# Master Course Computer Networks IN2097

**Prof. Dr.-Ing. Georg Carle**
**Oliver Gasser, M.Sc.**

**Chair for Network Architectures and Services**
**Department of Computer Science**
**Technische Universität München**
**http://www.net.in.tum.de**

# Chapter:
# Quality of Service Support

# Chapter outline – Quality-of-Service Support

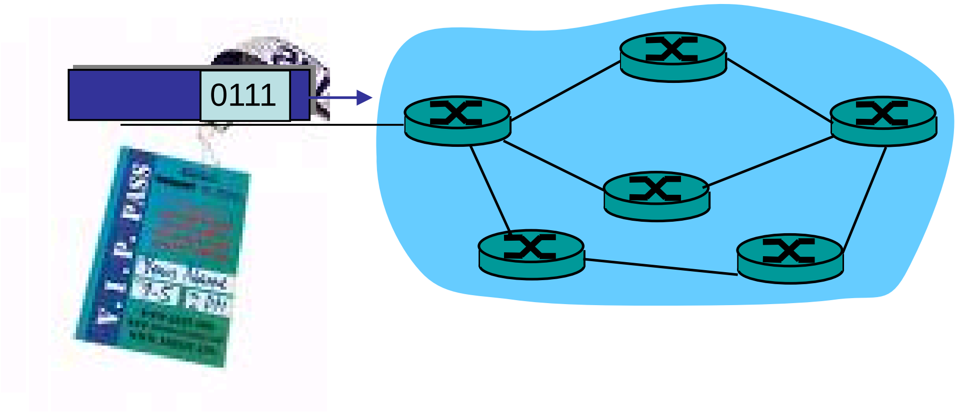
- Why providing multiple classes of service?
- Scheduling and Policing
- QoS guarantees
- Signaling for QoS

# Chapter outline – Quality-of-Service Support

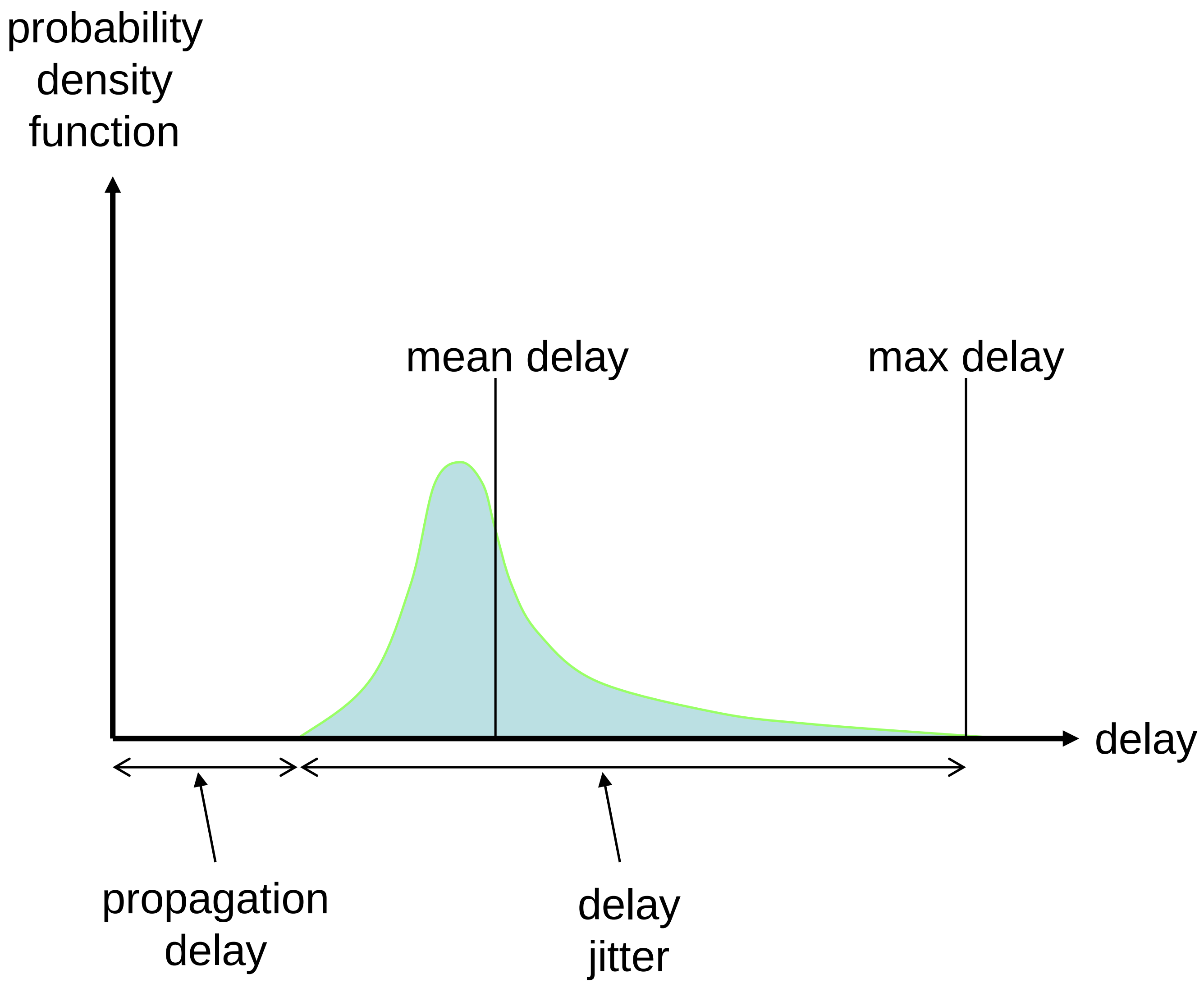
- **Why providing multiple classes of service?**
- Scheduling and Policing
- QoS guarantees
- Signaling for QoS

# Providing Multiple Classes of Service

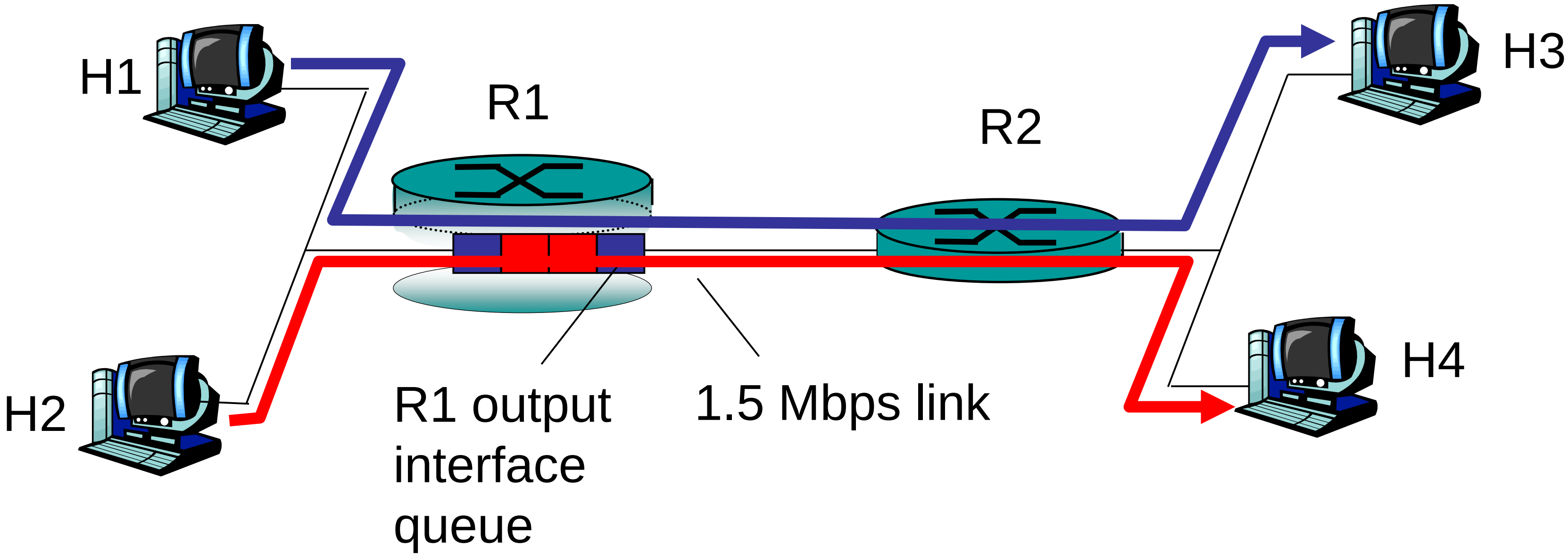
- Traditional Internet approach
  - Best effort, one-size fits all service model
- Alternative approach
  - Multiple classes of service
  - Partition traffic into classes, treated differently
  - Analogy: VIP service vs. regular service
- Granularity
  - Differential service among multiple classes, not among individual connections
- History
  - “Type of Service” bits in IP header



# Motivation: Delay Distribution

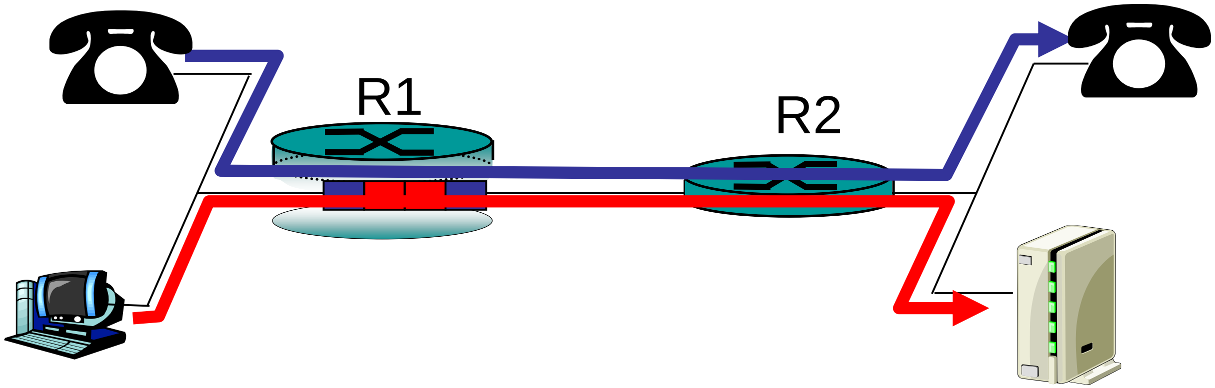


# Multiple classes of service: scenario



# Scenario 1: mixed FTP and audio

- Example: 1Mbps IP phone, FTP or NFS share 1.5 Mbps link.
  - Bursts of FTP/NFS cause congestion ⇨ audio loss
  - Idea: Give priority to audio over FTP

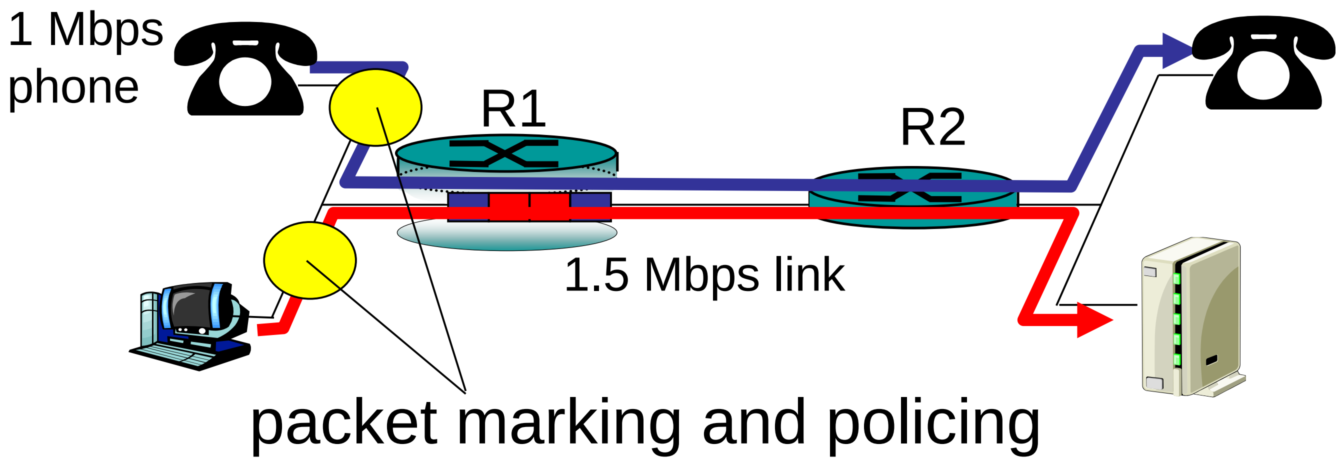



Principle 1

Packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly.

# Principles for QoS Guarantees (more)

- What if applications misbehave (audio sends higher than declared rate)
  - Policing: force source adherence to bandwidth allocations
- Marking and policing at network edge

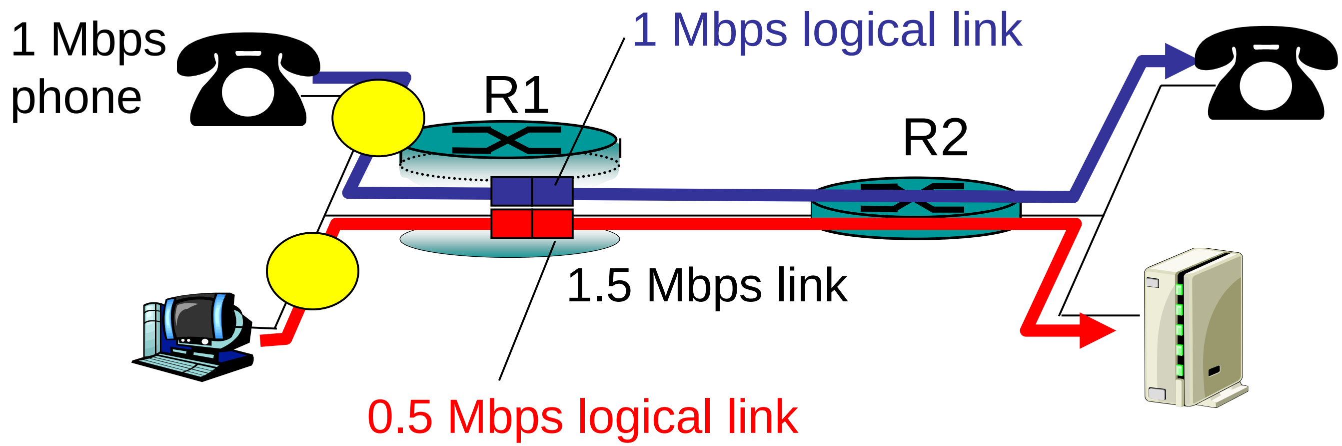



Principle 2

Provide protection (*isolation*) for one class from others.

# Principles for QoS Guarantees (more)

- Idea: Allocate *fixed* (non-sharable) bandwidth to flow
  - Result: *Inefficient* use of bandwidth if flows don‘t use their allocation




Principle 3

While providing **isolation**, it is desirable to use resources as efficiently as possible.

# Chapter outline – Quality-of-Service Support

- Why providing multiple classes of service?
- **Scheduling and Policing**
- QoS guarantees
- Signaling for QoS

# Scheduling vs. Policing

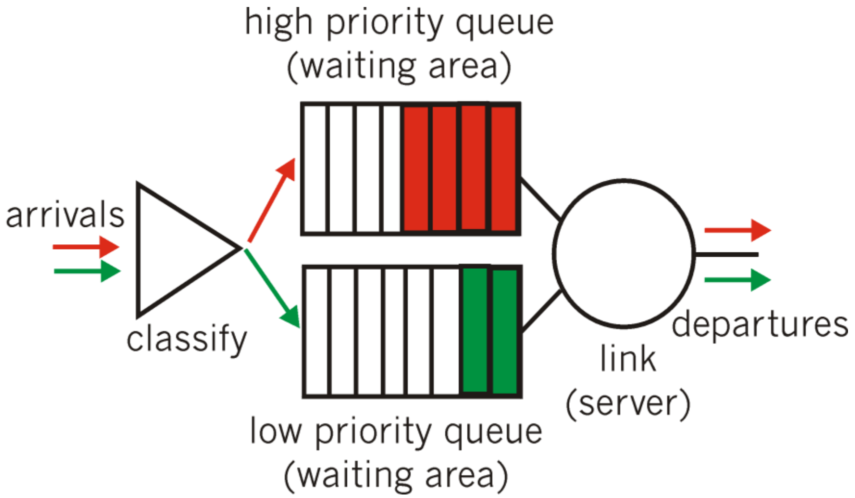
- **Scheduling policy**
  - Choose next packet to send on link
- **Discard policy**
  - Choose packet to drop when queue is full
- **Policing**
  - Limit traffic not to exceed declared parameters

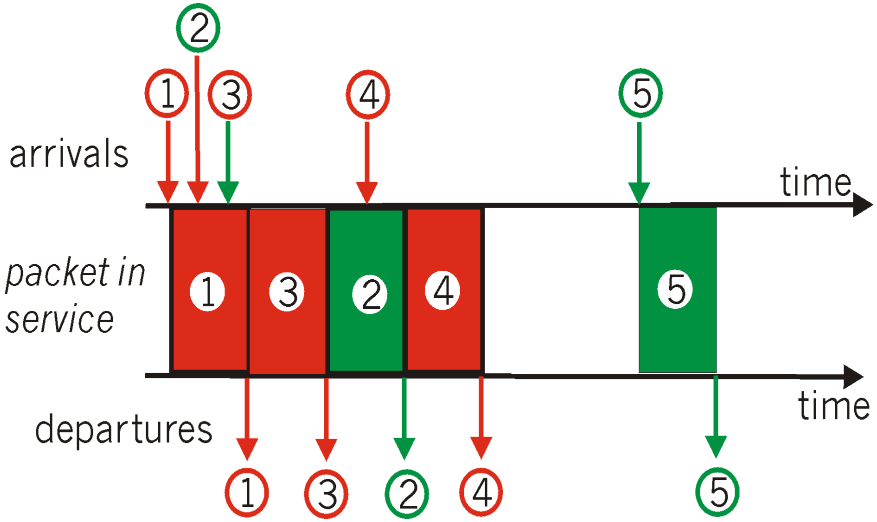
# FIFO Scheduling

- FIFO (first in first out) scheduling
  - Send in order of arrival to queue
  - Real-world example?
    - Queues at the cinema, bus stop,…

# Priority Queuing

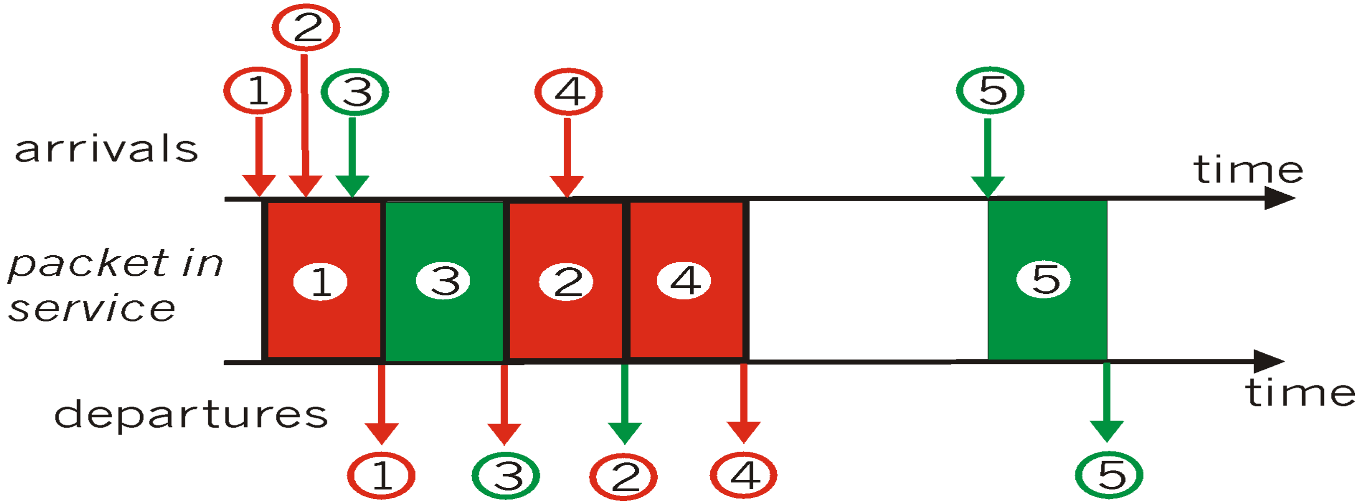
- Priority queuing
  - Multiple *classes*, with different priorities
  - Transmit packet from highest priority queue
  - Class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..
  - Preemptive vs. non-preemptive



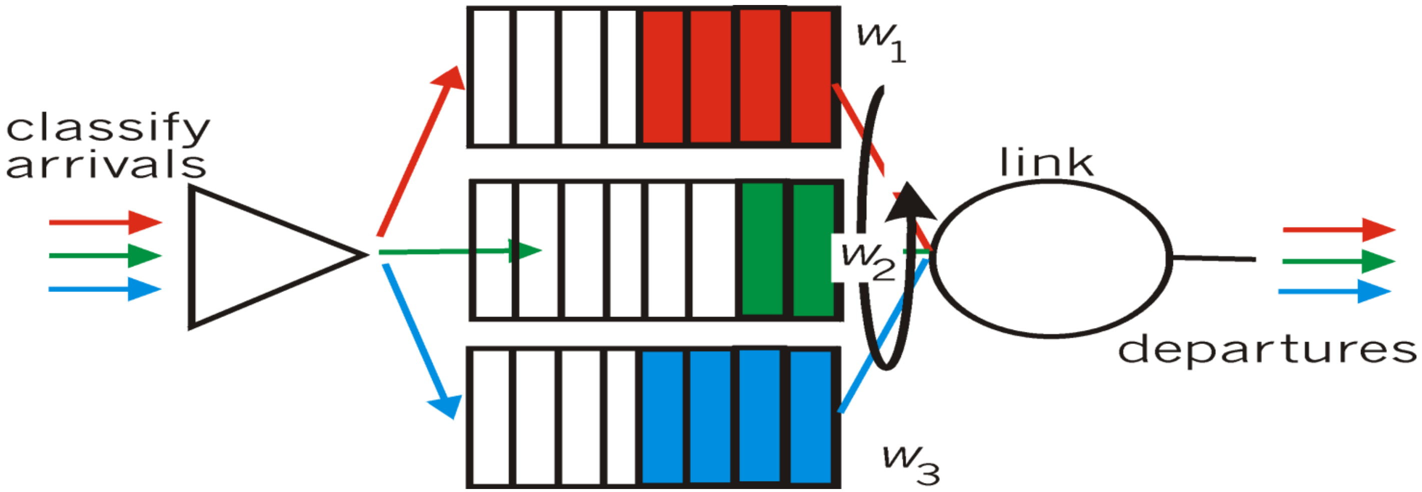
# Round Robin Queuing

- Round robin queing
  - Multiple classes
  - Cyclically scan class queues, serving one from each class (if available)



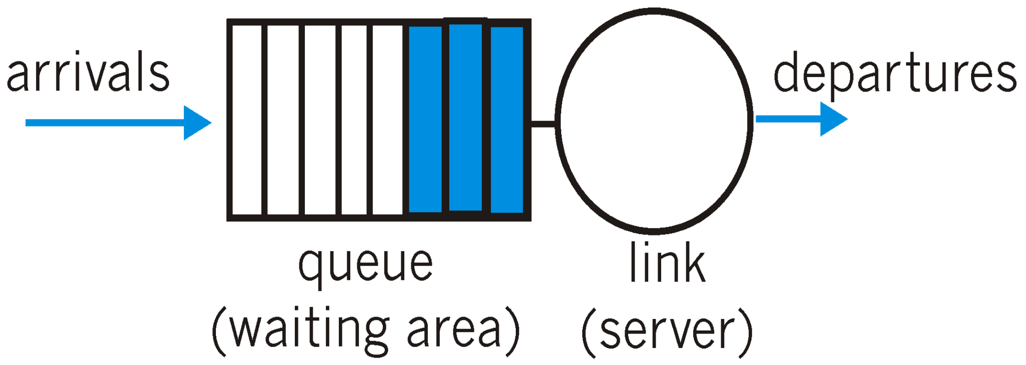
# Weighted Fair Queuing

- Weighted Fair Queuing (WFQ)
  - Generalized Round Robin
  - Each class gets weighted amount of service in each cycle: $w_i$
  - When all classes have queued packets, class i will receive a bandwidth ratio of $w_i/\Sigma w_j$
  - Min. throughput with rate R: $R * (w_i/\Sigma w_j)$



# Discard Policy

- Discard policy
  - If packet arrives to full queue: what to discard?




- Tail drop: drop arriving packet
- Front drop: drop first packet in queue
- Priority: drop/remove on priority basis
- Random: drop/remove randomly

# Random Early Detection

- Synchronization problem
  - Bursts fill up queues
  - Congestion
  - Lots of connections are interrupted

- Idea
  - Drop packets early
  - Drop probabilistically
  - Avoid interrupting many connections

- Random Early Detection (RED)
  - Act on average queue size
  - Two thresholds

# RED Algorithm

```
avg = calc_avg_queue_size()
if avg < TH_min:
     queue_packet()
else if TH_min <= avg < TH_max:
     p_a = calc_prob()
     with probability p_a:
          discard_packet()
     with probability (1-p_a):
          queue_packet()
else:
     discard_packet()
```

# Policing Idea

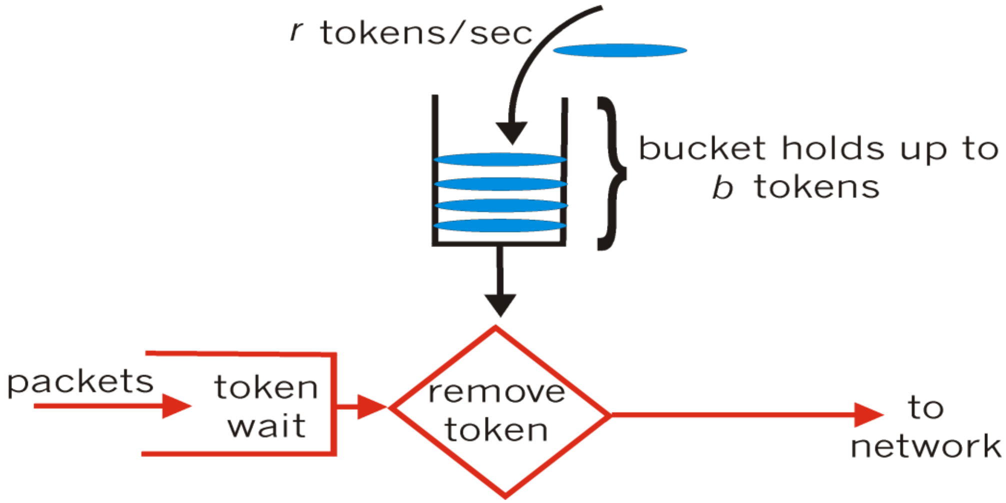
- Goal: Limit traffic to not exceed declared parameters

Three commonly used criteria:

- *(Long term) Average Rate:* How many packets can be sent per unit time (in the long run)
  - Crucial question: What is the interval length? (Flexibility)
    - 100 packets per sec
    - or 6000 packets per min have same average!
- *Peak Rate:* e.g., 6000 packets per min. (ppm) avg.; 1500 pps peak rate
- *(Max.) Burst Size:*
  - Max. number of packets sent consecutively

# Policing Mechanisms

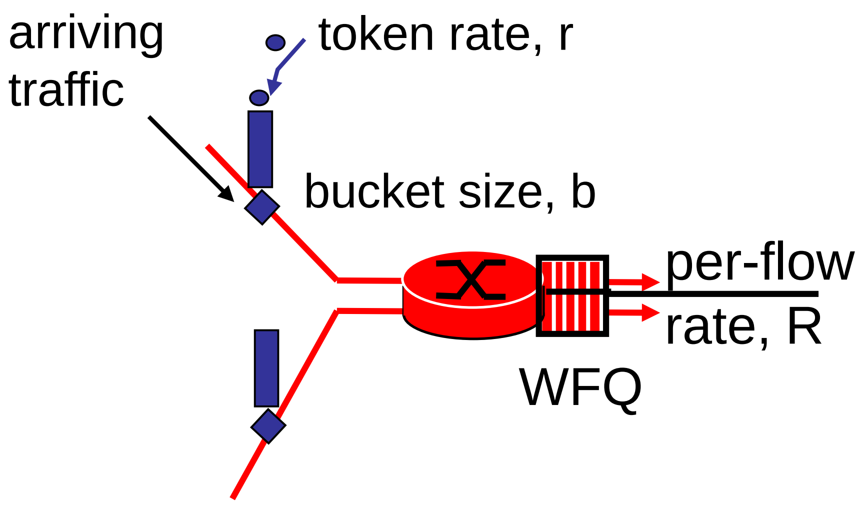
**Leaky Bucket:** Limit input to specified Burst Size and Average Rate.



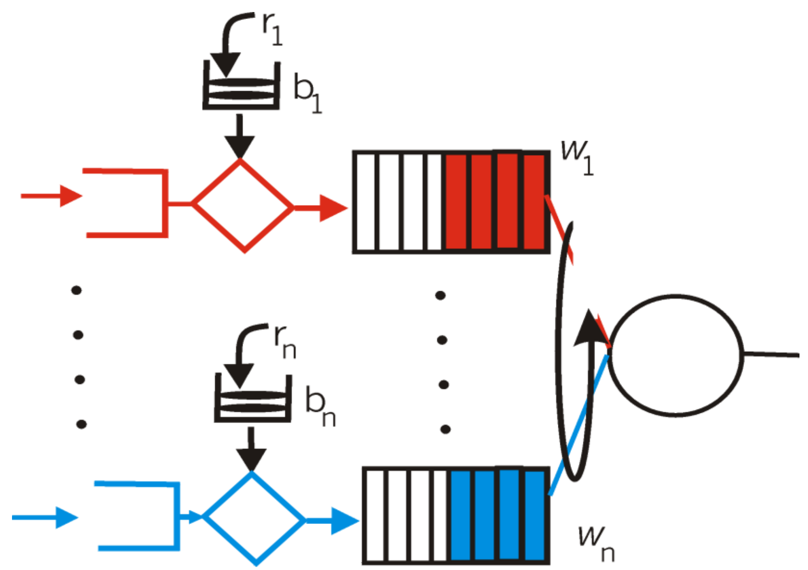

- Bucket can hold b tokens ⇒ limits maximum burst size
- Tokens generated at rate *r token/sec* unless bucket full
- *Over interval of length t:*
  - *Max. allowed number of packets: b + r*t*

# Policing Mechanisms (more)

- Leaky bucket and WFQ combined provide guaranteed upper bound on delay, i.e., *QoS guarantee*




$D_{max} = b/R$



# IETF Differentiated Services

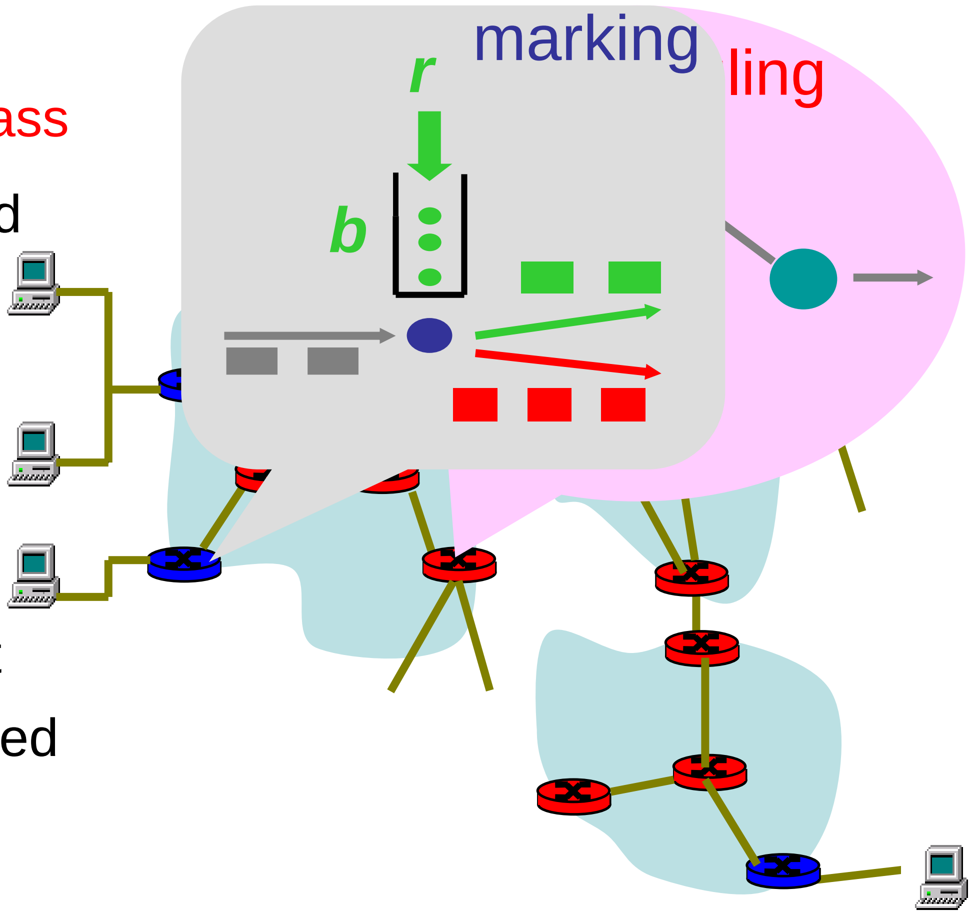
- Want "qualitative" service classes
  - "Behaves like a wire"
  - Relative service distinction: Platinum, Gold, Silver

- *Scalability:*
  - Simple functions in network core
  - Complex functions at edge routers (or hosts)
  - In contrast to IETF Integrated Services

- Don't define service classes, provide functional components to build service classes

## Edge router:

- Per-flow traffic management
- Marks packets according to class
- Marks packets as in-profile and out-profile

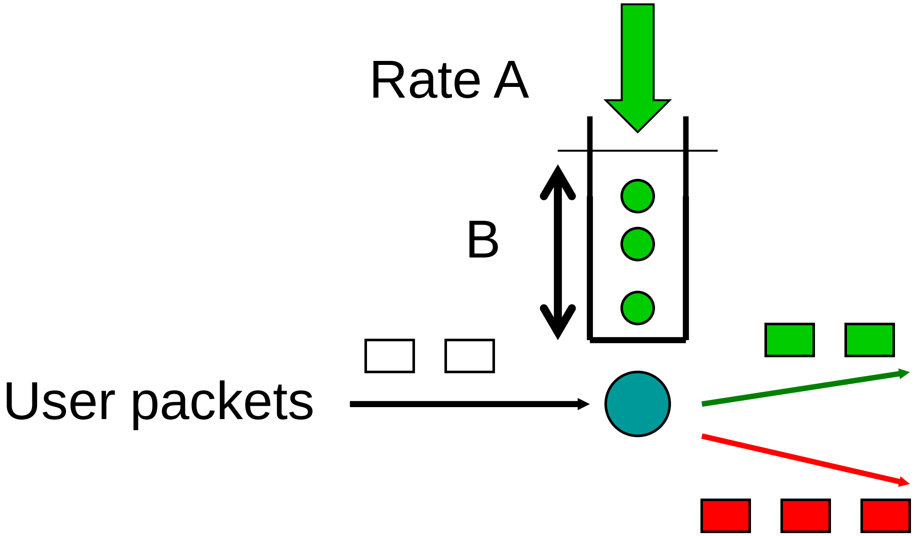



## Core router:

- Per class traffic management
- Buffering and scheduling based on marking at edge
- Preference given to in-profile packets

# Edge Router Packet Marking

- Traffic profile: pre-negotiated rate A, bucket size B
- Packet marking at edge based on per-flow profile




## Possible usage of marking:

- *Class-based* marking: Packets of different classes marked differently
- *Intra-class* marking: Conforming portion of flow marked differently than non-conforming one

# Classification and Conditioning

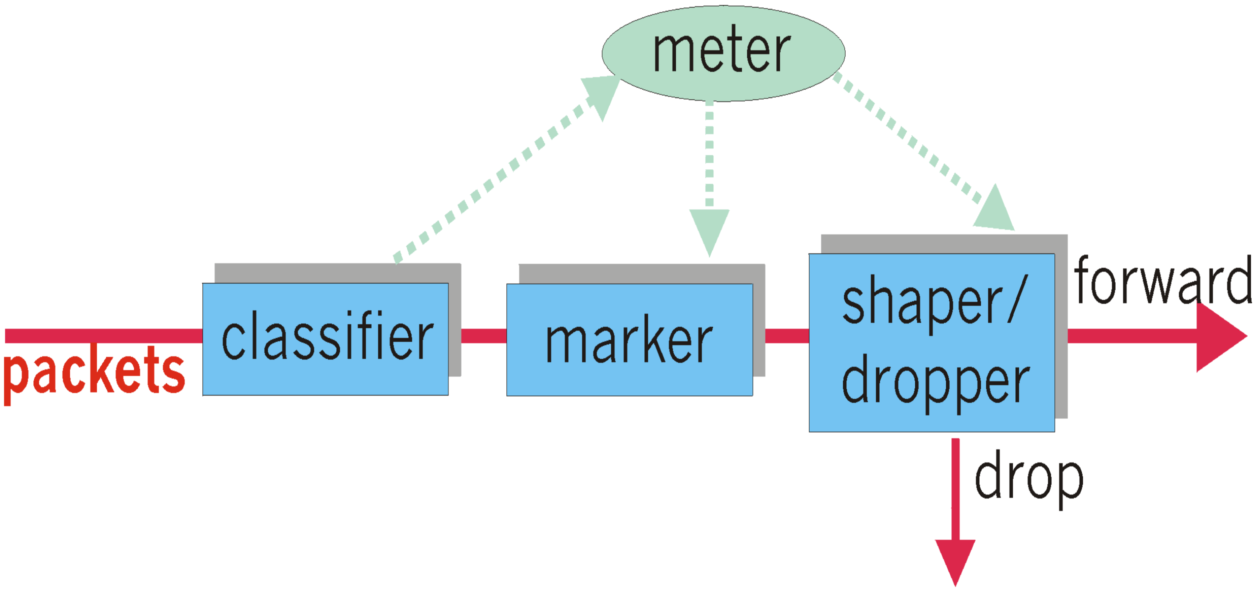
- Differentiated Service Code Point (DSCP)
  - 6 bits, RFC 2474
  - Determines Per-Hop Behavior (PHB) that the packet will receive

- Explicit Congestion Notification (ECN)
  - 2 bits, RFC 3168
  - Used for congestion notification

- Packet is marked in the Type of Service (ToS) in IPv4, and Traffic Class in IPv6

# Classification and Conditioning

- May be desirable to limit traffic injection rate of some class
  - User declares traffic profile (e.g., rate, burst size)
  - Traffic metered, shaped or dropped if non-conforming



# PHB Forwarding

- Different classes can have different performance
- Defines ‘outside’ behavior
  - Measurable performance
- Does not define mechanisms/queuing behavior to use
- Examples:
  - Class A gets x% of outgoing link bandwidth over time intervals of a specified length
  - Class A packets leave first before packets from class B

# Forwarding (PHB)

PHBs being developed:

- Expedited Forwarding
  - Packet departure rate of a class ≤ specified rate
  - Independent on traffic intensity → isolation
  - Logical link with a minimum guaranteed rate

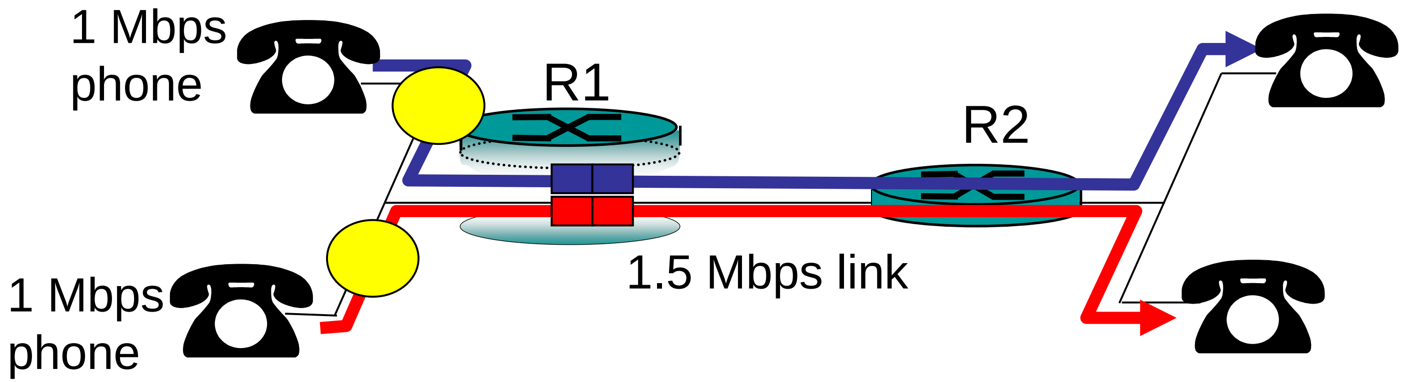
- Assured Forwarding
  - 4 classes of traffic
  - Each class guaranteed minimum bandwidth & buffering
  - 3 drop preferences

# Chapter outline – Quality-of-Service Support

- Providing multiple classes of service
- Scheduling and Policing
- **QoS guarantees**
- Signaling for QoS

# QoS Guarantees

- *Fact:* Can not support traffic demands beyond link capacity

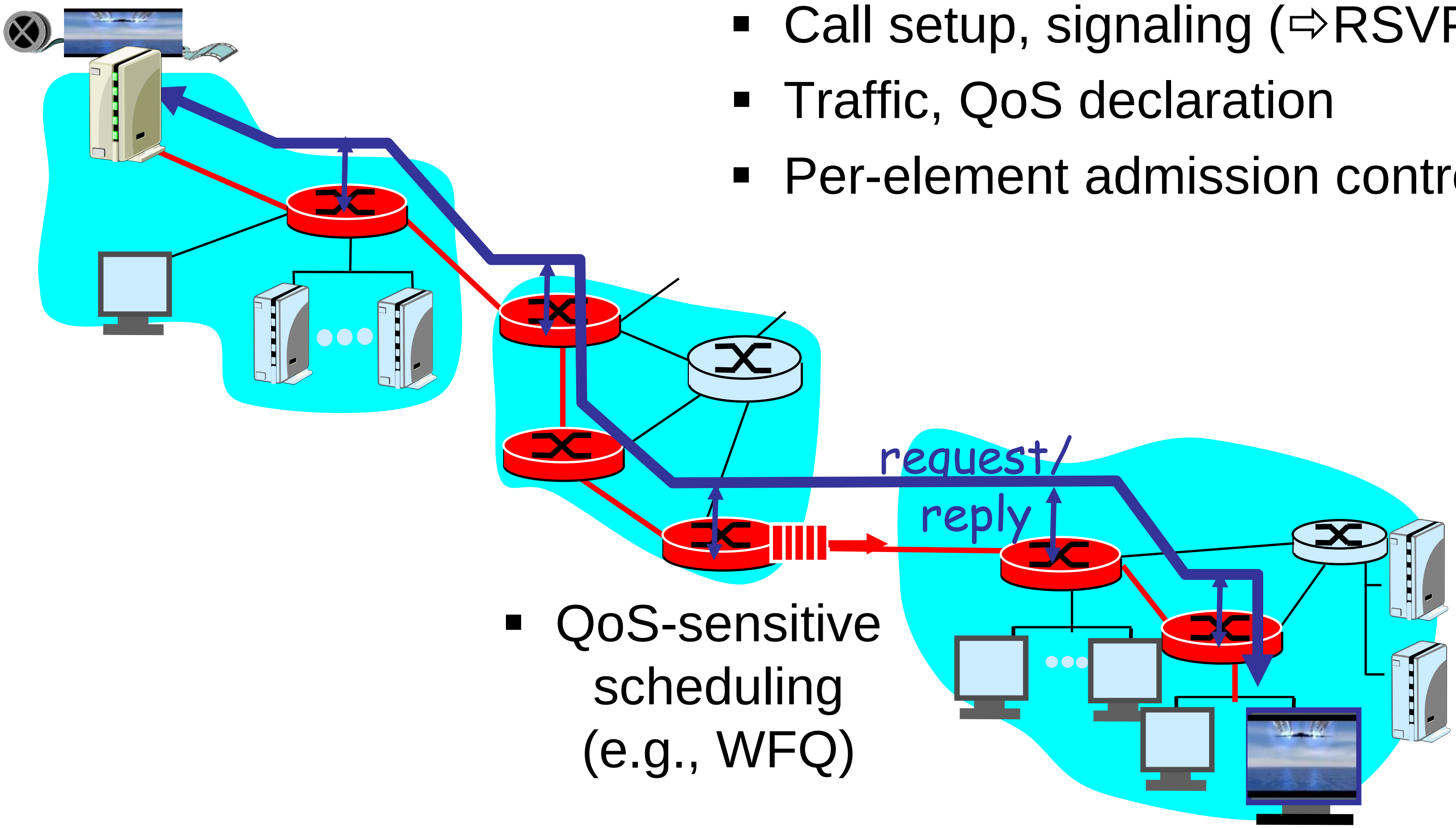



Principle

Call Admission: Flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs.

# QoS Guarantee Scenario

- Resource reservation
  - Call setup, signaling (⇨RSVP)
  - Traffic, QoS declaration
  - Per-element admission control




- QoS-sensitive scheduling (e.g., WFQ)

# IETF Integrated Services

- Architecture for providing QoS guarantees
- For individual application sessions
- Resource reservation
  - Routers maintain state info of allocated resources, QoS requests
- Admit/deny new call setup requests:

Question: Can newly arriving flow be admitted with performance guarantees while not violating QoS guarantees made to already admitted flows?

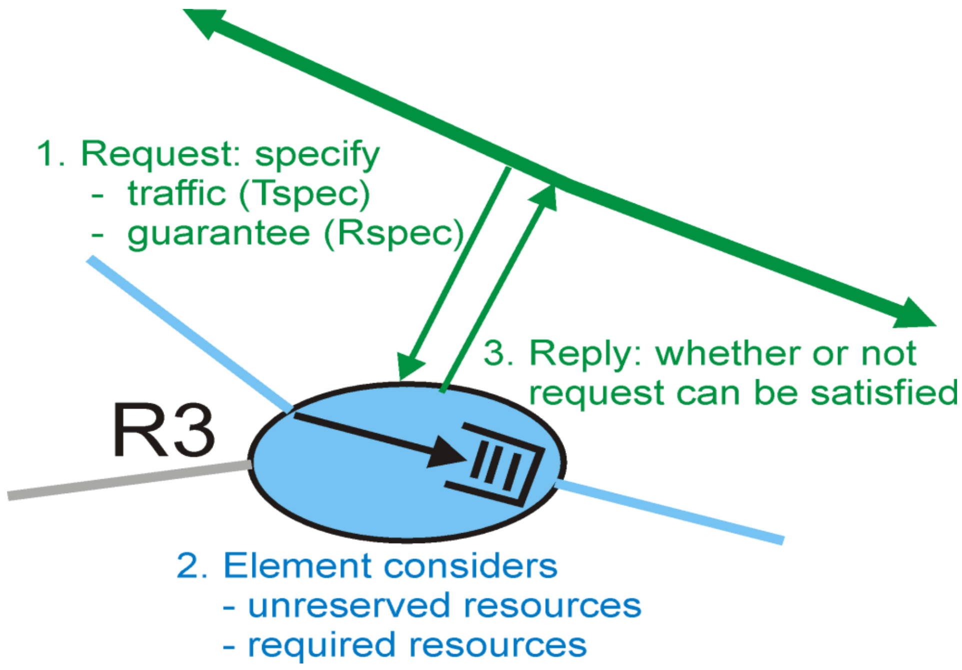
# Call Admission

Arriving session must

- Declare its QoS requirements
  - RSPEC: Defines the QoS guarantees being requested
- Characterize traffic it will send into network
  - TSPEC: Defines traffic characteristics
- Signaling protocol
  - Needed to carry RSPEC and TSPEC to routers (where reservation is required)
  - RSVP: Resource Reservation Protocol

# Call Admission

- Routers will admit calls based on
  - Flow characteristics
    - RSPEC and TSPEC
- The already allocated resources at the router



# Chapter outline – Quality-of-Service Support

- Providing multiple classes of service
- Scheduling and Policing
- QoS guarantees
- **Signaling for QoS**

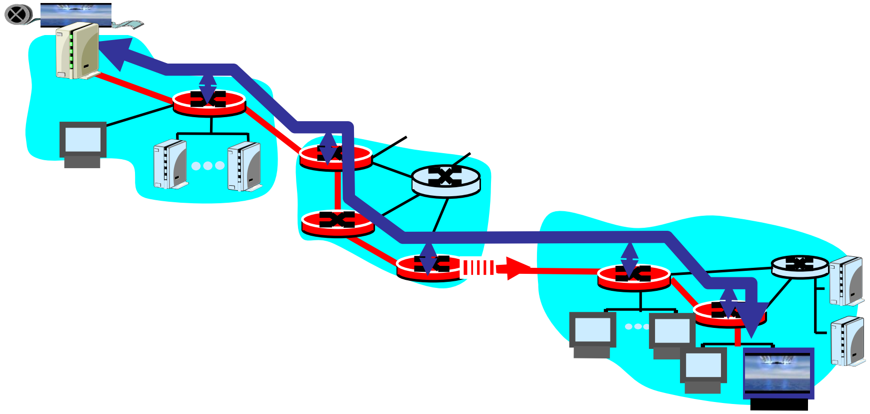
# Signaling in the Internet

| connectionless (stateless) forwarding by IP routers | + | best effort service | = | no network signaling protocols in initial IP design |
|---|---|---|---|---|

- New requirement
  - Reserve resources along end-to-end path (end system, routers) for QoS of multimedia applications
- RSVP: Resource Reservation Protocol [RFC 2205]
  - " … allow users to communicate requirements to network in robust and efficient way." i.e., signaling !
- Earlier Internet Signaling protocol: ST-II [RFC 1819]

# RSVP Design Goals

1. Accommodate heterogeneous receivers (different bandwidth along paths)
2. Accommodate different applications with different resource requirements
3. Make multicast a first class service, with adaptation to multicast group membership
4. Leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. Control protocol overhead to grow (at worst) linear in # receivers
6. Modular design for heterogeneous underlying technologies



# RSVP: does not…

- Specify *how* resources are to be reserved
  - Rather: Provides a mechanism for *communicating needs*

- Determine routes packets will take
  - That’s the job of routing protocols
  - Signaling decoupled from routing

- Interact with forwarding of packets
  - Separation of control (signaling) and data (forwarding) planes

# RSVP: overview of operation

- Senders, receiver join a multicast group
  - Done outside of RSVP
- Sender-to-network signaling
  - *Path message:* Make sender presence known to routers
  - Path teardown: Delete sender’s path state from routers
- Receiver-to-network signaling
  - *Reservation message:* Reserve resources from sender(s) to receiver
  - Reservation teardown: Remove receiver reservations
- Network-to-end-system signaling
  - Path error
  - Reservation error

# RSVP Messages

There are two primary types of messages:

- Path messages (*path*)
  - Sent from the sender host along the data path
  - Stores the *path state* in each node along the path
    - IP address of the previous node
    - *Sender template* describes the format of the sender data
    - *Sender TSPEC* describes the traffic characteristics of the data flow
    - ADSPEC advertises supported RSVP services

- Reservation messages (*resv*)
  - Sent from the receiver to the sender host along the reverse data path
  - At each node
    - IP destination address changes to the address of the next node IP source address to the address of the previous node address
  - The *resv* message includes the *flowspec* data object that identifies the resources that the flow needs.

# Further reading

- *Computer Networking: A Top-Down Approach*
  J. Kurose and K. Ross
  Pearson

- *High Speed Networks and Internets*
  W. Stallings
  Prentice Hall

- Respective RFCs