

# Common network/protocol functions

## Goals:

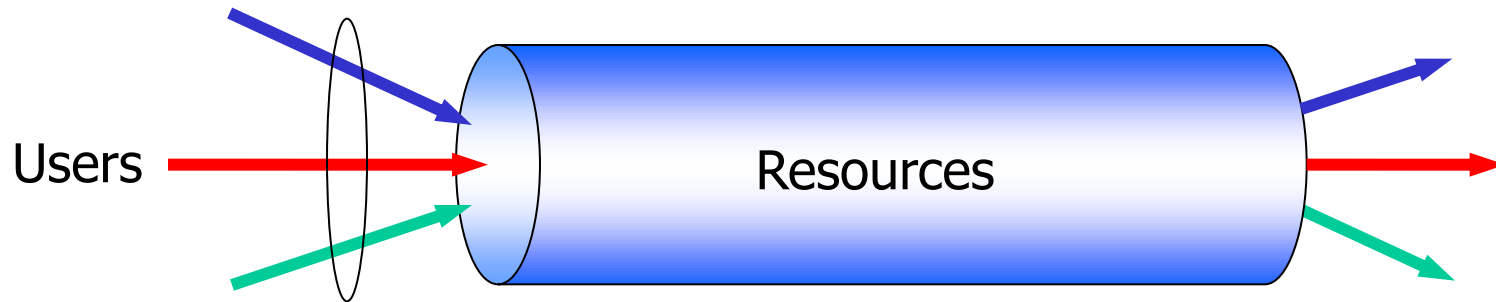
- ❑ Identify, study common architectural components, protocol mechanisms
- ❑ *Synthesis*: big picture
- ❑ *Depth*: important topics not covered in introductory courses

## Overview:

- ✓ Signaling
- ✓ State
- ❑ **Multiplexing / Resource Allocation**
- ❑ Randomization
- ❑ Indirection
- ❑ Service location
- ❑ Network virtualization

# Multiplexing

Multiplexing: *Sharing* resource(s) among users of the resource.



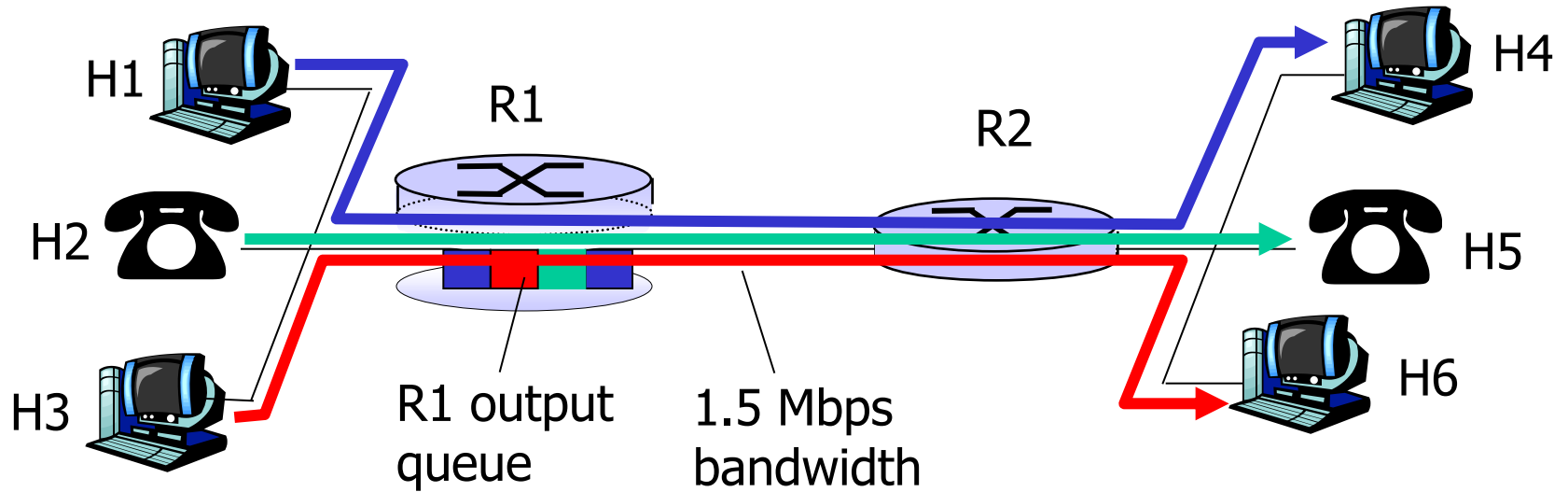
## In this lecture:

- ❑ The Resources are
  - Bandwidth (Link Capacity)
  - Queues (Buffers)
- ❑ The Users are
  - Phone Calls
  - TCP/UDP flows, packets

Other types of resources:

- ❑ CPU
  - ❑ Storage
  - ❑ Frequency spectrum
- ... and other types of users?

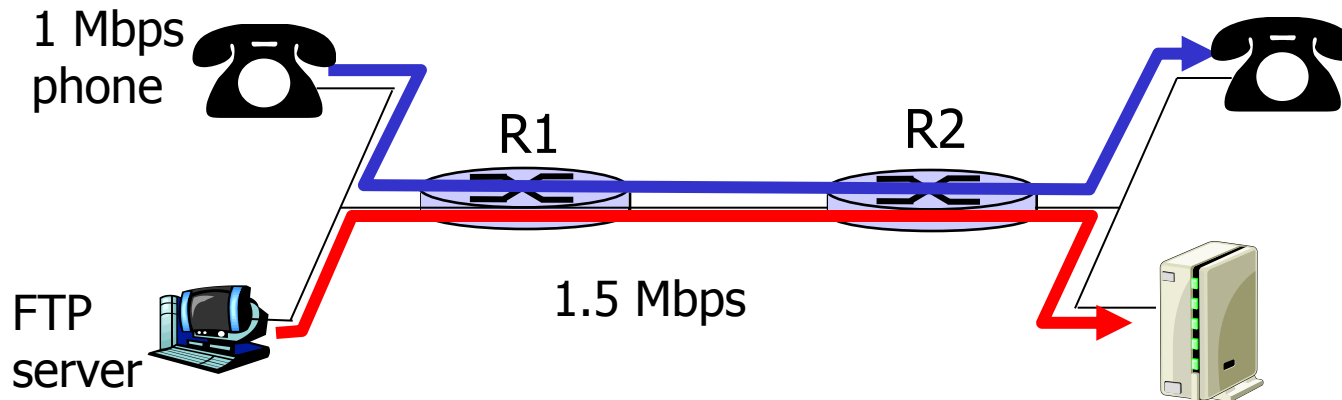
# The resources and the users



# From Multiplexing to QoS

## □ *Basic facts of life:*

- Bandwidth is **finite**
- Cannot support traffic demands beyond capacity



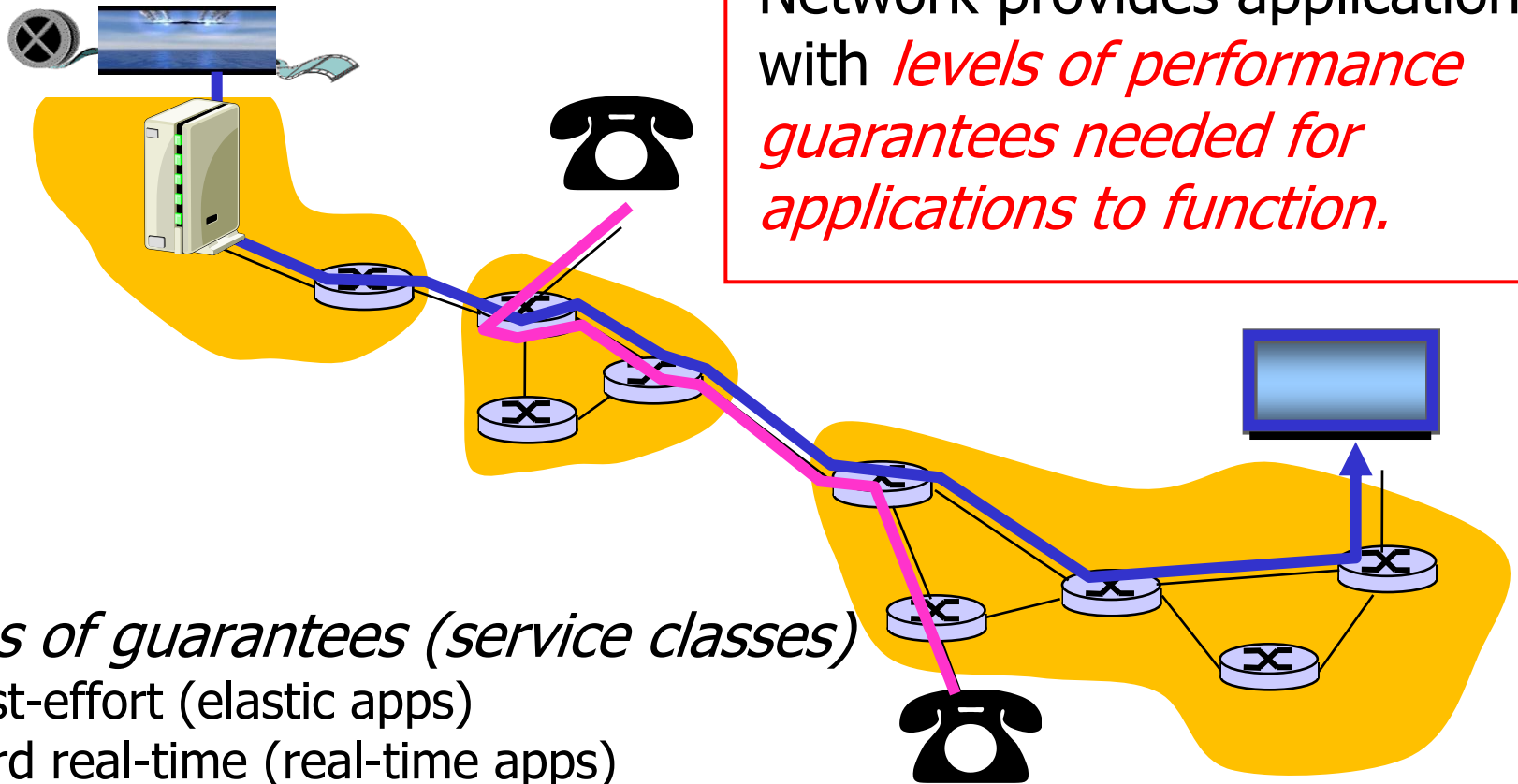
## □ Example: 1Mbps IP phone, FTP share 1.5 Mbps link

- bursts of FTP can congest router, cause **large delays/audio loss**

## □ What's to be done?

- Move away from the best-effort paradigm
- ... provide "Quality of Service (QoS)"

# QoS: What is it?



## QoS

Network provides applications with *levels of performance guarantees needed for applications to function.*

### *Types of guarantees (service classes)*

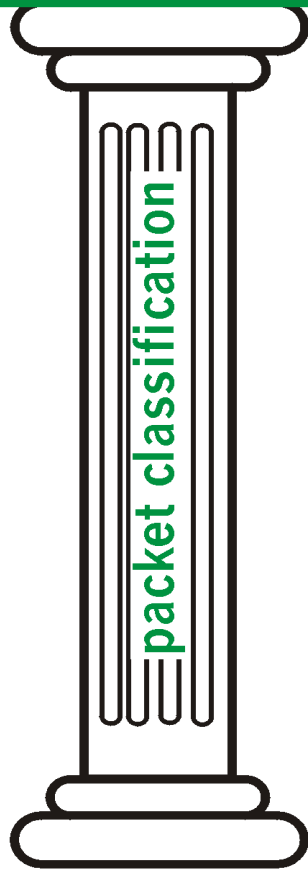
- ❑ Best-effort (elastic apps)
- ❑ Hard real-time (real-time apps)
  - e.g., bounded loss / delay
- ❑ Soft real-time (tolerant apps)
  - e.g., probabilistic loss / delay

### *How to implement QoS?*

- ❑ A set of five principles

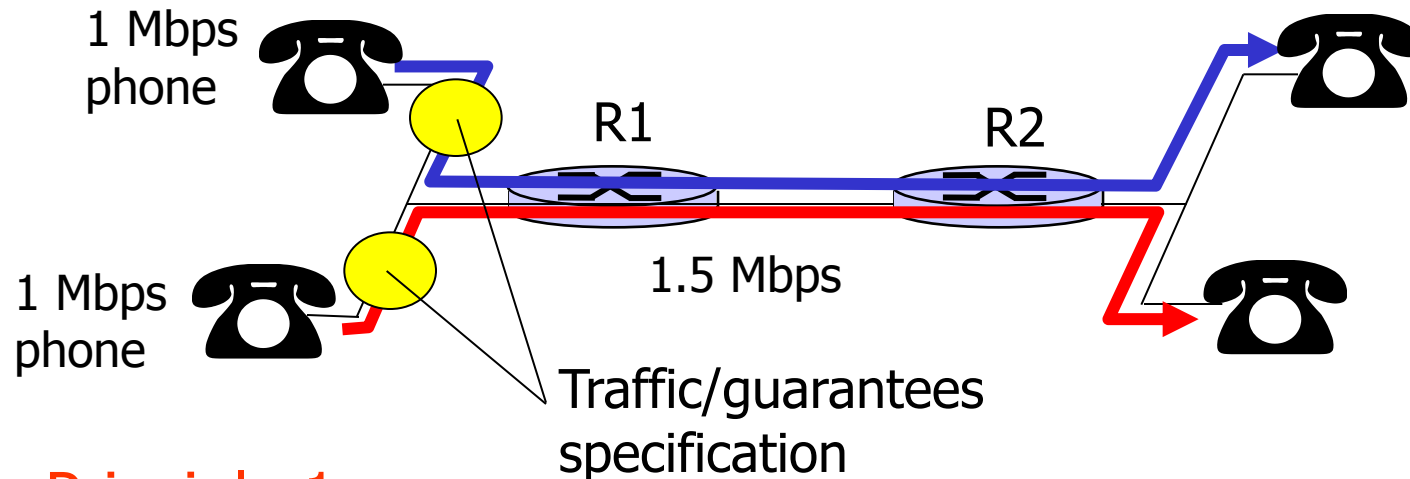
# Summary of QoS Principles

QoS for networked applications



# Principle 1. Traffic/Guarantees specification

- ❑ Two 1Mbps IP phones share 1.5 Mbps link
  - want applications to specify: 1) how much bandwidth they need, 2) what levels of guarantees they want

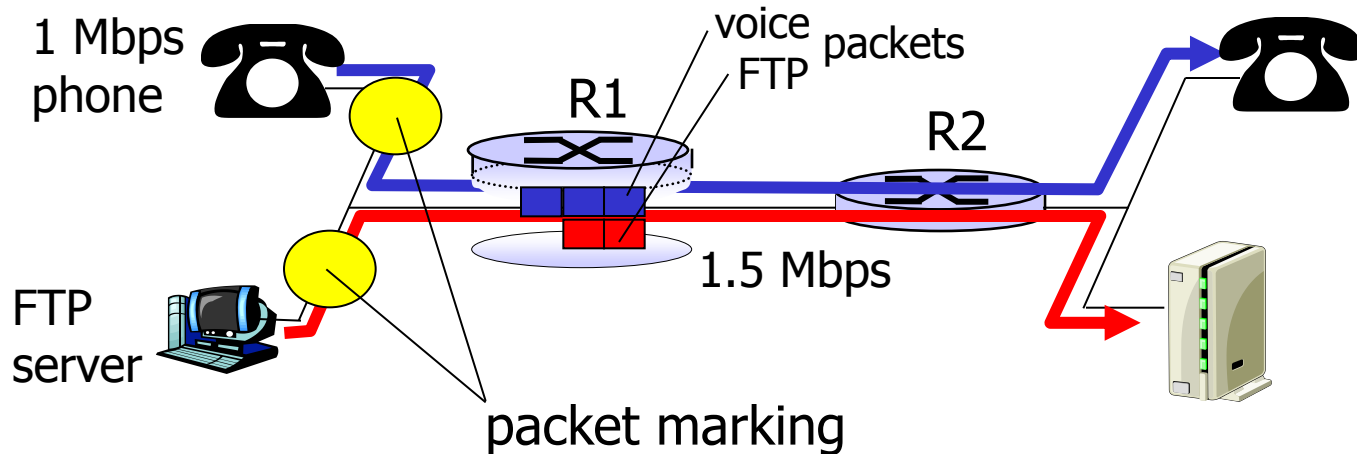


## Principle 1

Traffic/guarantees specification (**service contract**) needed for router to plan whether it can provide certain levels of performance guarantees

# Principle 2. Traffic classification

- ❑ 1Mbps IP phone, FTP share 1.5 Mbps link
  - bursts of FTP can congest router, cause audio loss
  - want to **give priority** to audio over FTP
  - Can FTP server declare how much bandwidth it needs?



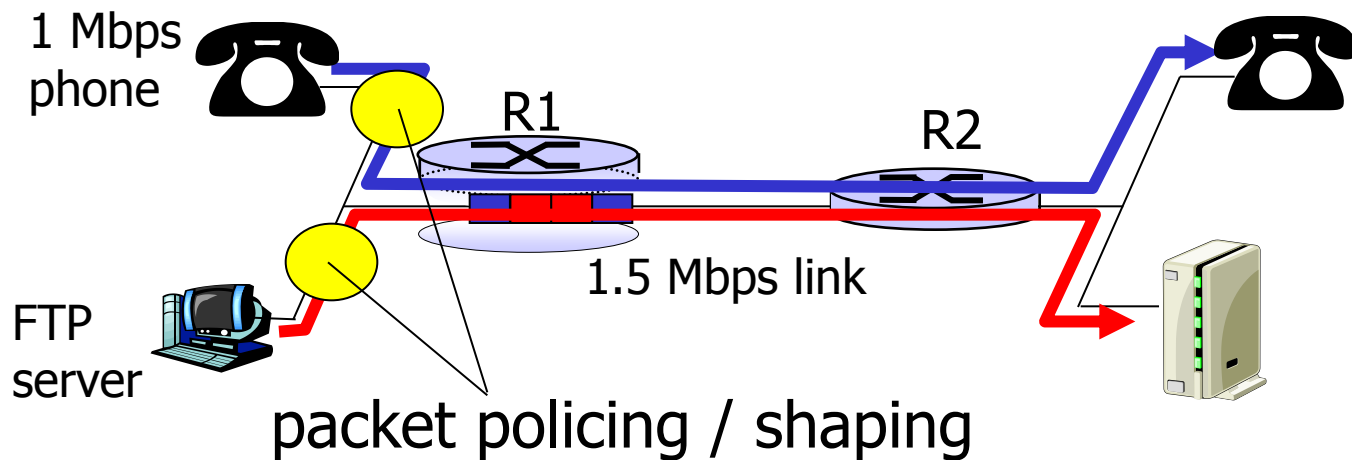
## Principle 2

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



# Principle 3. Traffic isolation

- ❑ what if applications misbehave (audio sends higher than declared rate)
  - Want to force source adhere to traffic specification

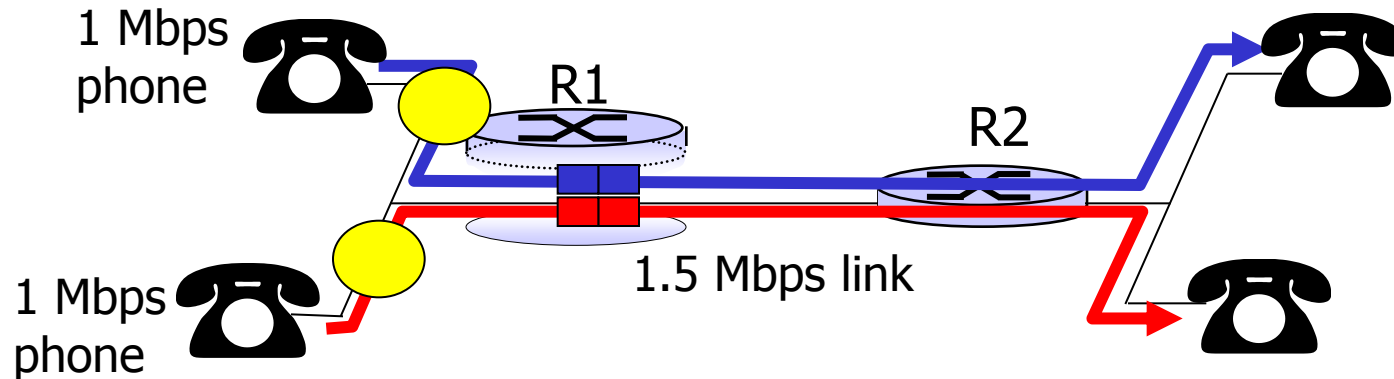


## Principle 3

provide protection (*isolation*) for one class from others

# Principle 4. Call admission

- Bandwidth is finite
  - Cannot support more than available
  - To provide isolation, some flows have to be sacrificed

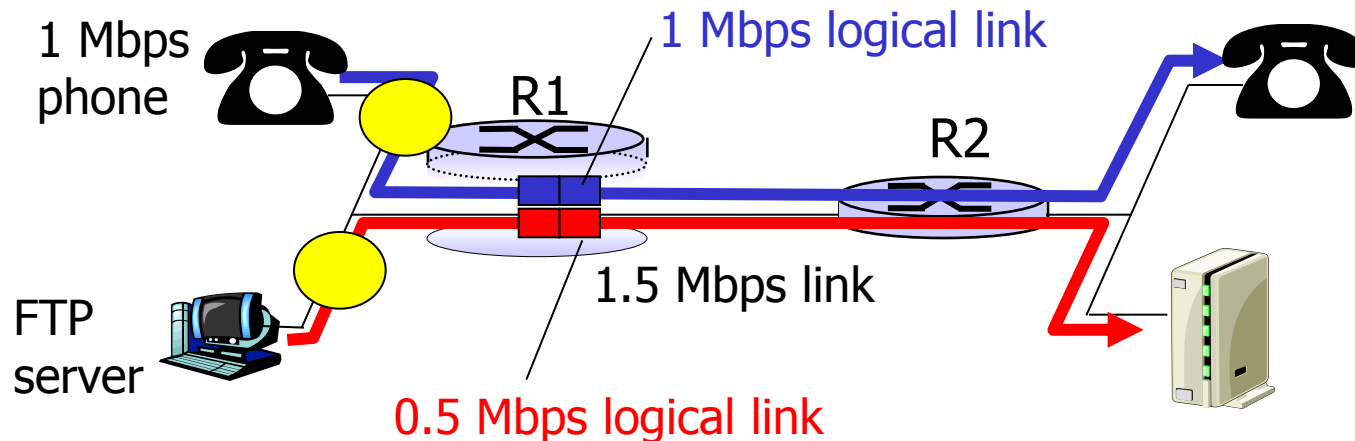


## Principle 4

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

# Principle 5. Resource sharing

- ❑ Allocating *fixed* (non-sharable) bandwidth
  - Inefficient if flows don't use it
  - Circuit/packet switching; scheduling



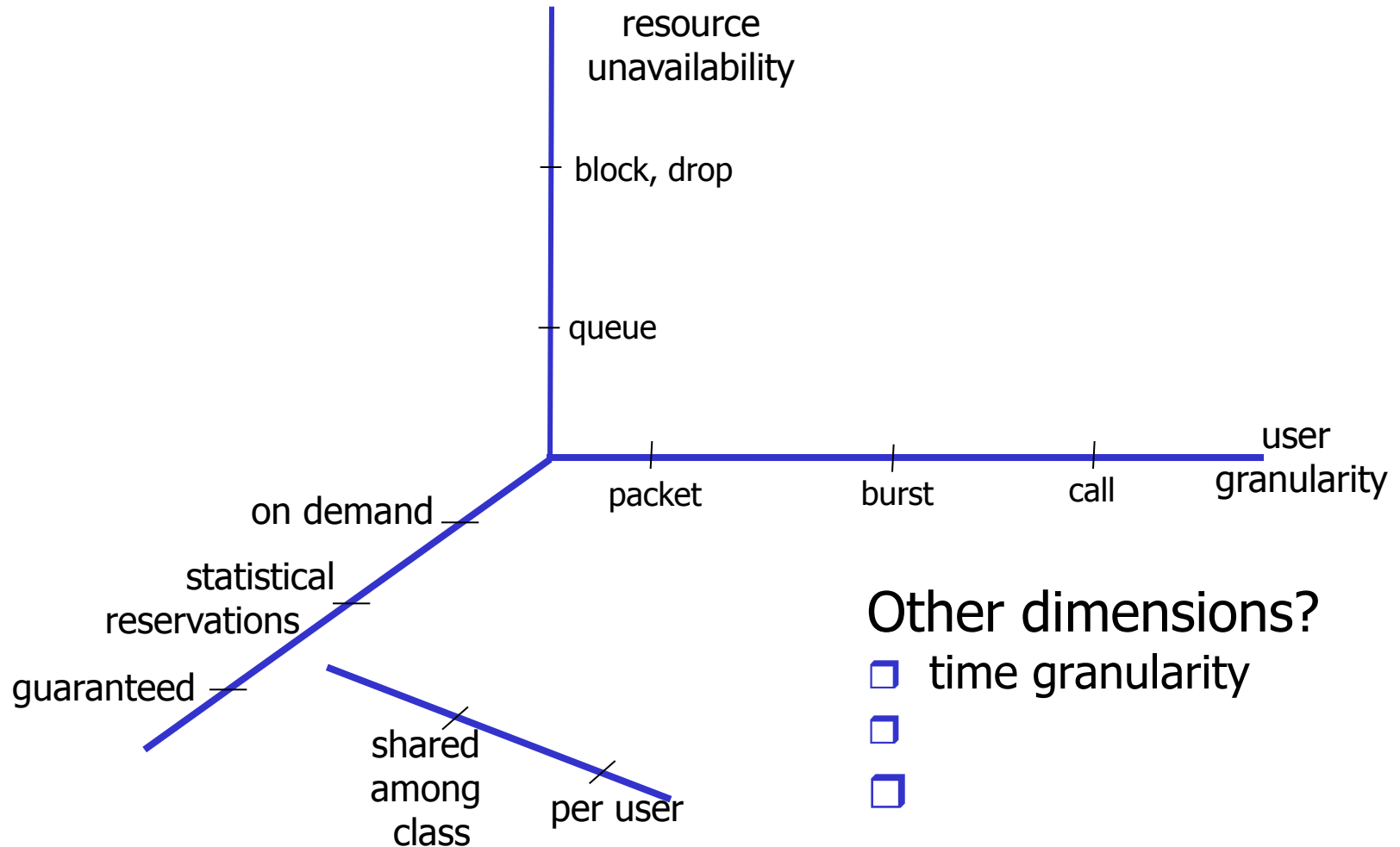
## Principle 5

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

# Service Classes vs. Principles

	Traffic / Guarantees Specification	Traffic Classification	Traffic Isolation	Call Admission	Resource Sharing
Best Effort	Yes/No?	?	?	?	?
Hard real-time	?	?	?	?	?
Soft real-time	?	?	?	?	?

# Back to Multiplexing: many dimensions

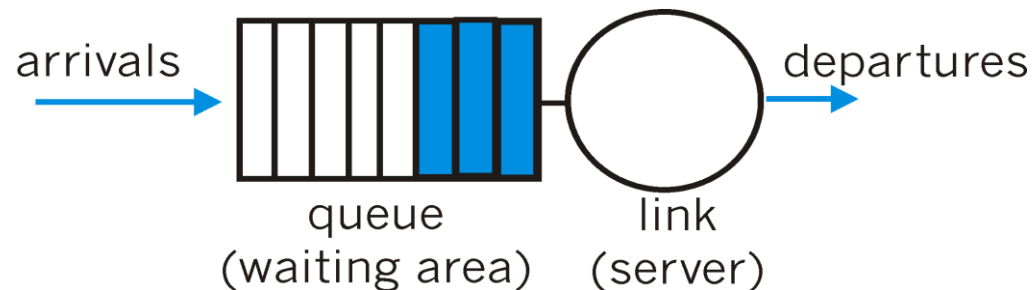


# Outline

- ❑ Scheduling
- ❑ Policing
- ❑ Admission Control
- ❑ General class-based link sharing (multiplexing)
- ❑ Concrete IETF proposals to do things in practice
  - IntServ
  - DiffServ
  
- ❑ Focus on packet-level multiplexing!

# Scheduling And Policing *Packets*

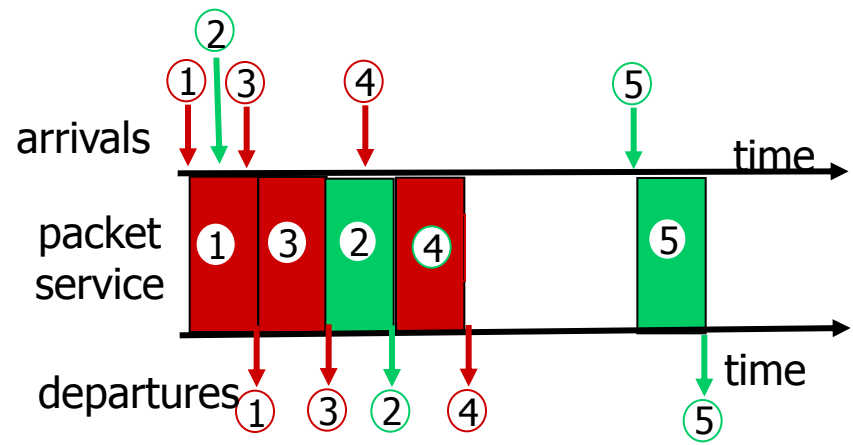
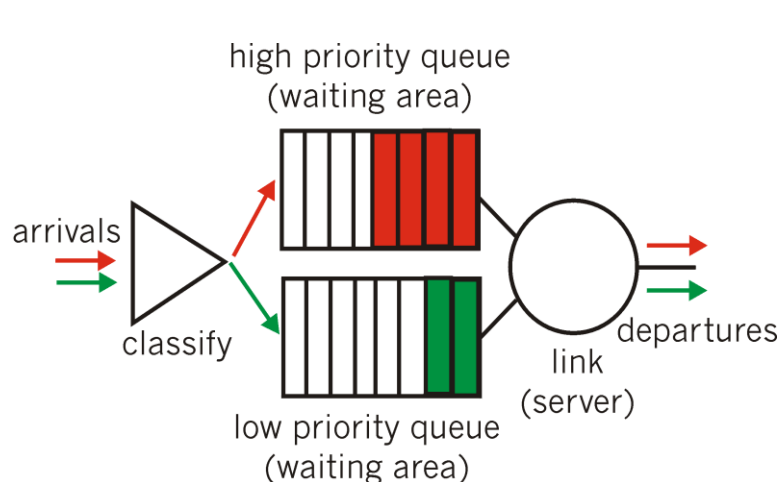
- ❑ **Scheduling:** choose next packet to send on link
- ❑ **FIFO (first in first out) scheduling:** send in order of arrival to queue
  - Real-world example: stop sign
  - **Discard policy:**
    - Tail drop: drop arriving packet
    - RED



# Scheduling Policies: more

**Strict Priority scheduling:** transmit highest priority queued packet

- Multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
  - real world example: reservations versus walk-ins

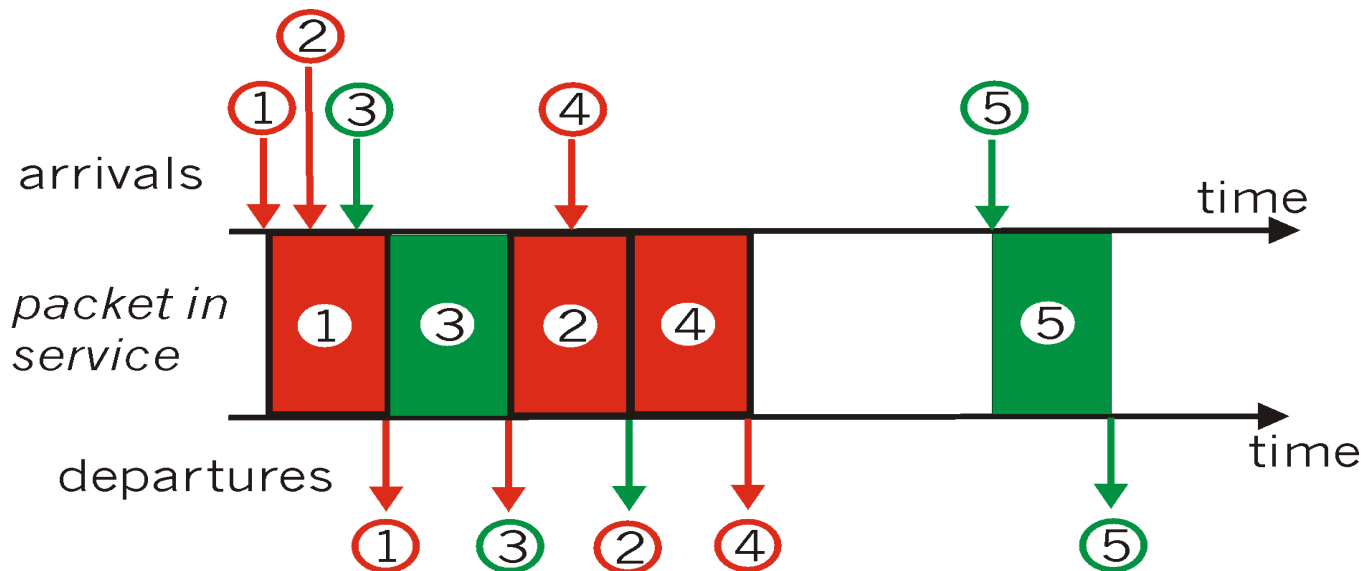




# Scheduling Policies: still more

## Round robin scheduling:

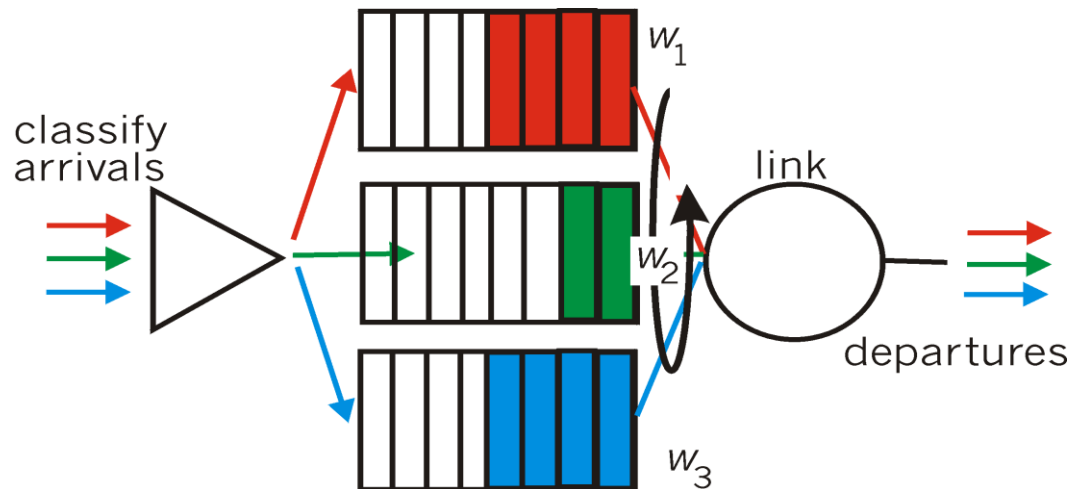
- ❑ Multiple classes
- ❑ Cyclically scan class queues, serving one from each class (if available)
- ❑ Real-world example: 4-way stop (distributed scheduling)



# Scheduling Policies: still more

## Weighted Fair Queuing:

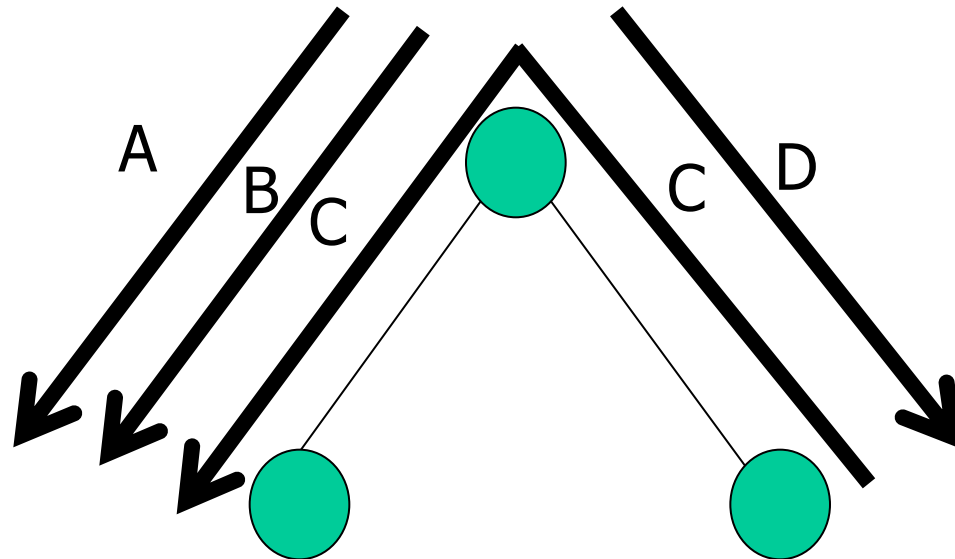
- generalized Round Robin
- each class gets weighted amount of service in each cycle



# Excursion: Fairness (1)

- How to divide a resource fairly?
- Philosophical question...
  - many notions of fairness exist!

Fair shares for flows A,B,C,D?



# Excursion: Fairness (2)

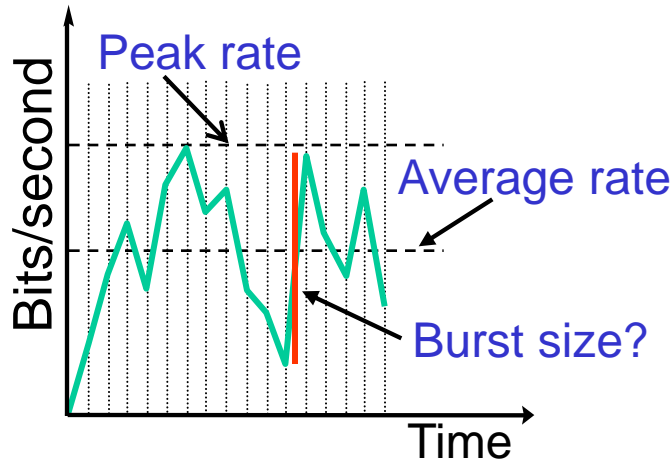
- ❑ Important concept: **max-min fairness**

— **Max-Min Fairness** —

A max-min fair scheduler maximizes the minimum rate of any data flow.  
(First, the smallest rate is maximized, then the second smallest, etc.)

# Policing Mechanisms

**Goal:** Limit traffic to not exceed declared parameters

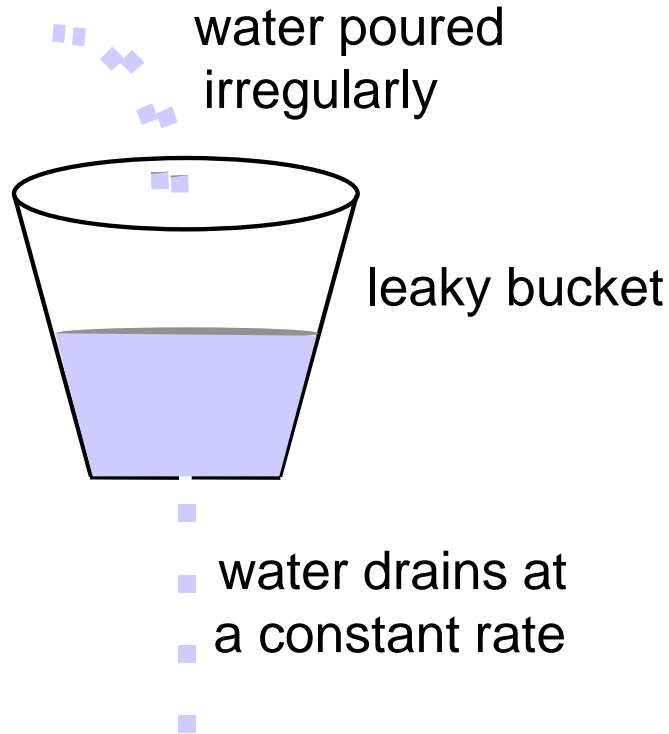


Three commonly-used criteria:

- ❑ *(Long term) Average Rate:* how many pkts can be sent per unit time (in the long run)
  - crucial question: what is interval length: 100 packets per sec or 6000 packets per min have same average!
- ❑ *Peak Rate:* e.g., 6000 pkts per min. (ppm) avg.; 15000 ppm peak rate
- ❑ *(Max.) Burst Size:* max. number of pkts sent consecutively (with no intervening idle)

# Leaky Bucket Algorithm

Used to police arrival rate + burst size of a packet flow(s)



Leak rate corresponds to long-term rate

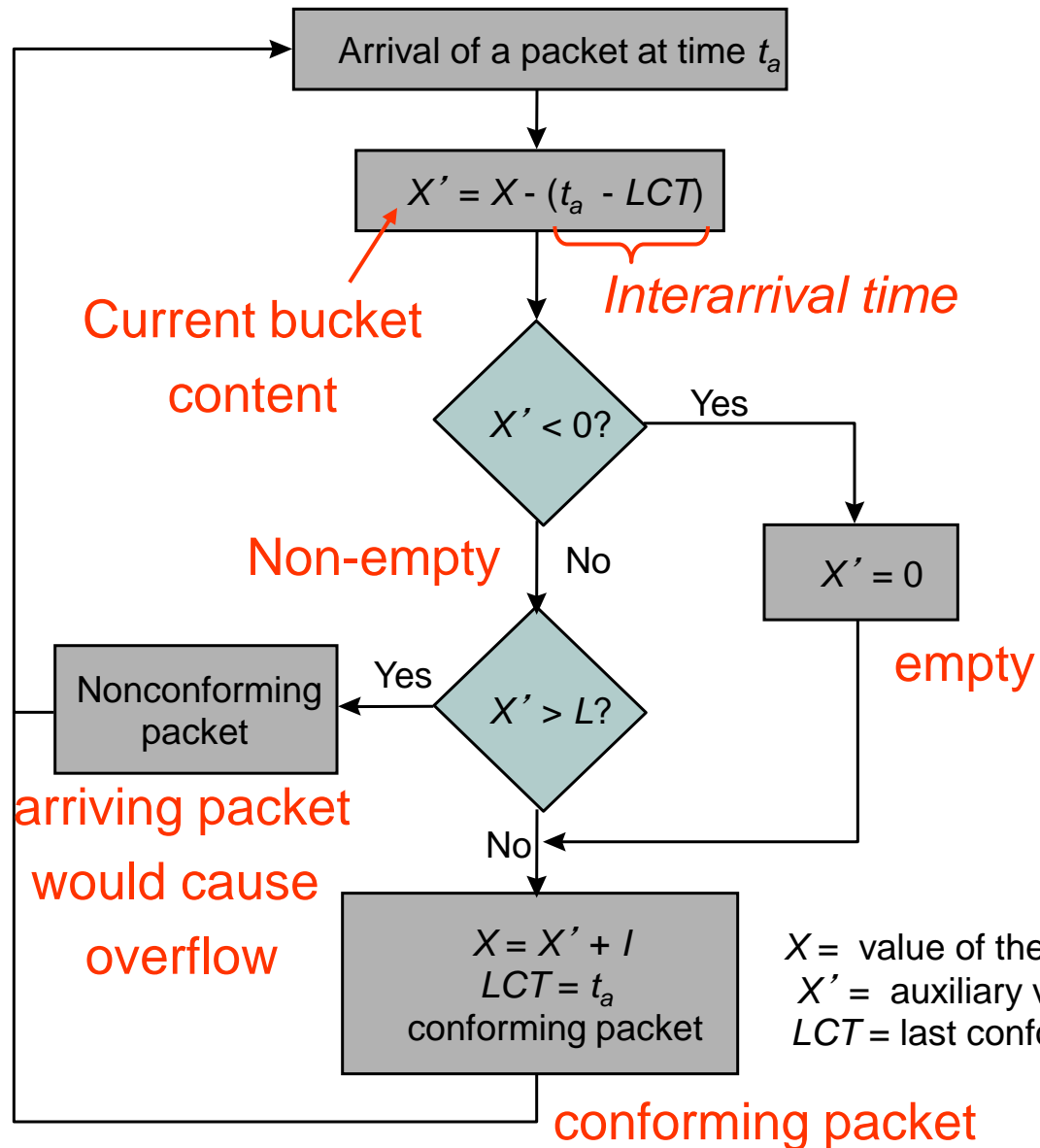
Bucket depth corresponds to maximum allowable burst arrival

1 packet per unit time  
Assume constant-length packet

Let  $X$  = bucket content at last conforming packet arrival

Let  $t_a$  – last conforming packet arrival time = depletion in bucket

# Leaky Bucket Algorithm



Depletion rate:  
1 packet per unit time

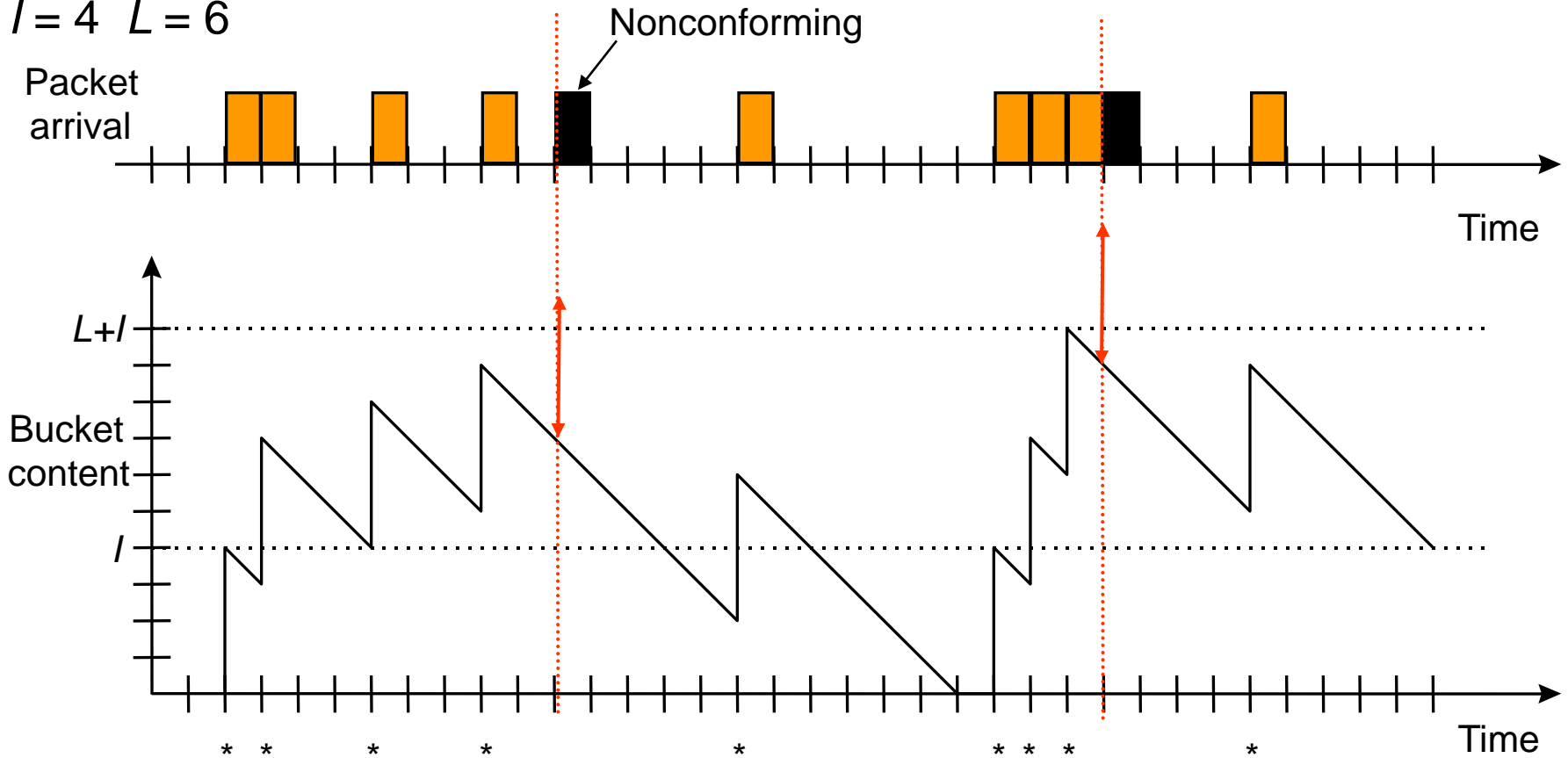
$b = L + l =$  Bucket Depth

$l =$  increment per arrival,  
nominal interarrival time

$X =$  value of the leaky bucket counter  
 $X' =$  auxiliary variable  
 $LCT =$  last conformance time

# Leaky Bucket Example

$I = 4$   $L = 6$

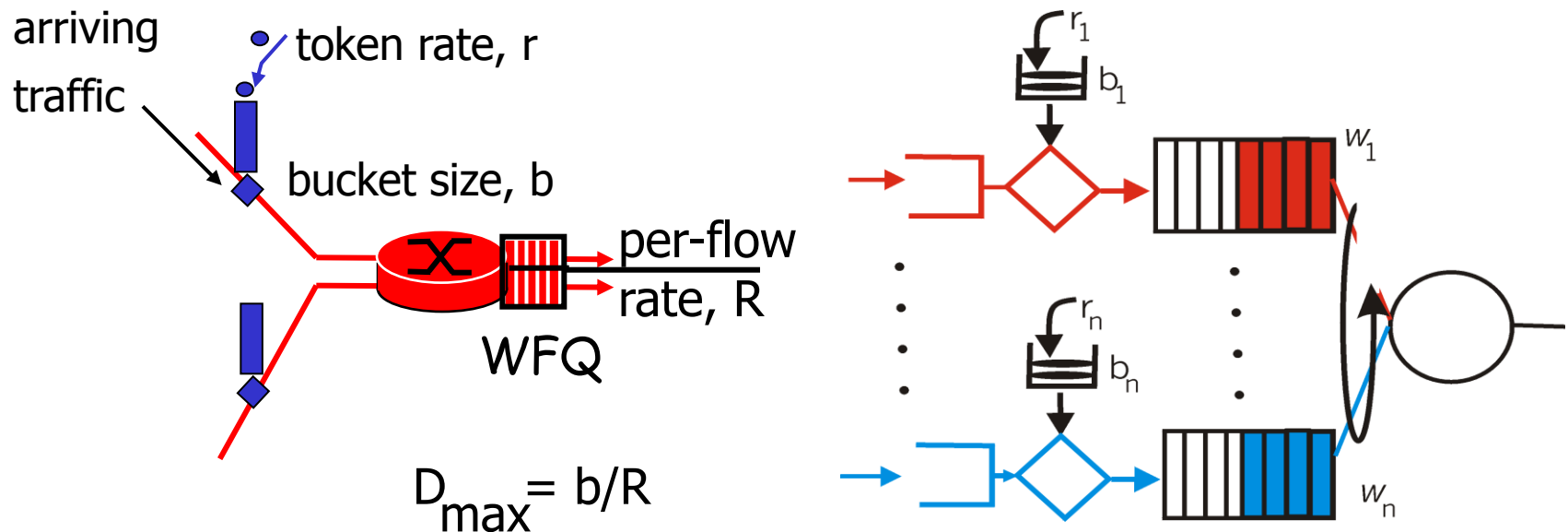


Non-conforming packets not allowed into bucket & hence not included in calculations



# Policing Mechanisms (more)

- Leaky bucket, WFQ combine to provide guaranteed upper bound on delay, i.e., *QoS guarantee!*



# Admission Control

## □ Users watch either one of two movies

- Star Wars (with the declared parameters)

$$A_1(s, t) \leq \min \{P_1(t - s), r_1(t - s) + b_1\}$$

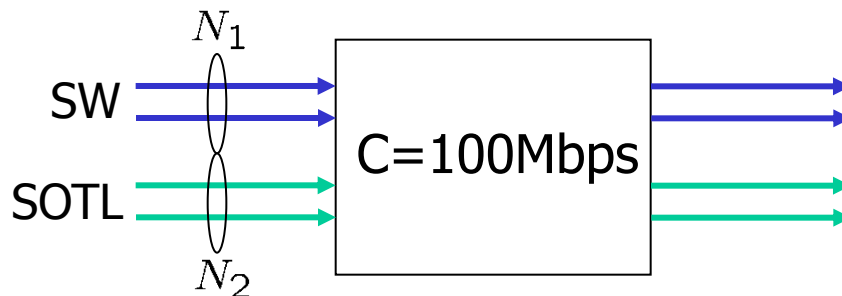
$$P_1 = 5Mbps, r_1 = 2Mbps, b_1 = .7Mb$$

- Silence of the Lambs (with the declared parameters)

$$A_2(s, t) \leq \min \{P_2(t - s), r_2(t - s) + b_2\}$$

$$P_2 = 4Mbps, r_2 = 1Mbps, b_2 = .5Mb$$

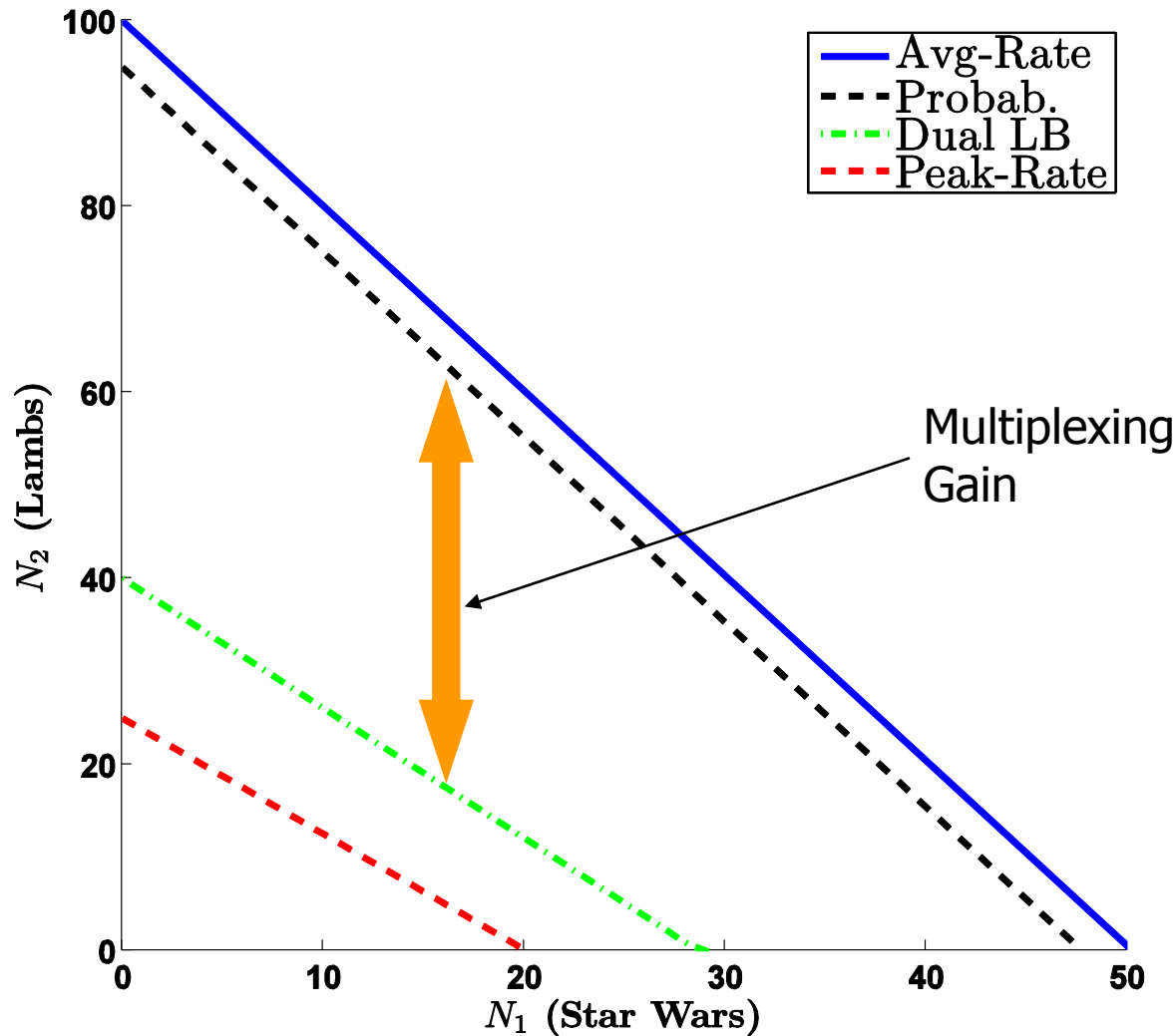
- ## □ *Concrete Problem:* How many users can be admitted at a $C=100Mbps$ link such that the delay for each user/movie is less than $d=200ms$ ?



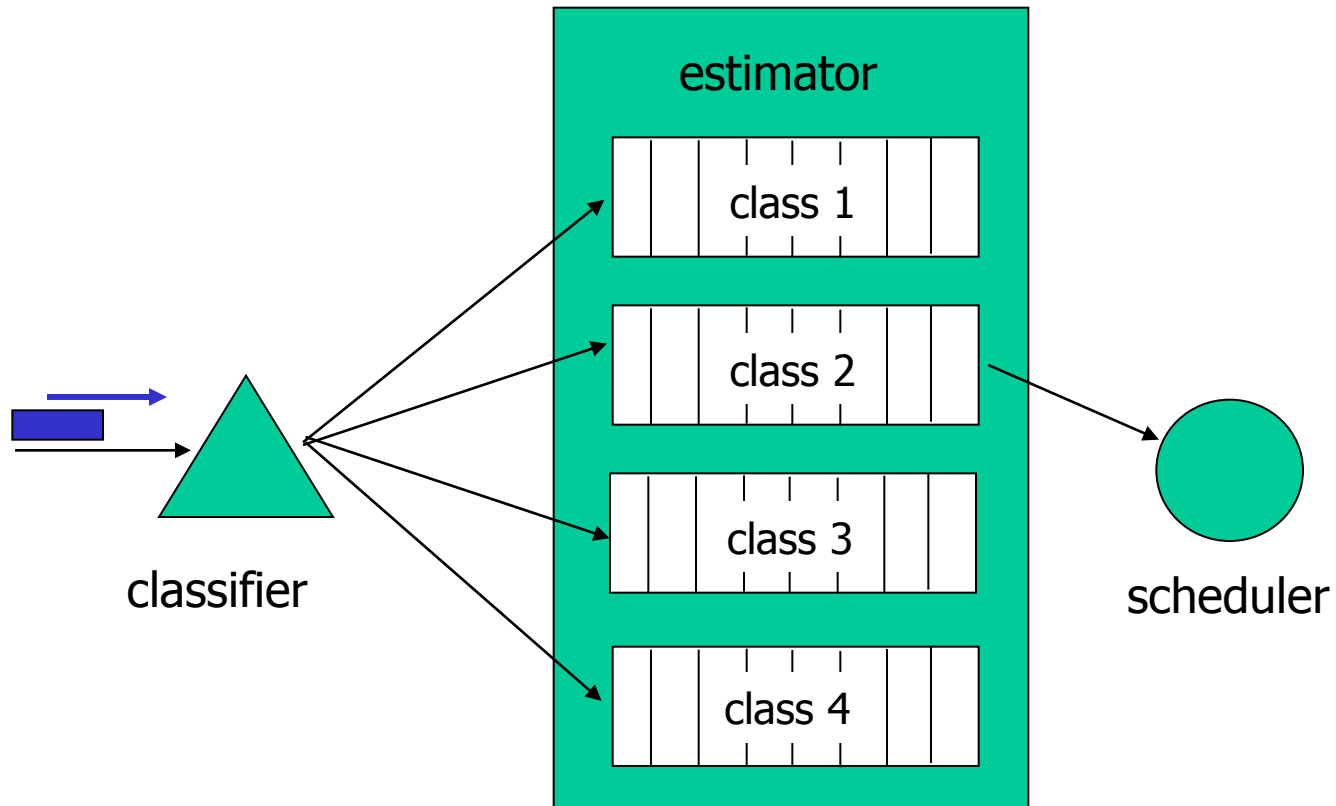
# Admission Control (Contd.)

- ❑ Peak rate admission control
  - Provides hard-guarantees
  
- ❑ Dual Leaky-bucket admission control
  - Provides hard-guarantees
  
- ❑ Statistical admission control
  - Provides soft-guarantees
  
- ❑ Average rate admission control
  - Only qualitative guarantees (e.g., delay is always finite)

# Admission Control. Numerical Example



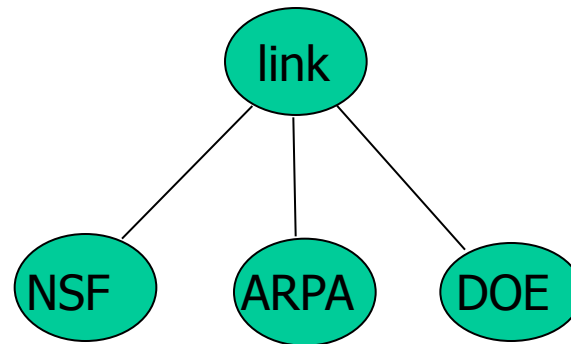
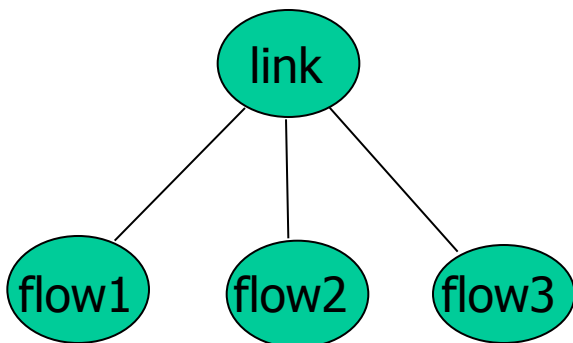
# General Model of Class-based Link Scheduling



# Link sharing among classes

The question: sharing link among classes of packets in times of overload

- isolation among classes
- sharing between classes when link not fully used



# Link Scheduling Framework

- ❑ each class guaranteed some share (fraction) of bandwidth (over suitable time interval) *if needed*
- ❑ *Use it or lose it*: unused bandwidth should be used by others who can use it

**Under-limit class:** used less than guaranteed share

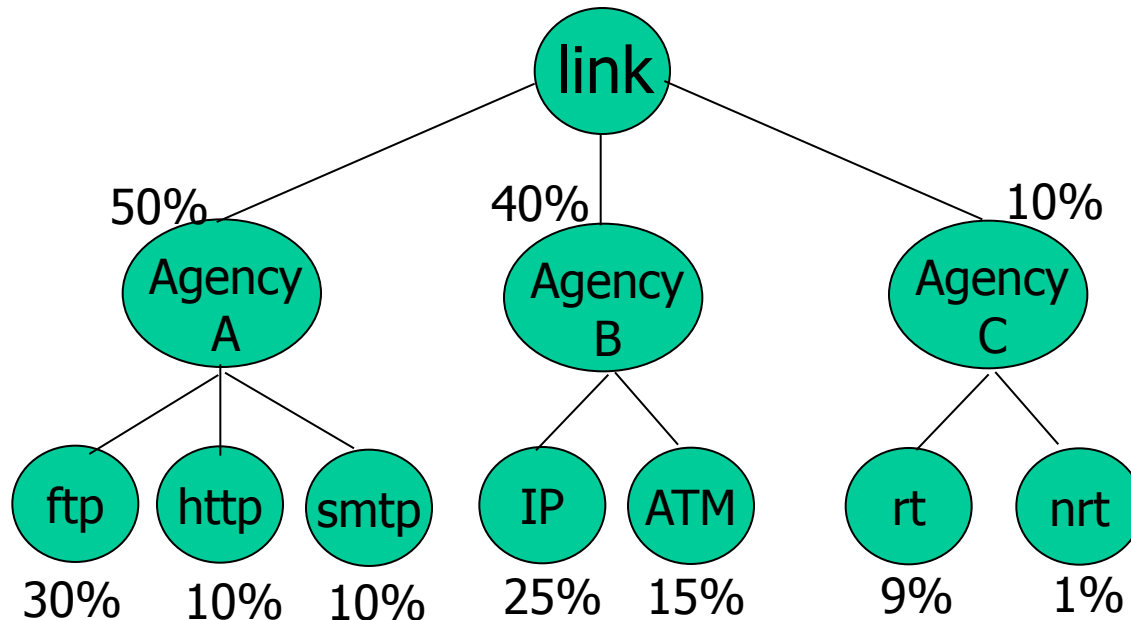
**Over-limit class:** used more than guaranteed share. Not a bad thing if not needed by others!

**At-limit class**

**Unsatisfied class:** has a persistent backlog and is under limit ☹️ (persistent backlog intentionally not defined)

**Satisfied:** not unsatisfied 😊

# Link Scheduling Hierarchies



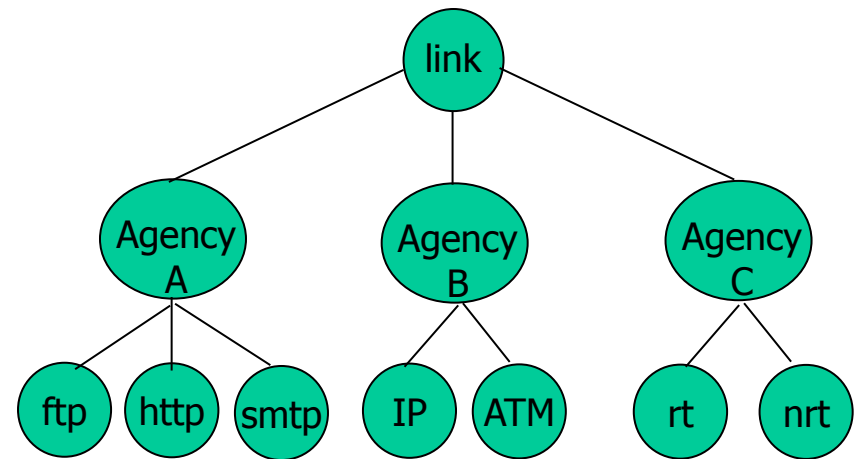
Classes may be divided into subclasses, which can then further divide class bandwidth according to link scheduling guidelines



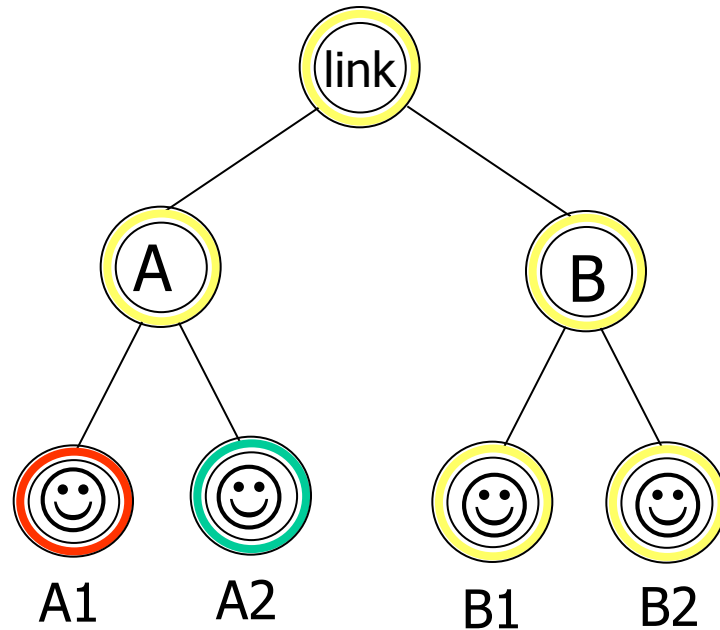
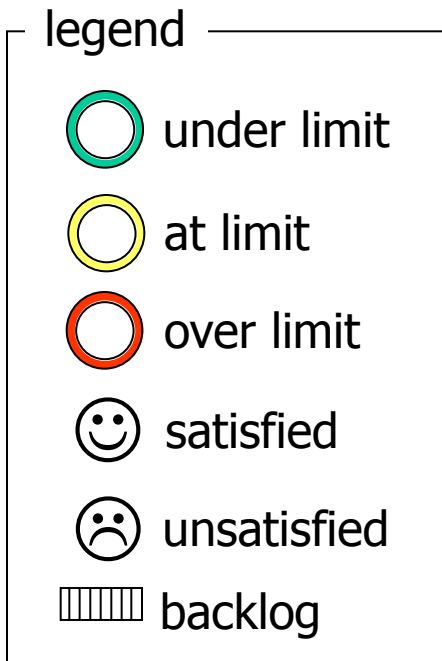
# Link Scheduling Guidelines

A class can continue unregulated if one of the following hold:

- ❑ the class is **not overlimit**
- ❑ the class has a **not-overlimit ancestor** at level  $i$ , and there are **no unsatisfied classes** in link sharing structure at levels lower than  $i$



# Example 1




Q; which classes need to be regulated?

# Example 2

legend

 under limit

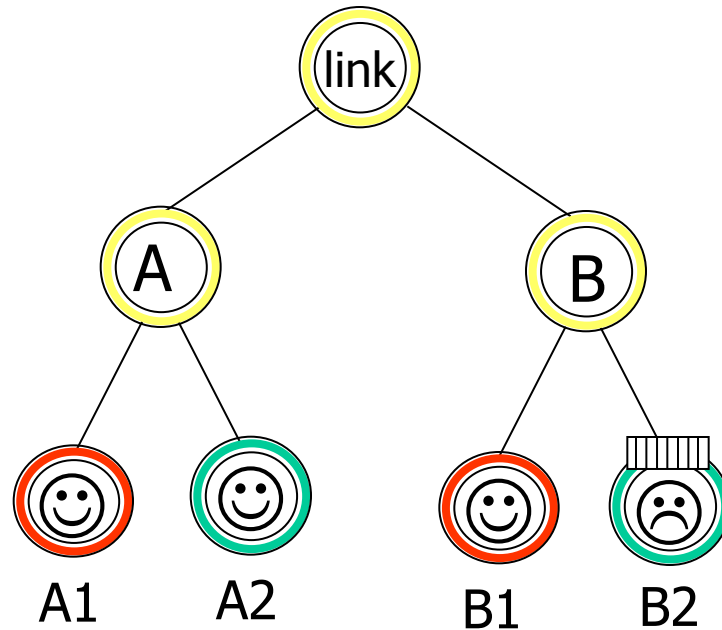
 at limit

 over limit

 satisfied

 unsatisfied

 backlog




Q; which classes need to be regulated?

# Example 3

legend

 under limit

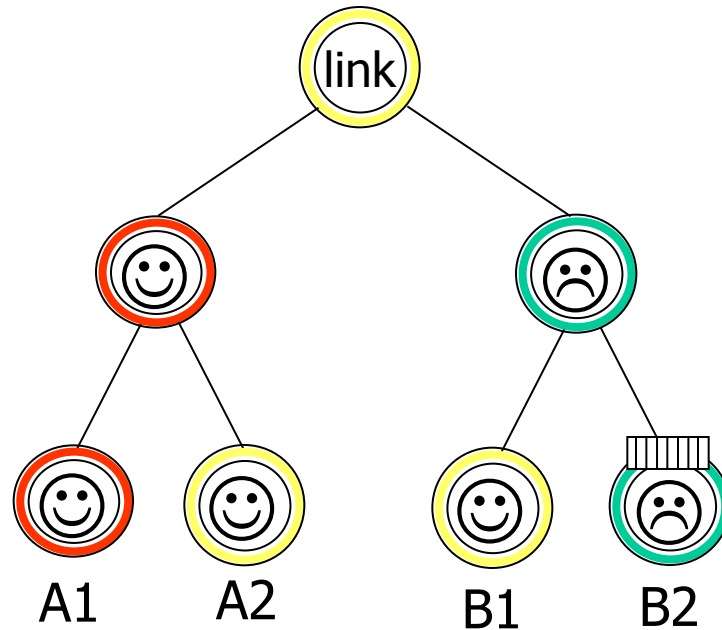
 at limit

 over limit

 satisfied

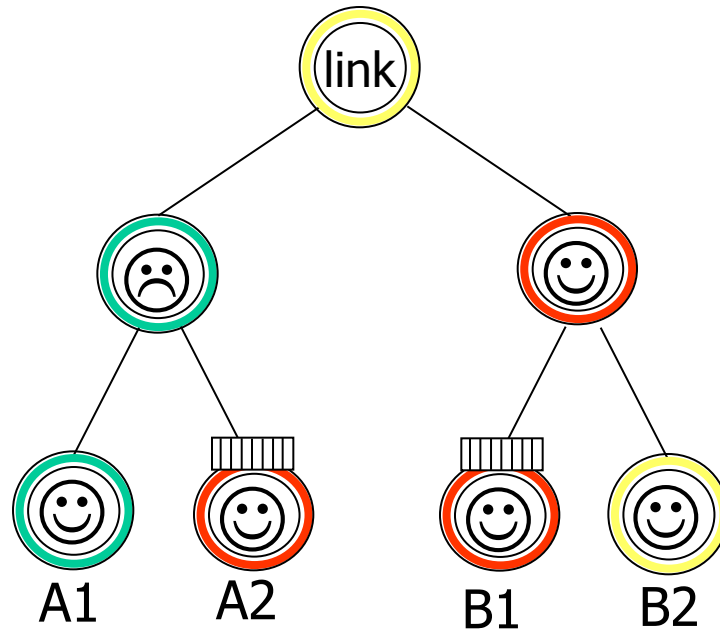
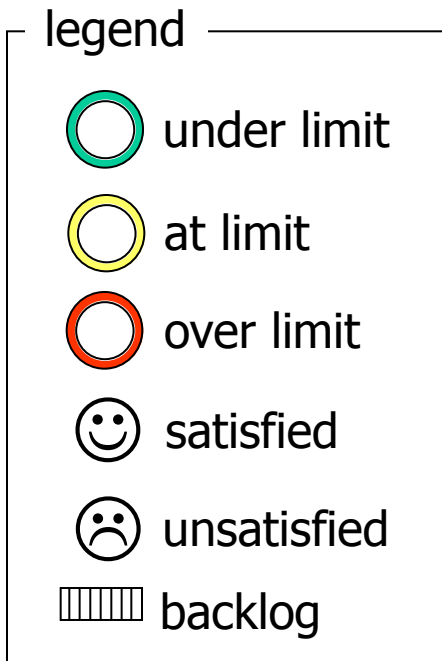
 unsatisfied

 backlog



Q; which classes need to be regulated?

# Example 4



Q; which classes need to be regulated?

# Link Sharing Summary

- ❑ Timescales over which persistent backlog and under/over limit defined are crucial
- ❑ Provide “rationale” basis for determining who is available to be scheduled in a multi-class, multi-objective system (elegantly)
- ❑ Devil is in the details
  - E.g., malicious users, parallel flows ...

# QoS in the Internet. Part I.

## IETF Integrated Services

- ❑ Architecture for providing QOS guarantees in IP networks for individual application sessions
- ❑ Resource reservation: routers maintain state info of allocated resources, QoS req' s
- ❑ 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 requirement
  - **R-spec**: defines the QOS being requested
- ❑ characterize traffic it will send into network
  - **T-spec**: defines traffic characteristics
- ❑ signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  - **RSVP**



# Intserv QoS: Service models

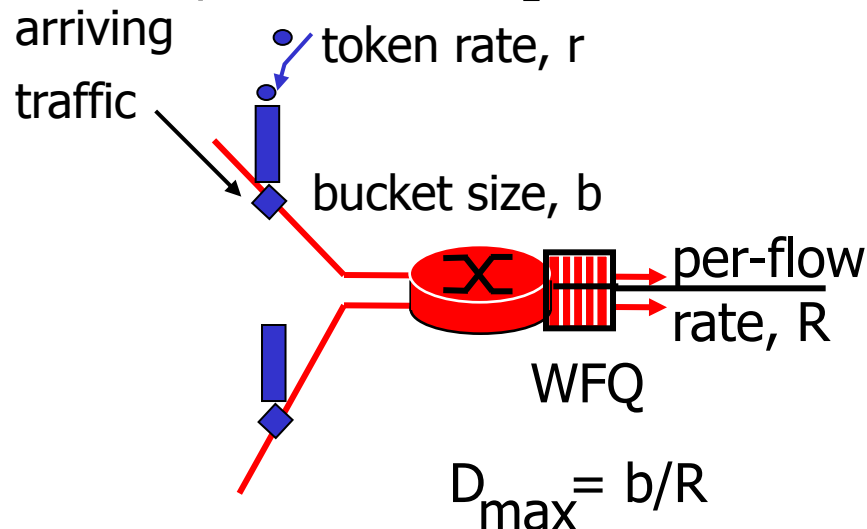
[rfc2211, rfc 2212]

## Guaranteed service:

- ❑ worst case traffic arrival:  
leaky-bucket-policed  
source
- ❑ simple (mathematically  
provable) *bound* on delay  
[Parekh 1992, Cruz 1988]

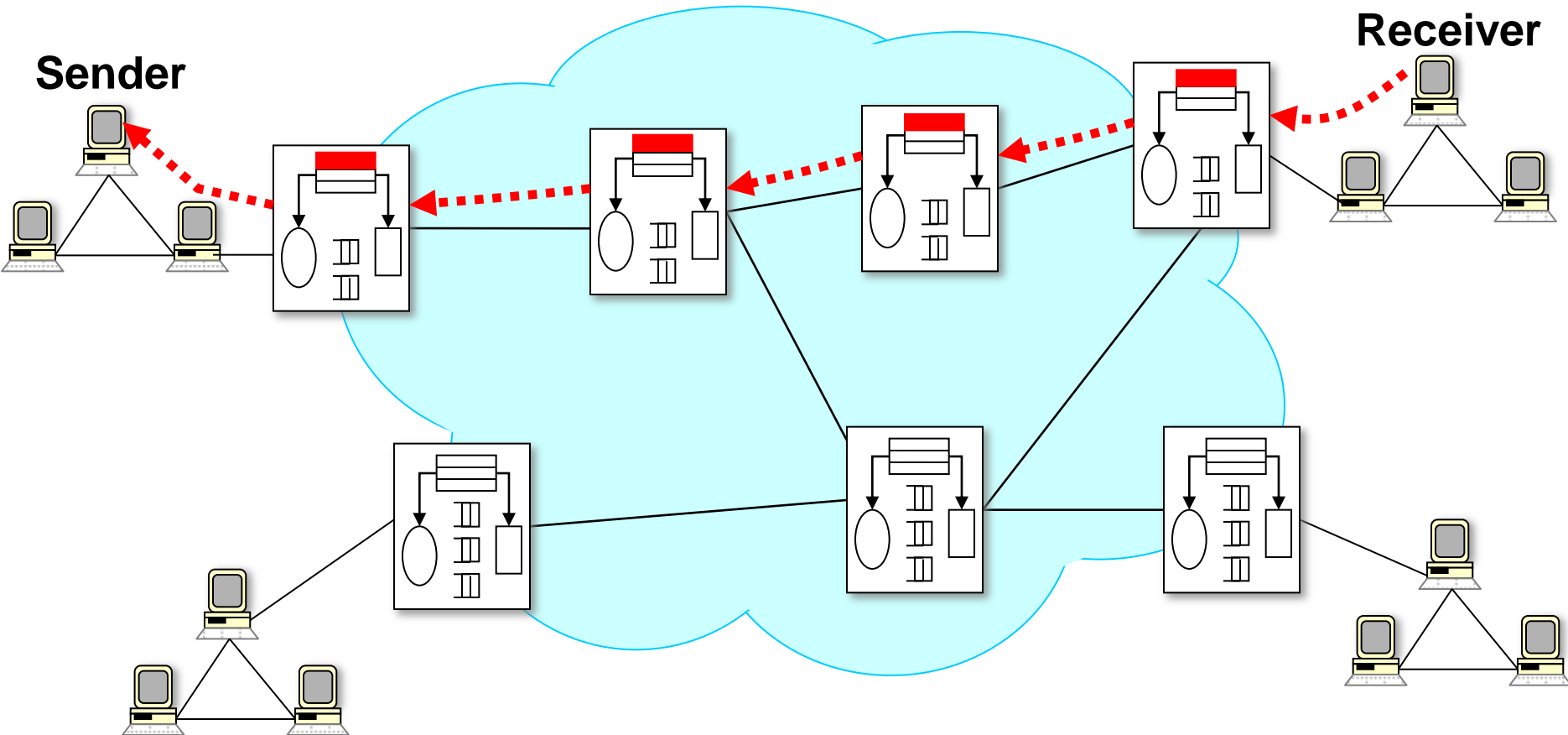
## Controlled load service:

- ❑ "a quality of service  
closely approximating the  
QoS that same flow  
would receive from an  
unloaded network  
element."



# Integrated Services Example

- Install per flow state

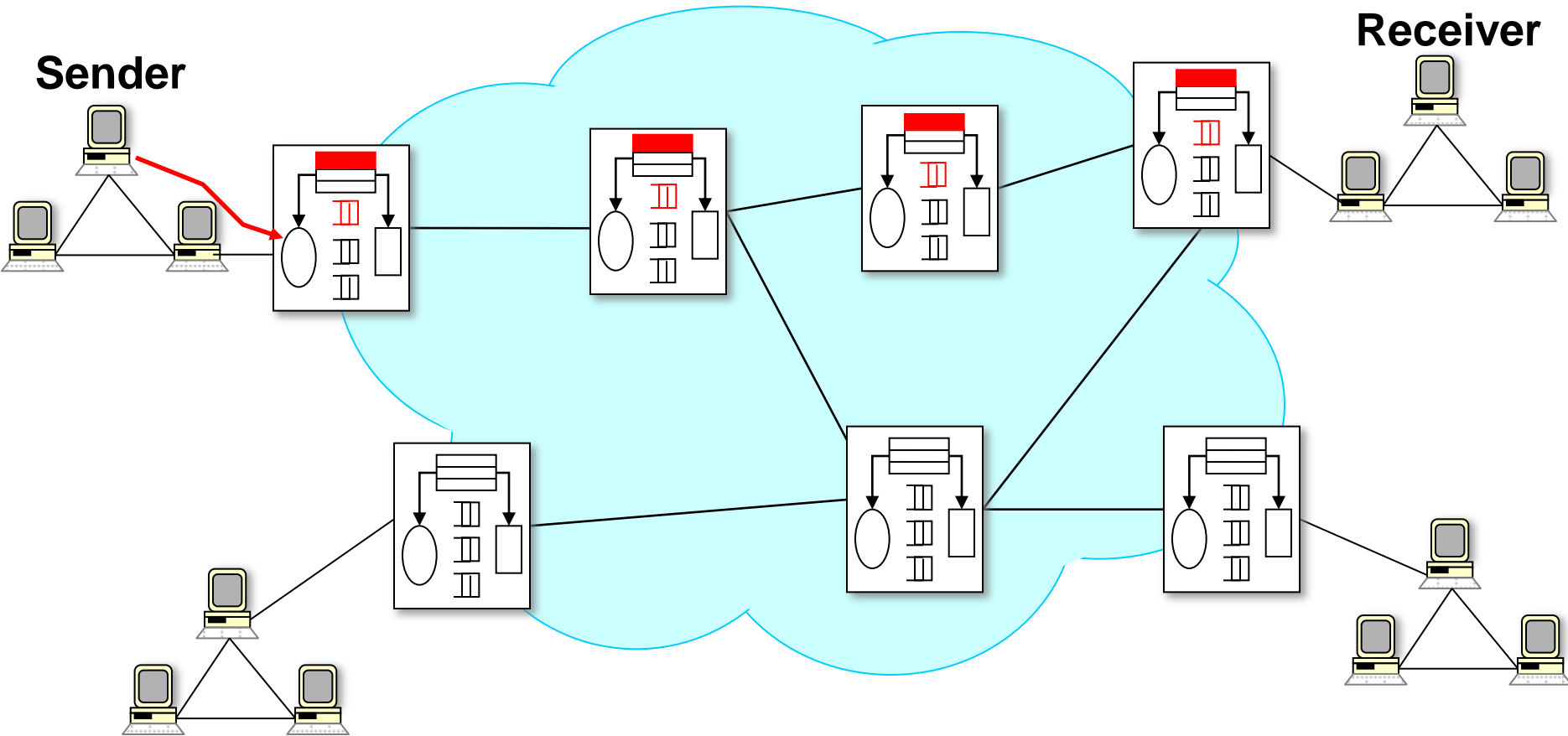


# Recall RSVP

- ❑ Signaling protocol for establishing per flow (soft) state
- ❑ Carry resource requests from hosts to routers
- ❑ Collect needed information from routers to hosts
- ❑ At each hop
  - Consult admission control and policy module
  - Set up admission state or informs the requester of failure
- ❑ Decouples routing from reservation

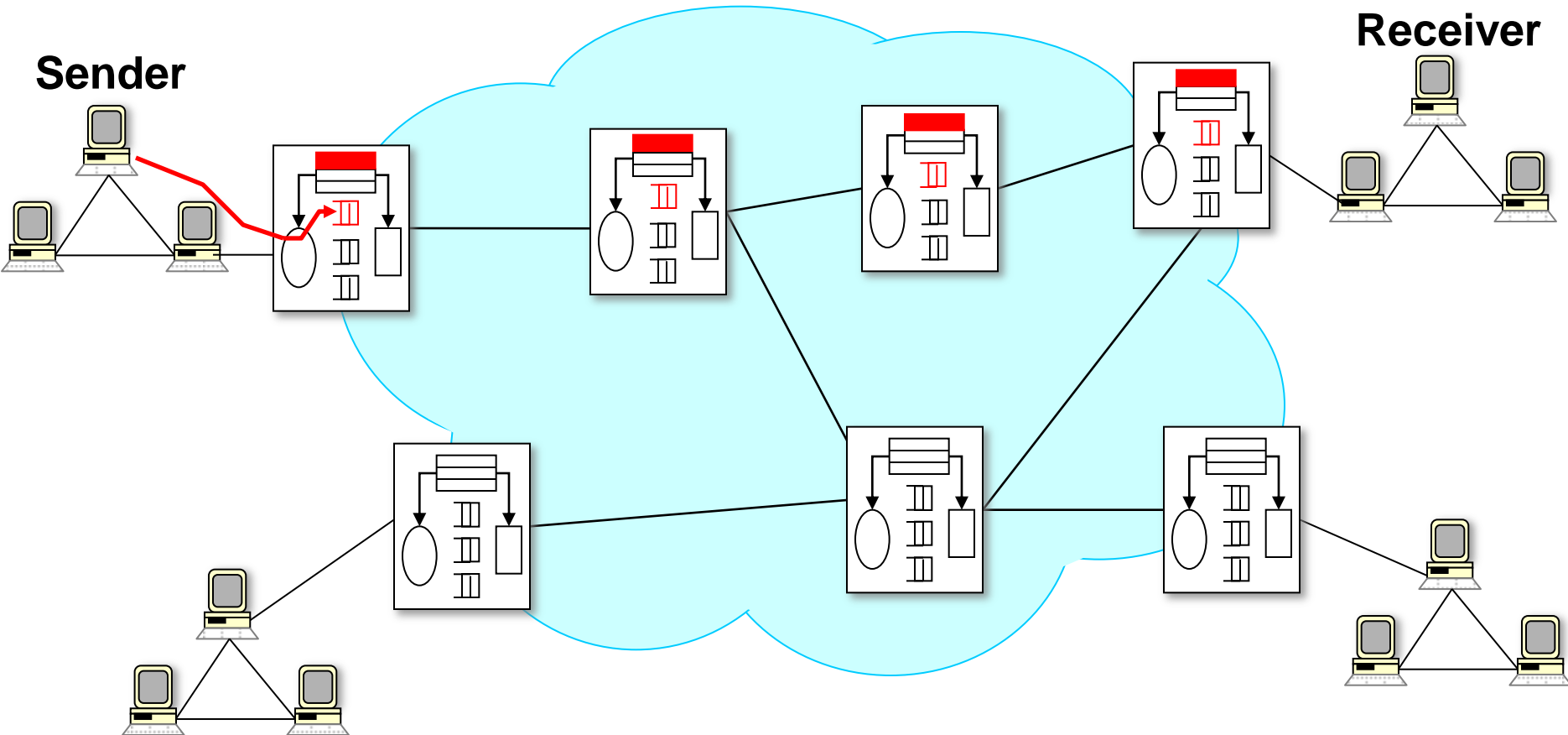
# Integrated Services Example: Data Path

## □ Per-flow classification



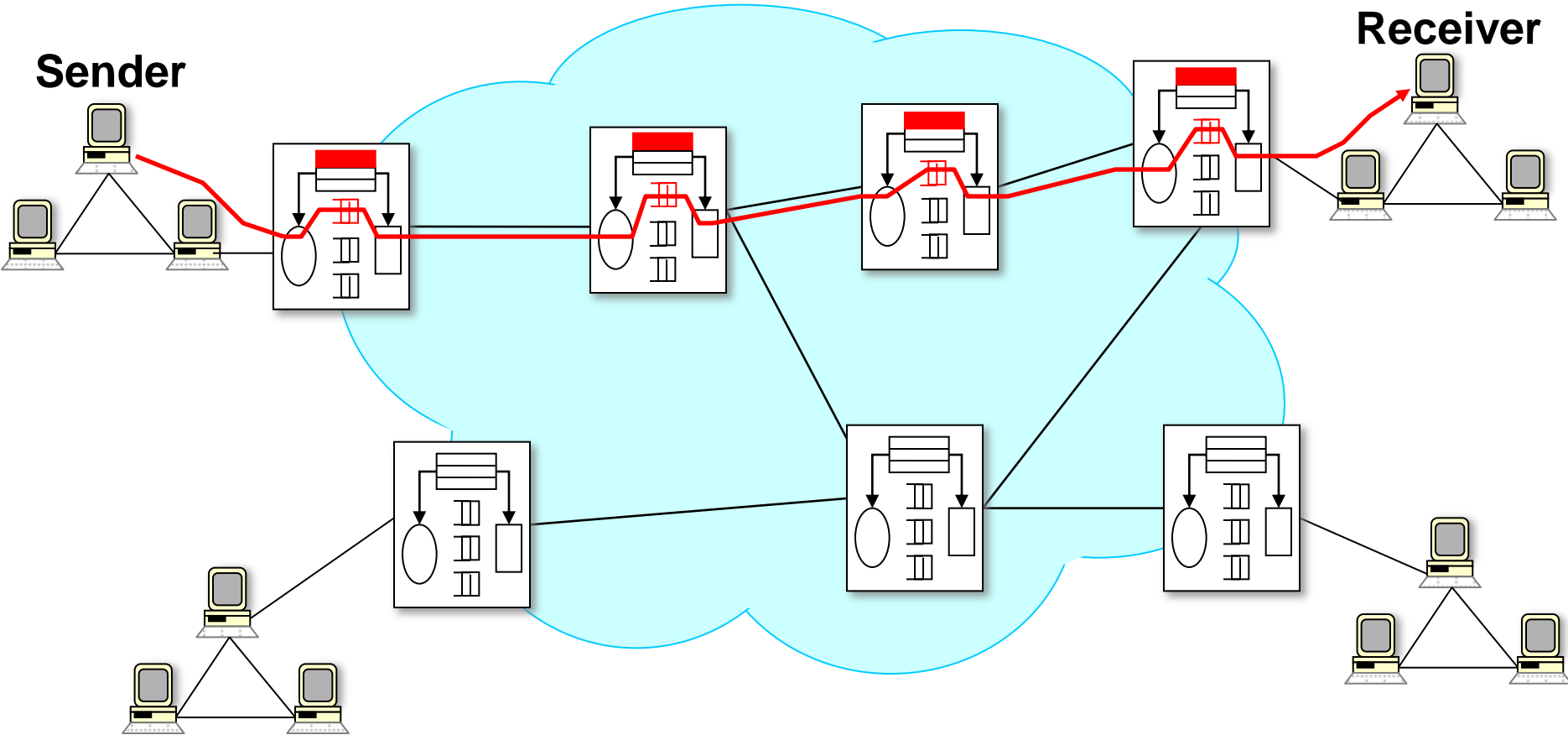
# Integrated Services Example: Data Path

- Per-flow buffer management

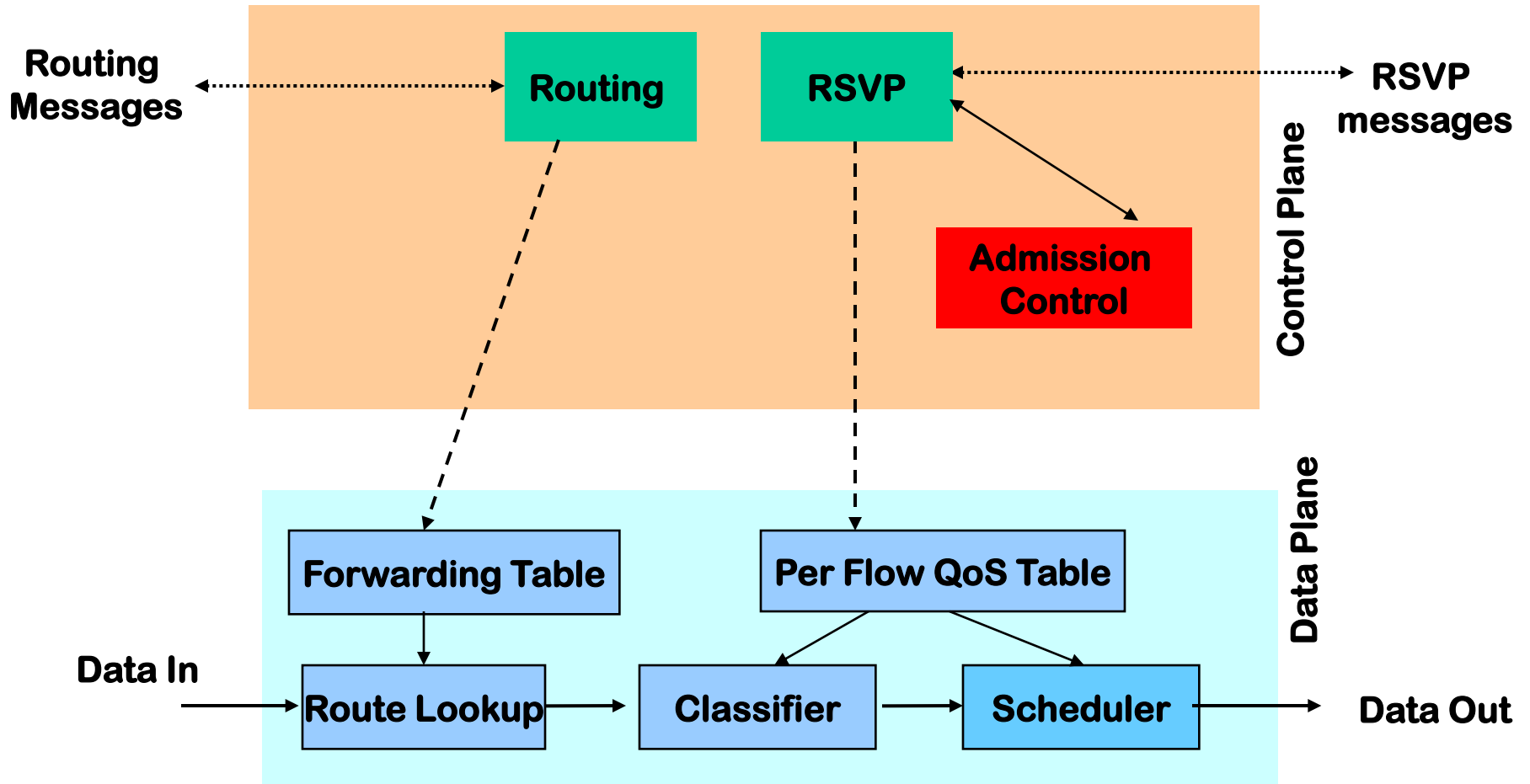


# Integrated Services Example

## □ Per-flow scheduling



# How Things Fit Together



# QoS in the Internet. Part II.

## IETF Differentiated Services

- ❑ Want “qualitative” service classes
  - “behaves like a wire”
  - relative service distinction: Platinum, Gold, Silver
- ❑ *Scalability*: Simple functions in network core, relatively complex functions at edge routers (or hosts)
  - signaling, maintaining per-flow router state difficult with large number of flows
- ❑ Don’ t define service classes, provide functional components to build service classes



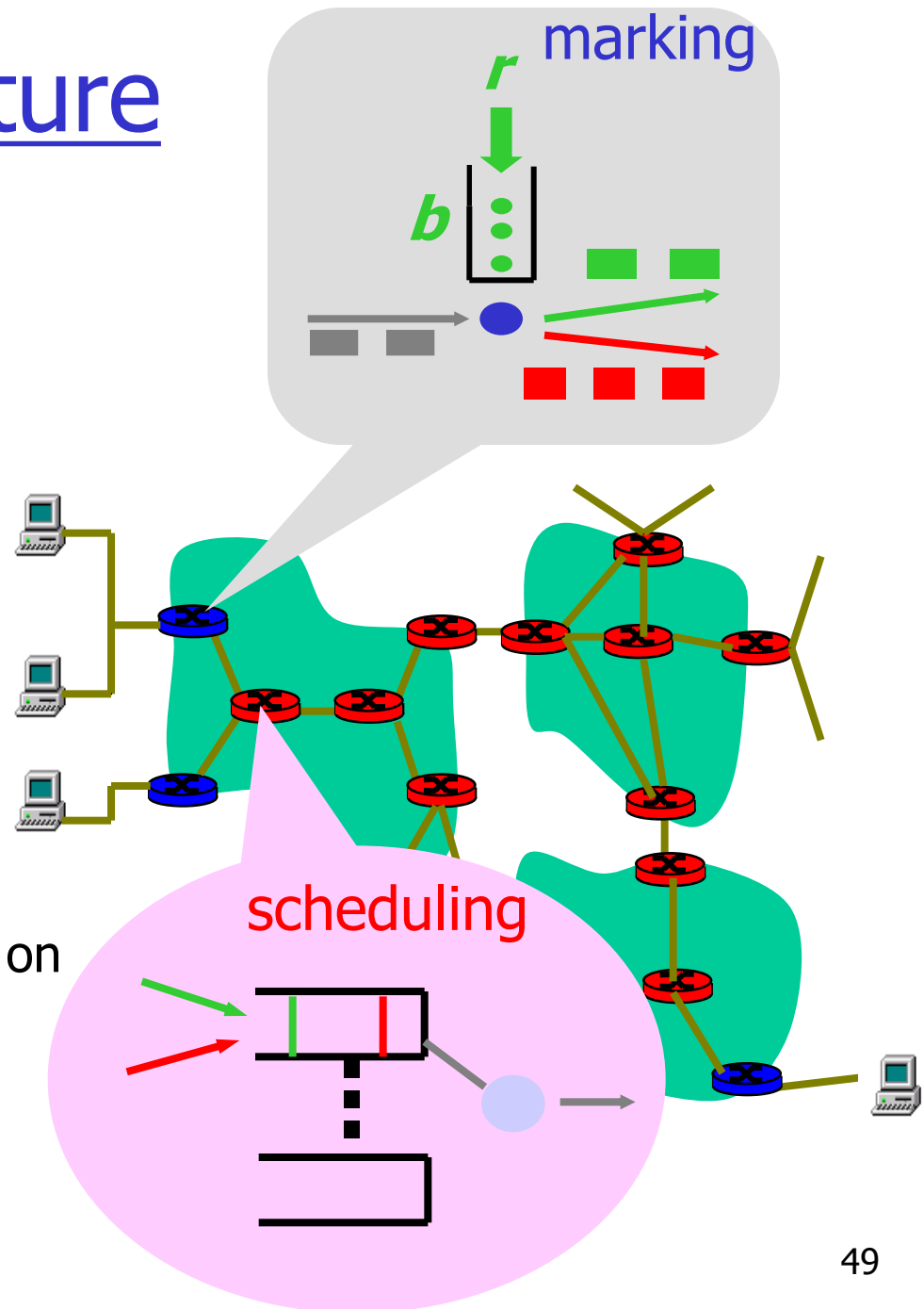
# Diffserv Architecture

## Edge router:

- per-flow traffic management
- marks packets as **in-profile** and **out-profile**

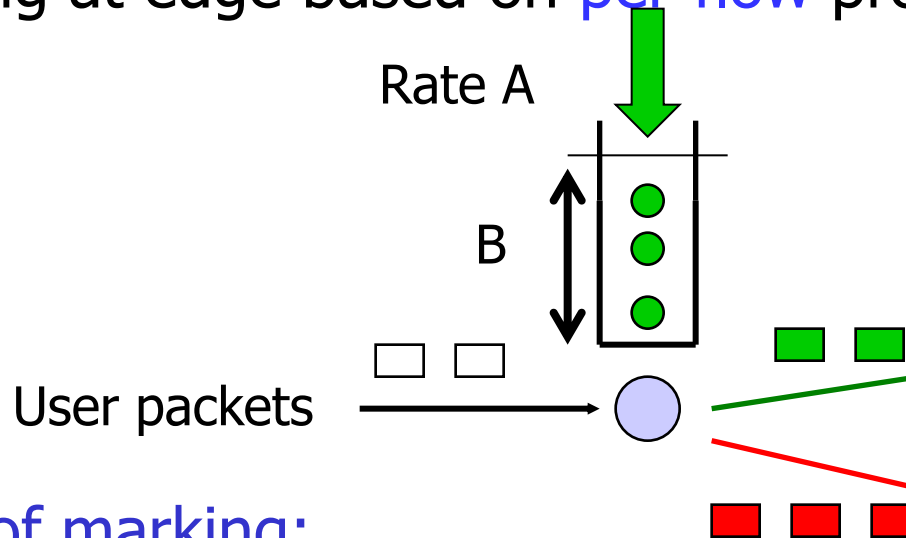
## Core router:

- **per class** traffic management
- buffering and scheduling based on **marking** at edge
- preference given to **in-profile** packets



# Edge-router Packet Marking

- 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

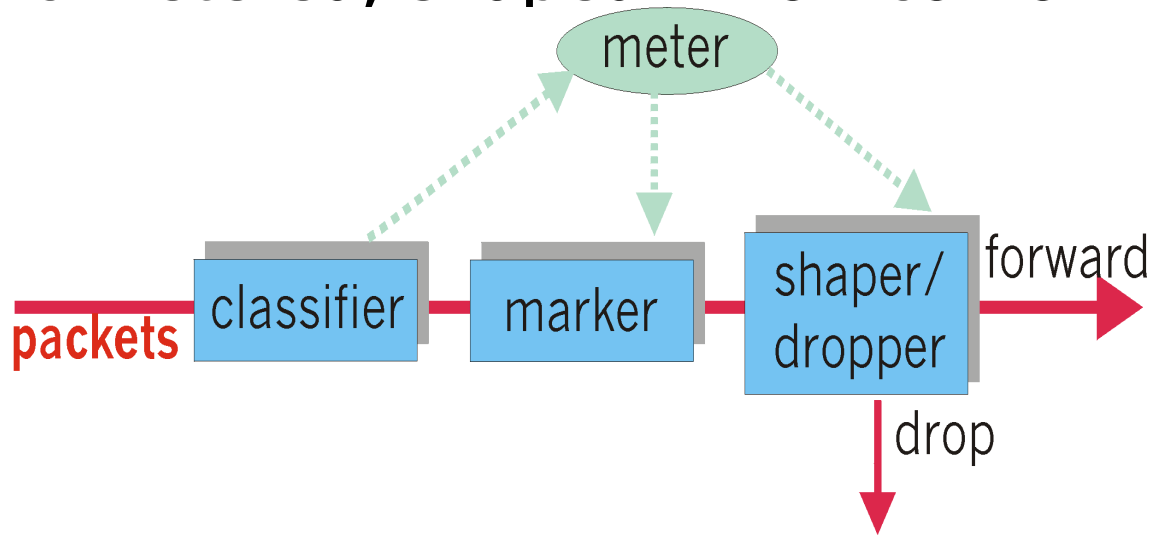
- ❑ Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- ❑ 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- ❑ 2 bits are currently unused



# 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 if non-conforming



# Forwarding – Per Hop Behavior (PHB)

- ❑ PHB result in a different observable (measurable) forwarding performance behavior
- ❑ PHB does not specify what mechanisms to use to ensure required PHB performance behavior
- ❑ 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:** pkt departure rate of a class equals or exceeds specified rate
  - logical link with a minimum guaranteed rate
- **Assured Forwarding:** 4 classes of traffic
  - each guaranteed minimum amount of bandwidth
  - each with three drop preference partitions