# Multimedia Communication and Internet QoS

**Hermann Hellwagner**

**Research Group Multimedia Communication (MMC)**

**Institute of Information Technology (ITEC)**
**Klagenfurt University, Austria**

**hermann.hellwagner@uni-klu.ac.at**
**http://www.itec.uni-klu.ac.at/~hellwagn**

**Györ, 29 April 2010**

# Who am I?

- **Professor of Information Technology**
  **Head of RG Multimedia Communication (MMC)**
- **Research interests:**
  - **Distributed multimedia systems**
  - **Multimedia communication, quality of service (QoS)**
- **Teaching:**
  - **Computer organization, operating systems, computer networks**
  - **Internet QoS, servers/clusters, advanced topics in multimedia communication**
- **Professional services:**
  - **Member of the Scientific Board of the Austrian Science Fund (FWF)**
  - **Deputy Head of Austrian Delegation to ISO/IEC JTC1/SC29 WG11 (Moving Picture Experts Group - MPEG)**

# Where do I come from?

# Where do I come from?

# Outline

1 Introduction: Motivation, QoS Requirements, Terminology, Principles

2 Integrated Services (IntServ) and the Resource Reservation Protocol (RSVP)

3 Differentiated Services (DiffServ)

4 Multiprotocol Label Switching (MPLS)

5 Real-Time Multimedia Data Transport: Basic Protocols

6 Conclusions

# Literature: Books

- **Ina Minei, Julian Lucek**
  ***MPLS-Enabled Applications: Emerging Developments and New Technologies***
  **Wiley 2008 (Second Edition)**
- **Mihaela van der Schaar, Philip A. Chou**
  ***Multimedia over IP and Wireless Networks. Compression, Networks, and Systems***
  **Elsevier / Academic Press 2007**
- **John Evans, Clarence Filsfils**
  ***Deploying IP and MPLS QoS for Multiservice Networks. Theory and Practice***
  **Elsevier / Morgan Kaufmann 2007**
- **Jitae Shin, Daniel C. Lee, C.-C. Jay Kuo**
  ***Quality of Service for Internet Multimedia***
  **Prentice Hall 2004**
- **Sanjay Jha, Mahbub Hassan**
  ***Engineering Internet QoS***
  **Artech House 2002**
- **Fred Halsall**
  ***Multimedia Communications***
  **Addison Wesley 2001**
- **Ralf Steinmetz**
  ***Multimedia-Technologie. Grundlagen, Komponenten und Systeme***
  **Springer 1998**

# Literature: Journals and Conferences

- IEEE Network
- IEEE Communications Magazine
- IEEE Communications Surveys & Tutorials:
  http://www.comsoc.org/livepubs/surveys/index.html
- IEEE Multimedia
- IEEE Internet Computing: http://computer.org/internet/ (IC Online)
- IEEE Transactions on Communications
- IEEE Journal on Selected Areas in Communications
- IEEE Transactions on Multimedia
- IEEE/ACM Transactions on Networking
- ACM Multimedia Systems
- Numerous Conferences on Networking Topics

# Literature: Some WWW Resources

- **Internet Engineering Task Force (IETF): http://www.ietf.org (incl. many *Working Groups* defining QoS mechanisms)**
- **IEEE Communications Society: http://www.comsoc.org**
- **Qbone Internet2 Initiative: http://qbone.internet2.edu/**
- **Survey on Internet QoS: http://user.chollian.net/~son6971/qos/qos.htm**

- **Prof. Raj Jain‘s Web page: http://www.cs.wustl.edu/~jain/**
- **Prof. Henning Schulzrinne‘s Web page: http://www.cs.columbia.edu/~hgs/**
- **“Classical“ *Networking Reading Lists* (incl. QoS): http://www.cs.columbia.edu/~hgs/netbib/standard.html**
- **Prof. Klara Nahrstedt‘s Web page: http://cairo.cs.uiuc.edu/~klara/home.html**

# 1

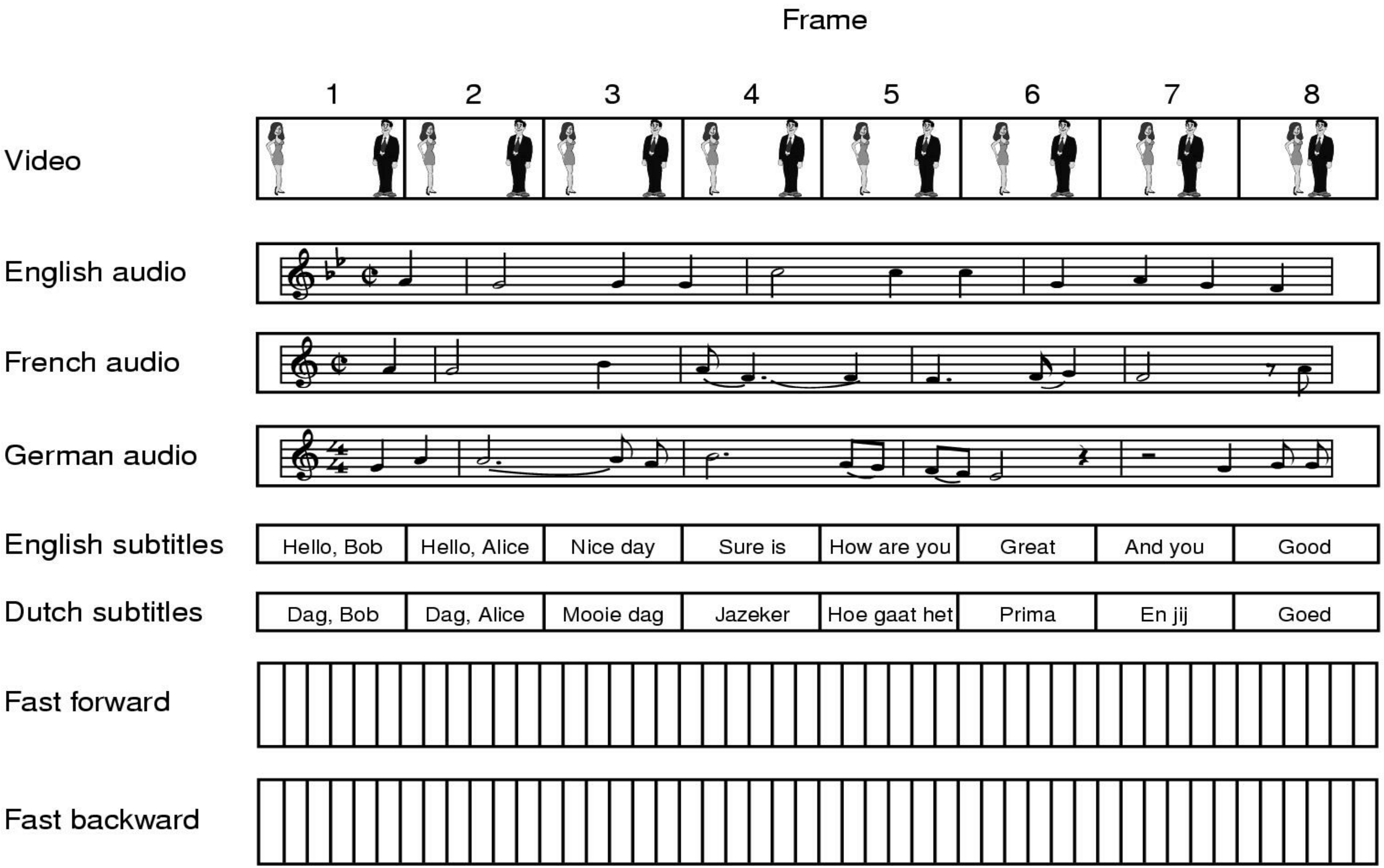
# Introduction

# Motivation, QoS Requirements, Terminology, Principles

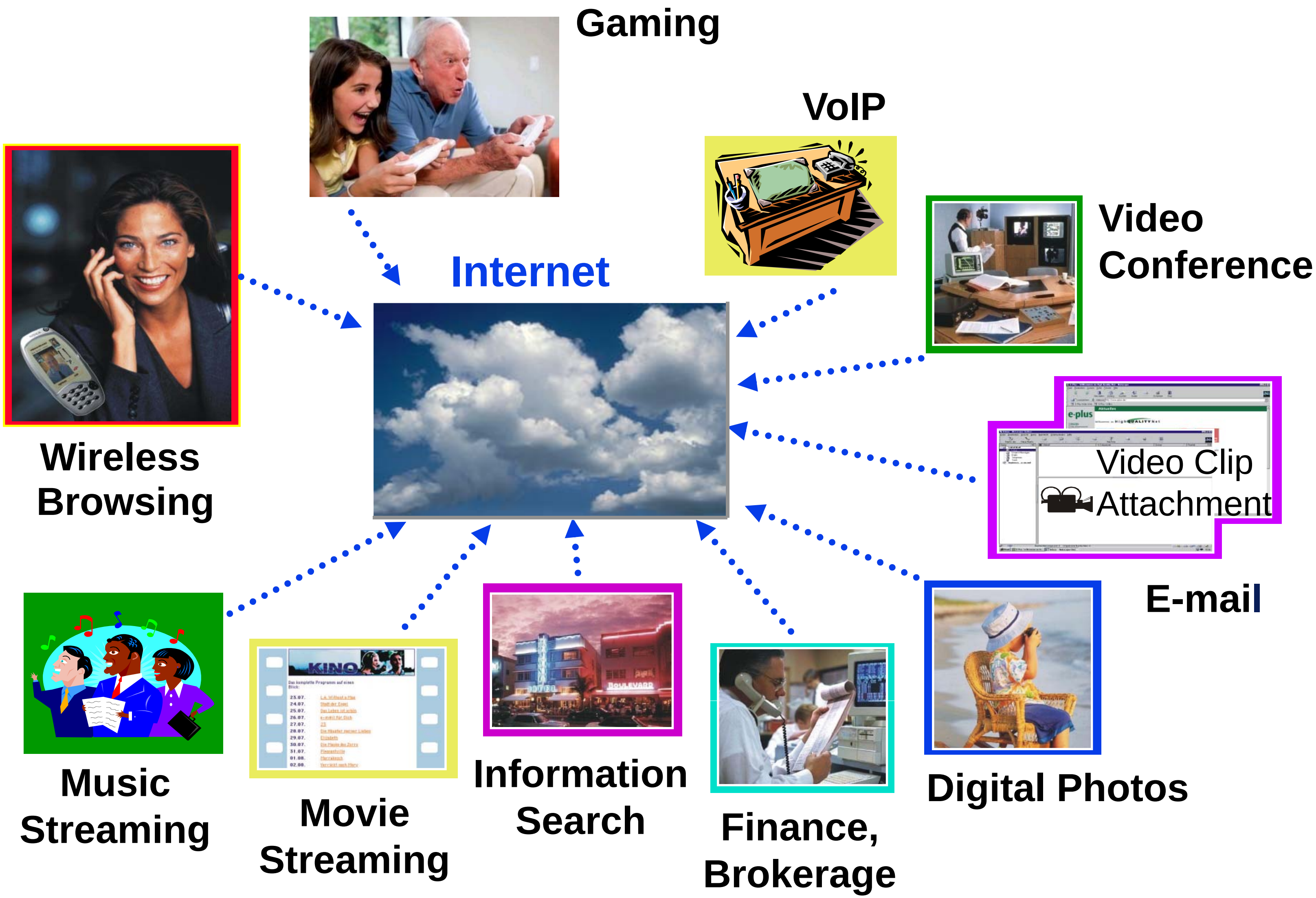
# The Challenge: Multimedia

- **Multimedia**
  - **Handling of a variety of presentation media**
  - **Acquisition, storage, retrieval, transmission, presentation, and perception of different data types:**
  - **Text, graphics, voice, audio, video, animation, VR/AR, .....**
  - **Movie: video + audio + script + descriptive (meta-) data + .....**
- **New applications**
  - **Multimedia has become pervasive in applications**
  - **Technology push and end user pull**
- **Challenges**
  - **Storage: many GBs per video**
  - ***Timely, continuous (real-time) and correct (synchronized) delivery***
  - **Indexing, searching, retrieval: 1000s of videos?**

# A Movie as a Set of Elementary Streams

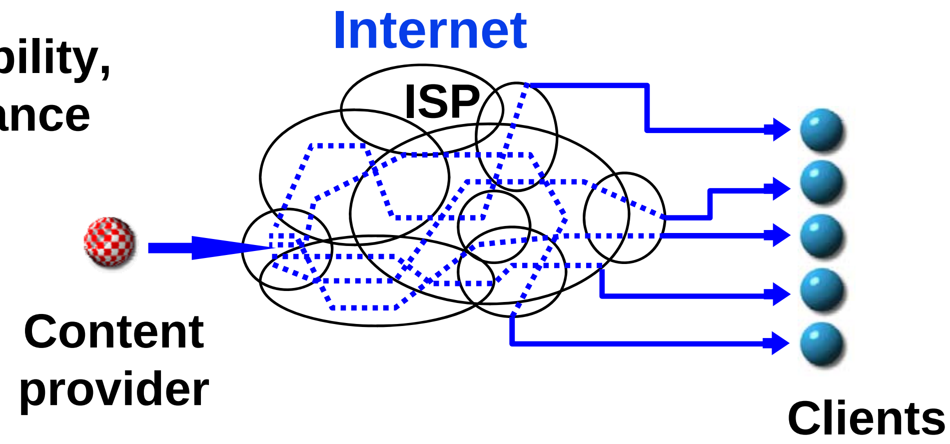


# The Real Challenge: Multimedia over IP Networks



# Some Applications: Broad-/Multicast and (N)VoD

- Broadcast or Multicast Video
  - Store-and-playback or live content
  - Enhanced services, reliability, availability, and maintenance

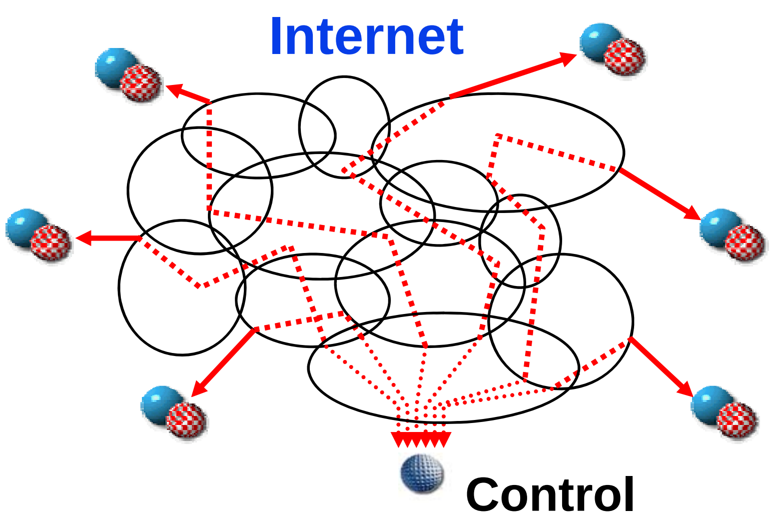



- Video-on-Demand (VoD)
  - Users can select from a list of choices (EPG), maybe preview
  - Interaction via set-top box + remote control or via PC
- Near Video-on-Demand (NVoD)
  - Popular videos are broadcasted periodically (“carousel”)
  - VCR control is more difficult

# Some Applications: Collaborative Work

- **Videoconferencing**
  - **Geographically distributed virtual meetings (presenters + audience)**
  - **Diverse audio/visual (A/V) input and output devices**
  - **Presenters can broadcast speech and graphics, maybe also real-time video**
  - **Hard requirements on the infrastructure, e.g., logging facility**
- **CSCW System**
  - **All of the above plus, e.g., joint work on a document**
- **Teleteaching / Telelearning**
  - **Potentially many participants**
- **Gaming**

**Distributed content providers**



# Some Data Rates and Technology Data

| *Multimedia Source* | *Mb/s* | *GB/h* |
|---|---|---|
| Telephone (PCM) | 0.064 | 0.003 |
| MP3 music | 0.14 | 0.06 |
| Audio CD | 1.4 | 0.62 |
| MPEG-1 movie | 1 - 1.5 | 0.66 |
| MPEG-2 movie | 4 | 1.76 |
| Digital camcorder (720 x 480) | 25 | 11 |
| Uncompressed TV (640 x 480) | 221 | 97 |
| Uncompressed HDTV (1280 x 720) | 648 | 288 |

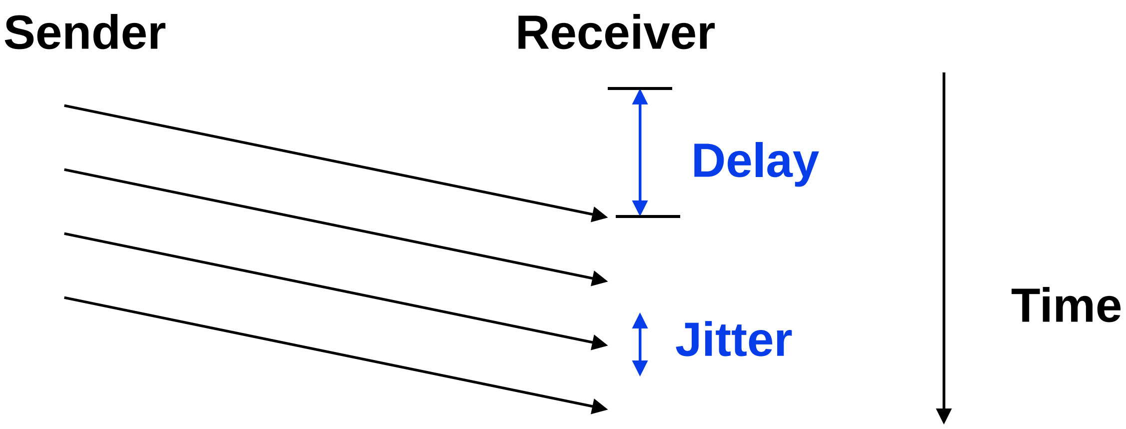
| *Device* | *Mb/s* |
|---|---|
| WiFi LAN (802.11b) | 11 |
| Fast Ethernet | 100 |
| EIDE disk | 133 |
| ATM OC-3 | 156 |
| SCSI Ultra Wide disk | 320 |
| IEEE 1394 (FireWire) | 400 |
| Gigabit Ethernet | 1000 |
| SCSI Ultra-160 | 1280 |

# How Much Does Compression Help?

| *Multimedia Source* | *Resolution* [cols x lines x fps] | *Raw bit rate* [Mb/s] | *Compressed bit rate* [Mb/s] |
|---|---|---|---|
| Film (USA and Japan) | 480 x 480 x 24 | 133 | 3 - 6 |
| NTSC video | 480 x 480 x 29.97 | 168 | 4 - 8 |
| PAL video | 576 x 576 x 25 | 199 | 4 - 9 |
| HDTV video | 1920 x 1080 x 30 | 1493 | 10 - 30 |
| ISDN videophone (CIF) | 352 x 288 x 29.97 | 73 | 0.064 - 1.92 |
| PSTN videophone (QCIF) | 176 x 144 x 29.97 | 18 | 0.01 - 0.03 |

# Is Raw Bandwidth the “Silver Bullet“?

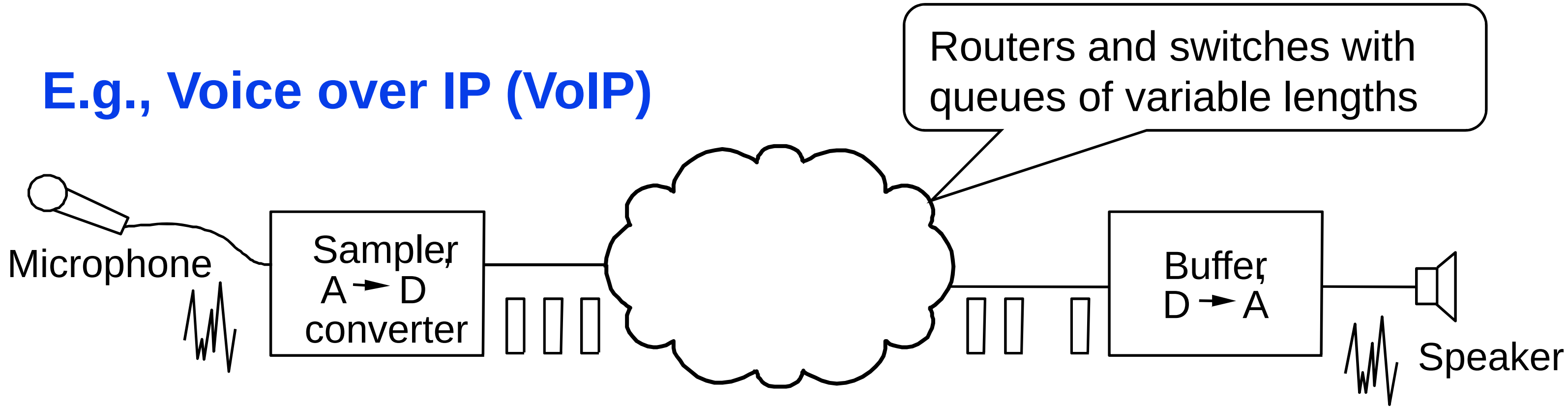
- Bandwidth (throughput): important, but not sufficient
  - Example: telephone over satellite:
    - Enough bandwidth, but:
    - Response not immediate
- Equally important: delay (latency) and jitter (delay variance):
  - Critical for timely, continuous delivery and (soft) real-time playback




- Additional criteria: stream synchronization, reliability, cost, .....

# Example Audio Application

E.g., Voice over IP (VoIP)

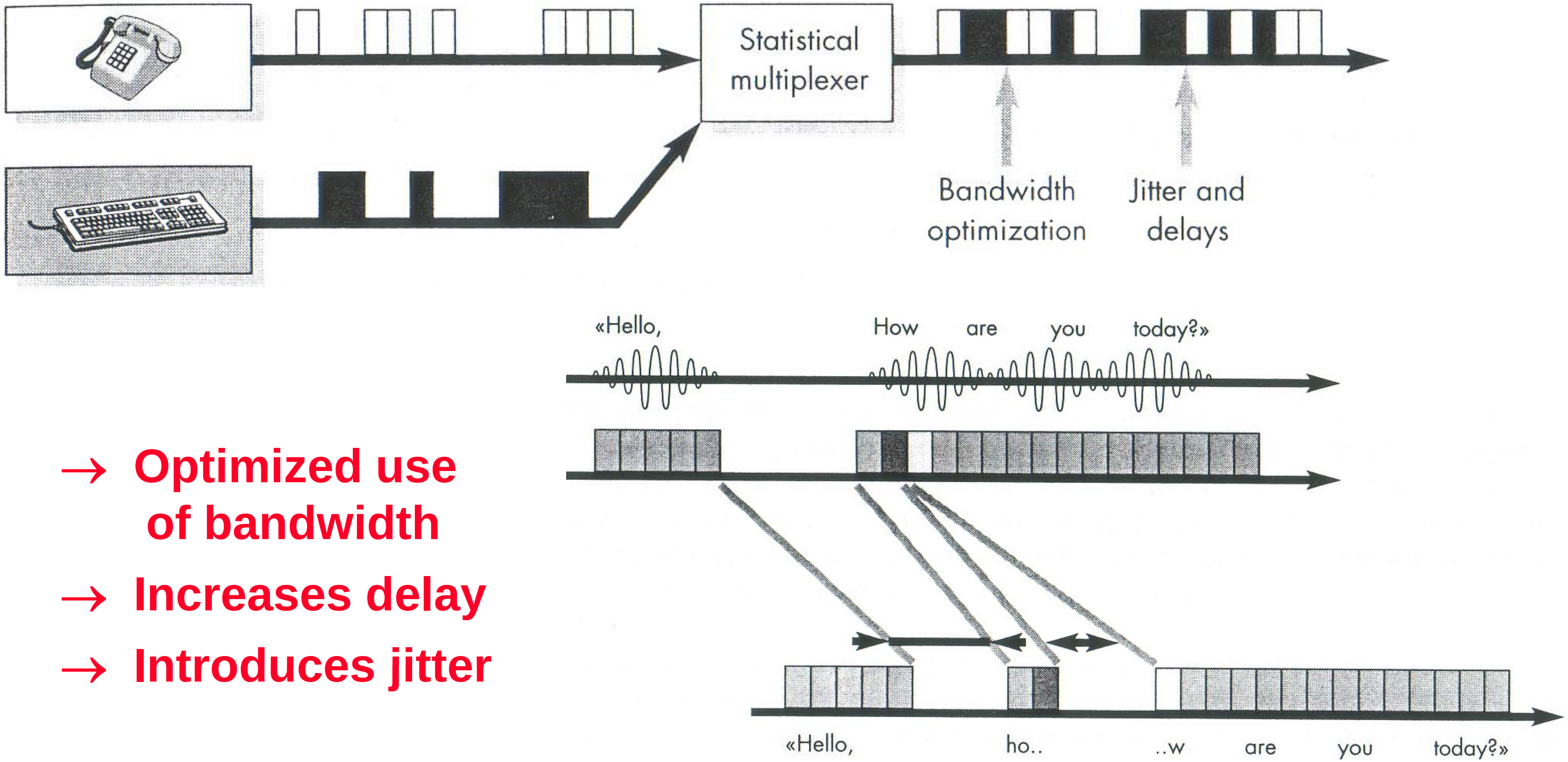



- **Voice sample once every 125µs**
- **Each sample has a playback time**
- **Packets experience variable delay in network**
- **Add constant "insurance" factor to playback time: playback point**
- **For voice, data arrival within ~300 ms is tolerable**

# Example Audio Application: Delay and Jitter

**Audio application (e.g., VoIP) will most probably use:**

- **Asynchronous transmission**
- **Statistical multiplexing**

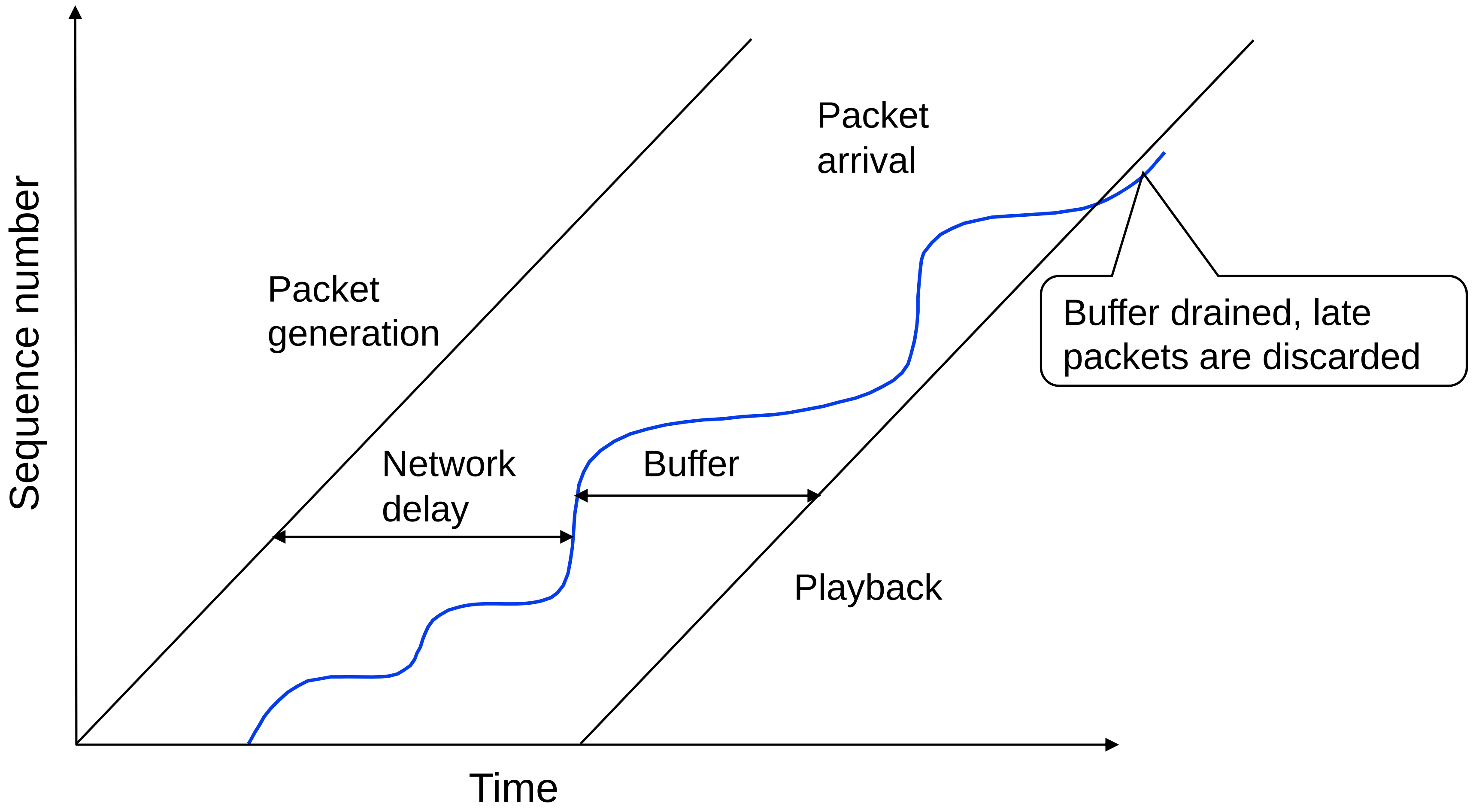



→ **Optimized use of bandwidth**

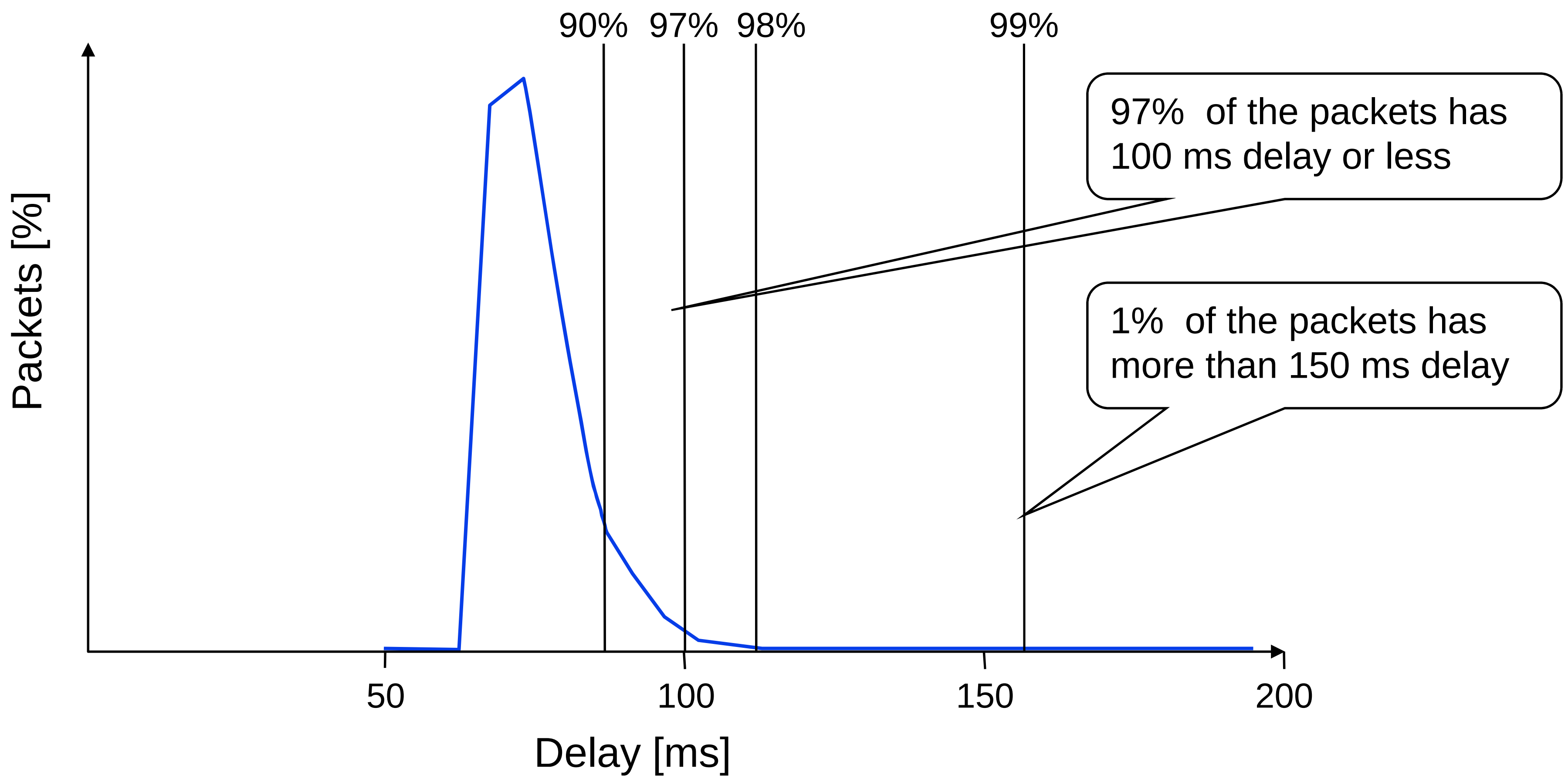
→ **Increases delay**

→ **Introduces jitter**

# Example Audio Application: Playback



# Distribution of Delays on the Internet

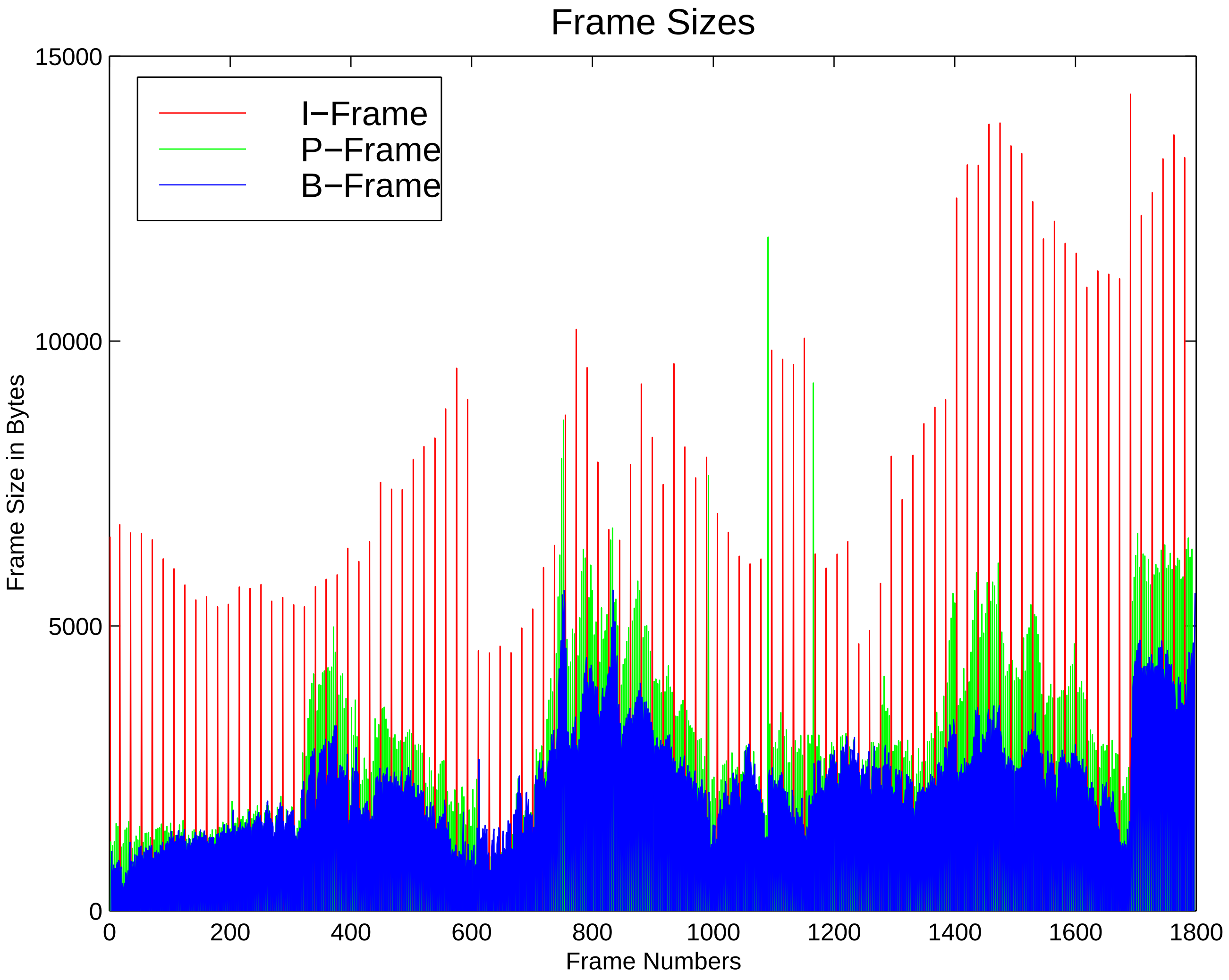


# Example MPEG-4 Video Stream

- **Frame size varies considerably over time:**
  - **Due to frame patterns (I, P, B)**
  - **Due to different scenes**

**→ Variable bit rate traffic**

**→ Burstiness**

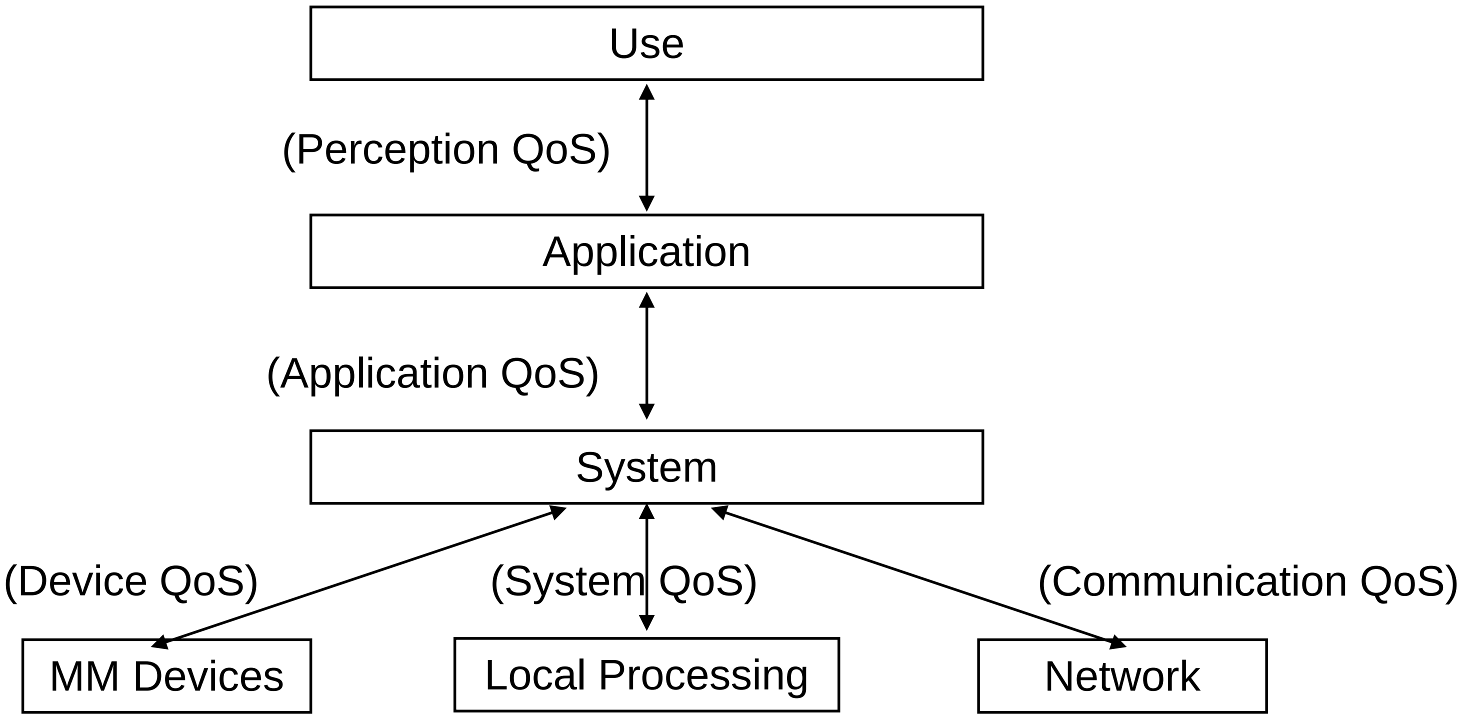


# QoS Definition

**Quality of Service :=**

**"Well-defined and controllable behavior of a system according to quantitatively measurable parameters." [ISO]**

**Layer model, due to variety of possible views:**



# QoS Parameters: Examples

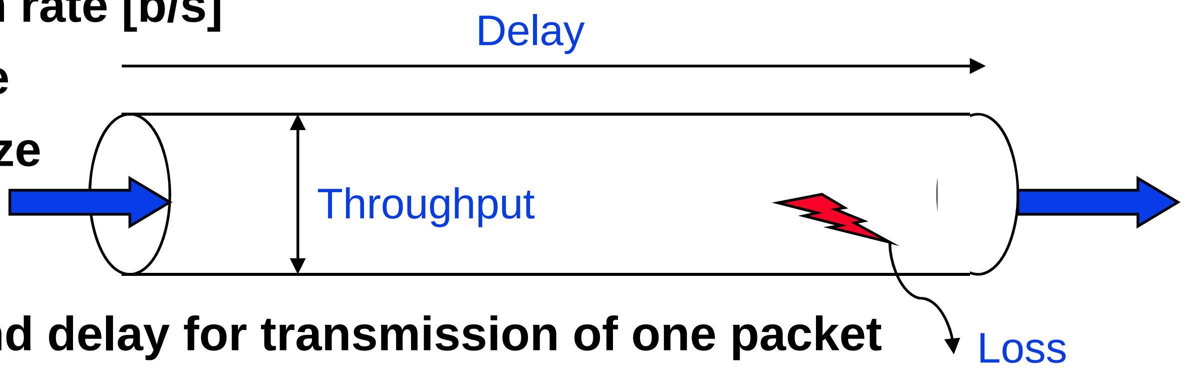
**Services can be described both qualitatively and quantitatively:**

| QoS Layer | QoS Parameters (Examples) | Description |
|---|---|---|
| **Application QoS** | Media quality | Parameters cover application-related requirements |
| | Media characteristics | |
| | Media transmission characteristics | |
| **System QoS** | Throughput | Parameters cover requirements on OS and communication services:<br>- Quantitative parameters |
| | Latency | |
| | Response time | |
| | Error detection and correction characteristics | |
| | Memory and buffer specification | |
| | Inter-stream synchronization | - Qualitative parameters |
| | Sequencing requirements | |
| | Error handling mechanisms | |
| | Scheduling mechanisms | |
| **Comm. QoS** | Packet size | Parameters cover low-level requirements on network services |
| | Average/maximum packet rate (throughput) | |
| | Burstiness | |
| | Processing time for packet in network node | |
| | Jitter - " - | |
| **Device QoS** | Average/maximum disk access time | Parameters cover time-related and performance requirements on individual devices |
| | Data transfer rate of disk | |

# QoS Parameters: Example Transport System

**Common parameters concerning the transport system:**

- **Throughput**
  - **Maximum long-term rate [b/s]**
  - **Maximum burst size**
  - **Maximum packet size**

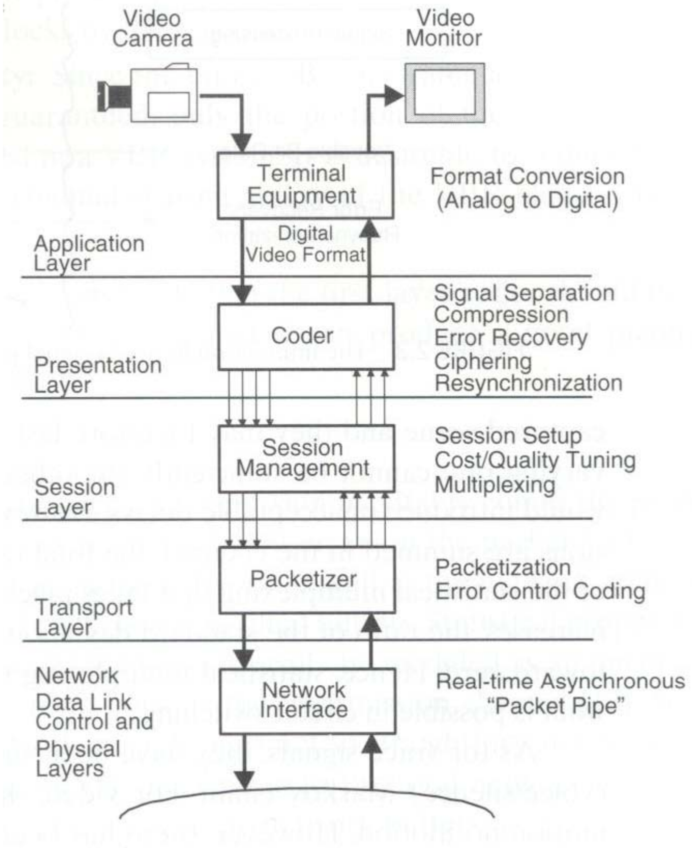



- **Delay / latency**
  - **Maximum end-to-end delay for transmission of one packet**
  - **Jitter := maximum variance of transmission times**
- **Loss / reliability**
  - **Loss rate := maximum number of losses per time interval**
  - **Loss size := maximum number of consecutively lost packets**
  - **Sensitivity class: ignore / indicate / correct losses**
- **But also: security, costs, stability (resilience)**

# Typical QoS Requirements

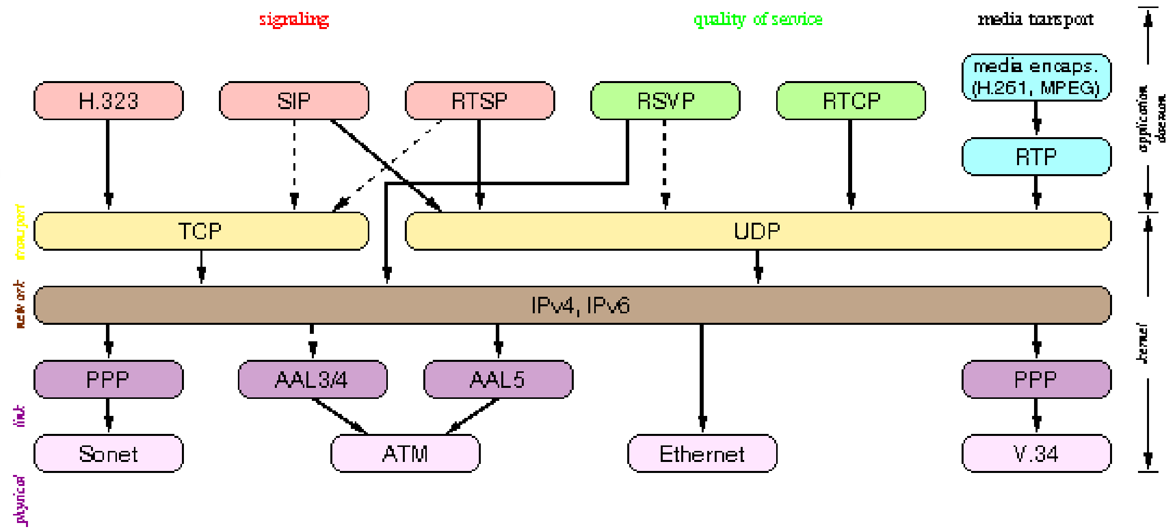
| *QoS* | *Max. latency [s]* | *Max. jitter [ms]* | *Throughput [Mb/s]* | *Bit error rate* | *Packet error rate* |
|---|---|---|---|---|---|
| Voice | 0.25 | 10 | 0.054 | $< 10^{-3}$ | $< 10^{-4}$ |
| Video (TV) | 0.25 | 100 | 100 | $< 10^{-2}$ | $< 10^{-3}$ |
| Compressed video | 0.25 | 100 | 2 - 10 | $< 10^{-6}$ | $< 10^{-9}$ |
| Image | 1 | - | 2 - 10 | $< 10^{-4}$ | $< 10^{-9}$ |
| Data (file transfer) | 1 | - | 2 - 100 | 0 | 0 |
| Real-time data (control) | 0.001 – 1 | - | < 10 | 0 | 0 |

# How to Provide End-to-End QoS in a Complex System?




**In the processing nodes and ...**

**... across the network (protocol zoo)**

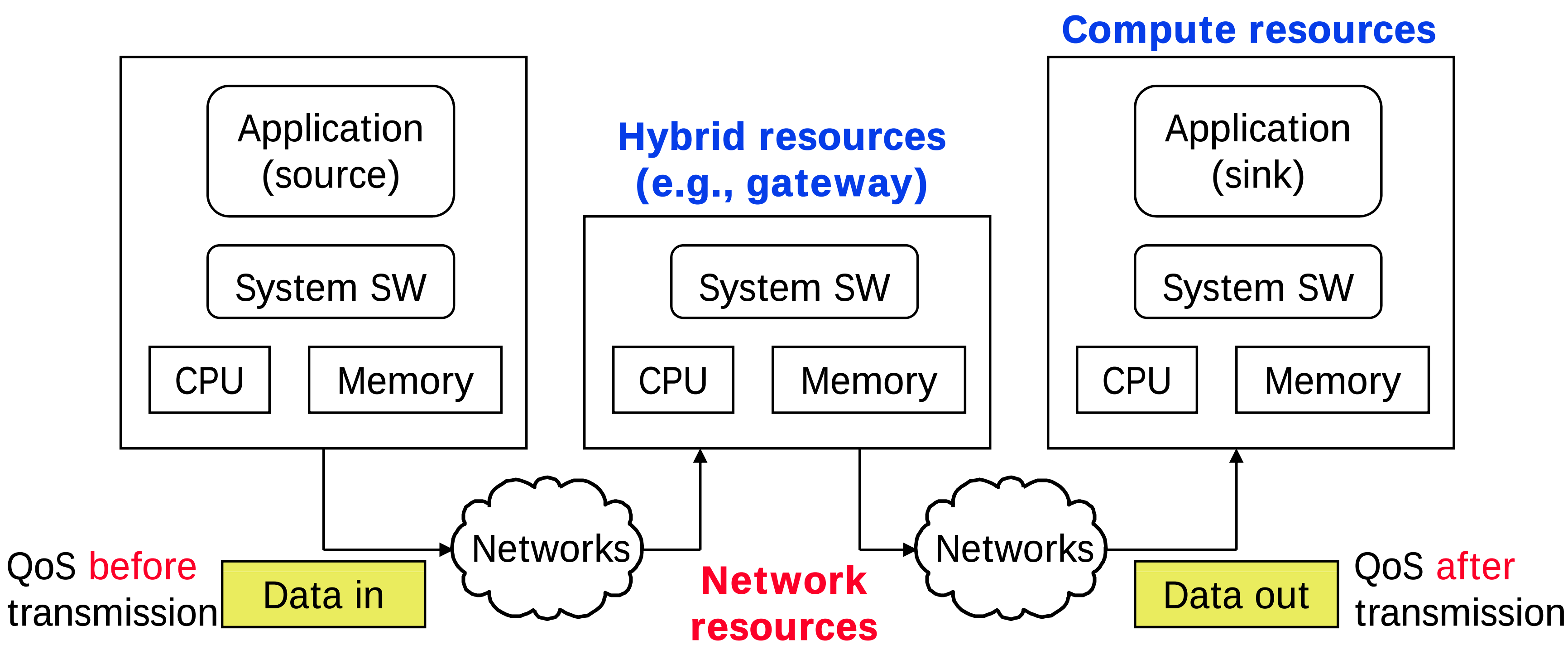


# By Managing Resources in an Appropriate Way!

**Services (and Qos) are provided by resources:**

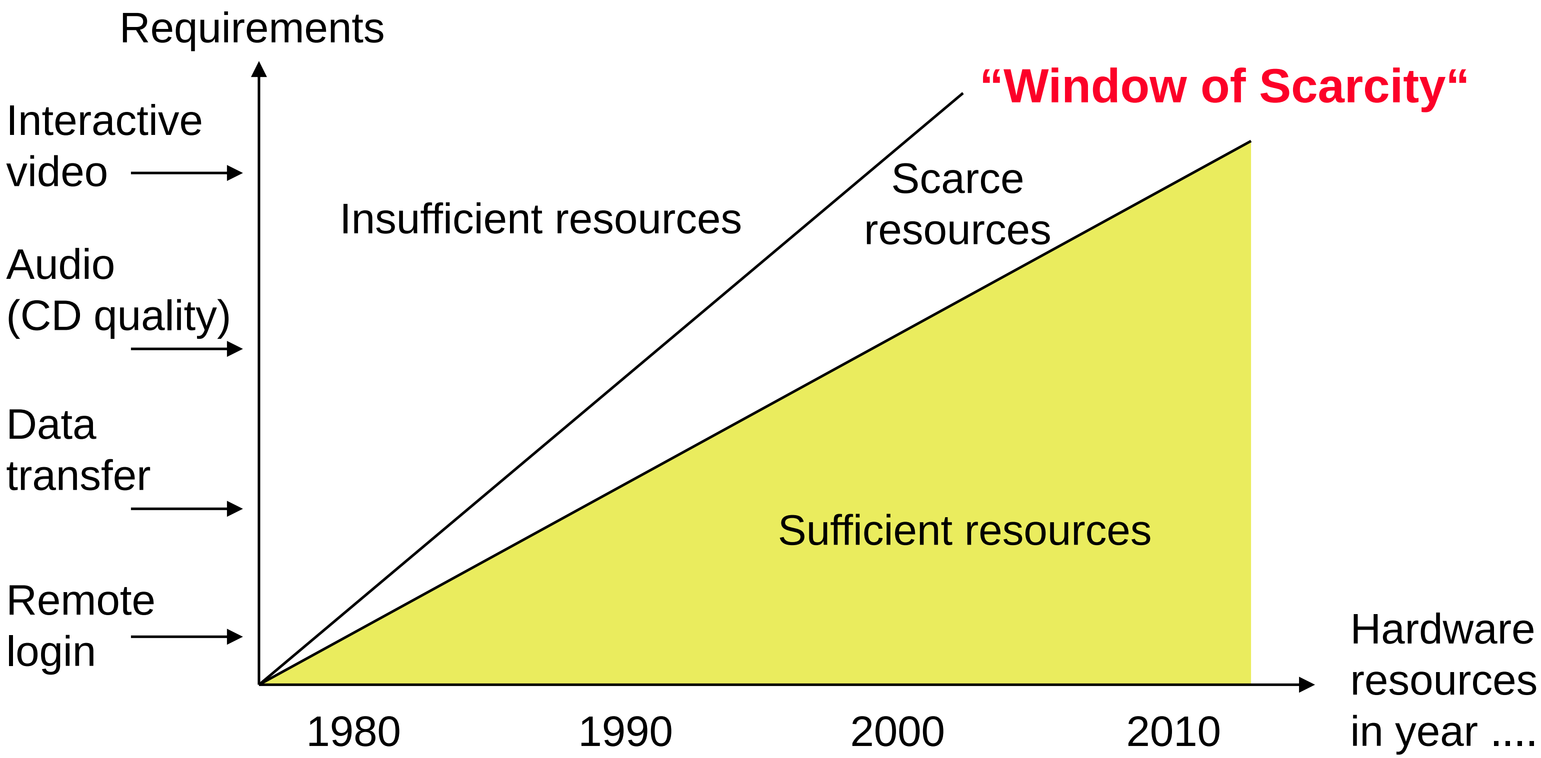
- **Active resources: CPU, I/O subsystem, network adapter, router, ...**
- **Passive resources: provide “space“; file system, buffer, link b/w, ...**

**Common parameter: capacity → allows quantitative description**



# Resource Management: Goal

**Starting point: scarce, but sufficient resources**



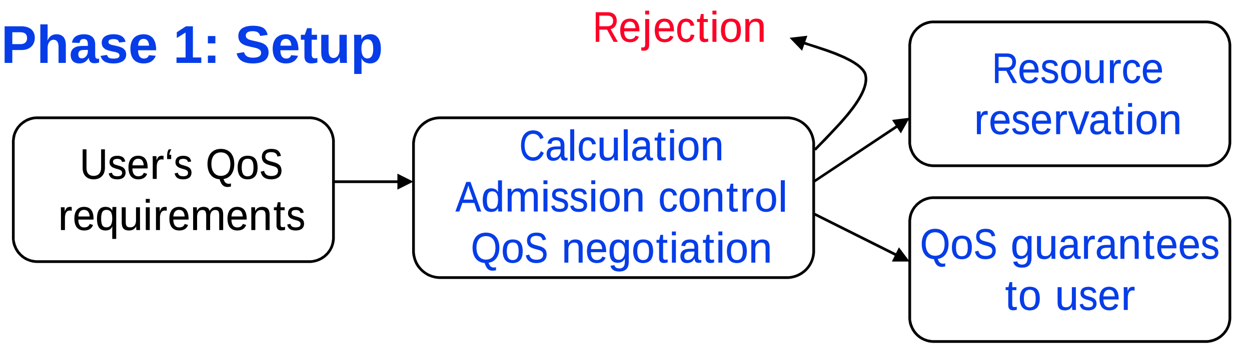

**Goal: provide best service at lowest possible resource consumption / costs**

# Resource Management: Approaches

**Approaches (in other words: relationship QoS – resources):**

1. **Resource reservation**

### Phase 1: Setup

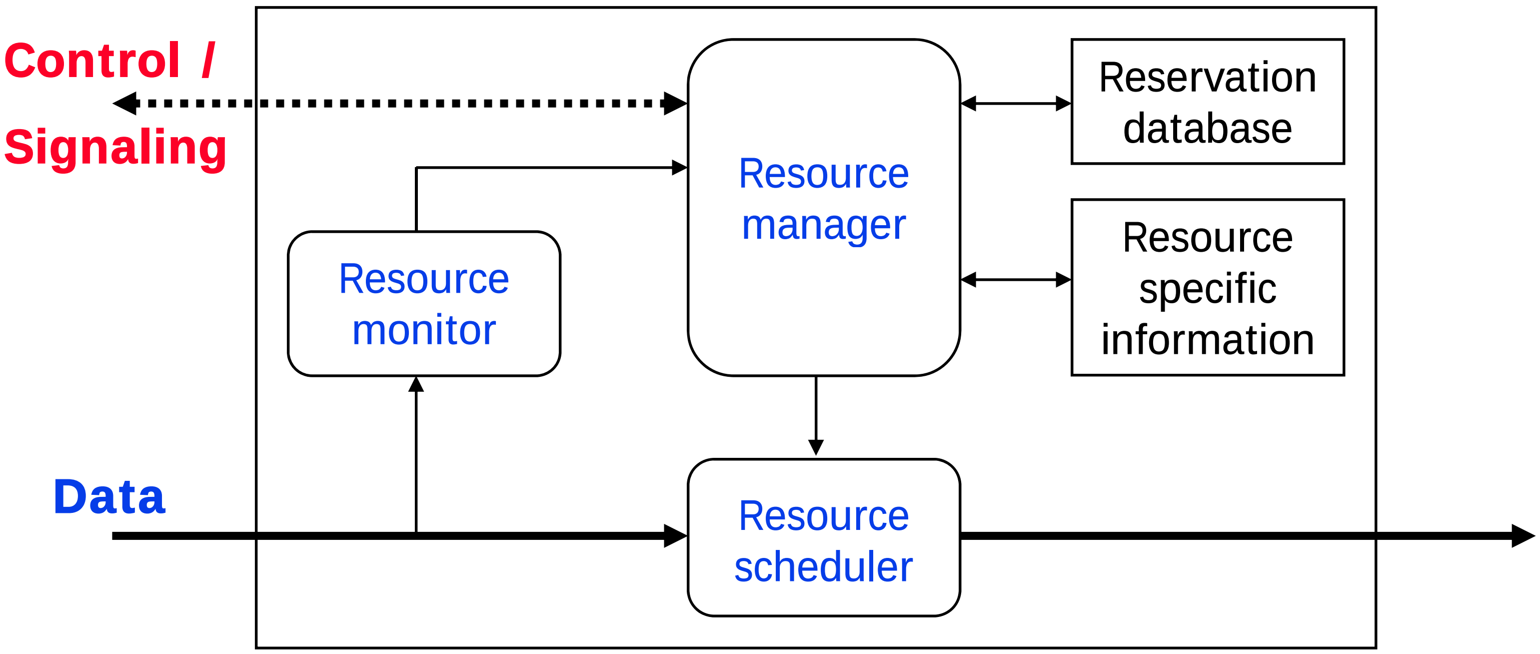



### Phase 2: Processing / Transmission


2. **Adaptation: data streams (and QoS) are adapted according to status of resources; e.g., CPU/network load, link bandwidth, congestion, ...**

# Resource Management: Architecture

## “Generic“ node architecture:



# Static Resource/QoS Management Functions

- **QoS specification**
  - **Qualitatively and quantitatively, on various layers**
  - **E.g., deterministic ranges for delay, throughput, and reliability parameters**
- **Negotiation**
  - **Goal: contract b/w user‘s app. and system (service/QoS provider)**
  - **Application may accept lower QoS level for lower cost**
- **Admission control**
  - **Are requested resources available and can QoS be provided?**
  - **If test is passed, the system has to guarantee the promised QoS**
- **Resource reservation**
  - **May be necessary to provide guaranteed QoS**
  - **Dominating approach today (conceptually, not in implementations)**

# Dynamic Resource/QoS Management Functions

- **Monitoring**
  - Notices deviation from QoS level
  - At a certain level of granularity (e.g. every 100 ms)
- **Policing**
  - Detects participants not adhering to the contract
  - E.g. source sends faster than negotiated (e.g. 30 fps)
- **Maintenance**
  - Attempt to sustain the negotiated QoS
  - E.g. the system acquires more resources
- **Renegotiation / adaptation**
  - User may be able to accept lower QoS and renegotiate with system
  - Or/and tries to adapt data stream(s)

# Setup Phase: QoS Specification

**Example: sample ATM QoS parameters (for IP, “cell“ ≅ “packet“)**

- **Traffic descriptor:** provided by user to describe traffic (via QoS API)

| | | |
|---|---|---|
| PCR | *Peak Cell Rate* | Max. cell rate input into network |
| SCR | *Sustained Cell Rate* | Avg. cell rate |
| MCR | *Minimum Cell Rate* | Min. cell rate (expected by user) |
| MBS | *Maximum Burst Size* | Max. number of back-to-back cells |

- **Service descriptor:** user‘s QoS requ‘s, negotiated b/w user & network

| | | |
|---|---|---|
| CTD | *Cell Transfer Delay* | Min. and max. cell delay |
| CDVT | *Cell Delay Var. Tolerance* | Max. acceptable jitter (for PCR, SCR) |
| CLR | *Cell Loss Ratio* | Max. ratio of cells lost |

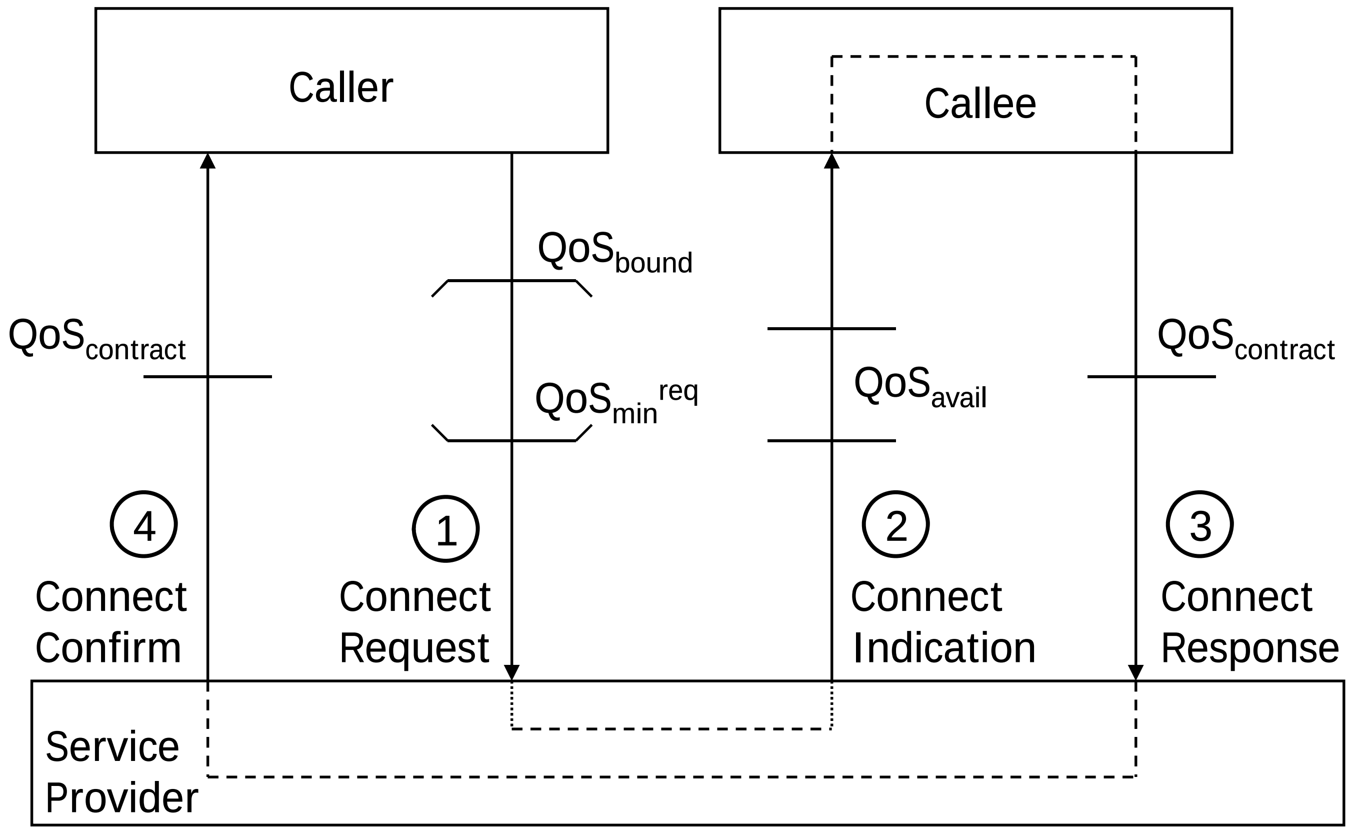
- **Service parameters:** QoS delivered, non-negotiatable, measured

| | | |
|---|---|---|
| CDV | *Cell Delay Variation* | Actual variance in cell delays |
| CER | *Cell Error Ratio* | (Max.) Ratio of erroneous cells |
| CMR | *Cell Misinsertion Ratio* | (Max.) Ratio of mis-delivered cells |

# Setup Phase: QoS Negotiation

## Example: Trilateral peer-to-peer QoS negotiation

- **QoS requirement:** $QoS_{min}^{req}$ and upper bound $QoS_{bound} > QoS_{min}^{req}$
- **Goal:** agreement of all participants on value $QoS_{contract}$



# Setup Phase: Admission Control

**Check whether requested resources are and will be available**

**Especially important for shared resources, like:**

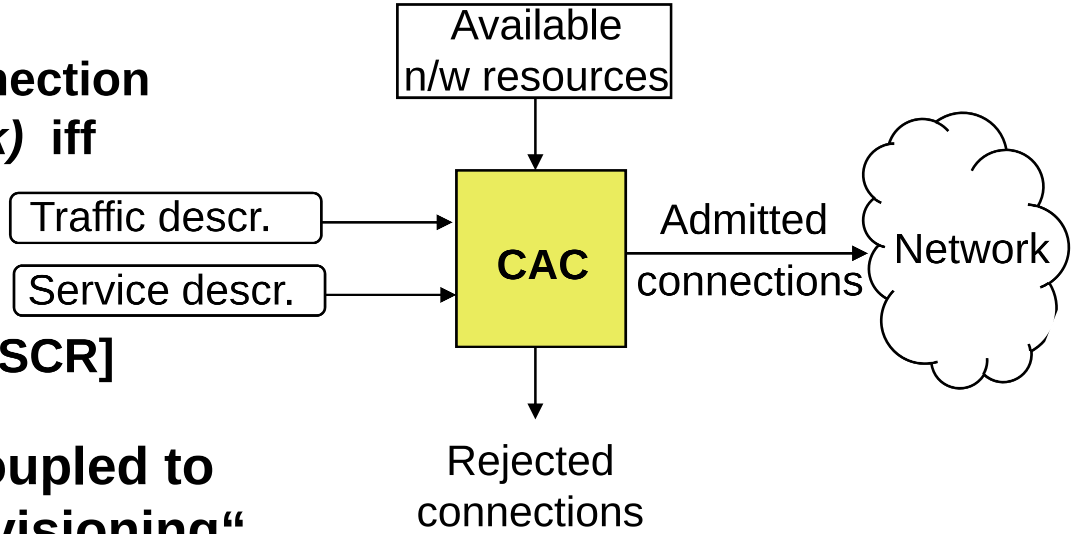
- **CPU**
- **Network paths**
- **Buffer space**

## Example: ATM connection admission control (CAC)

**Simple rule: admit new connection *k* with bandwidth requ. *B(k)* iff**

$$B(k) + \sum_i B(i) \leq B^{total}$$

**[B(k) may pertain to PCR or SCR]**

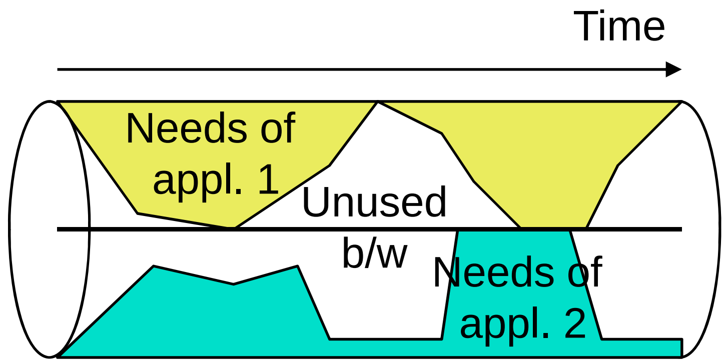



**Must be strongly related/coupled to costs, otherwise “over provisioning“**

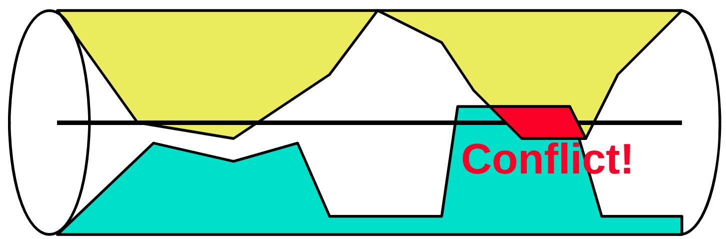
# Setup Phase: Resource Reservation

**Fundamental concept for reliable enforcement of QoS guarantees**

**Variants:**




- **Pessimistic reservation:**
  - **Relies on worst-case assumptions**
  - **Results in guaranteed QoS**
  - **E.g., bandwidth reservation using PCR**



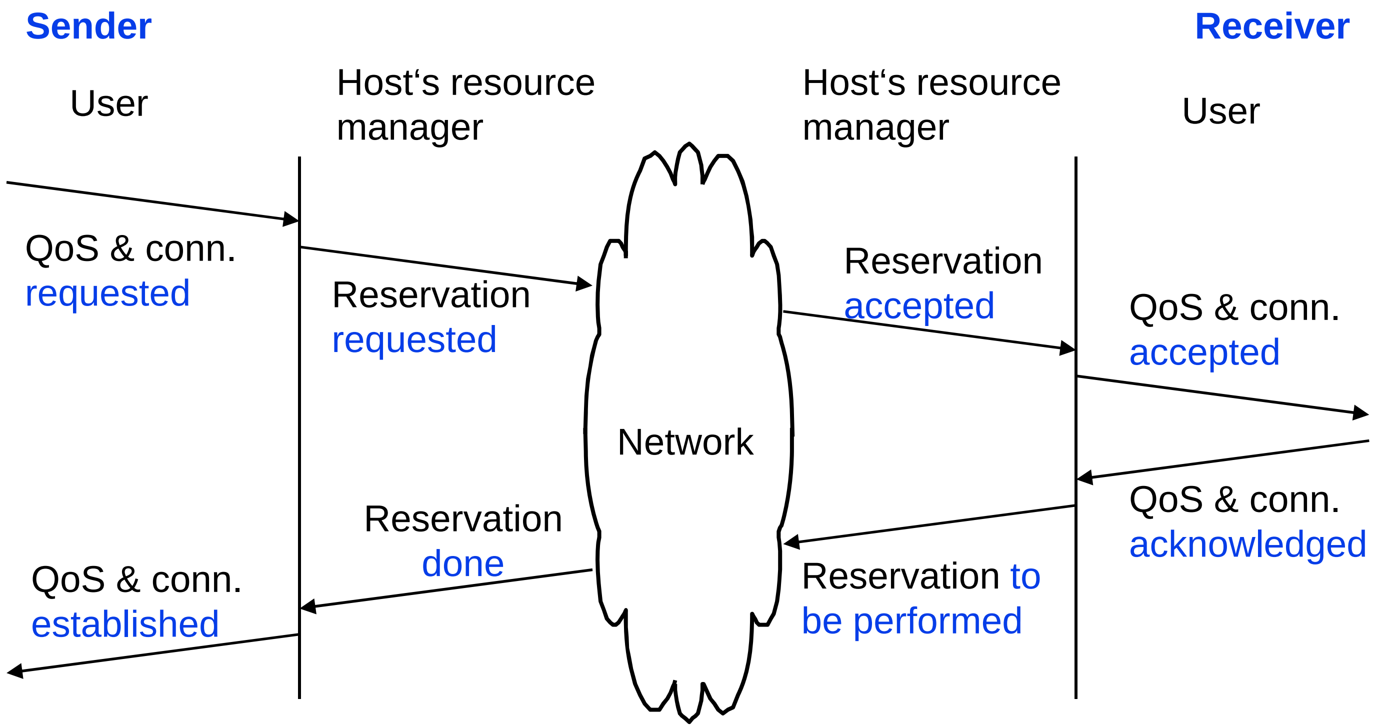

- **Optimistic reservation:**
  - **Relies on statistical multiplexing**
  - **Results in statistical QoS**
  - **Can lead to QoS violations and congestion**
  - **E.g., bandwidth reservation using SCR**

# Setup Phase: Reservation Protocols

**Reservation protocols in general referred to, and combined with, signaling protocols (e.g., for connection establishment)**

**Two phases:**

1. **Reservation is requested and checked (admission control)**
2. **Reservation is actually performed, if checks were successful**



# Processing Phase

**Enforce QoS by:**

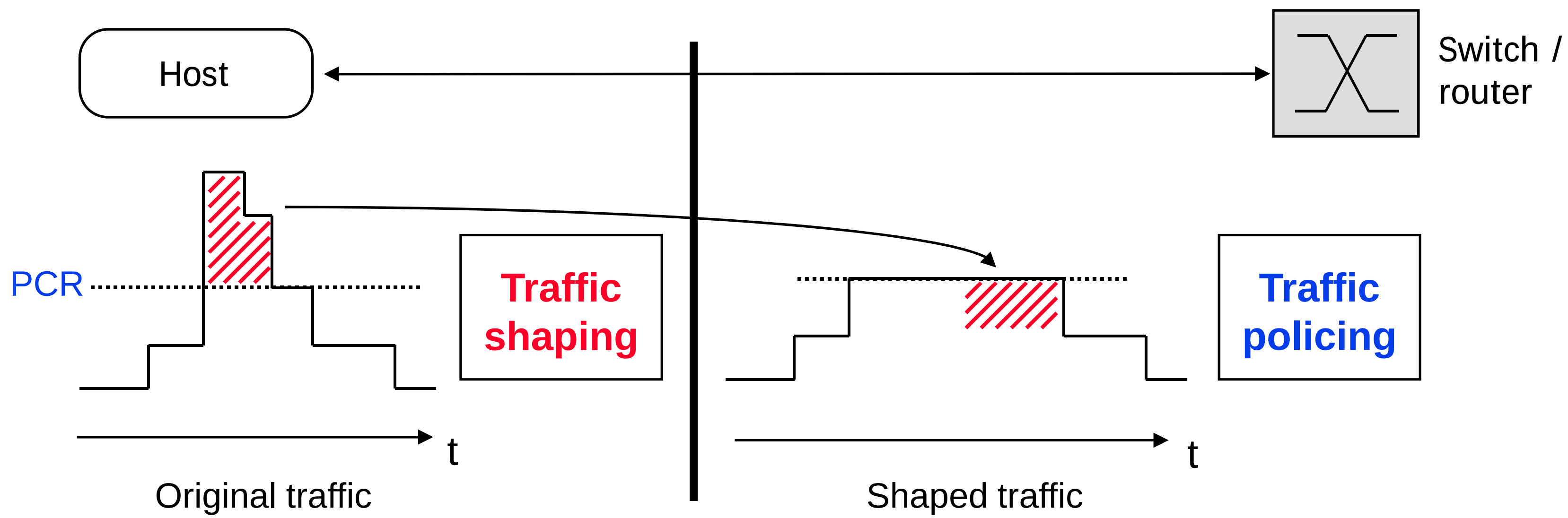
- **Resource scheduling**
- **QoS / traffic monitoring**
- **Maintenance of resource reservations**
- **Potential acquisition of new resources**
- **Traffic shaping**
- **Traffic policing**
- **QoS feedback and traffic adaptation**
- **Potential QoS renegotiation**

# Traffic Shaping and Policing (1)

- **Mechanisms to ensure that traffic conforms to QoS contract, does not overload network, etc.**

- **Traffic shaping:**
  - **Usually done in hosts**
  - **Change shape of traffic by e.g. postponing packets**

- **Traffic policing:**
  - **Usually done in switches/routers**
  - **Cut off traffic to conform to contract by e.g. discarding packets**



# Traffic Shaping and Policing (2)

- **Major goal: protect**
  - **network resources and**
  - **other traffic**

  **against congestion and conflicts, caused by ill-behaving traffic sources**

- **Mechanisms:**
  - **Marking packets/cells as candidates for later discarding**
  - **Immediate discarding of packets/cells (traffic policing)**
  - **Buffering of packets/cells in host (traffic shaping)**

- **How traffic can be shaped (ATM examples):**
  - **Reduction of PCR (Peak Cell Rate)**
  - **Reduction of CDV (Cell Delay Variation)**
  - **Reduction of MBS (Maximum Burst Size), i.e., limitation of bursts**

# Traffic Shaping / Policing: Leaky Bucket Algorithm (1)

**Single algorithm for both traffic shaping and policing:**
**Leaky Bucket Alg. or (in ATM terms) Generic Cell Rate Alg. (GCRA):**

**Each arriving packet/cell is classified as**

- **conforming (arriving within valid time interval)**

**or**

- **non-conforming (arriving too early)**

**Two parameters:**

- ***Increment T:*** **packet/cell interarrival time; e.g., *T* = 1 / PCR**
- ***Limit L:*** **tolerated variance thereof; e.g., *L* = CDVT**

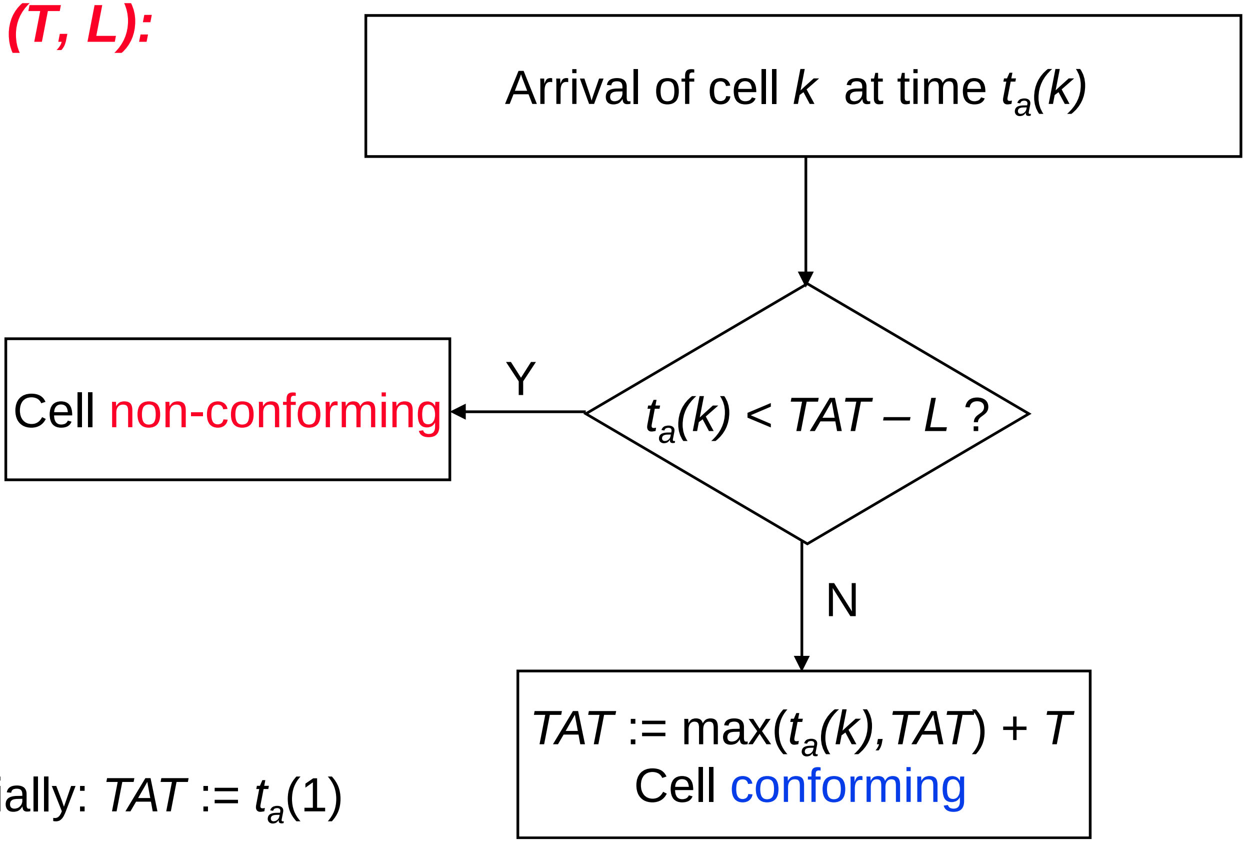
***→ GCRA (T, L)***

# Traffic Shaping / Policing: Leaky Bucket Algorithm (2)

**Notation:**

- ***$t_a(k)$:*** **arrival time of packet/cell *k***
- ***TAT:*** **theoretical arrival time of next packet/cell**

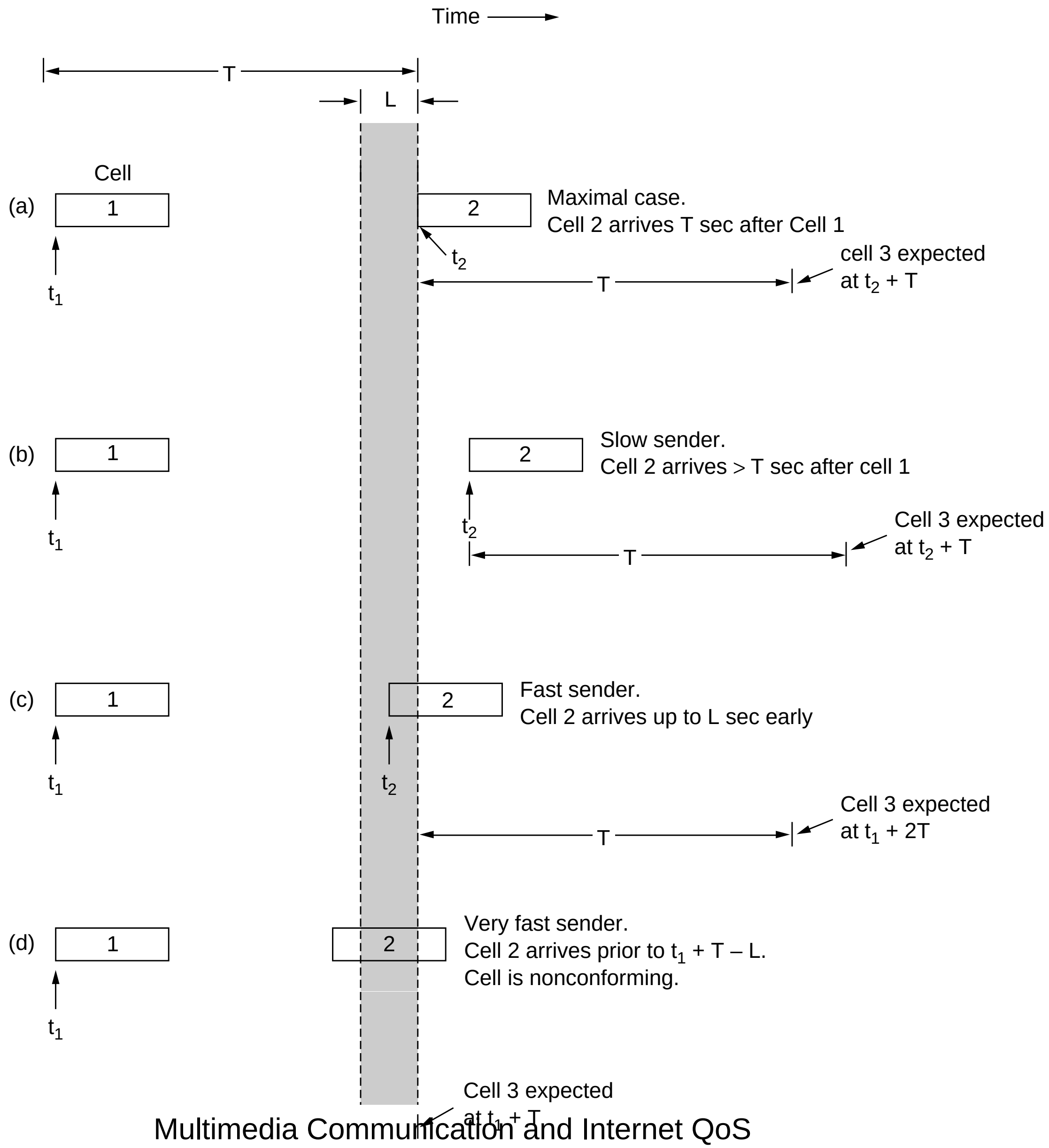
***GCRA (T, L):***




Initially: $TAT := t_a(1)$

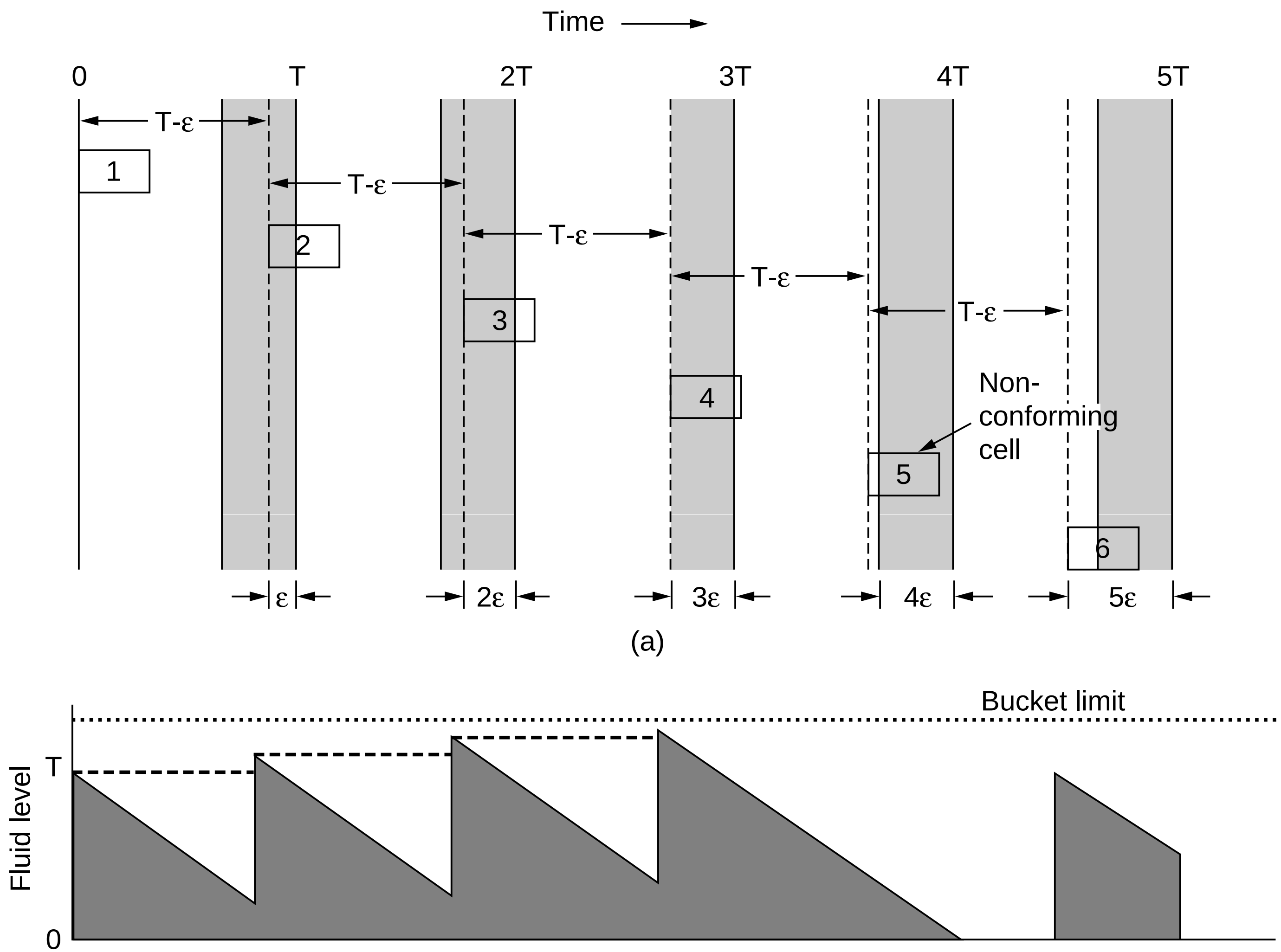
# Traffic Shaping / Policing: Leaky Bucket Algorithm (3)

**Illustration:**



# Traffic Shaping / Policing: Leaky Bucket Algorithm (4)

## “Leaky bucket“ metaphor and interpretation:




(a)

(b)

# Traffic Shaping / Policing: Leaky Bucket Algorithm (5)

**Use of *GCRA(T,L)*:**

**1. Formal definition and policing of CBR and VBR traffic (Constant / Variable Bit Rate) re. PCR and CDVT:**

**GCRA (1/PCR, CDVT)**

**2. Formal definition and optional policing of VBR traffic re. SCR and burstiness:**

**GCRA (1/SCR, BT)**

**where BT = (MBS – 1) (1/SCR – 1/PCR)**

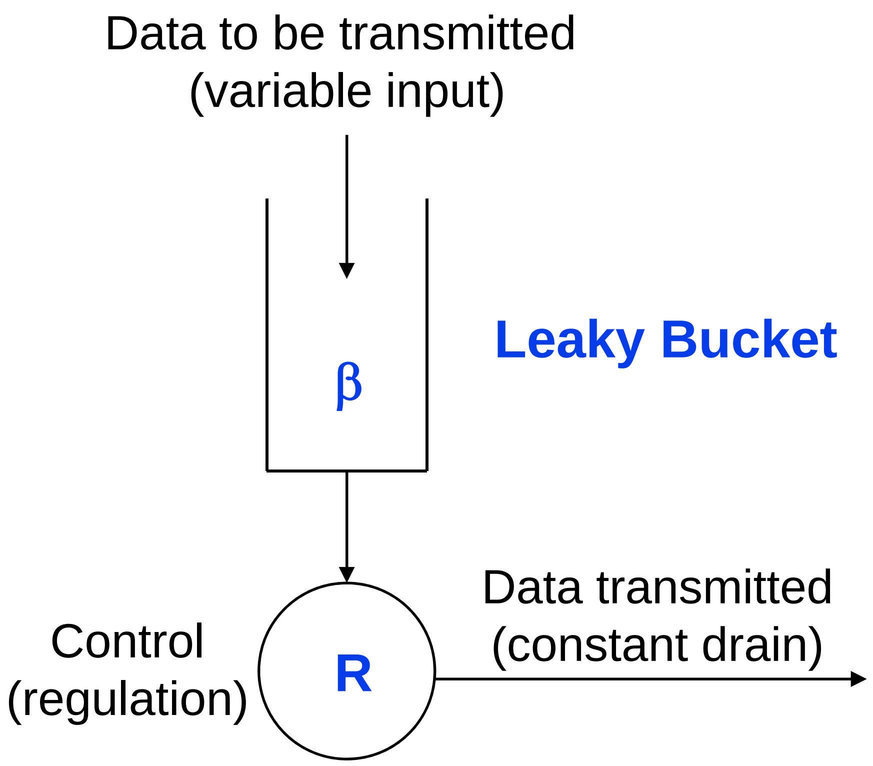
**(BT ... Burst Tolerance**
**MBS ... Maximum Burst Size)**

**3. Traffic shaping re. these parameters**

# Traffic Shaping / Policing: Leaky Bucket Algorithm (6)

**Characteristics and alternative notation / illustration:**

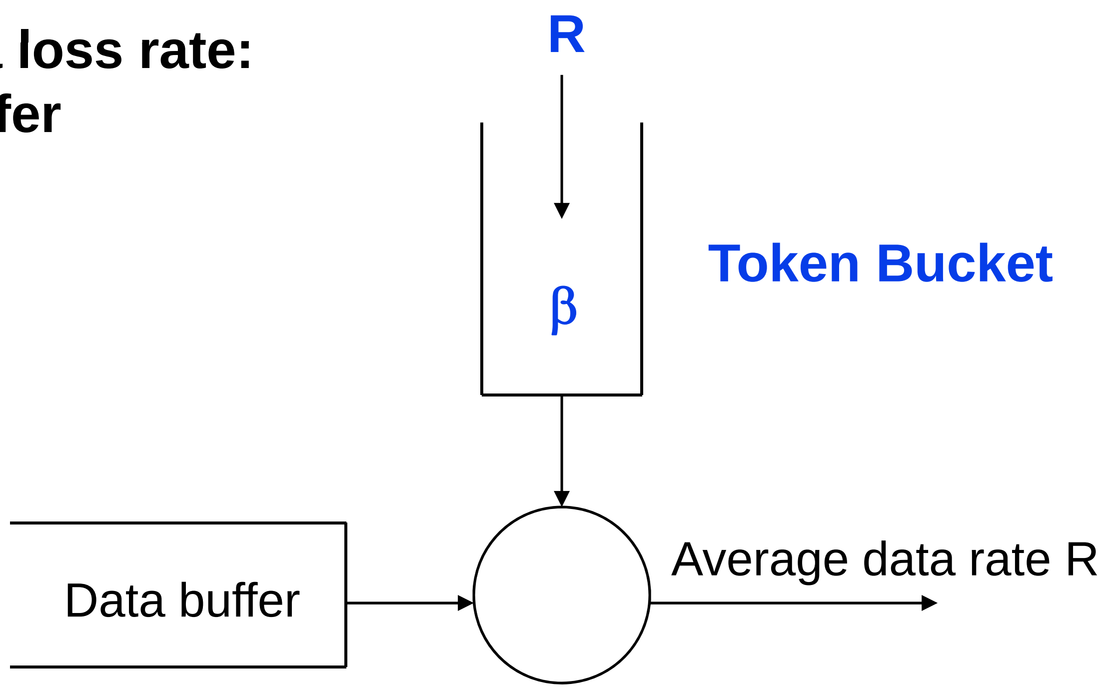
- **Simple isochronous algorithm**
- **β ... bucket size**
  **R ... constant output (data rate)**
- **Problems with bursts:**
  **→ packet losses**



# Traffic Shaping / Policing: Token Bucket Algorithm

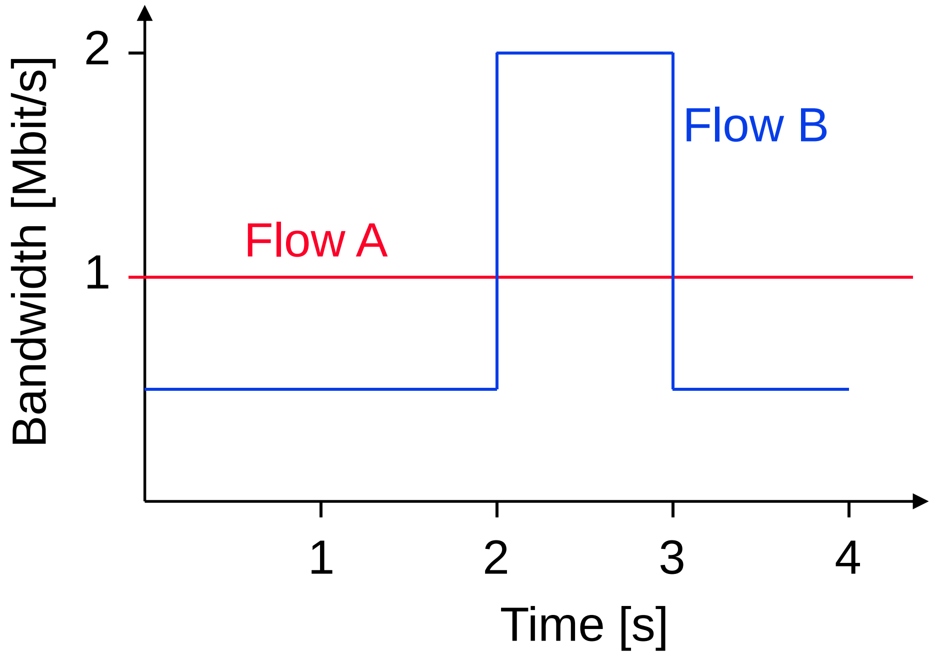
## Token Bucket Algorithm:

- Improvement over Leaky Bucket to tolerate limited burstiness
- Data transmission consumes tokens
- Bursts are limited in interval $T$ by: $\beta + T \cdot R$
  (R ... token placement rate = average packet rate)
- Reduction of data loss rate:
  separate data buffer



# Traffic Shaping / Policing: Token Bucket Example

**Example:** **two traffic flows with equal average data rate, but different Token Bucket descriptions**



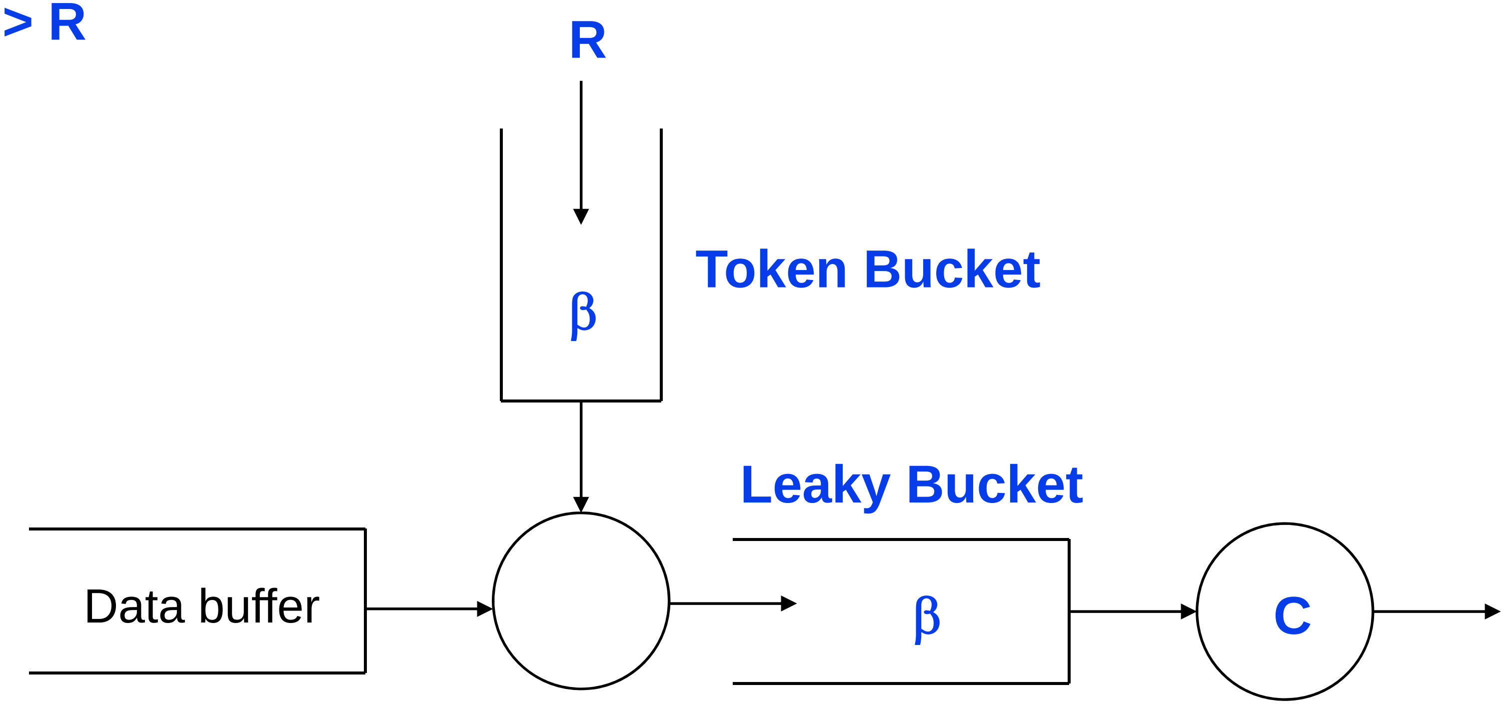

**Flow A:** **CBR traffic with R = 1 Mbit/s, β = 1 byte**

**Flow B:** **VBR traffic with R = 1 Mbit/s, β = 1 Mbit**

# Traffic Shaping / Policing: Token + Leaky Bucket

**Token Bucket with Leaky Bucket rate control:**

- **Smoothing of bursts of Token Bucket by Leaky Bucket at output**
- **C ... maximum data rate**
  **R ... average data rate**
- **C > R**



# 2

# Integrated Services (IntServ) and the Resource Reservation Protocol (RSVP)

# IntServ and RSVP

- Early (mid 90‘s) IETF standard for end-to-end QoS provisioning:
  - Extension to Internet architecture
  - For forthcoming real-time applications
  - E.g., packet voice, packet video, distributed simulation
- QoS for particular data / traffic flows
- Flows categorized into service classes
- Generally requires reservation of network resources in routers along the path(s) of the flows as well as in the end hosts

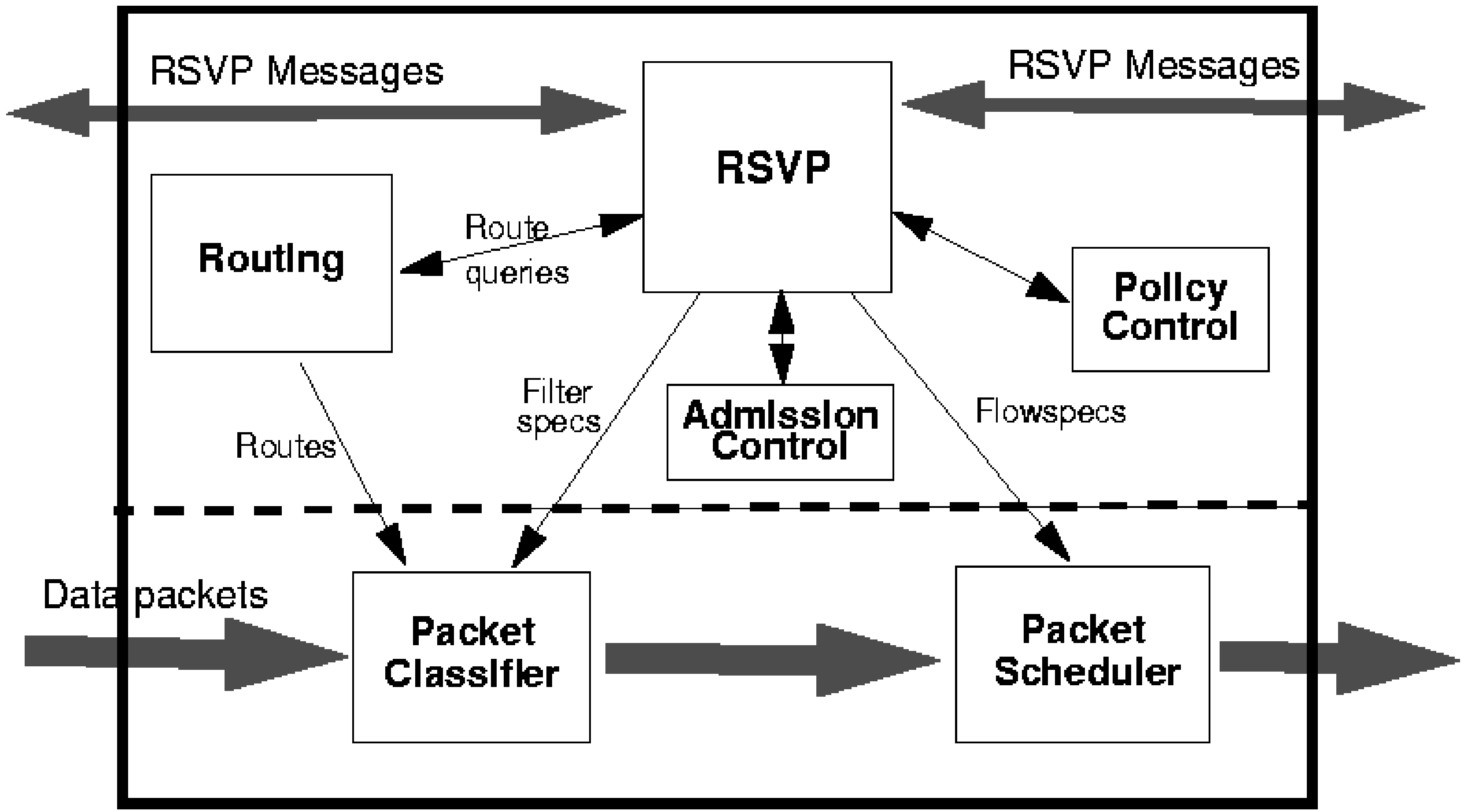
# IntServ Services Classes

1. **Guaranteed service**
   - **Fixed delay bound**
   - **For A/V, no packet arrives after its play back time**
   - **Early packets must be buffered**
   - **For “hard“ real-time applications**
2. **Controlled load service**
   - **Probabilistic delay bound**
   - **Gives the illusion of reserved channels, under reasonable load**
   - **Based on routers’ queuing alg. + admission control**
   - **For delay-adaptive applications**
3. **Best effort service**
   - **Available per default**
   - **For “elastic“ (adaptive) applications**

**Resource consumption**

# Components / Mechanisms (1)

**Setup, routing, background control** **(on end systems, impl. on user-level)**




**Traffic control** **(on end systems, impl. in the OS kernel)**

# Components / Mechanisms (2)

## 1. Reservation setup ($\rightarrow$ RSVP)

**Passes the QoS request from originating end system to each router along the data path or, in case of multicasting, along the branches of the delivery tree**

**Consists of:**

- **FlowSpec: defines the desired QoS**
- **FilterSpec: defines the subset of flows to receive this QoS**

## 2. Admission control

**Allocates the necessary node and link resources to satisfy requ. QoS**

**Establishes state for flow, if accepted**

## 3. Policy control

**Ensures that QoS request and reservation is administratively possible**

**(Routing is separate, not part of IntServ concept.)**

# Components / Mechanisms (3)

## 4. Packet classifier

Sorts incoming data packets forming flows into appropriate scheduling classes, based on FilterSpec

State (i.e., filter) established by RSVP

## 5. Packet scheduler

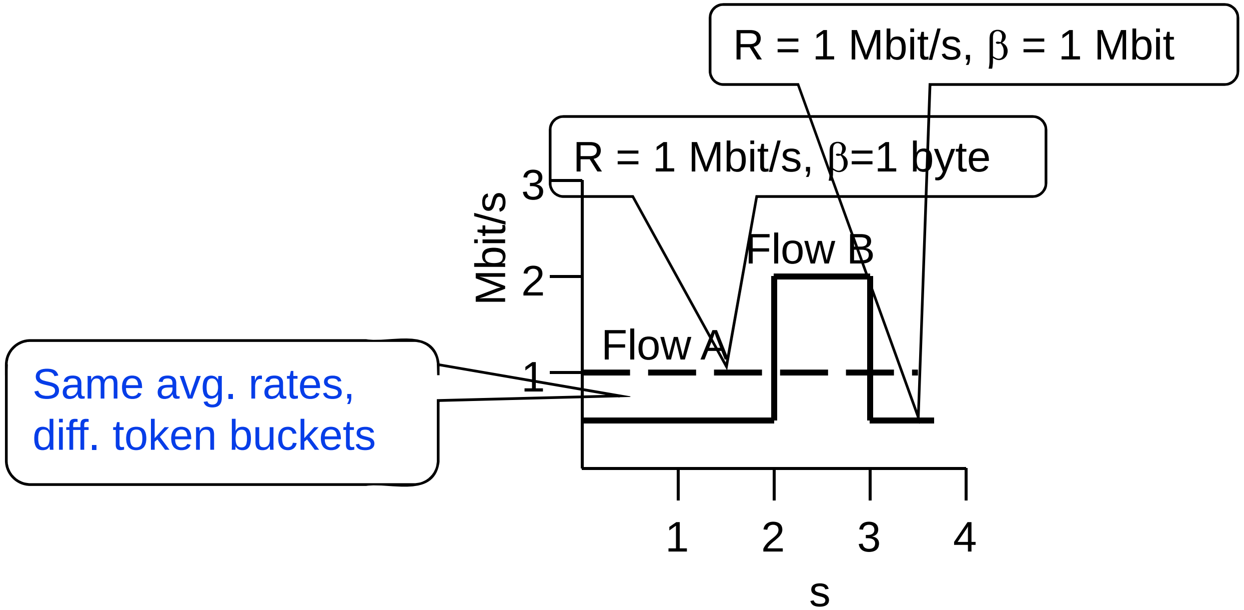
Link-layer QoS mechanism to provide requested QoS (queuing mechanisms in routers, not covered here)

Multiplexes packets from different reserved flows onto the outgoing links, based on FlowSpec

State established by RSVP

# FlowSpec

- **RSpec: characteristics requested from network, e.g.,**
  - **Guaranteed service: delay bound as parameter**
  - **Controlled load: no additional parameters**
- **TSpec: traffic characteristics of the flow, e.g.,**
  - **Average bandwidth and burstiness**
  - **Usually specified by a token bucket, e.g.,**



# RSVP: Features (1)

**RSVP ...**

- **... is a control (signaling) protocol, not a routing protocol**
- **... is used by hosts to request specific QoS from the network**
- **... is used by routers to deliver QoS requests and to establish and maintain states to provide the requested services**
- **... supports heterogeneous hosts, QoS requests, links, etc.**
- **... supports unicast and multicast (in fact, multipoint-to-multi-point communication), i.e., uses IP multicast for data distribution**
- **... does not deliver messages reliably**

# RSVP: Features (2)

**RSVP ...**

- **... uses receiver-initiated reservations**
  - **PATH and RESV messages**
  - **Receivers explicitly request/start reservation**
  - **No need for senders to keep track of many receivers (for multicast)**
  - **Merging of reservation requests possible**
- **... uses soft state in the routers and connectionless model**
  - **States time out automatically (e.g., after 1 minute)**
  - **Can be (and need to be) refreshed periodically**
  - **Responsibility of reservation maintenance is in the end hosts**
  - **Modifications of reservations possible**
- **is therefore robust and adaptive against route and multicast group membership changes**

# Multipoint-to-Multipoint Communication Model

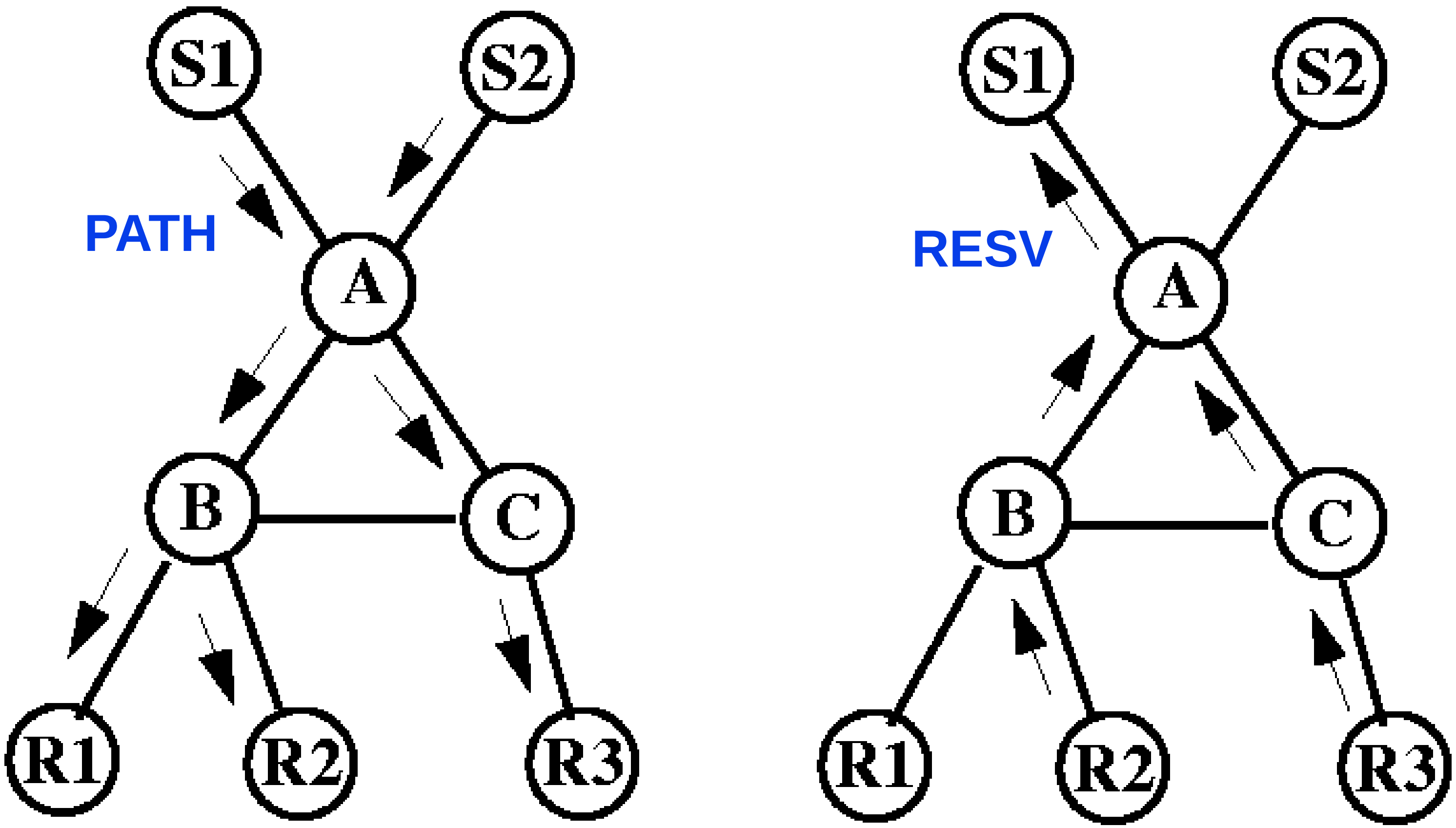
- **Basic communication model: simplex distribution of data from *m* sources to *n* receivers (same destination)**
- **Such an *m*-to-*n* flow is called RSVP session**
- **Examples: video conference: *m* = *n*; broadcast: *m* = 1, *n* >(>) 1**
- **Logical definition of an RSVP session:**

  **(Destination IP address, IP protocol ID, destination port)**
- **To select a subset of the traffic in a session, e.g., particular sender(s) (=: FilterSpec):**

  **(Source IP address, source port)**

# Reservations: Basic Protocol (1)

## PATH and RESV messages:

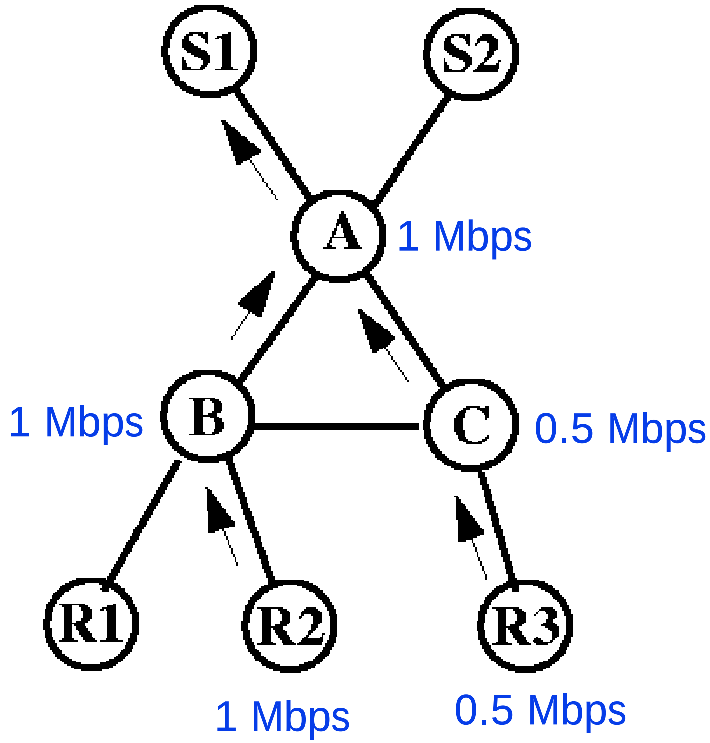


# Reservations: Basic Protocol (2)

- **Source transmits PATH message:**
  - **Conveys TSpec of the source**
  - **Routers figure out the reverse path**
  - **Sent e.g. every 30 seconds**
- **Destination responds with RESV message:**
  - **Conveys TSpec and RSpec (FlowSpec) of the receiver**
  - **Routers on the path try to make appropriate reservations**
- **In case of router or link failure:**
  - **New route will be automatically established by repeated PATHs**
  - **Reservation will be also automatically renewed on the new path**
  - **Soft state on old path will time-out**

# Reservations: Merging Reservations

- **In case of multicast, reservation requests can merge as they travel up the delivery tree**
- **Example:**




- **“Largest“ FlowSpec will arrive at sender(s): Least-Upper-Bound (LUB)**
- **Service-specific routines required to perform FlowSpec merging**

# RSVP Drawback: Poor Scalability

**State is required for each individual flow!**

**Example:**

- **Optical link @ 2.5 Gbps**
- **We can multiplex**

  $$(2.5 * 10^9) / (64 * 10^3) = 39000$$

  **ISDN flows (64 kbps, e.g., voice) onto this link**

**Per-flow management requires huge memory and CPU resources!**

**So revert to best-effort service model again?**

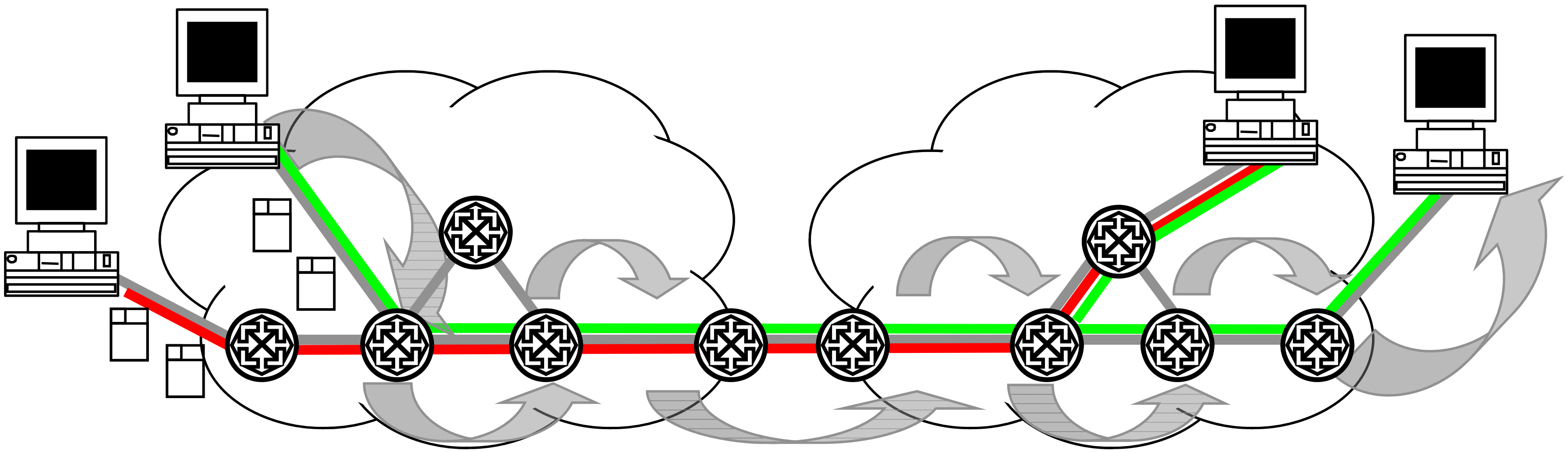
- **Requires (almost) no state about individual flows**
- **Scales well, since “only“ bandwidth and routing tables have to grow**

# 3

# Differentiated Services (DiffServ)

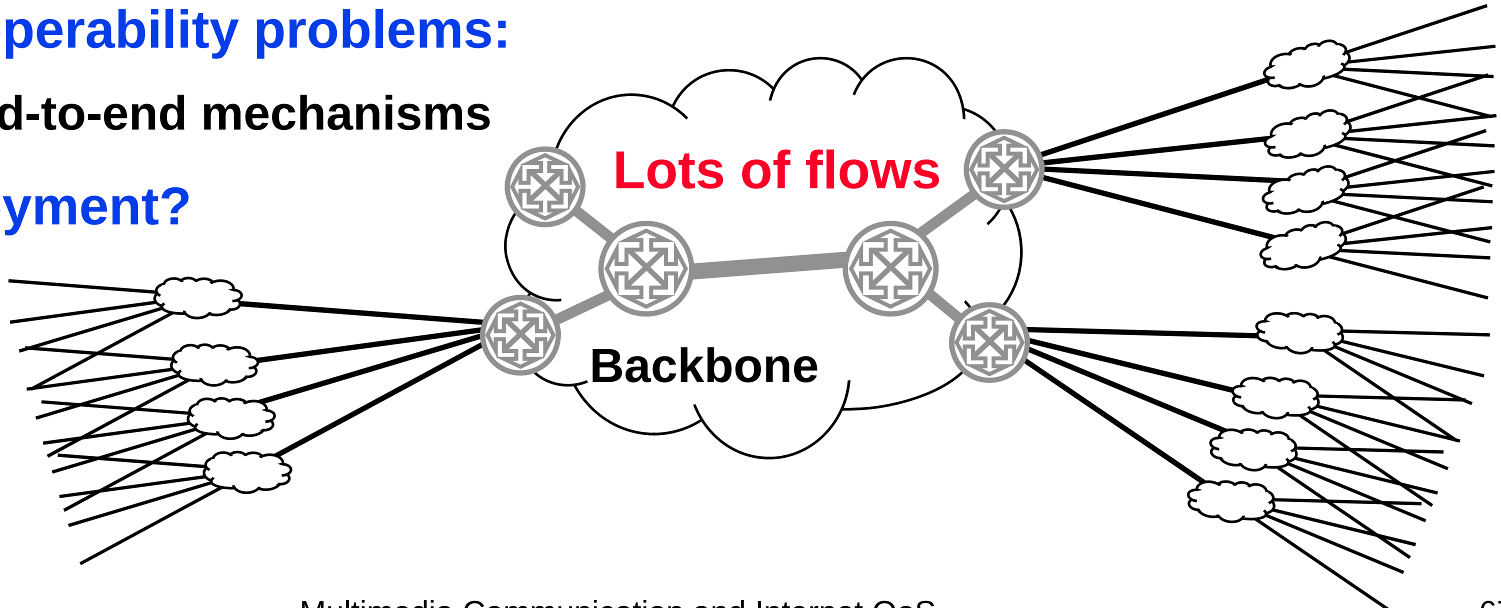
# IntServ/RSVP Review: Major Characteristics

- Per-flow signaling and reservation
- Per-flow service state at every hop (router)
- Strict, end-to-end QoS guarantees
- Focus on multicast
- Virtually connection oriented



# IntServ/RSVP Review: Problems and Drawbacks

- **Complexity:**
  - **Reservation protocols and structure complicated**
    - **Lot of message passing; coordination problems**
  - **Heavy-weight state at every router**
    - **Processing and memory resources required**
- **Scalability problem:**
  - **Lots of flows traverse core (backbone) routers**
- **Interoperability problems:**
  - **End-to-end mechanisms**
- **Deployment?**



# Basic Ideas Toward a More Light-Weight Model

- **Support QoS end-to-end**
- **But keep per-flow state and packet forwarding overhead out of the core**

→ **Keep the architecture simple within the backbone, permit higher complexity at edge routers**

- **Some data is more important than other data**
- **Specify relative priorities of packets**
- **Charged differently for high and low priority classes**

→ **Just provide for service differences, no explicit guarantees**

- **Get rid of complexity of per-flow signaling & state maintenance**

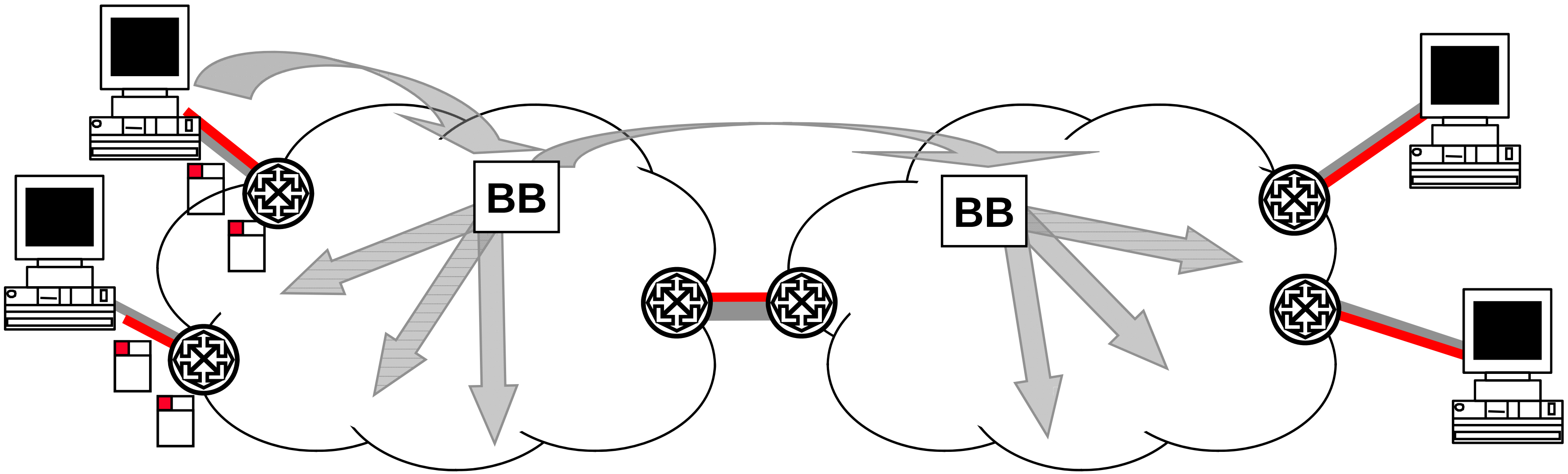
→ **Deal with flow aggregates rather than individual flows**

# DiffServ Goals

- Simpler than IntServ
- Lightweight, scalable service discrimination suitable for network backbones
  - No per-flow signaling and resource allocation
  - No per-flow state
  - Separation of service from signaling
- Aggregation of traffic into priority classes
  - Focus on aggregates, not flows
  - Customer agreements are relatively static
  - Ability to charge differently for different services
- Simple system at first, expand if needed in future
  - Allow for incremental deployment
  - Chosen for Internet2 QoS

# DiffServ Overview (1)

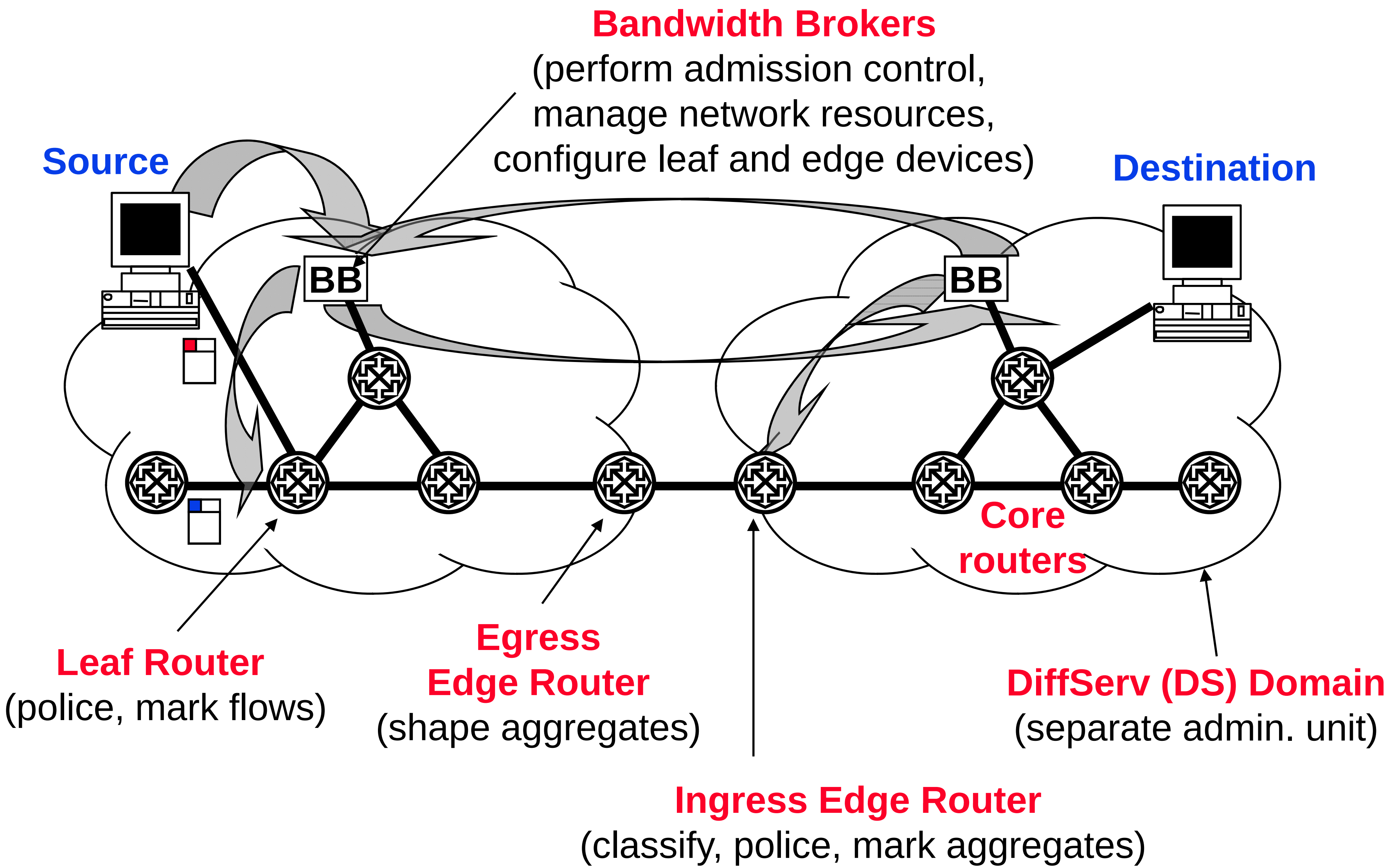
- Exploit edge/core router distinction for scalability
  - Policing at edge to get services (complex)
  - Forwarding in core to realize services (simple, fast)
- Relative-priority scheme
  - Packets are marked with “behavior” (essentially, priority) ...
  - ... and treated according to small set of packet-forwarding schemes
- Abstract/manage each network’s resources → bandwidth brokers (BBs)



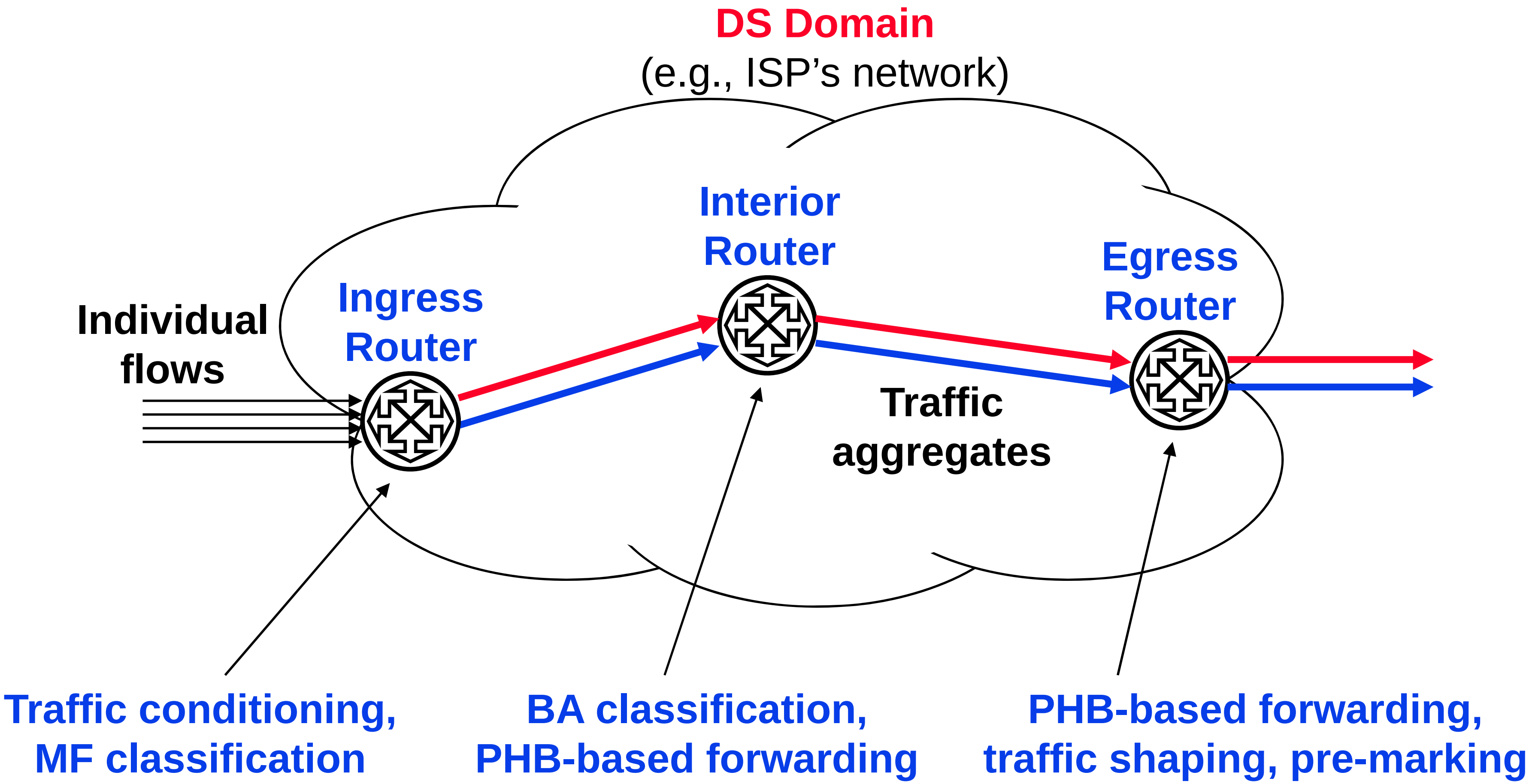
# DiffServ Overview (2)

- Applications contract for specific QoS traffic profiles
  - Traffic is policed at network periphery (leaf routers) ...
  - ... and assigned to different service classes (packets marked) ...
  - ... according to Service Level Agreements (SLAs) between customer and ISP
- Networks ("clouds") contract for aggregate QoS traffic profiles
  - Aggregates are policed at network-network boundaries (edge routers) ...
  - ... according to simple, bilateral business agreements
- Core routers apply few, simple per-hop fwd. behaviors (PHBs)
  - Indicated in packet header
  - Applied to PHB traffic aggregates (service classes)
- Policing rules + PHBs = range of services

# DiffServ Network Model



# DiffServ Domain



# Service Level Agreements (SLAs)

**Agreement customer/ISP — ISP about traffic and services:**

- **Economic and technical specifications (SLSs)**
- **Static vs. dynamic**
- **Quantitative vs. qualitative (relative priorities)**

## Service Level Specifications (SLSs):

- **Detailed technical specifications of QoS parameters**
- **Contain Traffic Conditioning Agreements/Specs. (TCAs/TCSs):**
  - **Detailed service parameters, e.g., throughput, max. delay, max. jitter, packet loss rate**
  - **Traffic profile, spec‘d e.g. by token bucket**
  - **Scope of service: ingress - egress routers**
  - **What happens to non-conforming packets?**
  - **Where is traffic marking and shaping being performed?**
  - **....**

# Example Service Levels

**Initial proposal, not standardized:**

- **Premium Service:** **low delay, low jitter**
- **Assured Service:** **better reliability than best-effort service**

**→ Initial two-bit DiffServ model (P-bit and A-bit)**

**In addition, only given as example:**

- **Olympic Service:** **three tiers with decreasing quality**
  - **Gold**
  - **Silver**
  - **Bronze**

**DiffServ defines only DS field and PHBs:**

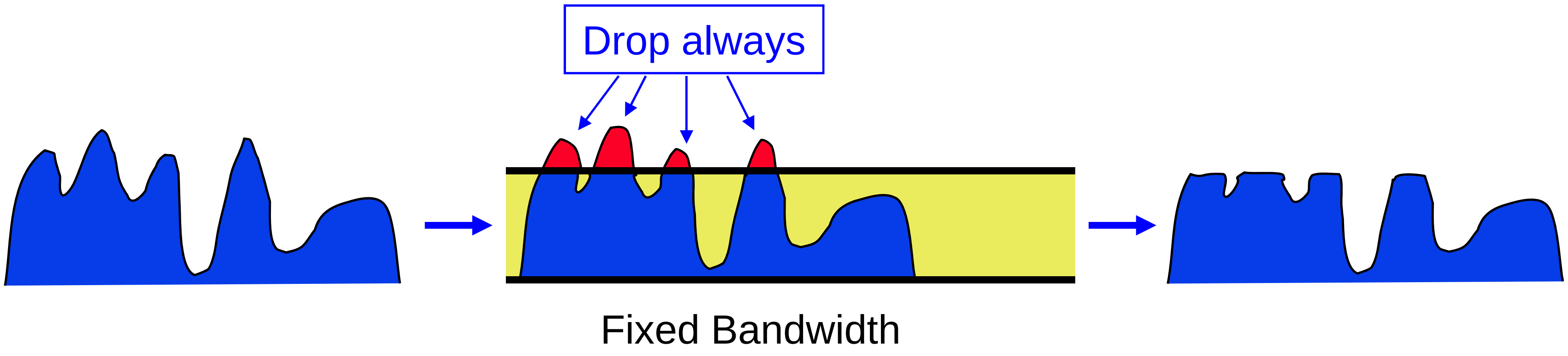
- **Expedited Forwarding PHB**
- **Assured Forwarding PHB**

**Support/mapping of these services levels to PHBs required**

# Premium Service

**Defines a virtual leased line:** **fixed maximum bandwidth, available when needed:**

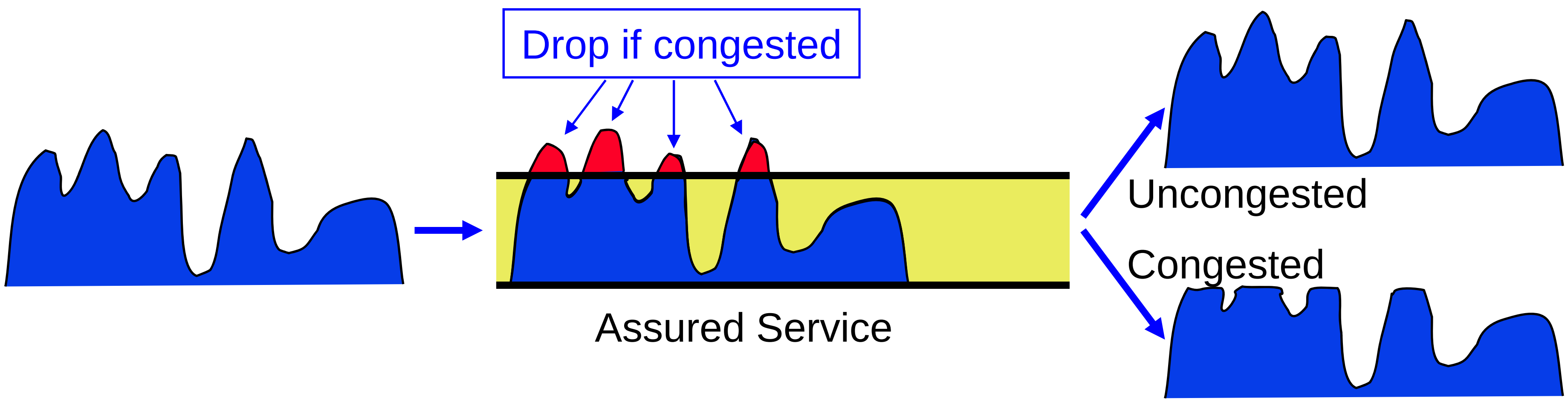
- **Conservative allocation of resources:**
  - **Provisioned according to peak capacity profiles**
- **Shaped at network boundaries to remove bursts**
- ***Out-of-profile* packets are dropped**
- **Qualitative**



# Assured Service

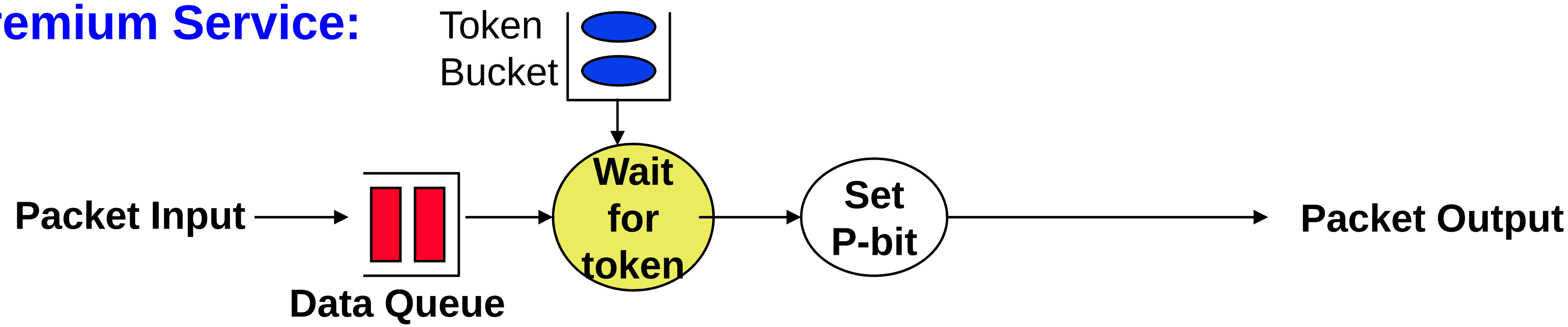
**Defines a “better” best-effort line:**

- **Resources statistically provisioned:**
  - **Provisioned according to expected capacity usage profiles**
- ***In-profile* traffic is “unlikely” to be dropped**
- ***Out-of-profile* packets get best-effort delivery**
- **Qualitative**

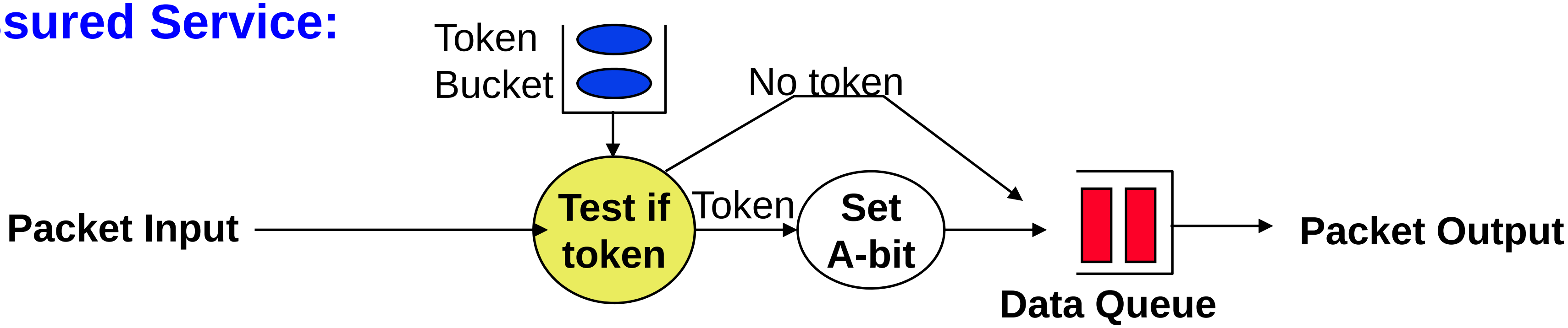


# Two-Bit DiffServ Border (Leaf) Router Functionality

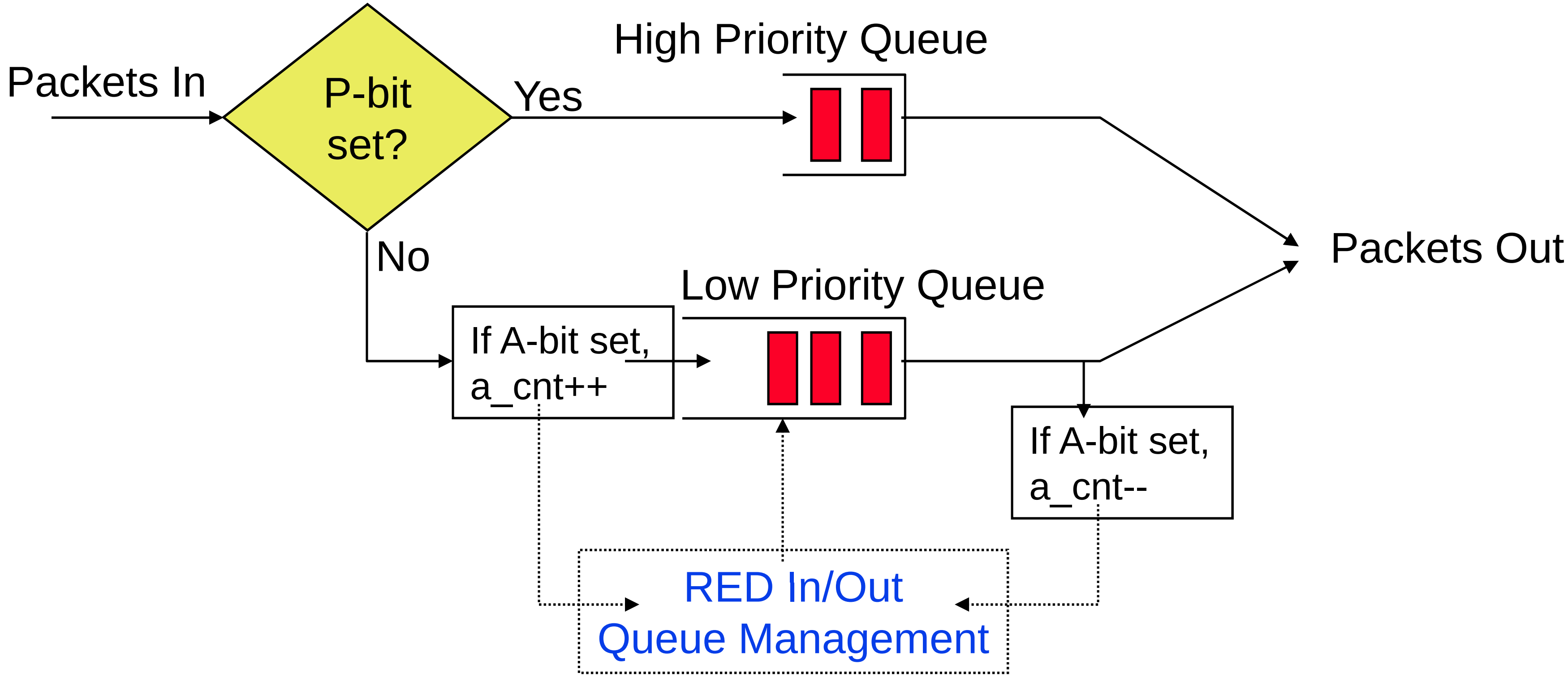
## Premium Service:



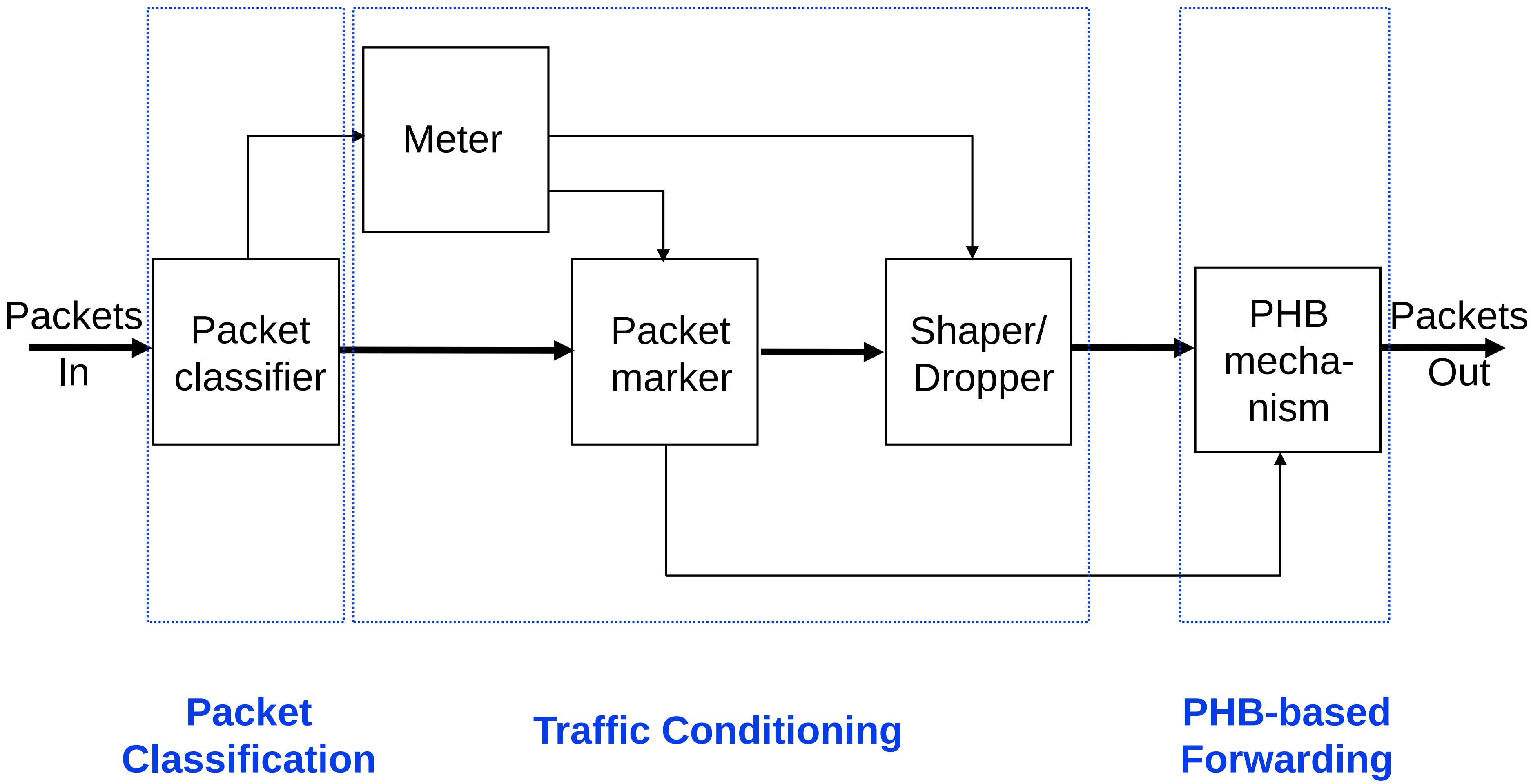

## Assured Service:



# Two-Bit DiffServ Interior Router Functionality



# Logical View of DiffServ-Capable Router




**Routers differ by functionality actually needed/provided !**

# Packet Classification

**Done in ingress routers, functionality based on position of router:**

- **Leaf router: multi-field (MF) classification**
  - **Process of classifying packets based on the content of multiple fields in the IP header, such as**
    - **Source and destination IP addresses**
    - **Source and destination port numbers**
    - **Protocol ID**
    - **TOS byte**
  - **Operates on individual flows, builds traffic aggregates**
- **Other ingress routers: behavior aggregate (BA) classification**
  - **Process of sorting packets based only on the contents of the DS field in the IP header**
  - **Operates on traffic aggregates**

# Traffic Conditioning

**Done in ingress routers, partially in egress routers; consisting of (a subset of):**

- **Metering:** **measurement of actual characteristics of a flow**
- **Marking:** **setting Differentiated Services field (DS field) in the IP header**

  **(Pre-marking: egress router sets DS field before packet leaves DS domain)**
- **Shaper:**
  - **Traffic shaping according to TCA**
  - **Packets buffered or dropped, if buffer full**
- **Dropper:** **special form of traffic shaper, without buffer**

# DS Field

## DS field in IP header:

- Information which service class (behavior aggregate) a packet belongs to
- Signaled by Differentiated Services Codepoint (DSCP)

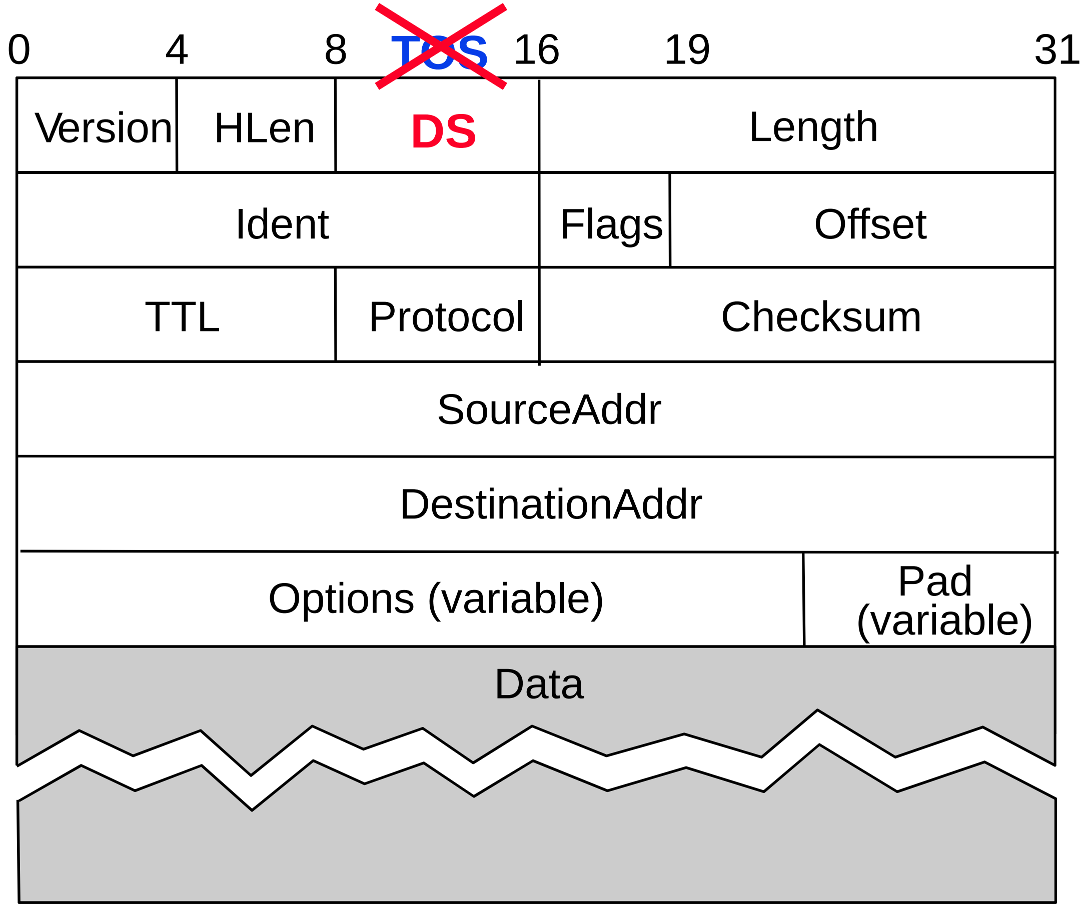
| DS Codepoint (6 bits) | Unused (2 bits) | |
|---|---|---|
| | | **DS field** |

## DSCP:

- Set by leaf / ingress routers
- Must be interpreted and used by DS-capable core / egress routers
- Is ignored by DS-unaware routers → best-effort service
- Predefined class selector codepoints (CSCs): xxx000, where $x \in \{0,1\}$
- Bits 0-2 define class, bits 3-5 relative priority within class
- Backward compatibility to IPv4 TOS field required

# DS Field in IP Header

**IPv4 header: TOS field → DS field**




**IPv6 header: Traffic class octet → DS field**

# Per-Hop Forwarding Behavior (PHB)

**Specifies a router‘s (hop‘s) externally observable behavior when forwarding (packets from) BAs**

**PHB based on DSCP; defined locally only!**

**Defined by:**

- **QoS parameters like delay, jitter, loss rate, ....**
- **Absolute or relative (to other PHB) values**
- **Rules how queues are shared among service classes**
- **....**

**Realized by:**

- **Queue management and packet scheduling techniques**
- **But not specified as their parameters**

**→ Allow for implementation flexibility**

# DSCP – PHB Mapping

**Simple table lookup, e.g.:**

| DSCP |
|---|
| 000000 |
| 001000 |
| 001111 |
| 111000 |
| 111001 |
| 111101 |

| PHB # | PHB Semantics |
|---|---|
| PHB 1 | Best Effort |
| PHB 2 | Premium Traffic |
| PHB 3 | Management |

**Some rules:**

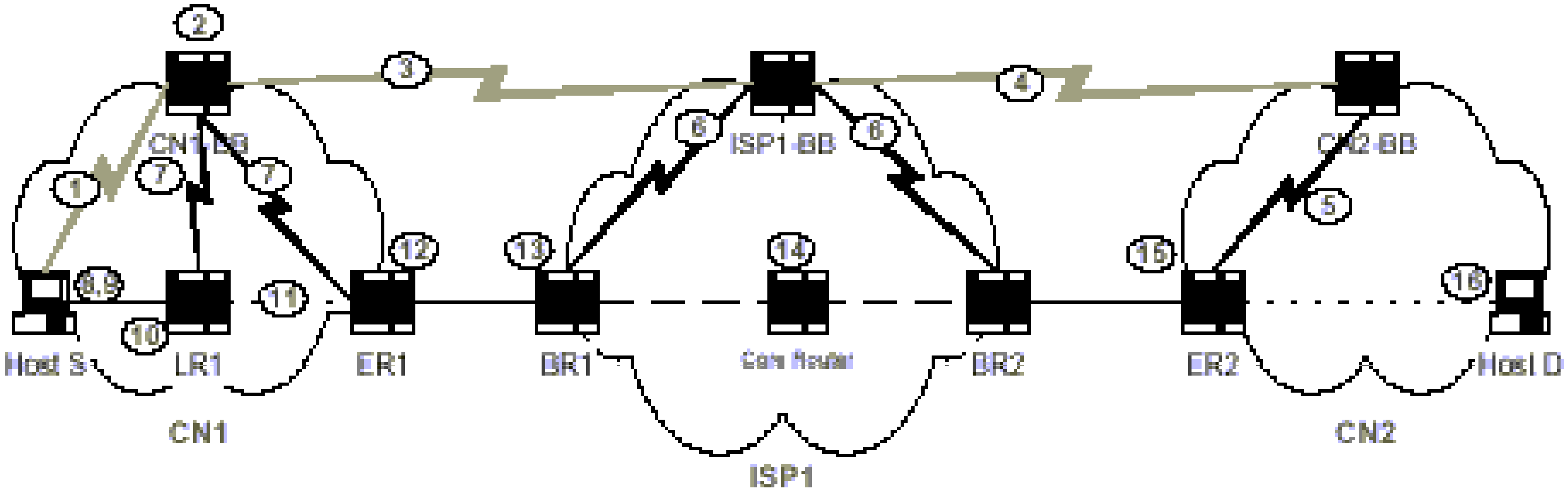
- **DSCP = 000000 → best-effort service class!**
- **≤ 8 service classes (CSCs) → ≥ 2 PHBs**
- **Numerically larger DSCP → better service**

**Caveat:**

**Packets from single source with different CSCs may arrive out of order, due to different PHBs encountered in routers!**

# End-to-End Service Architecture: Example

## Delivery of Premium Service with Dynamic SLA:




### Example scenario:

- Host S in Corporate Network 1 (CN1) wants to send data using Premium Service to Host D in CN 2.
- Host S has a dynamic SLA with ISP 1.

**The following shows potential behavior; nothing settled yet.**

# Phase 1: Signaling

**Step 1:** Host S sends an RSVP PATH Message to local Bandwidth Broker CN1-BB.

**Step 2:** CN1-BB makes an admission control decision.
If request is denied, an error message is sent back to S; signaling process ends.

**Step 3:** Request is accepted by CN1-BB. It sends the PATH Message to ISP1-BB.

**Step 4:** ISP1-BB makes an admission control decision.
- If request is denied, an error message is sent back to CN1-BB; S will be notified.
- If request is accepted, ISP1-BB sends the PATH Message to CN2-BB.

**Step 5:** CN2-BB makes an admission control decision.
- If request is denied, an error message is sent back to ISP1-BB; S will be notified.
- If request is accepted, CN2-BB will set classification and policing rules on router ER2 (using LDAP or RSVP). CN2-BB will then send RSVP RESV Msg. to ISP1-BB.

**Step 6:** When ISP1-BB receives the RESV Message, it will configure classification and policing rules on router BR1, and policing and reshaping rules on router BR2. It will then send the RESV Message to CN1-BB.

**Step 7:** When CN1-BB receives the RESV Message, it will set classification and policing rules on router LR1, and policing and reshaping rules on router ER1. CN1-BB will then send the RESV Message to S.

**Step 8:** When S receives the RESV Message, it can start transmitting data.

# Notes on Signaling Process

- Significant differences from IntServ/RSVP signaling process:
  - Sender requests for resources, not receiver.
  - Request can be rejected when a BB receives PATH Msg. from S. (In IntServ/RSVP, rejection only on RESV Message from receiver.)
  - A BB can aggregate multiple requests and make a single request to the next BB.
  - Each domain behaves like a single node, represented by its BB. ISP core routers are not involved in this process.
- State information installed by the BB on a BR is soft state. It must be regularly refreshed, or it will time out.
- Steps 4 and 6 repeated once for each intermediate ISP.
- If SLA between CN1 and ISP1 is static, Steps 3–6 are skipped.

# Phase 2: Data Transmission

**Step 9:** **Host S sends packets to leaf router LR1.**

**Step 10:** **Leaf router LR1 performs an MF classification.**
**If the traffic is non-conforming, LR1 will shape it.**
**LR1 will also set the P-bits of the packets. All packets enter the P-queue.**

**Step 11:** **Each intermediate router between LR1 and ER1 performs BA classification, puts the packets into the P-queue, and sends them out.**

**Step 12:** **ER1 performs a BA classification and reshapes the traffic to make sure that the negotiated peak rate is not exceeded. Reshaping is done for the aggregation of all flows heading toward BR1, not for each individual flow.**

**Step 13:** **BR1 classifies & policies the traffic. Excess premium packets are dropped.**

**Step 14:** **Intermediate routers between BR1 and BR2 (inclusive) perform BA classification. BR2 also reshapes the premium traffic.**

**Step 15:** **ER2 classifies & policies the traffic. Excess premium packets are dropped.**

**Step 16:** **The premium packets are delivered to host D.**

# DiffServ Advantages

- Limited number of service classes
- Aggregation of flows into traffic aggregates

  → Better scalability

- Thorough classification and policing in edge routers only

  → Easier implementation and deployment (than IntServ/RSVP)

- Leaf routers connected to slow links to customers

  → Room for per-flow classification (MF), policing, and shaping

- Core routers perform standard classification (BA) only,

  → Simple, fast packet forwarding

- Local rules and behavior only (PHBs; policies in DS domain)
- DS field ignored by DS-unaware routers (best-effort service)

  → Incremental deployment in the Internet

# Comparison of IntServ and DiffServ (1)

| Criteria | IntServ | DiffServ |
|---|---|---|
| Granularity of service differentiation | Individual flow | Aggregate of flows |
| State in routers (e.g., scheduling, buffer management) | Per flow | Per aggregate |
| Traffic classification basis | Several header fields | DS field |
| Type of service differentiation | Deterministic or statistical guarantees | Absolute or relative assurance |
| Admission control | Required | Required for absolute differentiation |
| Signaling protocol | Required (RSVP) | Not required for relative schemes |

# Comparison of IntServ and DiffServ (2)

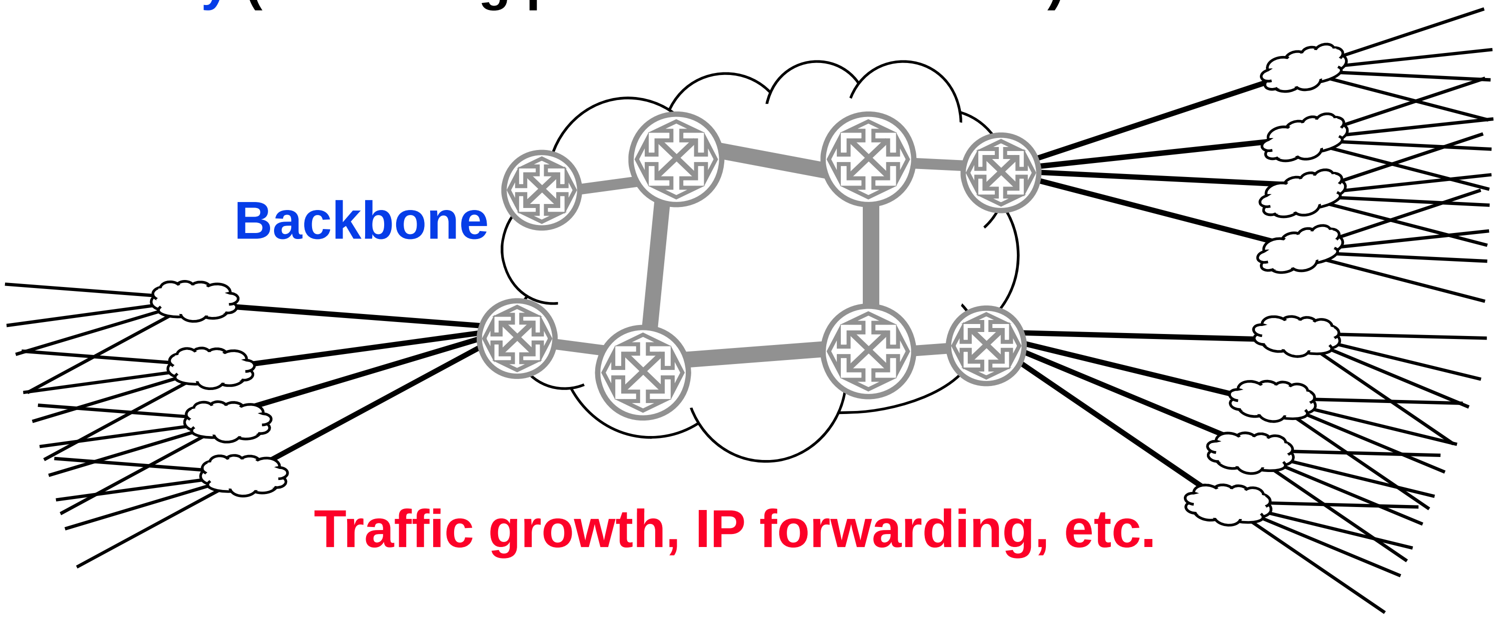
| Criteria | IntServ | DiffServ |
|---|---|---|
| Coordination for service differentiation | End-to-end | Local (per-hop) |
| Scope of service differentiation | A unicast or multicast path | Anywhere in a network or in specific paths |
| Scalability | Limited by the number of flows | Limited by the number of classes of service |
| Network accounting | Based on flow characteristics and QoS requirements | Based on class usage |
| Network management | Similar to circuit switching networks | Similar to existing IP networks |
| Inter-domain deployment | Multilateral agreements | Bilateral agreements |

# 4

# Multiprotocol Label Switching (MPLS)

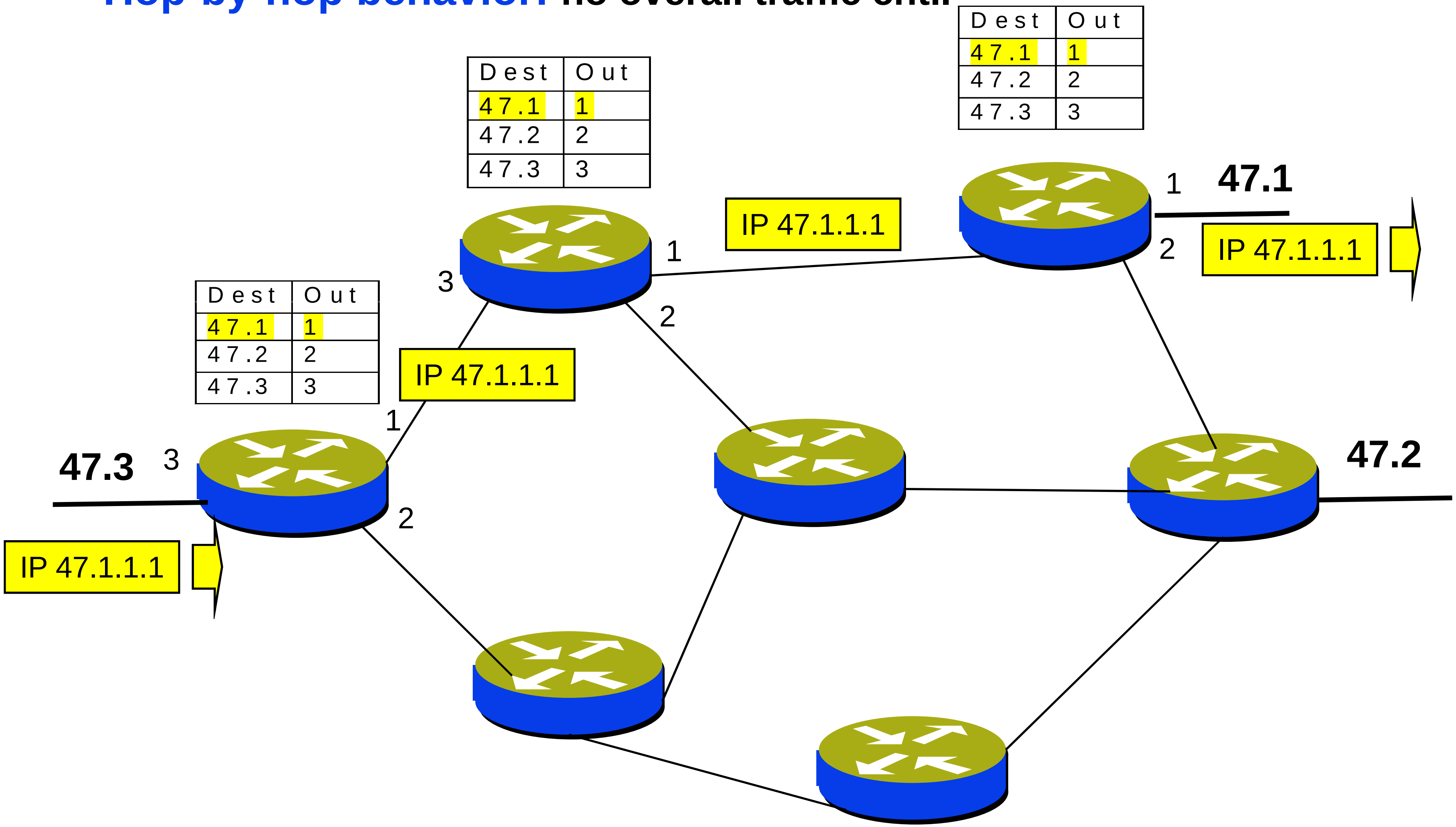
# Internet's Status Today

- **Traffic growth**
- **Complex, time-consuming IP datagram forwarding**
  - **Find IP address prefix matching with forwarding table entry**
- **Best-effort service model**
  - **No native QoS or multi-service capability in IP networks**
- **Scalability (of routing protocols and tables)?**



# Classical IP Datagram Forwarding

- **Forwarding: find longest matching prefix in fwd. table and fwd. packet**
- **Hop-by-hop behavior: no overall traffic cntl.**

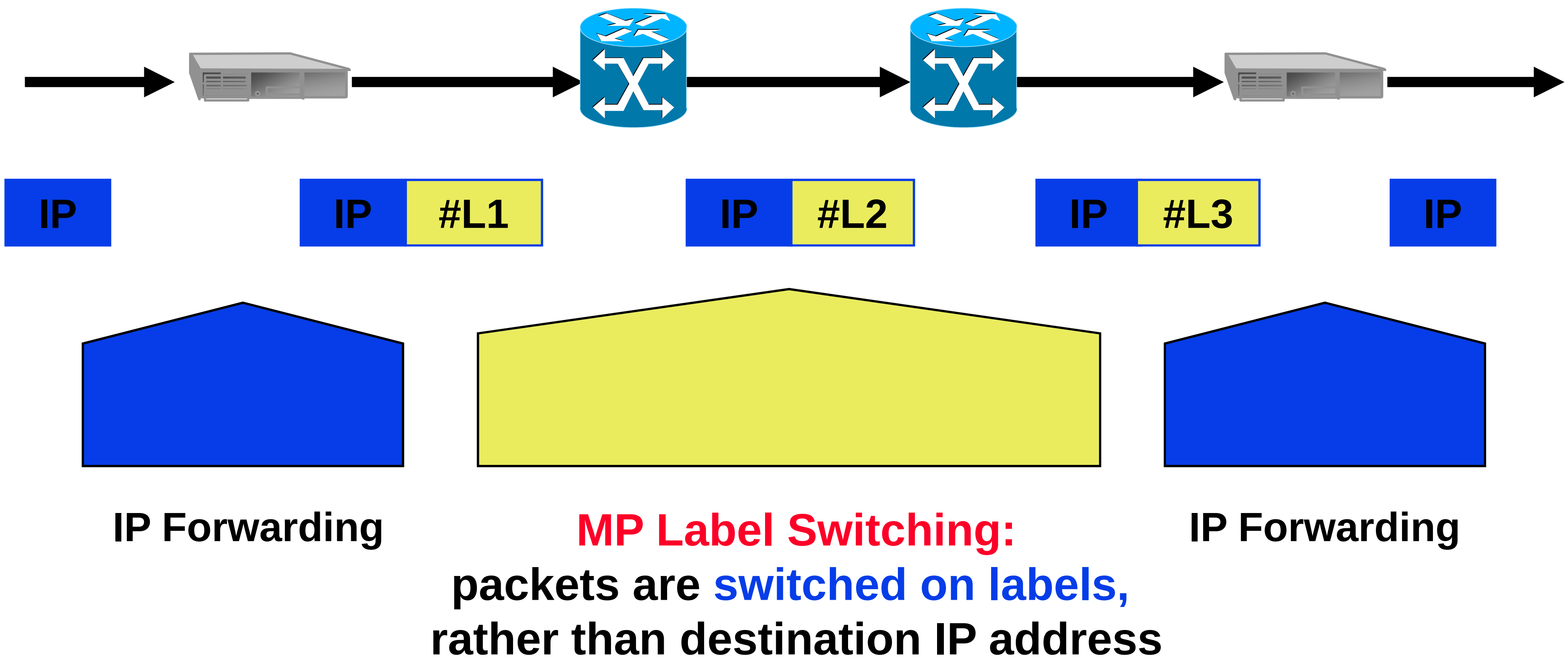


# Major Motivation and Goals behind MPLS

- Simplify and speed up IP forwarding significantly (10x ?)
- Support multiple service classes and QoS
- Support traffic engineering (TE)
  - TE := direct traffic to where the network capacity is
- Promote partitioning of functionality within the network
  - Detailed processing of packets at edge routers
  - Simple packet forwarding in core routers
- Thus, improve scalability of IP protocols (particularly, routing)
- Facilitate the integration of ATM switches and IP routers

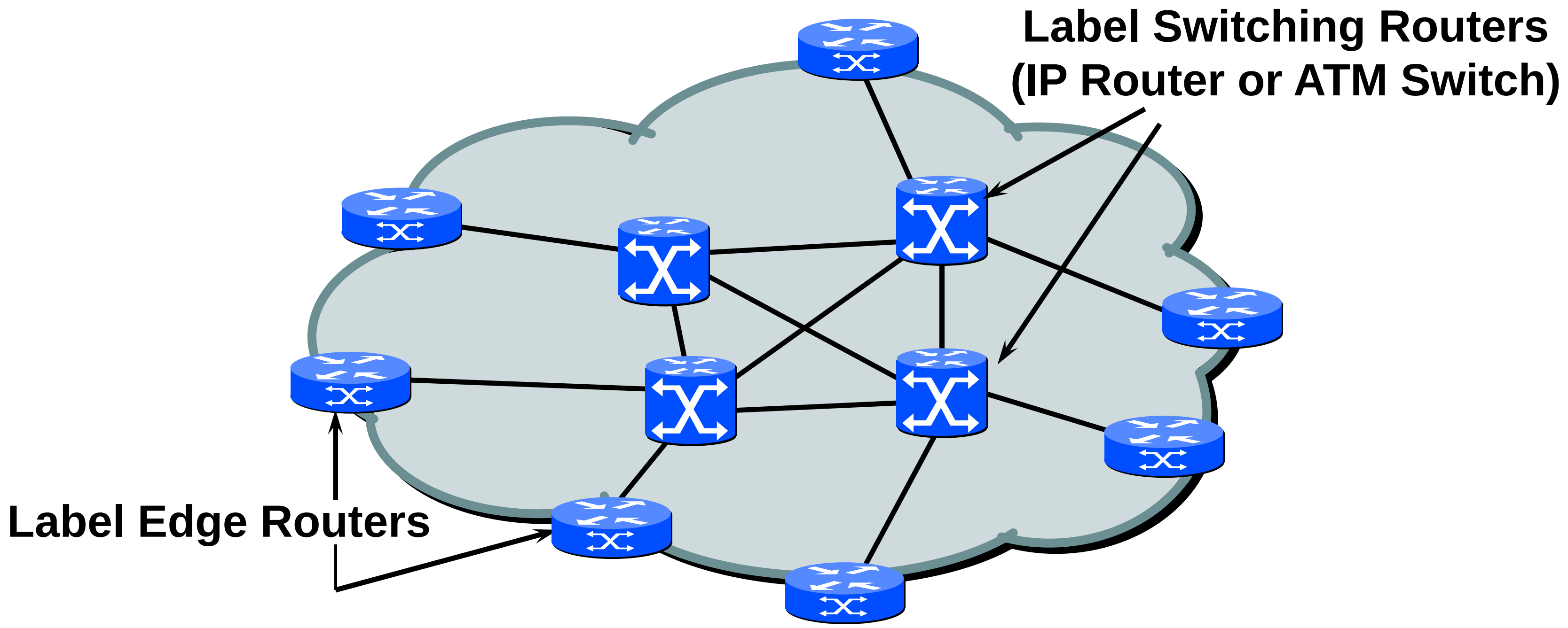
# Basic Idea: Route at Edge – Switch in Core

- **MPLS combines best of two worlds:**
  - **Flexible IP packet routing / forwarding**
  - **Fast ATM circuit switching**

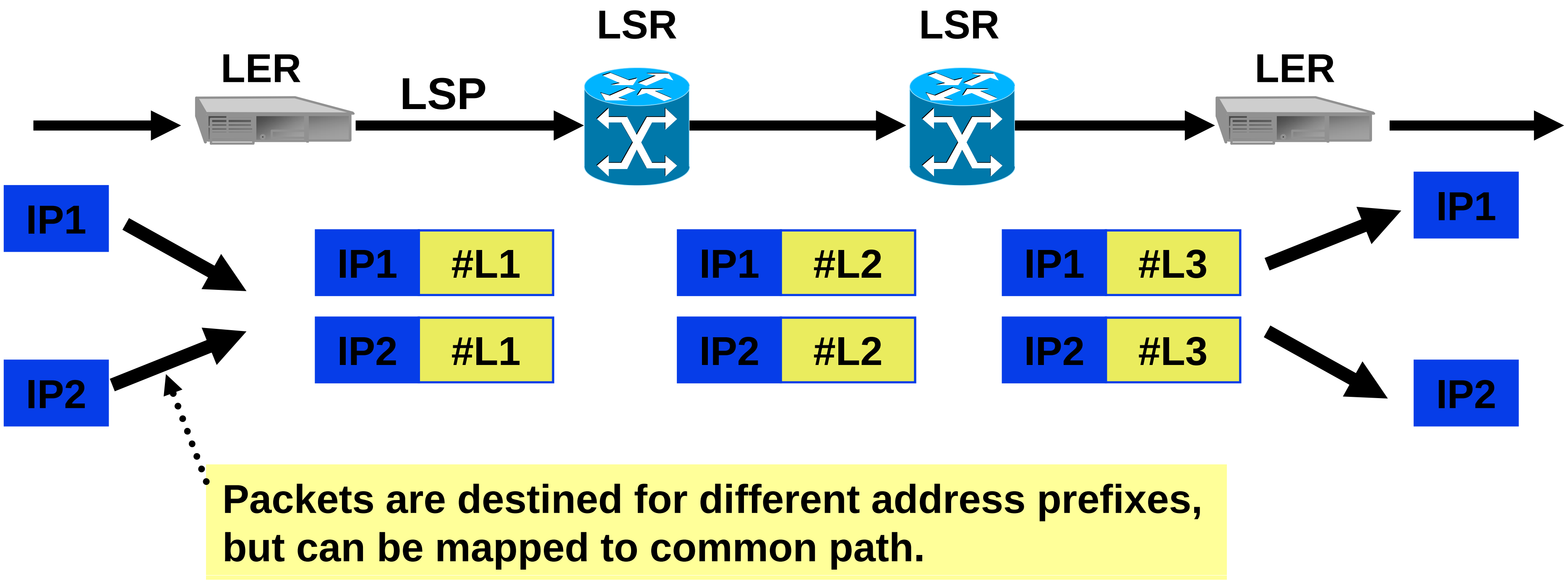


# Label Switching Devices

- **Label Edge Routers (LERs):** **LSRs at edge of MPLS network**
  - **Ingress LERs are responsible for classifying unlabelled IP packets and appending the appropriate label.**
  - **Egress LERs are responsible for removing the label and forwarding the unlabelled IP packet towards its destination.**
- **Label Switching Routers (LSRs):**
  - **Forward labelled packets based on the information carried by labels**



# Forwarding Equivalence Classes (FEC)




- FEC := subset of packets all treated the same way by a router
- Concept of FECs provides for flexibility and scalability
- In conventional routing, a packet is assigned to a FEC at each hop (i.e., L3 look-up), in MPLS it is only done once at the network ingress

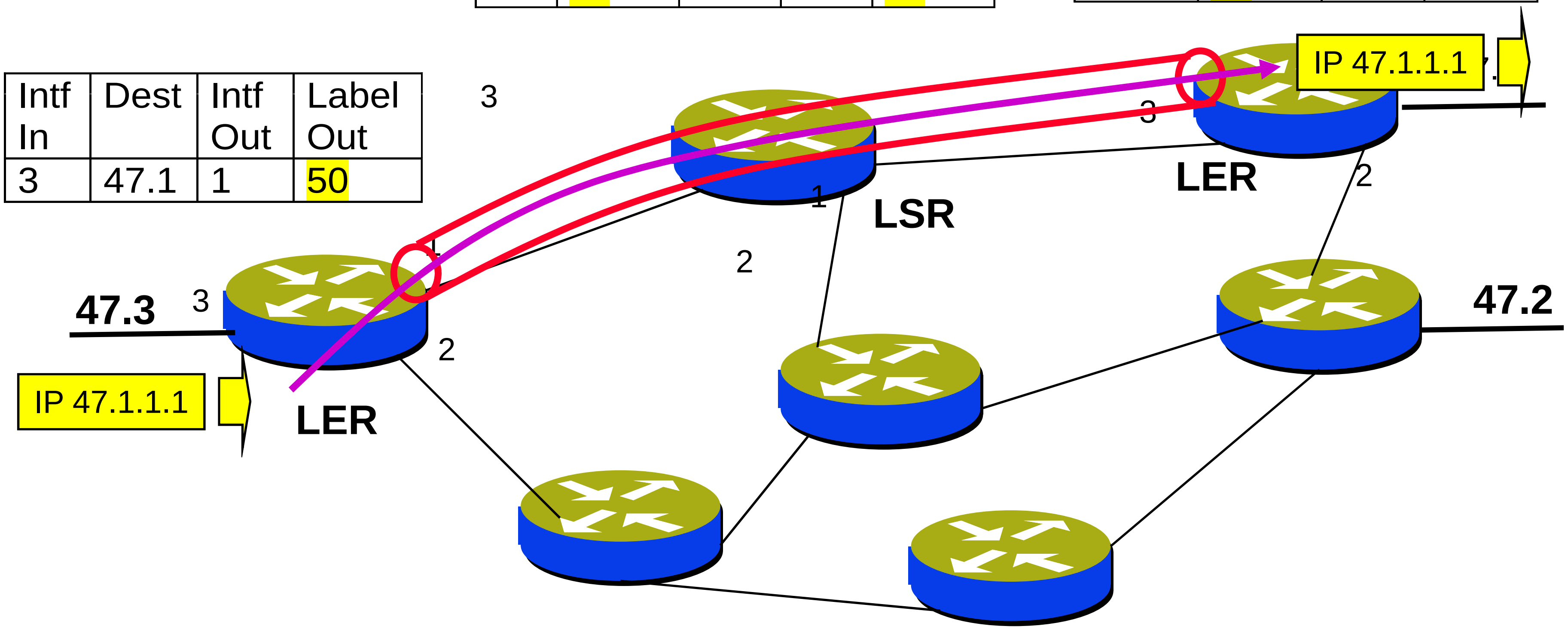
# Label Switched Path (LSP)

## Label Swapping (Substituion)

| Intf In | Dest | Intf Out | Label Out |
|---|---|---|---|
| 3 | 47.1 | 1 | 50 |

| Intf In | Label In | Dest | Intf Out | Label Out |
|---|---|---|---|---|
| 3 | 50 | 47.1 | 1 | 40 |

| Intf In | Label In | Dest | Intf Out |
|---|---|---|---|
| 3 | 40 | 47.1 | 1 |




**MPLS adds a connection-oriented paradigm to IP networks!**

# Components in Routers

- **Forwarding component:**
  - **Uses label information carried in a packet and label binding information maintained by a LSR to forward the packet**
- **Label Forwarding Information Base (LFIB):**
  - **Each entry consists of:**
    - **Incoming label**
    - **Outgoing label**
    - **Outgoing interface**
  - **LFIB is indexed by incoming label (→ fast forwarding, cf. ATM)**
  - **LFIB could be either per LSR or per interface**
- **Control component:**
  - **Responsible for maintaining correct label binding information among LSRs**

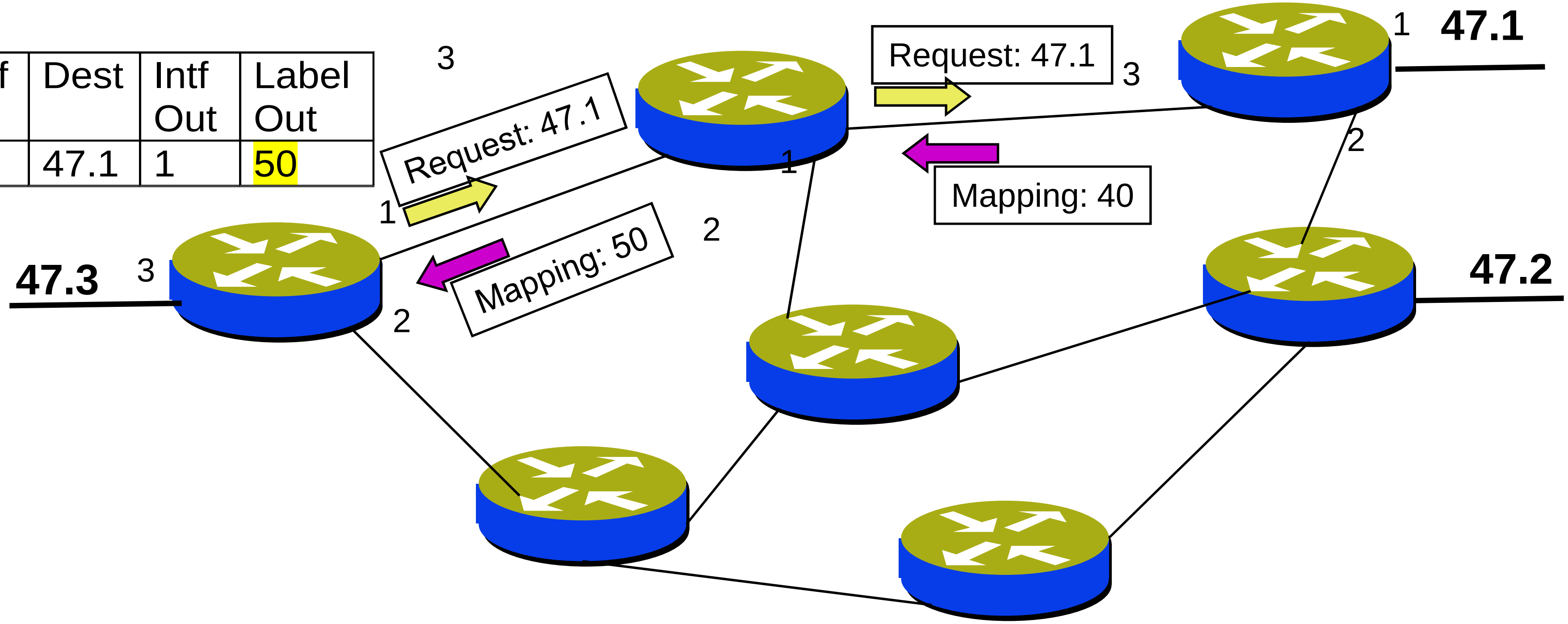
# Label Distribution

## Label Binding

| Intf In | Label In | Dest | Intf Out | Label Out |
|---|---|---|---|---|
| 3 | 50 | 47.1 | 1 | 40 |

| Intf In | Label In | Dest | Intf Out |
|---|---|---|---|
| 3 | 40 | 47.1 | 1 |

| Intf In | Dest | Intf Out | Label Out |
|---|---|---|---|
| 3 | 47.1 | 1 | 50 |



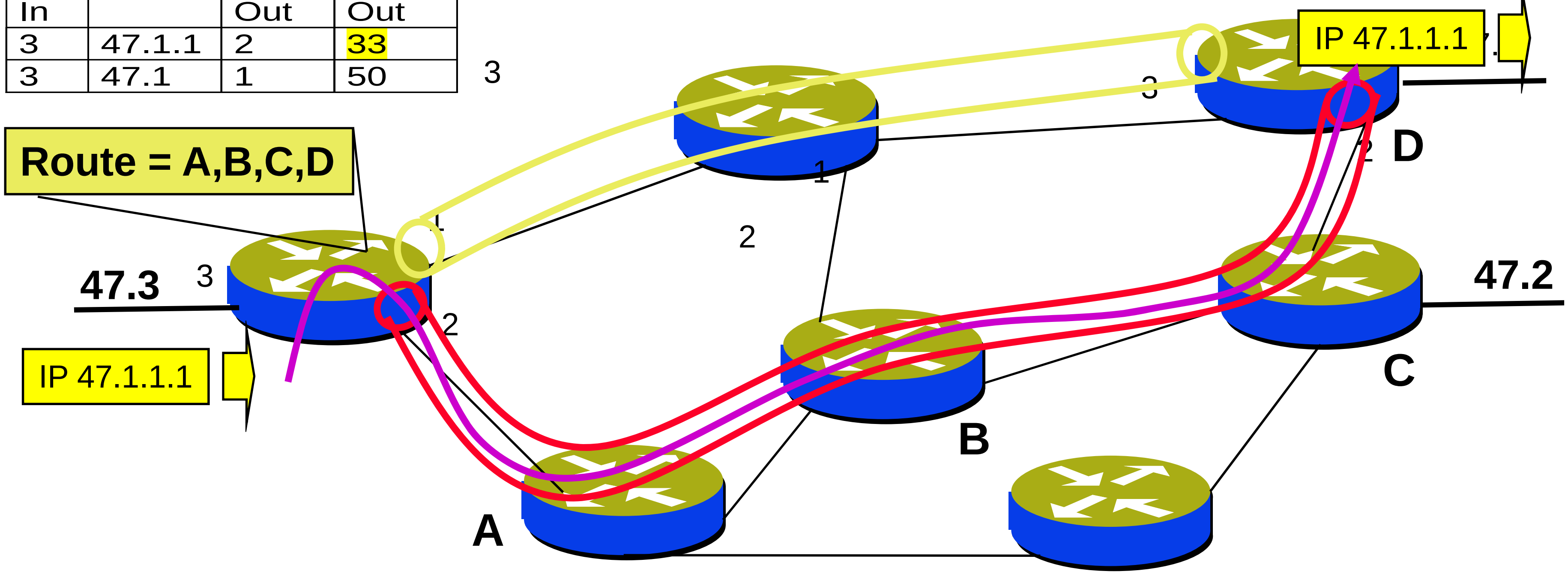

**Various** **Label Distribution Protocols (LDPs)** **under development/standard.**

# Example for LSP Setup: Explicitly Routed (ER) LSP

| Intf In | Label In | Dest | Intf Out | Label Out |
|---|---|---|---|---|
| 3 | 50 | 47.1 | 1 | 40 |

| Intf In | Label In | Dest | Intf Out |
|---|---|---|---|
| 3 | 40 | 47.1 | 1 |

| Intf In | Dest | Intf Out | Label Out |
|---|---|---|---|
| 3 | 47.1.1 | 2 | 33 |
| 3 | 47.1 | 1 | 50 |




**ER-LSP:** route chosen by source **(source routing),** via control messages:

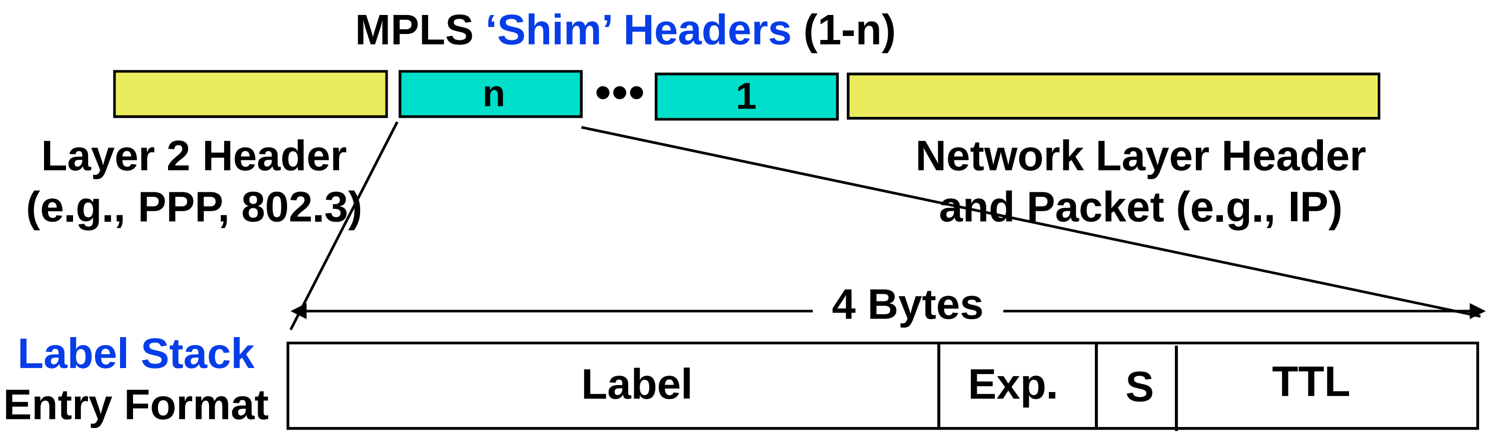
- **Can use routes other than shortest path**
- **Routing flexibility for network operator (policy-based, QoS-based)**

**Other routing protocols in use as well, e.g., constraint-based routing (CR)**

# MPLS Label Encapsulation

- **MPLS label encapsulation :=** **way to carry label information**
- **Defined over multiple link layers: Multiprotocol LS**
- **Specifications for the following link layers currently exist:**
  - **PPP/LAN: uses "shim" header inserted between L2 and L3 headers**
    **→ "Layer 2.5" technology**
  - **ATM: label contained in VCI/VPI field of ATM header**
  - **Frame Relay**
- **Translation between link layer types must be supported**

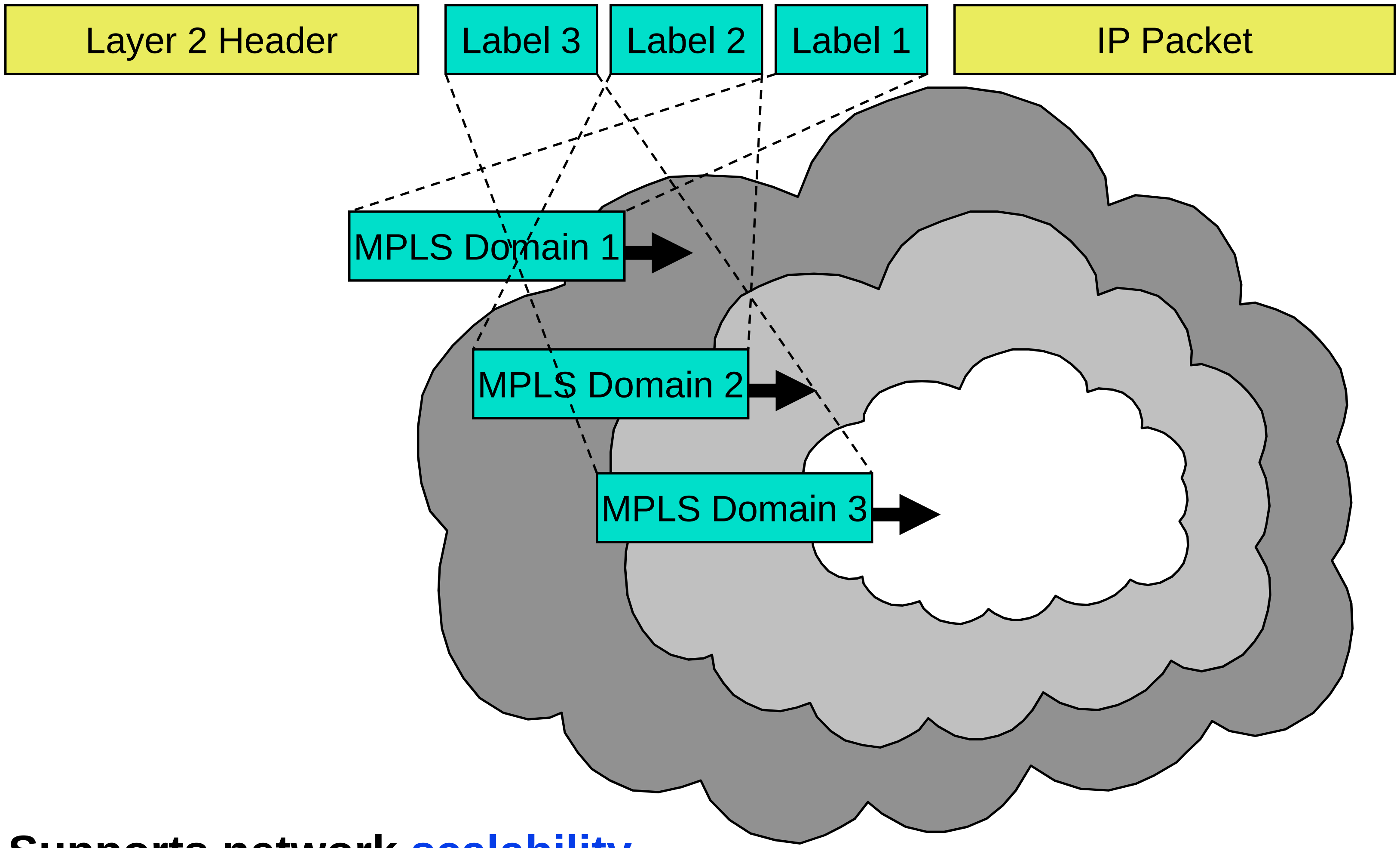
# MPLS Encapsulation – PPP Links & LANs




| | |
|---|---|
| Label: | Label Value, 20 bits (0-16 reserved) |
| Exp.: | Experimental, 3 bits (was Class of Service) |
| S: | Bottom of Stack, 1 bit (1 = last entry in label stack) |
| TTL: | Time to Live, 8 bits |

- **Multiple labels from multiple networks form label stack**
- **Network layer must be inferable from value of bottom label of the stack**
- **TTL must be set to the value of the IP TTL field when packet is first labelled**
- **When last label is popped off stack, MPLS TTL to be copied to IP TTL field**

# Hierarchy via Label Stack




**Supports network scalability**

# Summary: (Remaining) Key Elements of MPLS

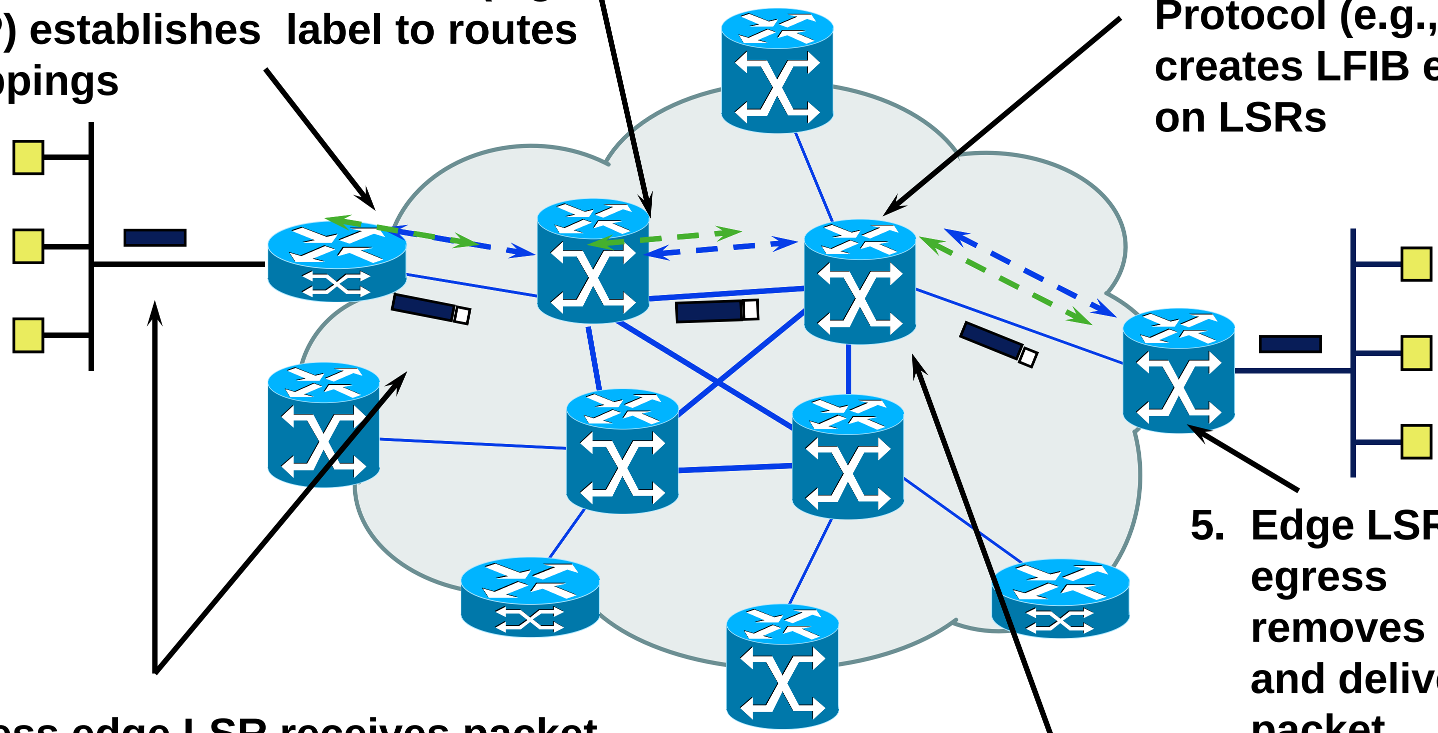
- **MPLS header stack**
  - **Contains the MPLS label on which LSRs forward the packets. Headers can be stacked.**
- **Differentiated behavior of routers**
  - **Packet labelling at edge, based on routing information**
  - **Packet forwarding in core, based on labels (much alike ATM cell switching)**
- **Enhanced IP routing protocols**
  - **Which distribute topology and constraint-based data**
- **Label distribution protocols**
  - **Standardized (or, yet to be) connection establishment protocols through which LSRs set up a complete path (LSP) from ingress LSR to egress LSR.**

# MPLS – The Big Picture

1. Existing routing protocols (e.g. OSPF, IGRP) establish routes.

2a. Label Distribution Protocol (e.g., LDP) establishes label to routes mappings

2b. Label Distribution Protocol (e.g., LDP) creates LFIB entries on LSRs



3. Ingress edge LSR receives packet, performs Layer 3 value-added services, and “label” packets

4. LSRs forward labelled packets using label swapping

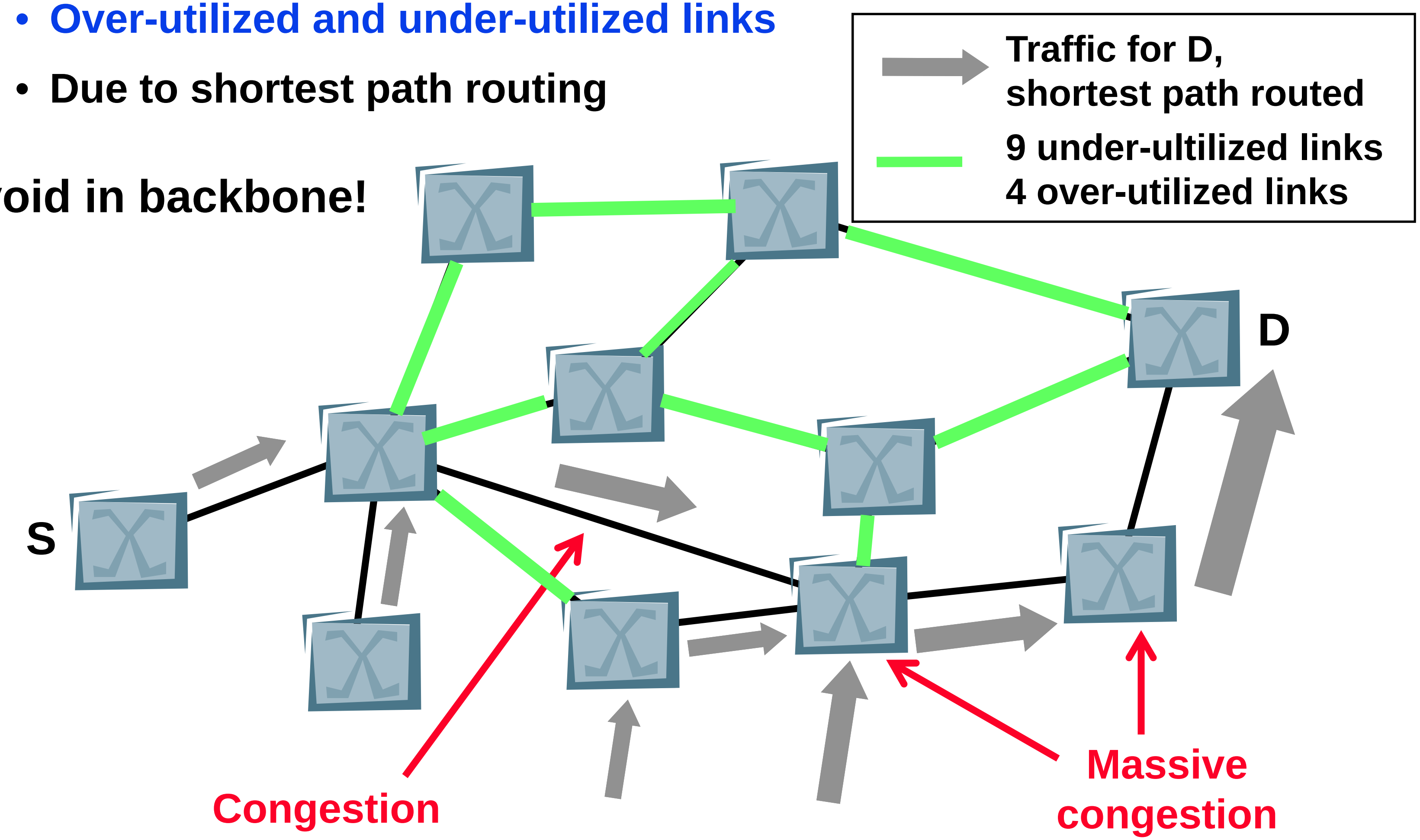
5. Edge LSR at egress removes label and delivers packet

# MPLS Traffic Engineering: Motivation

**Current interior gateway routing protocols (IGPs) can lead to unfavorable traffic distribution:**

- **Over-utilized and under-utilized links**
- **Due to shortest path routing**

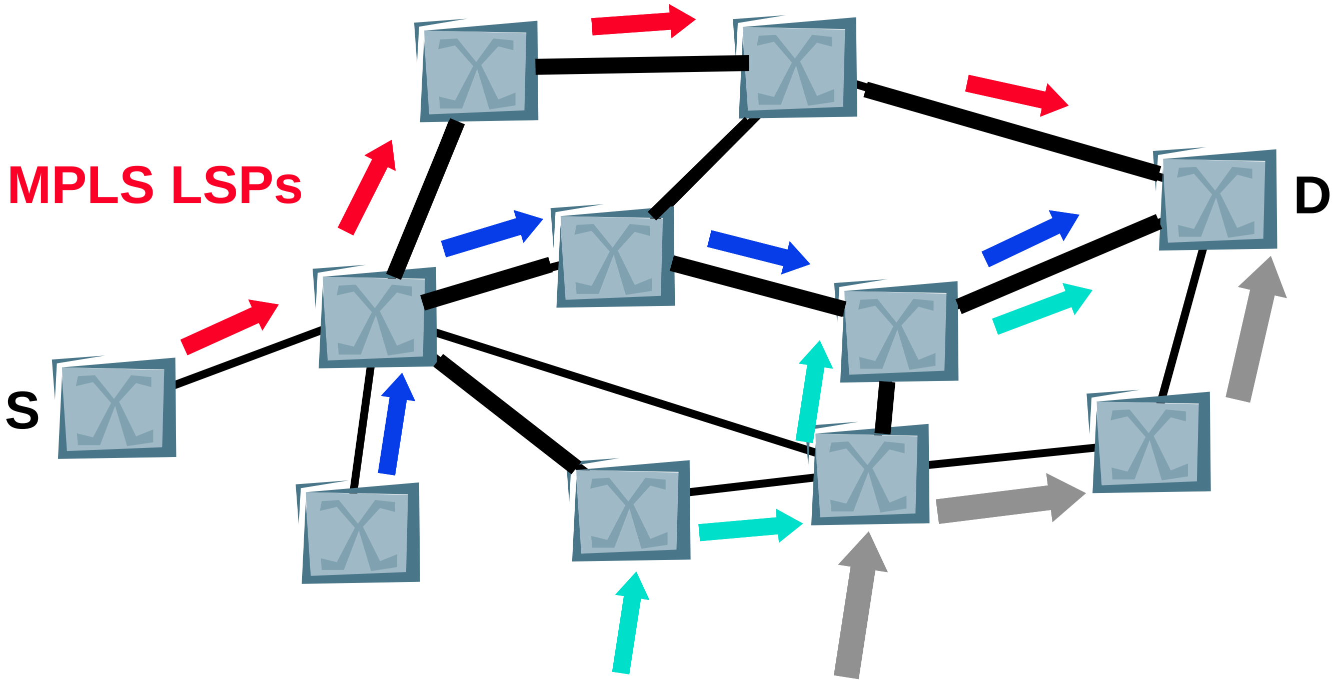
**Avoid in backbone!**



# MPLS Traffic Engineering: Objectives

- **Map/distribute actual traffic efficiently to available resources**
- **Use resources in a controlled way**
- **Redistribute traffic rapidly and effectively in response to changes in network topology (e.g., link or router failure)**

**Use of MPLS LSPs**




**This helps real-time traffic requiring QoS**

# MPLS Traffic Engineering: Some Methods

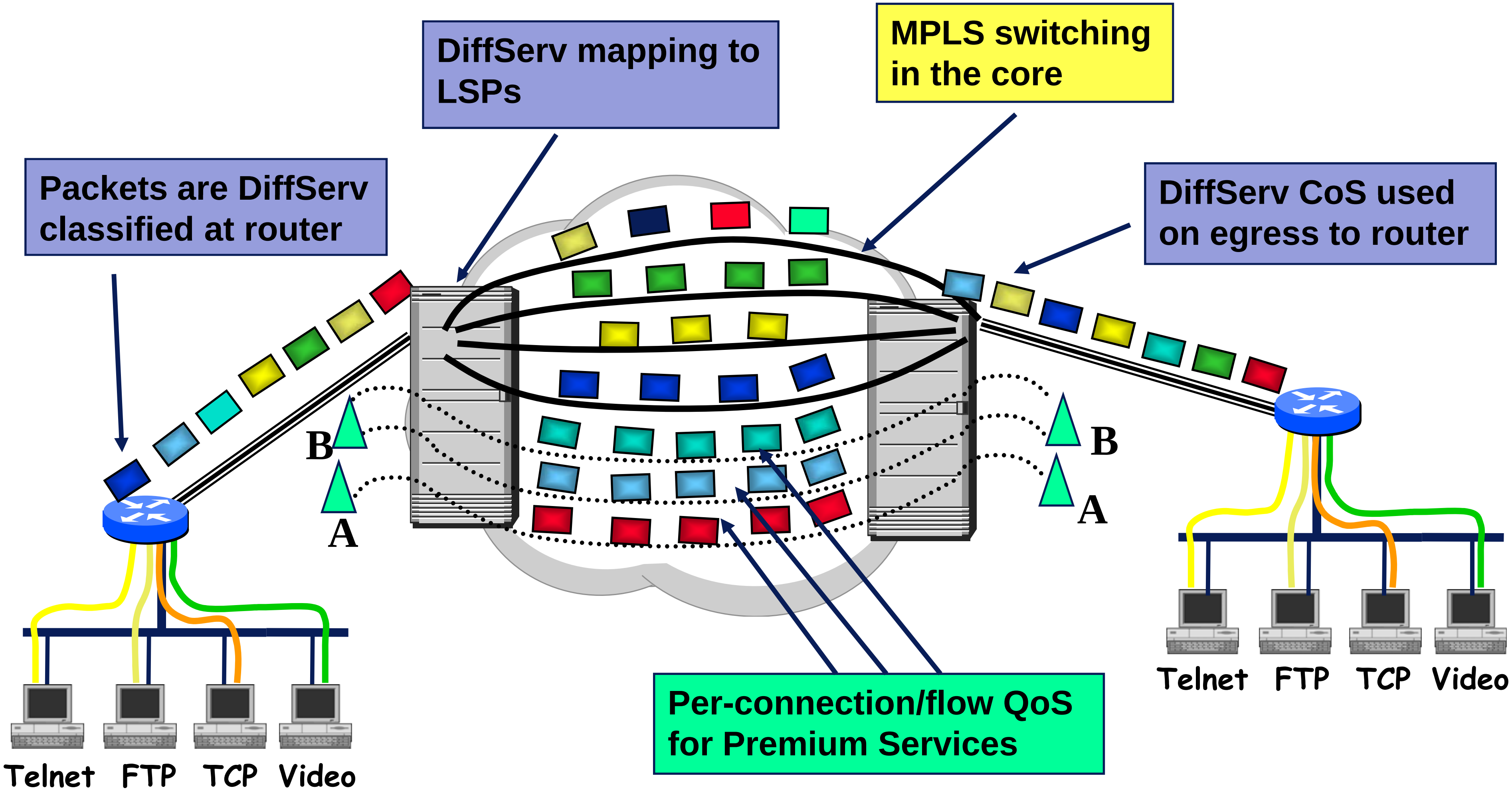
- **MPLS can use source routing capability to steer traffic on desired path**
- **Operator may manually configure these in LSRs along the desired path**
  - **Analogous to setting up PVCs in ATM switches**
- **Ingress LSR may be configured with the path, RSVP used to set up LSP**
  - **Some vendors have extended RSVP for MPLS path set-up**
- **Ingress LSR may be configured with the path, LDP used to set up LSP**
  - **Many vendors believe RSVP not suited**
- **Ingress LSR may be configured with one or more LSRs along desired path, hop-by-hop routing may be used to set up the rest of the path**
  - **A.k.a loose source routing, less configuration required**
- **If desired for control, route discov'd by hop-by-hop routing can be frozen**
- **In the future, constraint-based routing will offload traffic engineering tasks from the operator to the network itself**

# MPLS and DiffServ: Possible Combination

- **Both MPLS and DiffServ have same scalability goals/approach:**
  - **Aggregation of traffic on edge**
  - **Fast processing/forwarding of aggregates in core**
- **DiffServ can augment MPLS signaling and forwarding:**
  - **Signaling: augment LSP setup and distribution protocols (e.g., LDP, CR-LDP, RSVP-TE) with explicit CoS indication (e.g., EF, AF)**
  - **Forwarding: DiffServ Code Points (DSCP) can be mapped to the EXP (former CoS) bits in the MPLS label, to indicate packets' priorities**
- **PHB enforcement in MPLS DiffServ:**
  - **Exact same PHB mechanisms as in IP DiffServ**

    **DiffServ queues with DiffServ drop profiles; EF, AF *i*, default PHBs**
  - **Only difference is packet classification**
    - **For IP DiffServ, packets classified by DSCP**
    - **For MPLS DiffServ, packets classified by label / EXP bits**
- **MPLS DiffServ can be thought of undistinguishable from IP DiffServ**

# Example: MPLS End-to-End CoS and QoS



# 5

# Real-Time Multimedia Data Transport

# Basic Protocols

# Requirements (1)

- **Interoperability between different applications**
  - **E.g., two different audio conference systems**
- **Negotiation about coding issues**
  - **Agree on media type, compression method, etc.**
- **Timing for proper playback (in a single stream)**
- **Synchronization (among multiple streams)**
  - **E.g., between video and corresponding audio**
- **Indication of packet loss, thus also congestion**
  - **RTP runs typically over non-reliable transport (UDP)**
  - **This enables applications to do something, e.g., adapt to congestion situation**

# Requirements (2)

- **Framing**
  - **Enable applications to mark start and end of frames**
  - **E.g., mark the beginning of a "talkspurt" — the application may shorten or lengthen silences**
- **Sender identification**
  - **IP address is not extremely user-friendly**
- **Efficiency**
  - **No long headers are acceptable**
  - **E.g., audio data packets are typically short, a great overhead is undesirable**

# Real-Time Transport Protocol (RTP) [RFC 3550] (1)

- **Origins in the *vat* audio conferencing tool's application protocol**
- **RTP is a "transport" protocol on application level, running over the usual transport protocols, typically UDP**
- **Protocol stack for multimedia application using RTP:**

| |
|---|
| Application |
| RTP (Real-time Transport Protocol) |
| UDP (User Datagram Protocol) |
| IP (Internet Protocol) |
| Network |

# RTP (2)

- **Twin-standard by IETF (with RTCP)**
  - **RTP: exchange of multimedia data**
  - **RTCP: exchange of periodic control information**
  - **They use consecutive even–odd port numbers**
- **Application Level Framing (ALF)**
  - **Applications understand their needs best**
  - **Profile**
    - **Defines the meaning of certain fields in the header**
  - **Payload formats**
    - **Interpretation of data following the header,
e.g., as simple stream of bytes or some more complex structure (MPEG)**

# RTP Header Format (1)

Number of bits

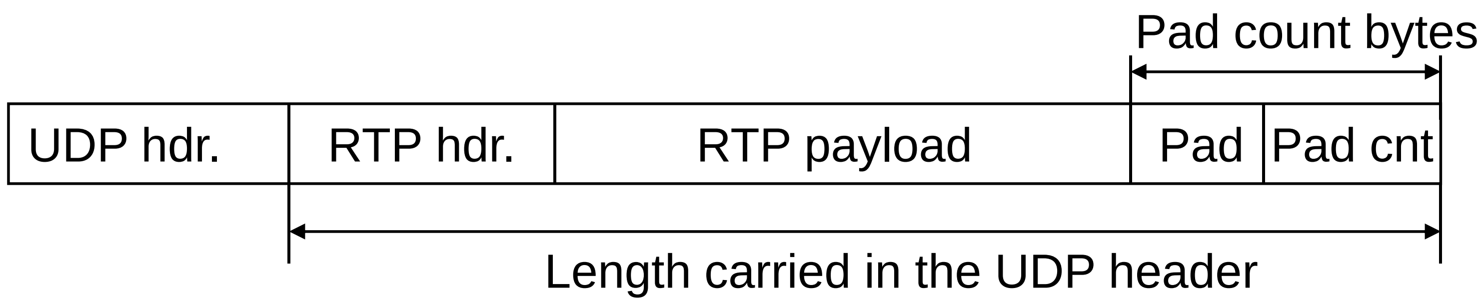
| $V_2$ | $P_1$ | $X_1$ | $CC_4$ | $M_1$ | $PT_7$ | Sequence $\text{number}_{16}$ |
|---|---|---|---|---|---|---|
| $\text{Timestamp}_{32}$ | | | | | | |
| Synchronization source (SSRC) identifier | | | | | | |
| Contributing source (CSRC) identifier<br>… | | | | | | |
| Extension header | | | | | | |

- **V:** **version no. – 2 bits might be few, later extensions possible (subversion)**
- **P:** **Padding is used**
- **X:** **Extension header exists**
- **CC:** **Contributing sources – needed only if the RTP streams are “mixed”**
- **M:** **Marks the packet, e.g., as start of frame – the app. uses it at wish**
- **PT:** **Payload type**

# RTP Header Format (2)

- **Payload type**
  - **Generally not used as demultiplexing key**
  - **Transport mechanisms are used for multiplexing, e.g., different UDP ports for each stream**
- **Sequence number**
  - **The sender just increments it – handling by application**
  - **E.g., replay the last frame for a lost video frame**
- **Time stamp**
  - **Tick is defined by the application, e.g., 125 μs for audio**
- **Synchronization source (SSRC)**
  - **Random number uniquely identifying single sources**

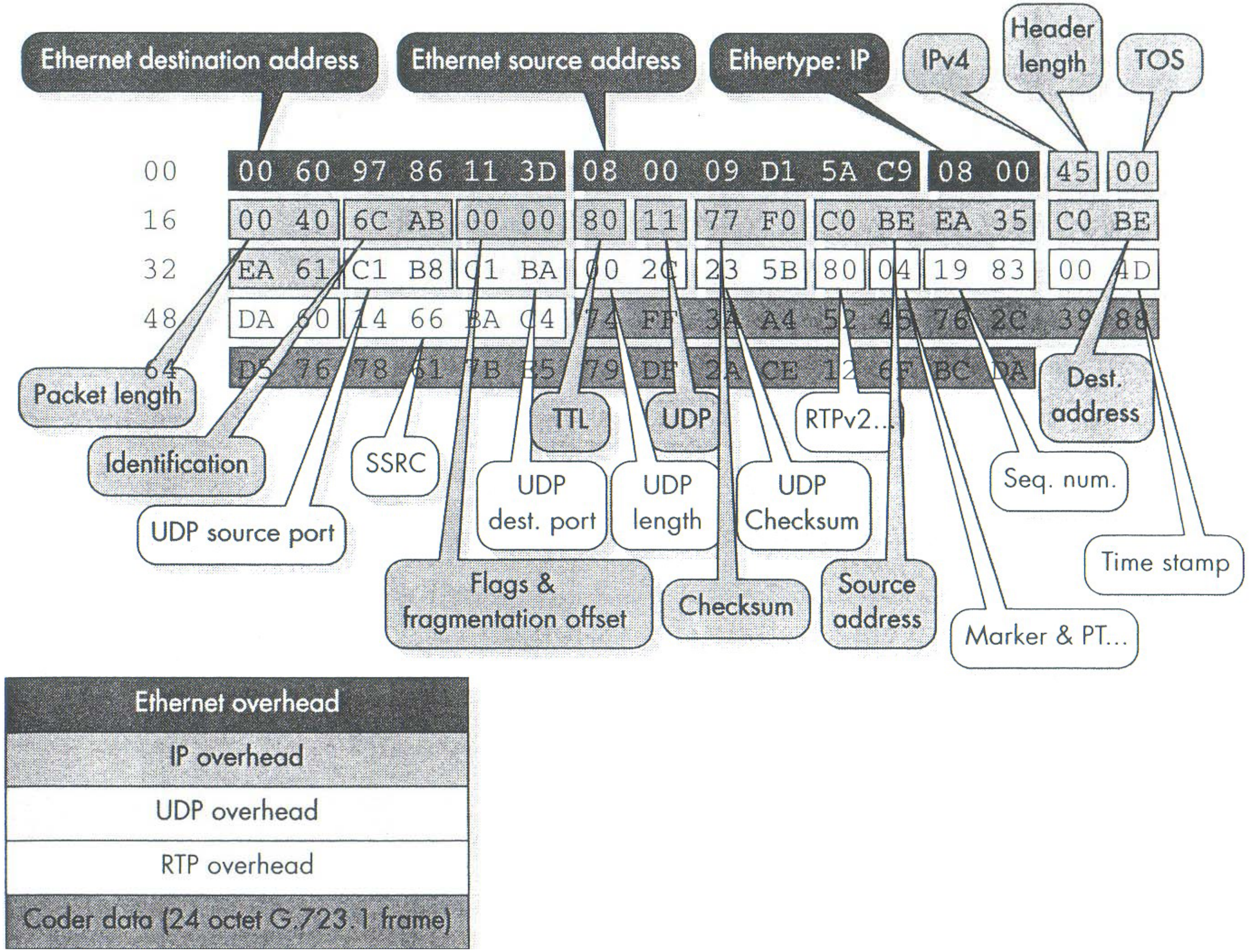
# RTP Packet Format




- **The length of the RTP data is coded in the UDP header**
- **If the RTP payload is less than this length, then padding is used**
- **Advantage**
  - **The RTP header remains short: 1 bit suffices**
  - **The Pad count byte is used only if this place is unused by the payload anyway**

# RTP Packet Example

**Example: single frame (24 bytes) of G.723.1 encoded voice**



# Real-Time Control Protocol (RTCP) [RFC 3550]

**RTCP provides a control stream associated with the data stream, with the functions:**

- **Performance feedback**
  - **E.g., for adaptive applications**
- **Correlate and synchronize media streams of the same sender**
  - **The same sender may have several SSRC values**
  - **Canonical name (CNAME) assigned to a sender**
  - **Different clocks must be synchronized**
- **Convey the identity of the sender**

# RTCP Packet Types

- **Sender and receiver reports**
  - **SSRC (synchronization source) identifier**
  - **Statistics of lost data packets from this source**
  - **Highest sequence number from this source**
  - **Estimated interarrival jitter for this source**
  - **Last actual timestamp received via RTP for this sender**
  - **Delay since last sender report received via RTP for this sender**
- **Extra sender information, enabling synchronization**
  - **Timestamp of the actual date time of report generation**
  - **RTP timestamp of report generation**
  - **Cumulative packet and byte counts of this sender**
- **Source descriptions**
  - **CNAME and other sender description information**
- **Application-specific control packets**

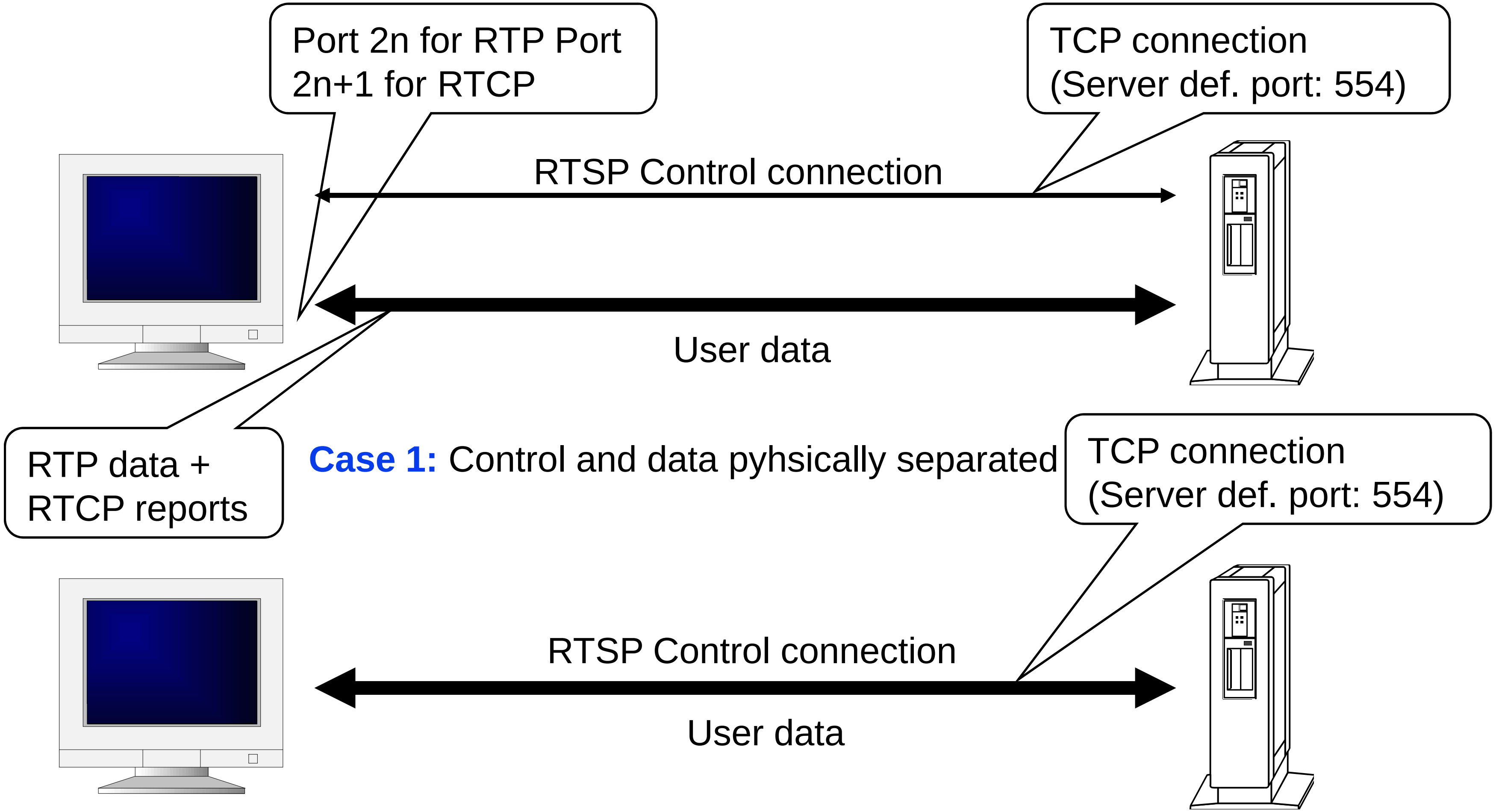
# RTCP Operation

- **RTCP traffic is limited to ~ 5% of RTP traffic**
  - **The report generation slows down if necessary**
- **Recipients and sender may react on the reports**
  - **Recipient may require resource reservation noticing that other recipients have better QoS**
  - **Sender may reduce rate if too many packets are lost**
  - **RTP timestamp + time of day enable synchronization of streams, even with different clock granularity**
  - **CNAME enables the identification of the media stream with different SSRC values**

# Real-Time Streaming Protocol (RTSP) [RFC 2326]

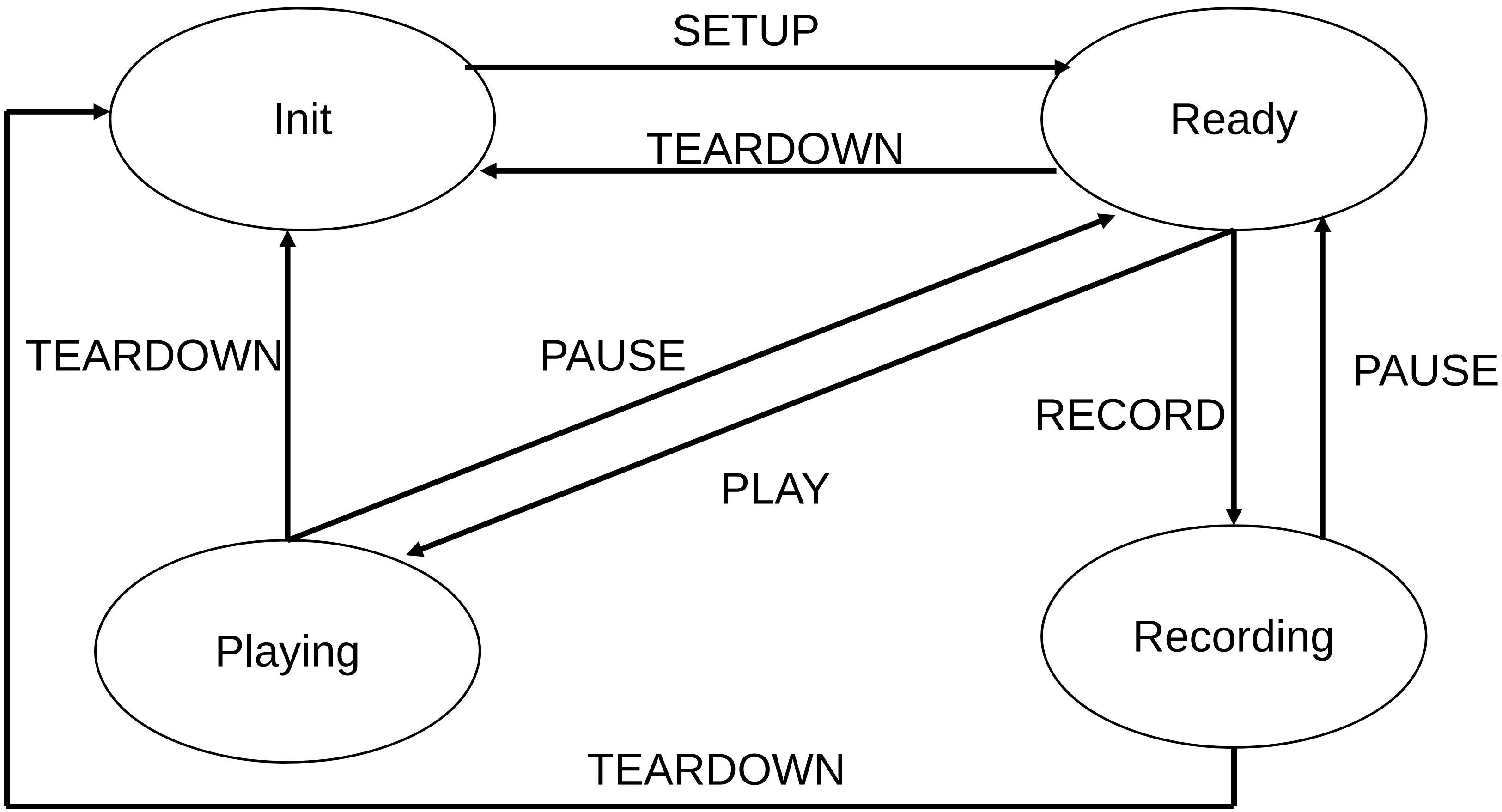
- **User-level protocol**
- **No assumption about the transport level**
- **No explicit QoS mechanisms**
- **Delivers only control data, no payload**
- **Syntax similar to HTTP, but the server does have state**
- **Extensible by new methods and/or parameters**
- **Supported operations**
  - **Download of media data from a media server**
  - **Invitation of a media server into a conference**
  - **Adding of media to an existing presentation**

# Control and User Data

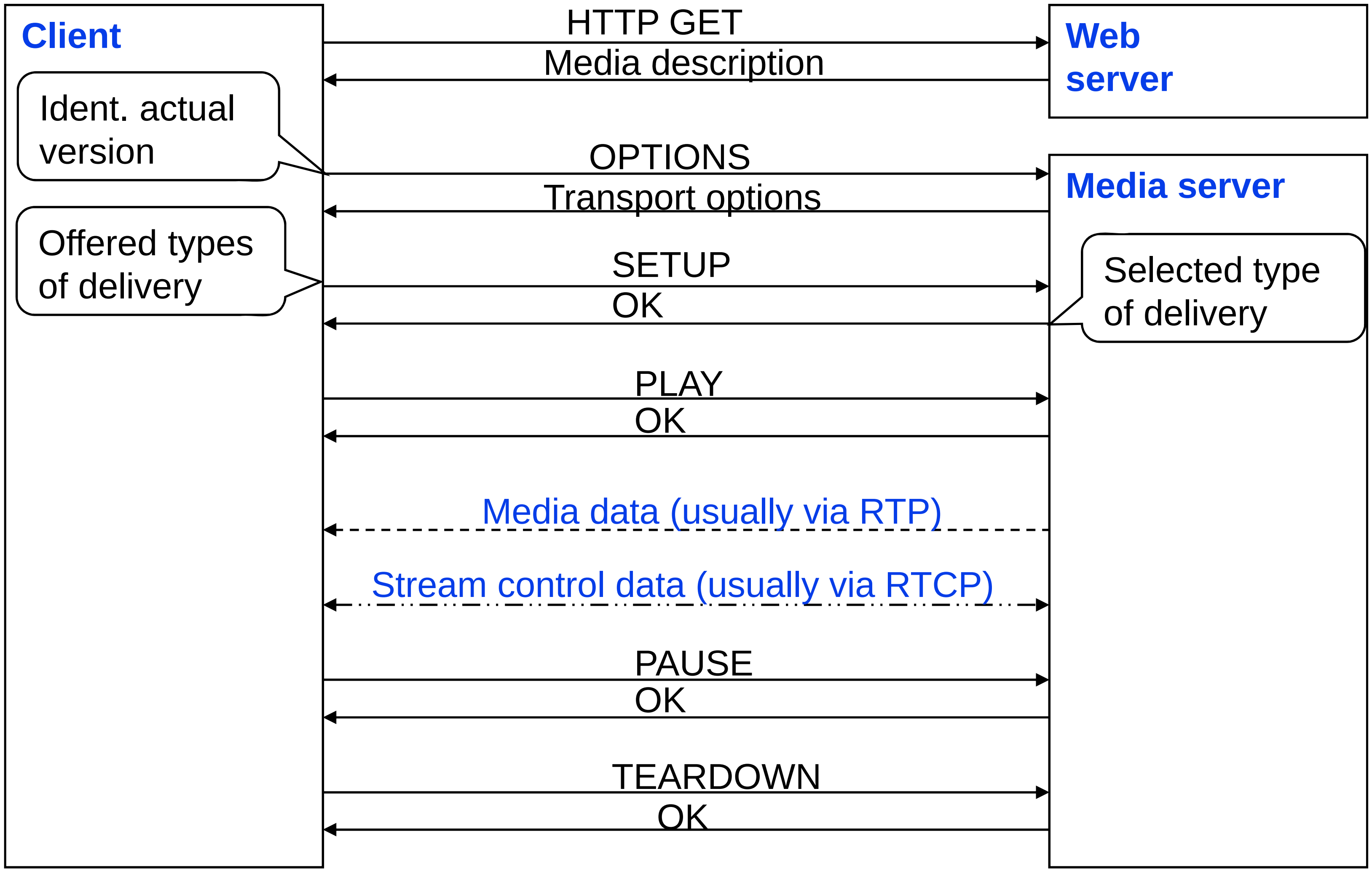



**Case 2:** Control and data only logically separated

# RTSP State Diagram



# RTSP Session Protocol



# RTSP Methods

| Method | Direction | Object | Availability |
|---|---|---|---|
| DESCRIBE | $C \rightarrow S$ | P, S | suggested |
| ANNOUNCE | $C \leftrightarrow S$ | P, S | optional |
| GET_PARAMETER | $C \leftrightarrow S$ | P, S | optional |
| OPTIONS | $C \leftrightarrow S$ | P, S | mandatory<br>($S \rightarrow C$: optional) |
| PAUSE | $C \rightarrow S$ | P, S | suggested |
| PLAY | $C \rightarrow S$ | P, S | mandatory |
| RECORD | $C \rightarrow S$ | P, S | optional |
| REDIRECT | $S \rightarrow C$ | P, S | optional |
| SETUP | $C \rightarrow S$ | S | mandatory |
| SET_PARAMETER | $C \leftrightarrow S$ | P, S | optional |
| TEARDOWN | $C \rightarrow S$ | P, S | mandatory |

# RTSP: OPTIONS and SETUP

```
C -> S:     OPTIONS * RTSP/1.0
            Cseq: 1
            Require: implicit-play
            Proxy-Require: gzipped-messages

S -> C:     RTSP/1.0 200 OK
            Cseq: 1
            Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

---

```
C -> S:     SETUP rtsp://server.com/media RTSP/1.0
            Cseq: 302
            Transport: RTP/AVP;unicast;client_port=4588-4589

S -> C:     RTSP/1.0 200 OK
            Cseq: 302
            Date: 11 Jan 2001 10:30:06 GMT
            Session: 447745
            Transport: RTP/AVP;unicast;client_port=4588-4589;
            server_port=6256-6257
```

# RTSP: PLAY and PAUSE

```
C -> S:      PLAY rtsp://server.com/media RTSP/1.0
             Cseq: 825
             Session: 447745
             Range: npt=10-15 (normal play time, 10.-15. sec)
S -> C:      RTSP/1.0 200 OK
             Cseq: 825
             Date: 11 Jan 2001 10:30:06 GMT
```

---

```
C -> S:      PAUSE rtsp://server.com/media RTSP/1.0
             Cseq: 834
             Session: 447745
S -> C:      RTSP/1.0 200 OK
             Cseq: 834
             Date: 11 Jan 2001 10:30:06 GMT
```

# RTSP and TCP Interleaved

```
C -> S:        SETUP rtsp://server.com/media RTSP/1.0
               Cseq: 2
               Transport: RTP/AVP/TCP;interleaved=0-1

S -> C:        RTSP/1.0 200 OK
               Cseq: 2
               Date: 11 Jan 2001 10:36:30 GMT
               Transport: RTP/AVP/TCP;interleaved=0-1
               Session: 123456

C -> S:        PLAY rtsp://server.com/media RTSP/1.0
               Cseq: 3
               Session: 123456

S -> C:        RTSP/1.0 200 OK
               Cseq: 3
               Session: 123456
               Date: 11 Jan 2001 10:31:24 GMT
               RTP-Info: url=rtsp://server.com/media;
               seq=232433; rtptime=972948234

S -> C: $\000(2 byte length)(“length“ bytes data, w/RTP header)
S -> C: $\001(2 byte length)(“length“ bytes  RTCP packet
```

# Session Initiation Protocol (SIP) [RFC 3261]: Features

- An application level, end-to-end, client-server session signaling (control) protocol for creating, modifying, and terminating sessions with one or more participants
- Can be used for voice, video, instant messaging, gaming, etc.
- Follows on HTTP:
  - Text-based messaging
  - URIs, e.g. sip:dbaron@mit.edu
- Supports user location, call setup, call transfers
- Supports mobility by proxying and redirection
- Works together with other IP protocols:
  - SAP (S. Advertisement P.) for advertising multimedia sessions
  - SDP (S. Description P.) for describing multimedia sessions
  - RSVP for network-resource reservation
  - RTP, RTCP, RTSP for real time data transport

# SIP Components

**SIP End Devices:**

- **User Agent Clients (originating calls)**
- **User Agent Servers (listening for incoming calls)**

**SIP Registrar:**

- **Accepts registration requests from users**
- **Maintains user‘s whereabouts at a Location Server**

**SIP Proxy Server:**

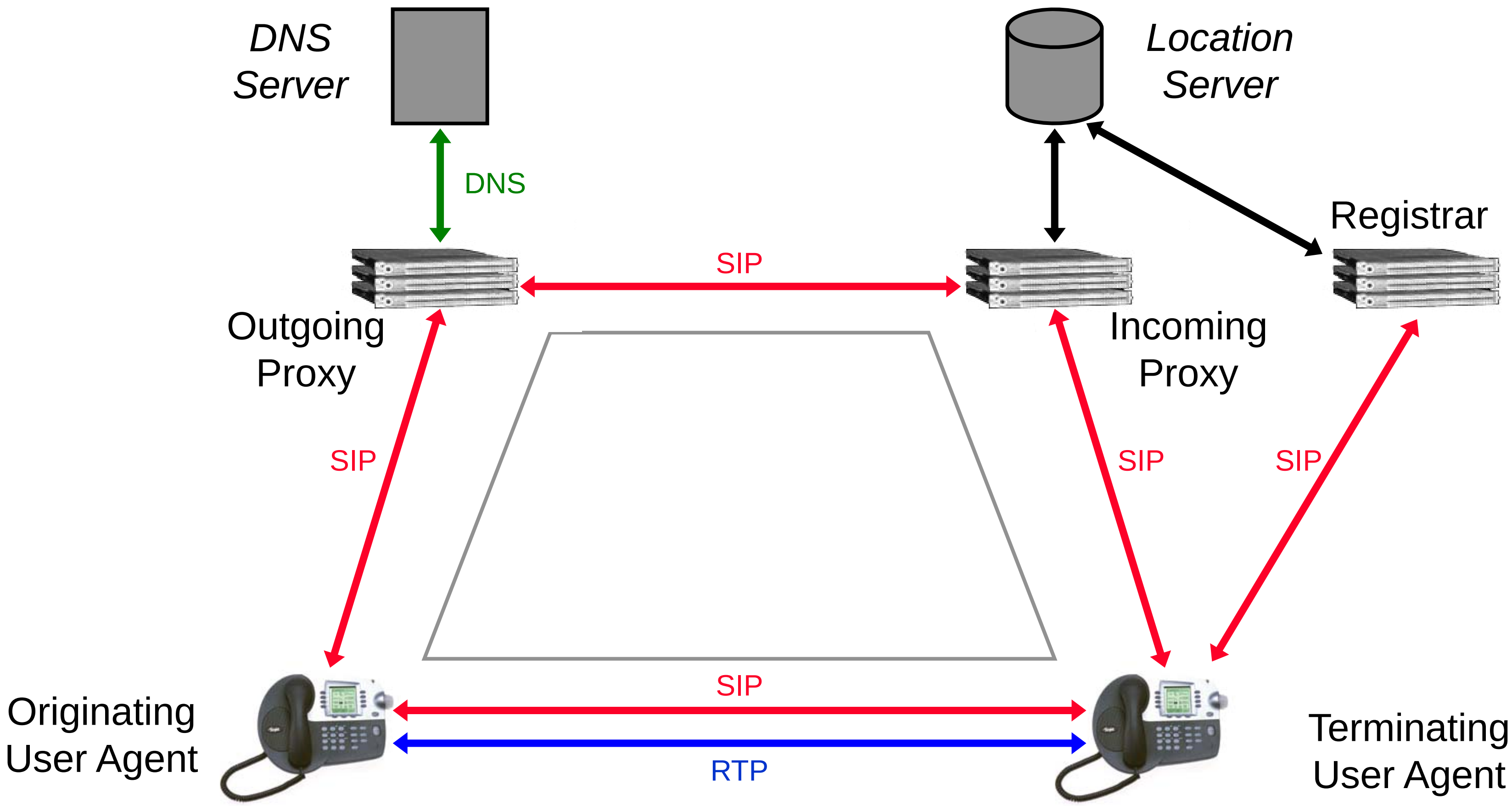
- **Relays call signaling (acting as client and server)**
- **Keeps no session state**

**SIP Redirect Server:**

- **Redirects callers to other servers**

**SIP Gateways**

# SIP Trapezoid




**Can simplify to triangle or P2P signaling**

# SIP Addresses

**SIP uses globally reachable addresses (URIs):**

- **Callees bind to this address using the**

  **SIP REGISTER method**

- **Callers use this address to establish real-time communication with callees**

**Examples:**

- **sip:jiri@iptel.org**
- **sip:voicemail@iptel.org?subject=callme**
- **Non-SIP URLs/URIs can be used as well (mailto:, http:, ...)**

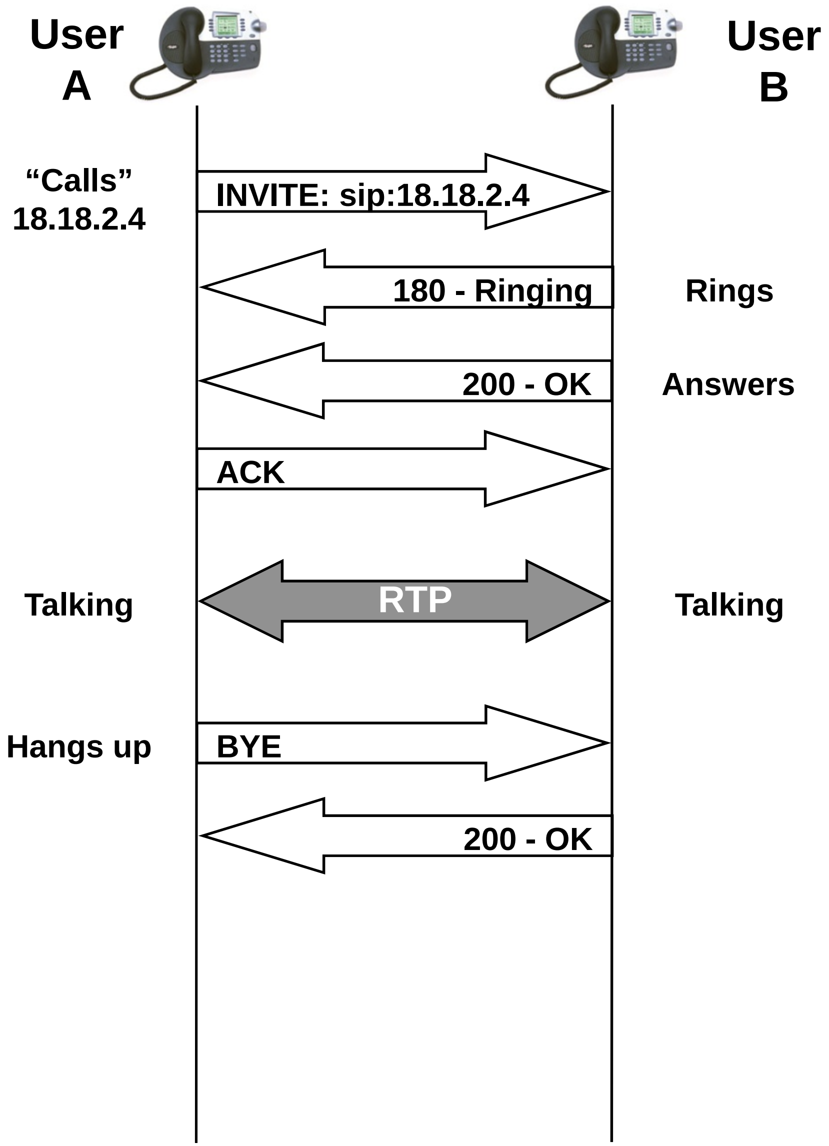
# Important SIP Methods

| | |
|---|---|
| INVITE | Requests a session |
| ACK | Final response to INVITE |
| OPTIONS | Asks for server capabilities |
| CANCEL | Cancels a pending request |
| BYE | Terminates a session |
| REGISTER | Sends user's address to server |

# SIP Responses

| | | |
|---|---|---|
| 1XX | Provisional | 100 Trying |
| 2XX | Success | 200 OK |
| 3XX | Redirection | 302 Moved Temporarily |
| 4XX | Client Error | 404 Not Found |
| 5XX | Server Error | 504 Server Time-out |
| 6XX | Global Failure | 603 Decline |

# SIP Flows - Basic



# SIP INVITE

INVITE sip:e9-airport.mit.edu SIP/2.0

From: "Dennis Baron"<sip:6172531000@mit.edu>;tag=1c41

To: sip:e9-airport.mit.edu

Call-Id: call-1096504121-2@18.10.0.79

Cseq: 1 INVITE

Contact: "Dennis Baron"<sip:6172531000@18.10.0.79>

Content-Type: application/sdp

Content-Length: 304

Accept-Language: en

Allow: INVITE, ACK, CANCEL, BYE, REFER, OPTIONS, NOTIFY, REGISTER, SUBSCRIBE

Supported: sip-cc, sip-cc-01, timer, replaces

User-Agent: Pingtel/2.1.11 (WinNT)

Date: Thu, 30 Sep 2004 00:28:42 GMT

Via: SIP/2.0/UDP 18.10.0.79

# Session Description Protocol (SDP) [RFC 2327]

- **Intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation**
- **Included:**
  - **Type of media (video, audio, etc.)**
  - **Transport protocol (RTP/UDP/IP, H.320, etc.)**
  - **Format of the media (H.261 video, MPEG video, etc.)**
  - **Information to receive those media (addresses, ports, formats and so on)**

# SDP

**v=0**

**o=Pingtel 5 5 IN IP4 18.10.0.79**

**s=phone-call**

**c=IN IP4 18.10.0.79**

**t=0 0**

**m=audio 8766 RTP/AVP 96 97 0 8 18 98**

**a=rtpmap:96 eg711u/8000/1**

**a=rtpmap:97 eg711a/8000/1**

**a=rtpmap:0 pcmu/8000/1**

**a=rtpmap:8 pcma/8000/1**

**a=rtpmap:18 g729/8000/1**

**a=fmtp:18 annexb=no**

**a=rtpmap:98 telephone-event/8000/1**

# 6

# Conclusions

# Concluding Remarks

- **Challenging QoS requirements,**
  **basically emerging from multimedia communication**

- **Many Internet QoS approaches/mechanisms proposed and tested, many not adopted, not feasible in practice, refused**
- **Currently most successful: MPLS**
  **(although not a “native“ QoS approach)**

- **Still a relevant topic in academia, industry, and standardization**
- **Important component in “Future Internet“ initiatives**

- **Still ongoing dispute: raw bandwidth vs. QoS mechanisms?**