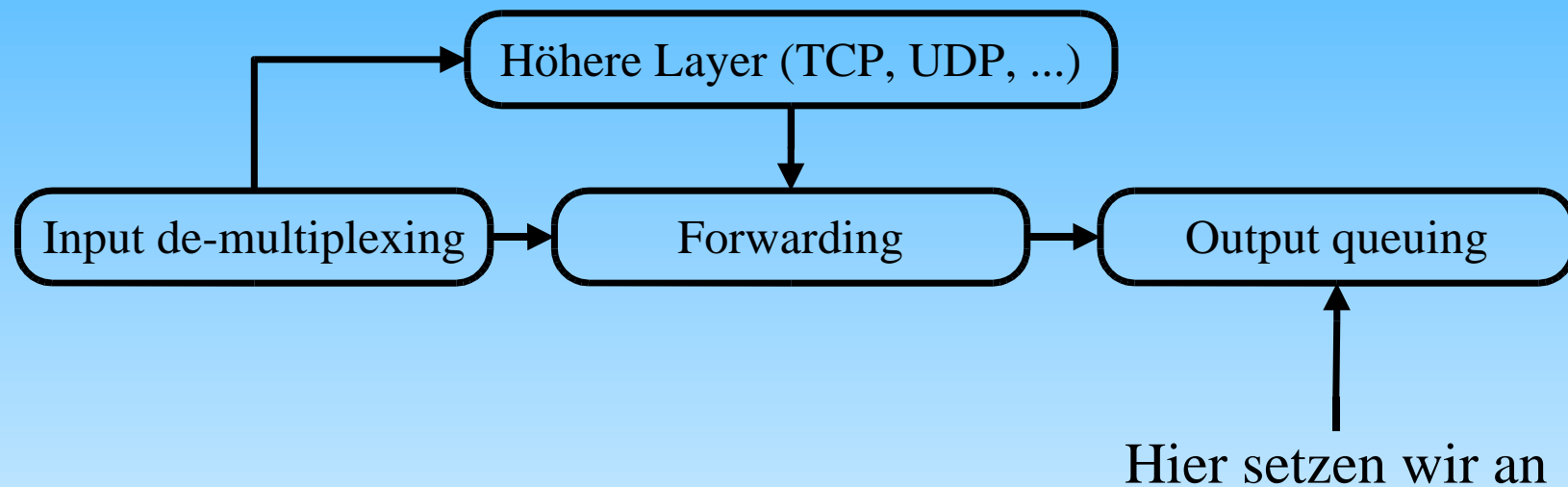
- ? Kosten bei entsprechenden Tarifen
- ? Fairness bei mehreren Benutzern sicherstellen
- ? Komfort (z.B. SSH priorisieren, damit's nicht so träge ist)
- ? Andere Leute ärgern
- ? Fun
- ? ...

# Wie?

- ? QoS beim kompilieren des Kernels aktivieren
- ? ändern der Parameter mittels `tc` irgendwo in den Startup-Skripten
- ? evtl. noch an der Firewall basteln

# Wie genau?

? Weg eines Pakets durch den Kernel:



? Wir können nur Pakete beeinflussen, die wir selber verschicken!

# Wie genau #2?

*Zitat aus dem „Linux Advanced Routing & Traffic Control HOWTO“:*

With the way the internet works, we have no direct control of what people send us. It's a bit like your (physical!) mailbox at home. There is no way you can influence the world to modify the amount of mail they send you, short of contacting everybody.

However, the internet is mostly based on TCP/IP which has a few features that help us. TCP/IP has no way of knowing the capacity of the network between two hosts, so it just starts sending data faster and faster ('slow start') and when packets start getting lost, because there is no room to send them, it will slow down. In fact it is a bit smarter than this, but more about that later.

This is the equivalent of not reading half of your mail, and hoping that people will stop sending it to you. With the difference that it works for the Internet :-)

# Queuing Disciplines

- ? Eine qdisc sendet „in letzter Instanz“ die Pakete für ein Netzinterface
- ? Es gibt simple „classless“ qdiscs und verschachtelbare „class-based“ qdiscs
- ? letzere können mit der Firewall (ipchains, iptables) kombiniert werden



# Classless qdiscs

- ? Pakete werden nur:
  - ? weitergeleitet
  - ? Umsortiert
  - ? verzögert
  - ? verworfen
- ? Jedem Netzinterface wird standardmässig die qdisc **pfifo\_fast** zugewiesen

# Classless qdisc: pfifo\_fast

- ? eine einfache FIFO, die aus mehreren priorisierten Bändern („bands“) besteht
- ? Pakete werden nach einer einfachen „priority map“ (TOS-Flag wird überprüft) einsortiert
- ? solange noch Pakete in Band 0 vorhanden sind, werden Band 1... nicht abgearbeitet
- ? die Anzahl der Bänder und die „priority map“ können mit **tc** verändert werden, die Länge der Bänder muss beim Anlegen des Netzinterfaces angegeben werden (also mit **ifconfig** oder **ip**)

# Classless qdisc: Token Bucket Filter

- ? Pakete werden nur bis zu einem bestimmten Datendurchsatz weitergeleitet, alles darüber wird verworfen
- ? Kleine Bursts sind erlaubt
- ? tbf ist sehr präzise sowie netzwerk- und prozessorfreundlich. Es stellt die beste Möglichkeit dar, die Geschwindigkeit eines Netzinterfaces zu verringern.
- ? Algorithmus: Ein Puffer (bucket) wird mit einer bestimmten Rate (token rate) mit sog. tokens gefüllt. Jedes Token „schnappt“ sich ein Paket und wird aus dem bucket entfernt.
- ? Sehr gut geeignet in Kombination mit einem DSL-Modem, um bei vielen Uploads die Latenz niedrig zu halten...

# Classless qdisc: sfq (Stochastic Fair Queueing)

- ? Jede Session (TCP-Session oder UDP-Stream) bekommt eine eigene FIFO.
- ? Alle FIFOs werden per Round Robin abgearbeitet
- ? Realisierung: Eine Hashfunktion verteilt alle Sessions auf wenige FIFOs.
- ? Um eine faire Verteilung der Sessions auf die FIFOs zu gewährleisten, wird nach einer kurzen Zeit (default: 10sec) die Hashfunktion ausgewechselt.

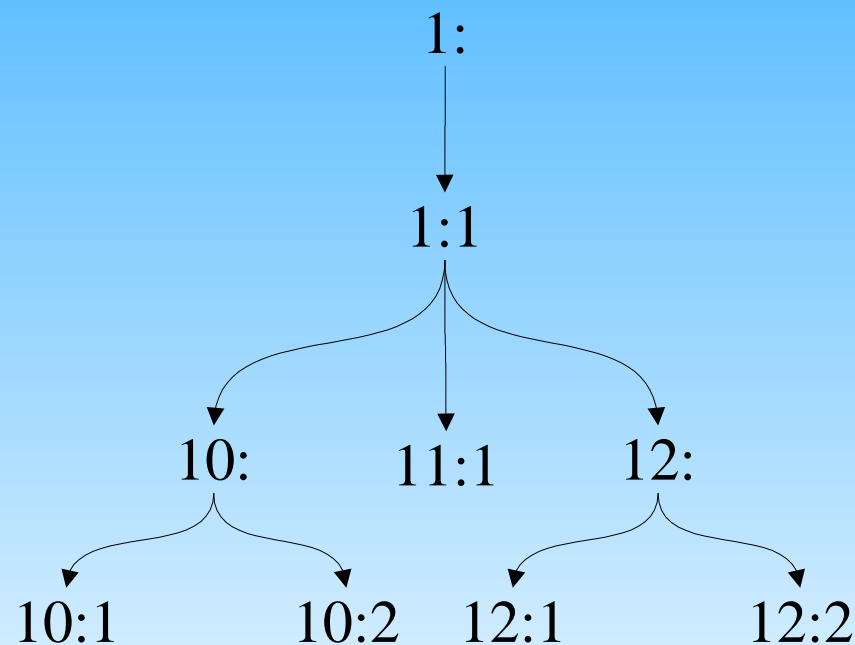
# Ratschläge zur Benutzung (aus dem Adv-Rout Howto geklaut)

- ? Summarizing, these are the simple queues that actually manage traffic by reordering, slowing or dropping packets.
- ? The following tips may help in choosing which queue to use. It mentions some qdiscs described in the 'Advanced & less common queueing disciplines'.
- ? To purely slow down outgoing traffic, use the Token Bucket Filter. Works up to huge bandwidths, if you scale the bucket.
- ? *If your link is truly full and you want to make sure that no single session can dominate your outgoing bandwidth, use Stochastic Fairness Queueing.*
- ? *If you have a big backbone and know what you are doing, consider Random Early Drop (see Advanced chapter).*
- ? To 'shape' incoming traffic which you are not forwarding, use the Ingress Policer.
- ? If you *\*are\** forwarding it, use a TBF on the interface you are forwarding the data to.
- ? *If you don't want to shape, but only want to see if your interface is so loaded that it has to queue, use the pfifo queue (not pfifo\_fast). It lacks internal bands but does account the size of its backlog.*

# Classful Queueing Disciplines

- ? Manche qdiscs können andere qdiscs enthalten, diese heissen dann plötzlich „classes“, „inner qdisc“ oder „sub-qdisc“.
- ? Sub-qdiscs müssen sich die Bandbreite ihrer äusseren qdisc teilen
- ? Traffic wird durch sog. filter auf die classes verteilt.
- ? Classful qdiscs werden in einem Baum angeordnet
- ? Jede qdisc bekommt eine eindeutige Nummer, bestehend aus einer Major-Number und einer Minor-Number; z.B. 1:10
- ? Jedes Interface hat mindestens eine root-qdisc mit der ID 1:

# Beispiel: Klassenhierarchie



- ? Die Hierarchie legt nur fest, welche qdiscs welche Klasseneigenschaften übernehmen (Bandbreite u.ä.).
- ? Der „Weg“ der Pakete wird nur über die Filter gesteuert

# PRIO qdisc

- ? Ähnlich pfifo\_fast, nur das die einzelnen Bänder wieder qdiscs sind
- ? Aufteilung wieder über TOS-Field
- ? Macht zusätzlich Accounting, kann also leicht für Messungen/-Benchmarks verwendet werden



# CBQ qdisc

- ? „almost famous“
- ? Kompliziert
- ? Ungenau
- ? Unschön
- ? Basiert auf Idletime-Berechnungen, setzt genau Kenntnis des Interfaces voraus (Probleme z.B. bei PCMCIA-Karten)
- ? Alternative: HTB

# HTB qdisc

- ? Hierarchical Token Bucket („Classful Token Bucket Filter“)
- ? Ist nicht Bestandteil vom Kernel, muss also als Patch hinzugefügt werden
- ? Benötigt nur wenige Parameter
  - ? rate: die gewünschte Übertragungsrate
  - ? ceil: die maximale Übertragungsrate
  - ? burst: wieviel darf kurzzeitig überzogen werden
- ? Die Verhältnisse der inner qdiscs werden bei „Überlast“ weniger inner qdiscs berücksichtigt.

# Filter: Übersicht

- ? Filter werden an qdiscs „gehängt“ und beim Einfügen von Paketen der Reihe nach abgearbeitet, bis eine Regel zutrifft oder nicht (dann geht's in die Defaultqueue)
- ? Filter können auf fast alles reagieren
  - ? Alle Felder im TCP Header
  - ? Auf markierte Pakete: ipchains und iptables können Pakete markieren, diese Markierung kann dann später zum „Einsortieren“ verwendet werden
  - ? Routing Table (Linux kann mehrer Routingtabellen verwenden)
  - ? Eigene Filter können leicht geschrieben werden

# Alternativen

- ? DSMARK, Ingress Policer, RED, ... (siehe [1])
- ? Diffserv
- ? es gibt einen Shaperdaemon im Userspace, ich hab aber die URL nicht mehr gefunden :-)
- ? Kabel von Alex W. verwenden

# Beispiele

- ? Jetzt im Real Life!
- ? ... oder in der Doku.

# Anhang: Quellen und Anregungen

- ? Linux Advanced Routing & Traffic Control
  - ? <http://ds9a.nl/lartc/>
- ? HTB Linux queuing discipline
  - ? <http://luxik.cdi.cz/~devik/qos/htb/>
  - ? <http://luxik.cdi.cz/~devik/qos/htb/htbtheory.htm>
  - ? <http://luxik.cdi.cz/~devik/qos/htb/htbmeas1.htm>
- ? Linux Network Traffic Control - Implementation Overview - Werner Almesberger
  - ? <ftp://icaftp.epfl.ch/pub/people/almesber/pub/tcio-current.ps.gz>
- ? Linux - Advanced Networking Overview: QoS Support in Linux
  - ? <http://qos.ittc.ukans.edu/howto/node3.html>
- ? Introduction to Linux Advanced Routing and Traffic Management
  - ? [http://www.cceenet.org/workshops/lectures2000/Rafal\\_Maszkowski/routing-traffic/](http://www.cceenet.org/workshops/lectures2000/Rafal_Maszkowski/routing-traffic/)
- ? Paketfilter für QoS unter Linux
  - ? <http://www.ind.uni-stuttgart.de/Content/u32h/u32h-filter.htm>
- ? Differentiated Services on Linux
  - ? <http://diffserv.sourceforge.net/>
- ? Ein paar allgemeine Links
  - ? <http://users.pandora.be/stef.coene/qos/docs/>
- ? Dynamische Bandbreitenbeschränkung mit Linux
  - ? <http://archiv.tu-chemnitz.de/pub/2001/0100/data/diplom.pdf>
- ? Paketfilter für QoS unter Linux
  - ? <http://www.ind.uni-stuttgart.de/Content/u32h/u32h-filter.pdf>
- ? QoS-Fähigkeit in Endsystemen
  - ? <http://www.rvs.uni-hannover.de/arbeiten/diplom/da-siemens/node6.html>
- ? Traffic Control Next Generation
  - ? <http://tcng.sourceforge.net/>
- ? IP Command Reference
  - ? <http://snafu.freedom.org/linux2.2/docs/ip-cref/>
- ? TC Docs
  - ? [http://www.fibrespeed.net/~mbabcock/linux/qos\\_tc/tc\\_info.php](http://www.fibrespeed.net/~mbabcock/linux/qos_tc/tc_info.php)
- ? Linux QoS & tc
  - ? [http://www.fibrespeed.net/~mbabcock/linux/qos\\_tc/](http://www.fibrespeed.net/~mbabcock/linux/qos_tc/)
- ? Docum
  - ? <http://www.docum.org/>
- ? Der Link is net so ganz erst gemeint:
  - ? <http://www.linux.or.jp/JF/JFdocs/traffic-control.html>
- ? An API for Linux QoS Support
  - ? [http://www.ittc.ukans.edu/~pramodh/courses/linux\\_qos/mainpage.html](http://www.ittc.ukans.edu/~pramodh/courses/linux_qos/mainpage.html)
- ? Traffic Shaping
  - ? <http://linux.oreillynet.com/pub/a/linux/2000/08/24/LinuxAdmin.html>
- ? <http://www.tm.uka.de/lehre/SS01/praktika/unterlagen/diffserv.ps.gz>
- ? <http://www.adsl4linux.de/forum/read.php?TID=457#1452>