

IP QoS mit Linux

LINUXFORUM – INTERNETWORLD *Berlin, 15.-17. Mai 2001*

MATTHIAS KRANZ

m.kranz@linux-ag.com

Linux Information Systems AG

Agenda

- ✍ Einleitung
 - ⇒ Was heißt *Quality of Service*?
 - ⇒ Warum brauchen wir es?
- ✍ Technologien
 - ⇒ RSVP
 - ⇒ DiffServ
- ✍ Linux Traffic Control

Einleitung

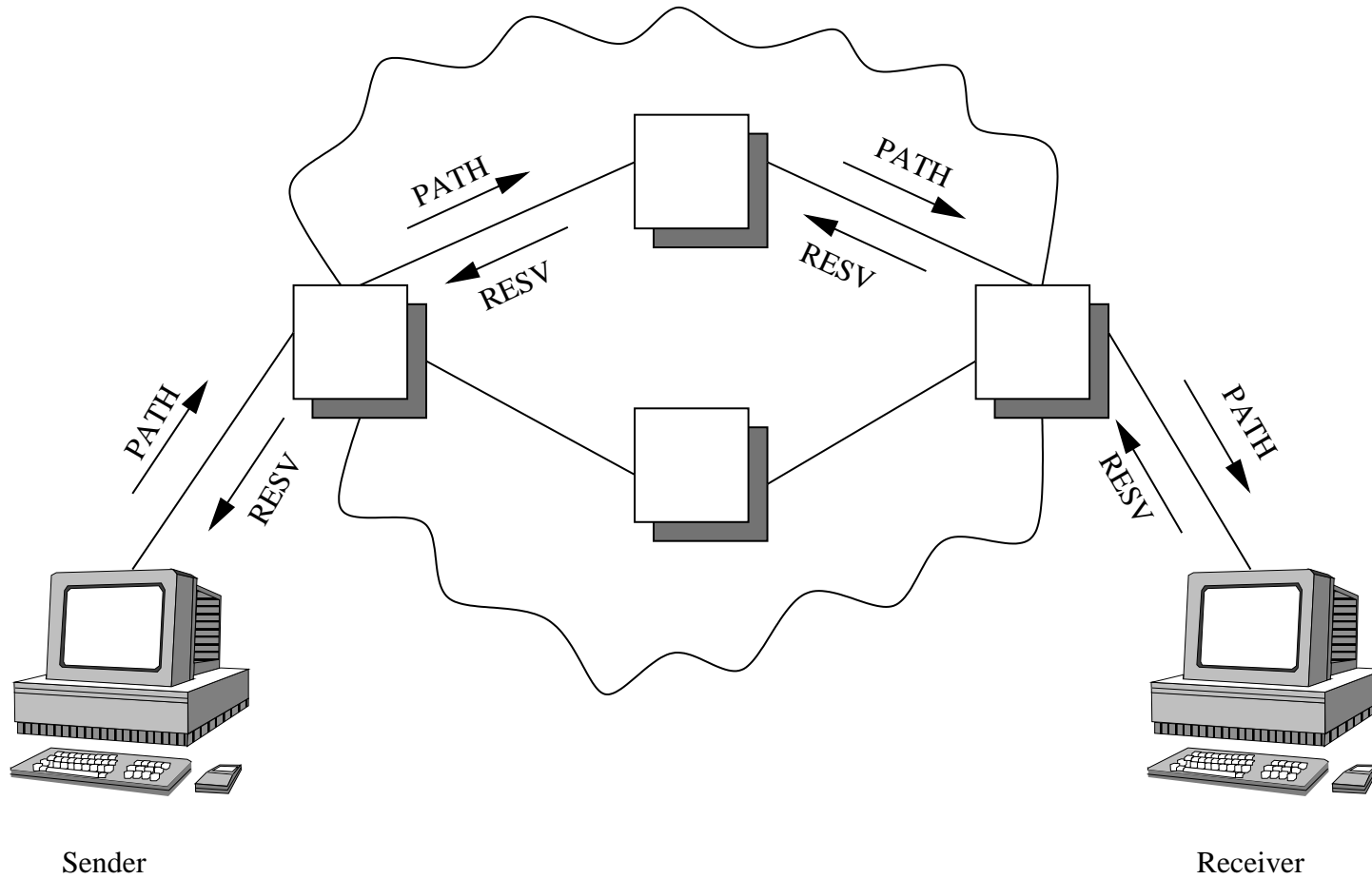
Bandwidth is the answer! What was the question?

- ✍ Seit dreißig Jahren fahren wir mit "Best Effort" – Warum etwas Neues?
- ✍ Der Traffic im Internet ist nicht nur gewachsen, er hat auch seine Charakteristika verändert.
- ✍ Keine Probleme mit *Mail, File Transfer und Web*, aber ...
- ✍ ... wie sieht es mit *IP-Telephony und Video on Demand* aus?

Was heißt QoS?

- ✎ Unterschiedliche Anwendungen haben unterschiedliche Anforderungen
- ✎ Diese Anforderungen lauten:
 - ⇒ Bandwidth
 - ⇒ Latency
 - ⇒ Jitter
 - ⇒ Reliability
- ✎ Unterschiedliche Anforderungen führen zu unterschiedlichen Service-Klassen und damit zu unterschiedlichen Preisen
- ✎ Vergleiche: Fliegen in der *Economy*, *Business* oder *First Class*

RSVP



RSVP unter Linux

- ✎ USC ISI's RSVPd Version 4.2a4 (portiert von Alexey Kuznetsov)
<ftp://ftp.sUNET.se/pub/os/Linux/ip-routing/rsvp/>
- ✎ Columbia University NY (Phil Wang)
- ✎ API für Userspace Applikationen verfügbar
- ✎ Der `rsvpd` übernimmt dann die Kommunikation mit weiteren Knoten im RSVP-sprechenden Netzwerk

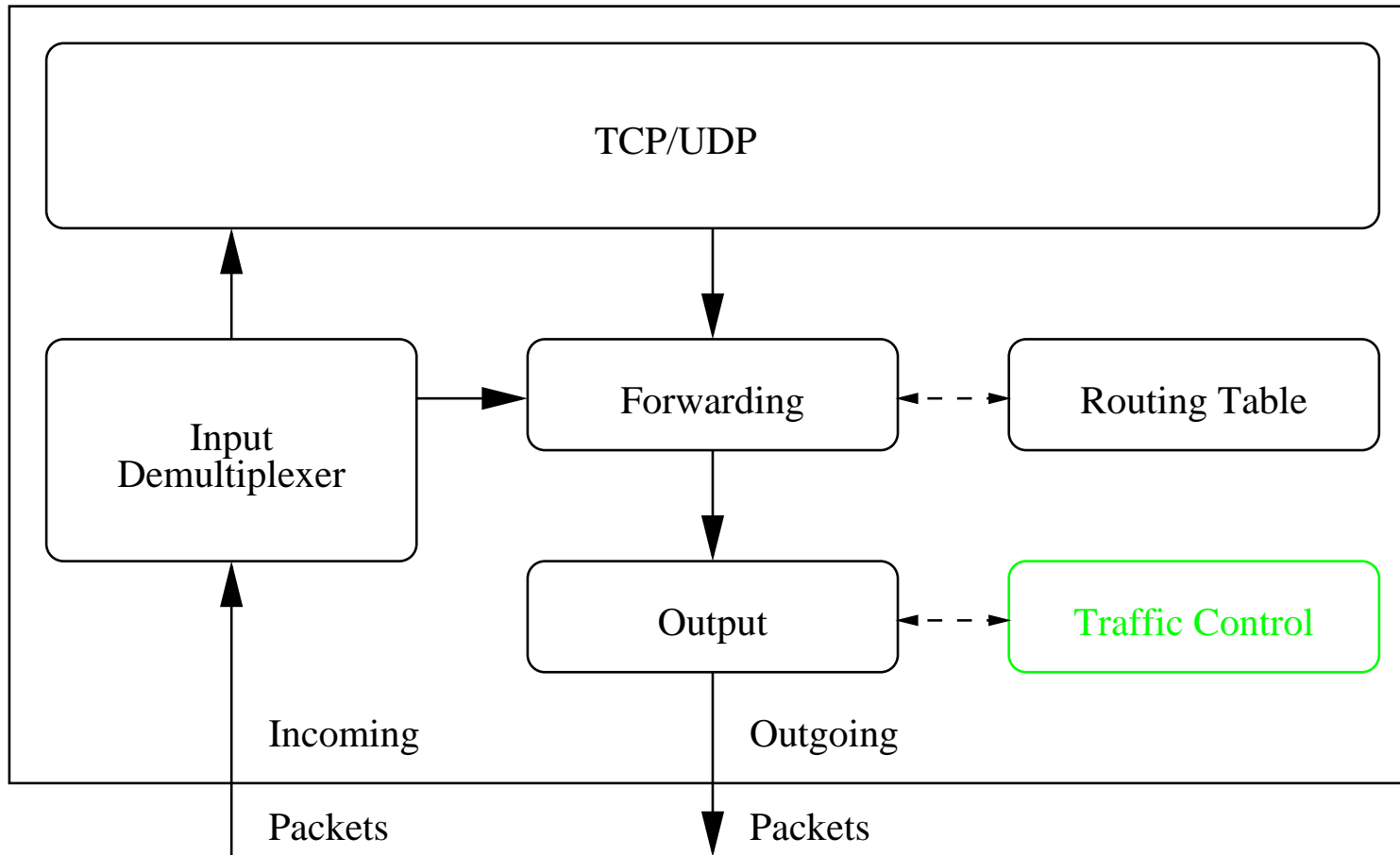
DiffServ

- ✍ Pakete werden klassifiziert und markiert
- ✍ Classification, Policing und Conditioning werden nur an den Netzwerk-Aussengrenzen angewendet
- ✍ Knoten im Netz wenden einfache Forwarding-Regeln an
- ✍ DS-Feld im IP-Header
 - ⇒ TOS Byte (IPv4)
 - ⇒ Traffic Class Byte (IPv6)

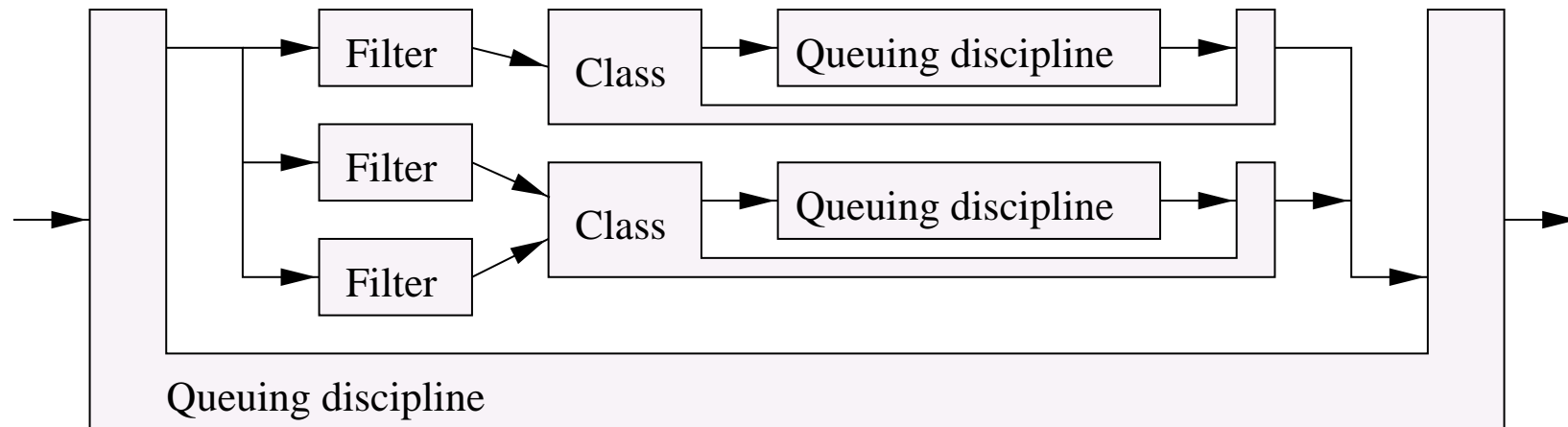
DiffServ unter Linux

- ✎ Diffserv on Linux
 - ⇒ <http://diffserv.sourceforge.net/>
- ✎ K.I.D.S. (University of Karlsruhe)
 - ⇒ <http://www.telematik.informatik.uni-karlsruhe.de/forschung/diffserv/KIDS/>
- ✎ ITT Center University of Kansas
 - ⇒ <http://qos.ittc.ukans.edu/>

Traffic Control unter Linux



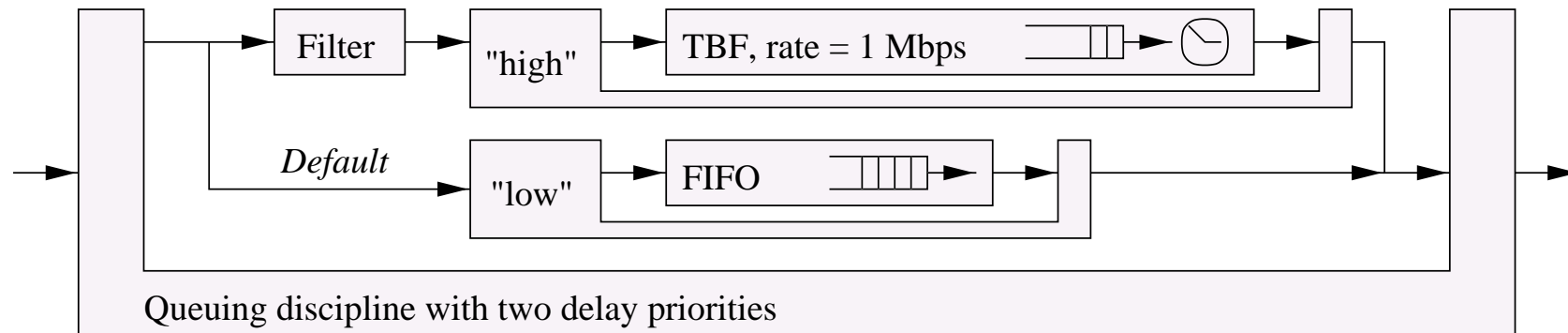
Traffic Control unter Linux



- ✎ Filter werden zur Differenzierung unterschiedlicher Klassen von Paketen angewendet
- ✎ Zu jeder Klasse gehört wiederum eine Queuing Discipline

Traffic Control unter Linux

✍ Beispiel



Traffic Control - Konfigurations-Management I

- ✎ Die Konfiguration erfolgt mit `tc`
- ✎ `tc` läuft im Userspace, verwendet Netlink Interface
- ✎ `tc` erlaubt die Konfiguration von
 - ⇒ Qdiscs
 - ⇒ Klassen
 - ⇒ Klassifizierern
- ✎ Its general syntax is

```
tc <component> add|del|change|get dev <devname> <component specs>  
<component type> <component parameters>
```

Beispiel

✎ Very simple example using the special qdisc ingress

```
# tag all incoming packets from host 10.2.0.24 to value 1
# tag all incoming SYN packets to value 2
ipchains -A input -i eth0 -s 10.2.0.24/32 -m1
ipchains -A input -i eth0 -y -m2
# install the ingress qdisc on the ingress interface (eth0)
tc qdisc add dev eth0 handle ffff: ingress
# for tag 1 allow up to at least 60 packets to burst
# (assuming maximum packet size of 1.5KB) in the long run and up to
# about 6 packets in the short run
tc filter add dev eth0 parent ffff: protocol ip prio 50 handle 1 fw\
police rate 1.5kbit burst 90 k mtu 9k drop
# SYN packets are 40 bytes (320 bits) so 10 SYNs equals
# 3Kbps, and we therefore limit below the incoming SYNs to 10/sec
tc filter add dev eth0 parent ffff: protocol ip prio 50 handle 2 fw\
police rate 3kbit burst 40 mtu 9k drop
```