

# IP QoS with Linux

*LINUXTAG Magdeburg, 18.+19. Mai 2001*

**MATTHIAS KRANZ**

m.kranz@linux-ag.com

Linux Information Systems AG

## Agenda

- ✍ Introduction
  - ⇒ What is *Quality of Service*?
  - ⇒ Why do we need it?
- ✍ Different Approaches
  - ⇒ RSVP
  - ⇒ DiffServ
- ✍ Linux Traffic Control
- ✍ Future
- ✍ Resources and References

## Introduction

*Bandwidth is the answer! What was the question?*

- ✍ Up to now, "Best effort" service was fine, so why do we need to change it?
- ✍ Traffic has not only increased, but changed its character
- ✍ No problems with *Mail, File Transfer and Web*, but ...
- ✍ ... everybody is talking about *IP-Telephony and Video on Demand* :)
- ✍ What we (and especially the service providers) need is a totally different technology approach – *IP Quality of Service*

## Quality of Service

- ✍ The problem is: Quality of Service means different things to different people
- ✍ From here we will use the following definition in this talk
  - Def.: QoS is the ability of a network element to provide some level of assurance for consistent network data delivery.*
- ✍ This could mean
  - ⇒ Consistent data transfer rate
  - ⇒ Consistent latency and jitter
- ✍ Different requirements lead to different classes of service and different service often means different prices
- ✍ Compare this to flying by airplane - *Economy, Business or First Class*

## View From 10.000 ft

- ✍ ATM is the answer, isn't it?
  - ⇒ Of course, it is *one* solution and Linux does support it well
- ✍ The IETF proposes two different approaches
  - ⇒ Resource Reservation (Integrated Services)  
End points request a particular bandwidth amount  
RSVP-enabled routers will provide those resources
  - ⇒ Prioritization (Differentiated Services)  
Differentiation and marking of traffic classes  
Not dynamically at the moment

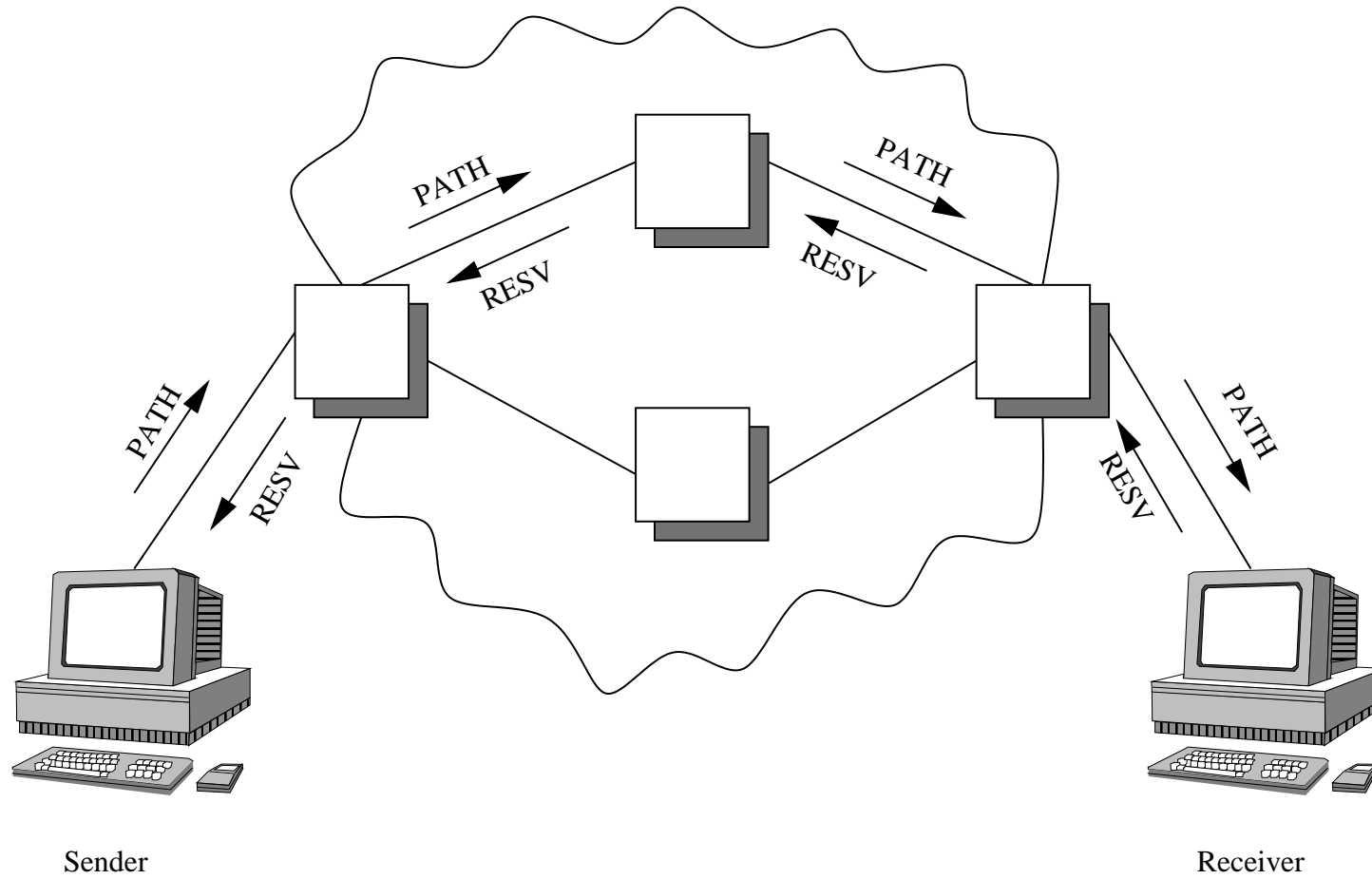
## Kernel Configuration

- ✎ Actually it is easy to configure a Linux box as an end device as well as a router enabled for
  - ⇒ ATM
  - ⇒ IntServ
  - ⇒ DiffServ
  
- ✎ Another mechanism available on Linux is Alan's *Shaper device*
  - ⇒ It's in > 2.0
  - ⇒ Download the configuration software from <ftp://shadow.cabi.net/pub/Linux>

## RSVP I

- ✍ Resource **ReSerVation Protocol**
- ✍ Signaling Protocol
- ✍ Kind of Circuit Emulation on IP networks
- ✍ Complex and far away from *traditional IP*
- ✍ Enables **Integrated Services**
  - ⇒ Guaranteed
  - ⇒ Controlled Load

# RSVP II



## RSVP III

- ✎ Sender characterizes outgoing traffic in terms of the upper and lower bounds of bandwidth, delay, and jitter (PATH)
- ✎ Each router along the downstream route establishes a "path-state"
- ✎ Receiver sends a reservation request message (RESV)
- ✎ Each router along the upstream uses the admission control and allocates the necessary resources
- ✎ Last router sends a confirmation message back to the receiver

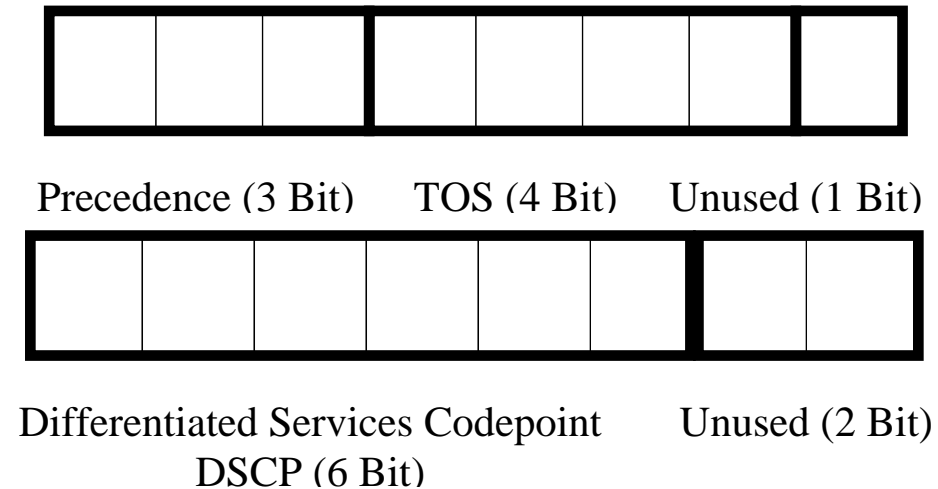
## RSVP On Linux

- ✍ The most used RSVP version is USC ISI's 4.2a4
  - ⇒ One version was ported by Alexey Kuznetsov
  - ⇒ The other is of Phil Wang (Columbia University NY)
- ✍ There is a RSVP API available which applications could directly use to request reservations
- ✍ A `rsvpd` talks to the RSVP-enabled network
- ✍ A request could possibly being denied
- ✍ The `rsvpd` could also adjust your local traffic handling

## DiffServ

- ✍ Packets are classified and marked
- ✍ Classification, Policing and Conditioning is only required at network boundaries
- ✍ Inner network nodes only apply certain forwarding rules
- ✍ DS field in the IP header
  - ⇒ TOS Byte (IPv4)
  - ⇒ Traffic Class Byte (IPv6)
- ✍ Two service levels
  - ⇒ Expedited Forwarding (EF)
  - ⇒ Assured Forwarding (AF)

## DiffServ II

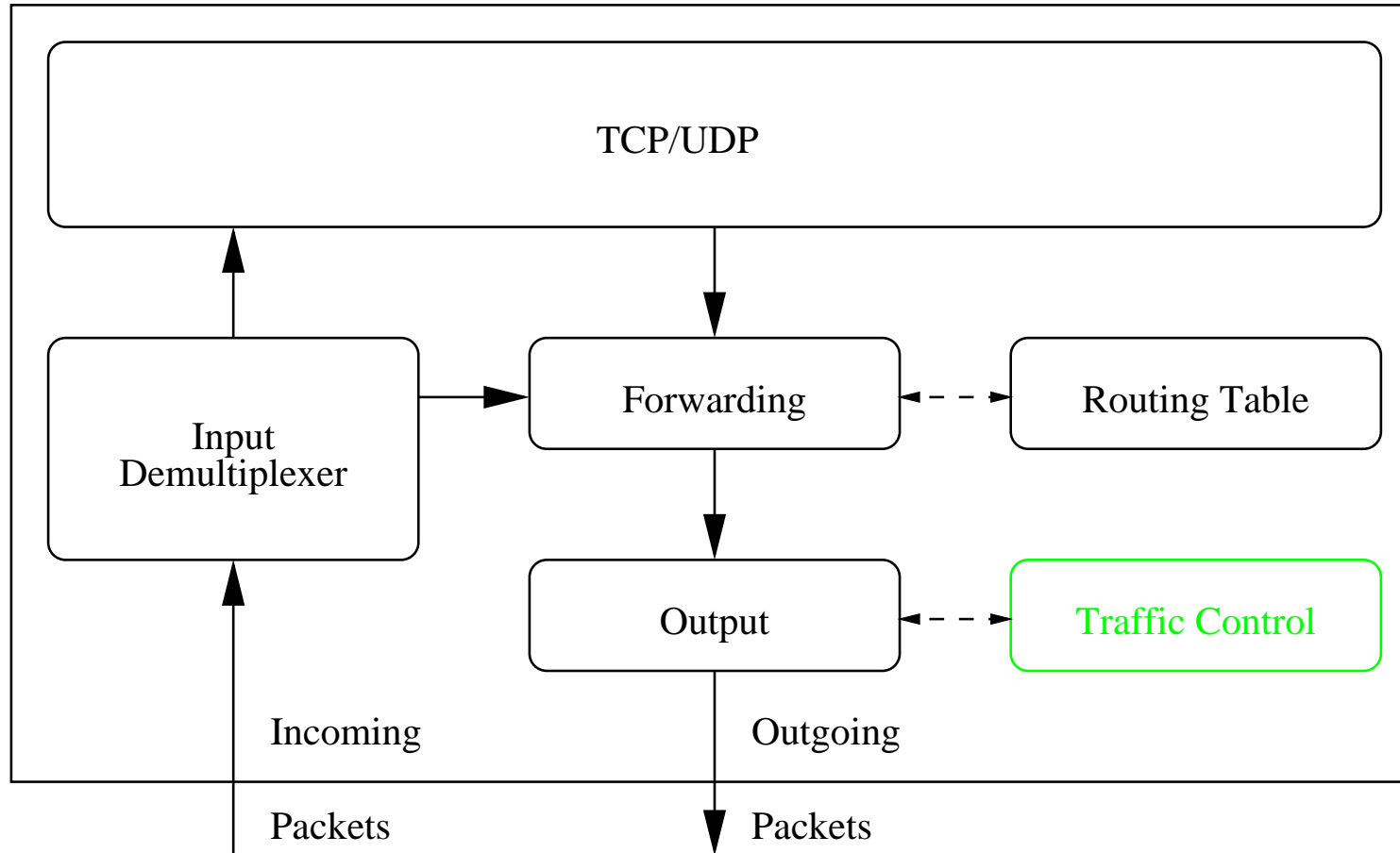


- ✍ DSCP is used to select the PHB a packet experiences at each node
- ✍ Structure is incompatible with IPv4

## Differentiated Services on Linux

- ✎ Diffserv on Linux
  - ⇒ <http://icawww1.epfl.ch/linux-diffserv/>
- ✎ 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/>

## The way Linux is doing it

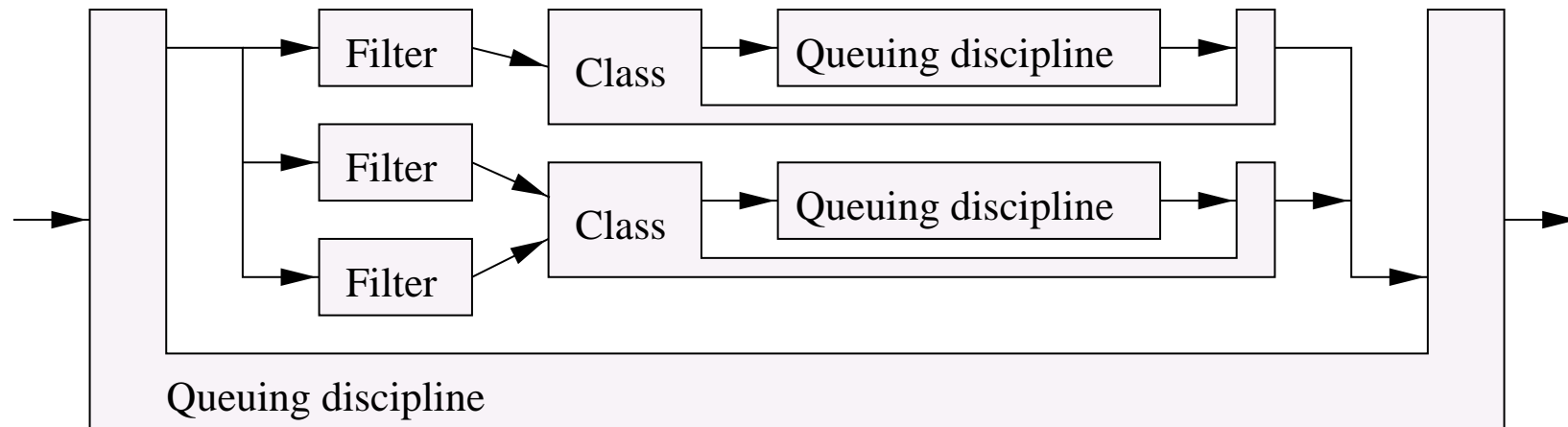


## Traffic Control I

- ✎ The TC code consists of the following components
  - ⇒ Queuing Disciplines (QDiscs)
  - ⇒ Classes
  - ⇒ Filters
  - ⇒ Policing
- ✎ Each network device has a Queuing Discipline



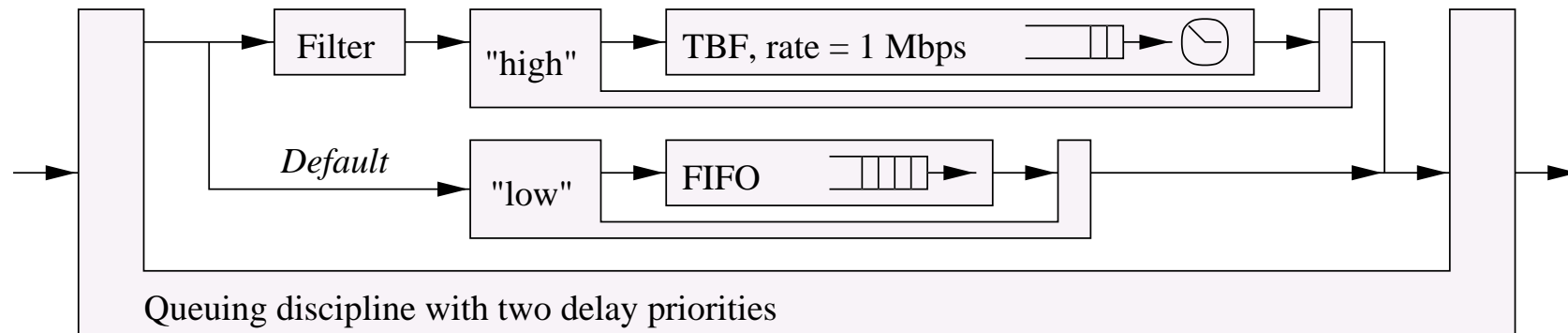
## Traffic Control II



- ✎ Filters are used to distinguish between different classes of packets
- ✎ Each class consists of a queuing discipline in turn

## Traffic Control III

### Example



## Traffic Control - Configuration Management I

- ✍ The configuration is done with `tc`
- ✍ `tc` runs in userspace, using the netlink interface
- ✍ `tc` enables you to configure
  - ⇒ `qdiscs`
  - ⇒ `classes`
  - ⇒ `classifiers`

- ✍ Its general syntax is

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

## Metering And Policing

- ✎ A meter can be attached to a filter
- ✎ It provides the following actions
  - ⇒ drop
  - ⇒ continue
  - ⇒ reclassify
- ✎ Use ipchains option `-m` for marking
- ✎ ipchains provides hooks to
  - ⇒ input, forward, and output chain
- ✎ Netfilter provides even more hooks

## Example

### 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
```

## Future

- ✍ Linux Traffic Control Design
  - ⇒ High modularity holds complexity but offers very high flexibility
  - ⇒ Offers solid basis for implementations like Linux DiffServ
- ✍ Redevelopment of the whole code is actually done
- ✍ Special emphasis on a completely new traffic control configuration language
- ✍ Stay tuned ...

*Thanks to Alexey Kuznetsov, Werner Almesberger, and many more ...*

## Resources and References I

✍ DiffServ on Linux Web Site

<http://icawww1.epfl.ch/linux-diffserv/>

✍ 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/>

✍ RSVP

<ftp://ftp.sunet.se/pub/os/Linux/ip-routing/rsvp/>

✍ Download site iproute2+tc

<ftp://ftp.inr.ac.ru/ip-routing/>

## Resources and References II

- ✎ RFC 2205 - Resource Reservation Protocol  
Braden, Zhang, Herzog, Berson, Jamin
- ✎ RFC 2210 - The Use of RSVP with IETF Integrated Services  
Wroclawski
- ✎ RFC 2475 - An Architecture for Differentiated Service  
Blake, Black, Carlson, Davies, Wang, Weiss
- ✎ RFC 2474 - Definition of the Differentiated Services Field (DS Field)  
Nichols, Blake, Baker, Black
- ✎ Linux Network Traffic Control - An Implementation Overview  
Almesberger
- ✎ Differentiated Services on Linux  
Almesberger, Salim, Kuznetsov
- ✎ Linux - Advanced Networking Overview  
Radhakrishnan