Network Support for Quality of Service (QoS)

CS 352, Lecture 24

http://www.cs.rutgers.edu/~sn624/352-S19

Srinivas Narayana

(heavily adapted from slides by Prof. Badri Nath and the textbook authors)

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Review: Streaming multimedia

- Watching media prerecorded at servers at multiple quality levels, ex: Netflix
- Client downloads an initial portion and starts viewing/listening
- Need continuous playout:
  - Manage network delays through a client buffer
- Playout rate must be not larger than average download rate
- Predominant model: Dynamic Adaptive Streaming over HTTP
  - DASH video is divided into chunks
  - Each chunk can be retrieved with an independent bitrate and from an independent (CDN) location

Review: Conversational multimedia

- Two parties making a real-time conversation, ex: Skype
- Important to bound playout delays & adapt to loss
  - More than ~400 ms audio delay provides for a very poor experience
- Fixed and adaptive playout delays at the granularity of "talk spurts"
- Retransmissions not really effective to conceal loss
  - Forward error correction mechanisms
- Relay-based call routing: used by Skype
  - Useful to overcome NATs
  - But need extra infrastructure to make it work

Network support

How can the network benefit multimedia network transfers?

Network support for Multimedia

- A best effort Internet architecture does not offer any guarantees on delay, bandwidth, and loss
  - Network may drop, reorder, corrupt packets
  - Network may treat transfers randomly regardless of their “importance”
- However, multimedia apps require delay and loss bounds
- How to provide quality of service (QoS) for multimedia applications through network mechanisms?
  - Provision enough resources: make the best of best effort service
  - Mechanisms to handle traffic differently based on importance
- Result: specific protocols and architectures for QoS

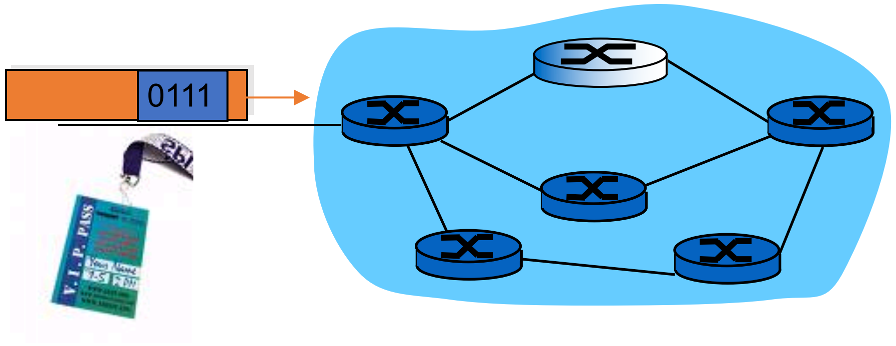
Dimensioning best effort networks

- *approach:* deploy enough link capacity so that congestion doesn't occur, multimedia traffic flows without delay or loss
  - low complexity of network mechanisms (use current "best effort" network)
  - high bandwidth costs
- challenges:
  - *network dimensioning:* how much bandwidth is "enough?"
  - *estimating network traffic demand:* needed to determine how much bandwidth is "enough" (for that much traffic)

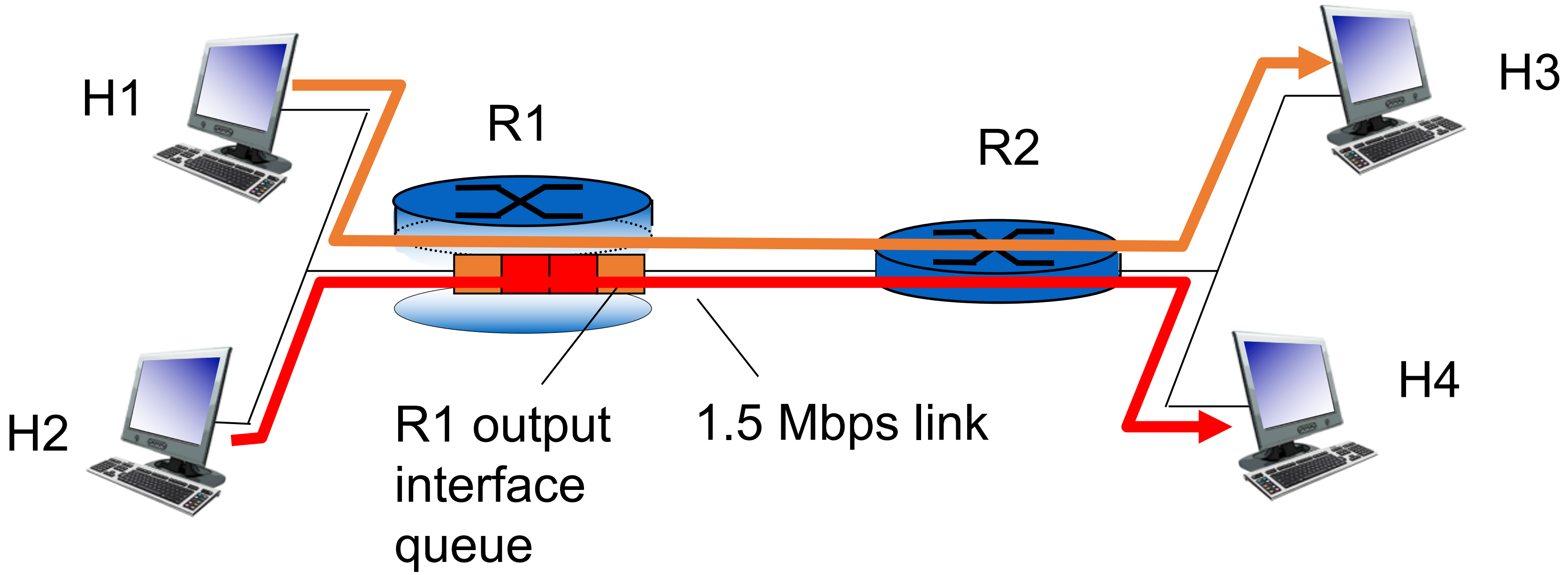
Providing multiple classes of service

- thus far: making the best of best effort service
  - one-size fits all service model
- alternative: multiple classes of service
  - partition traffic into classes
  - network treats different classes of traffic differently (analogy: VIP service versus regular service)

- granularity: differential service among multiple classes, not among individual connections
- history: ToS bits

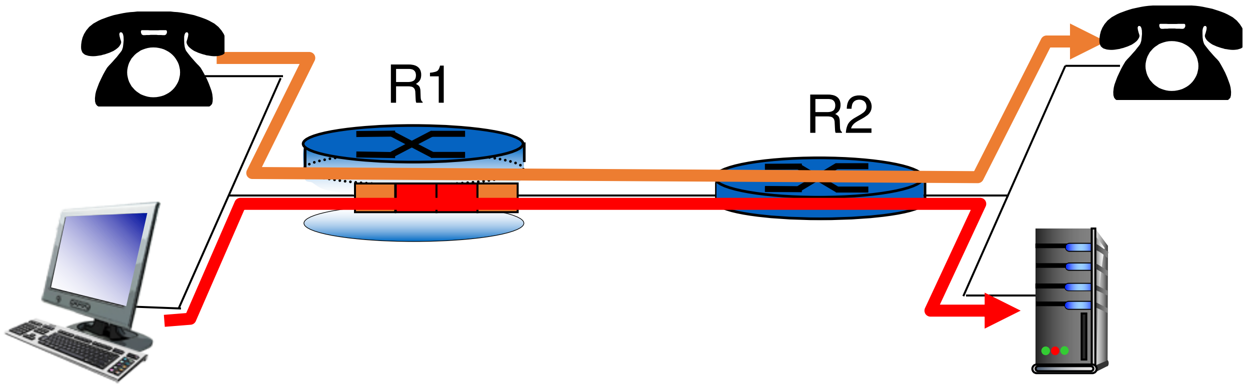


Multiple classes of service: scenario



Scenario 1: mixed HTTP and VoIP

- example: 1Mbps VoIP, HTTP share 1.5 Mbps link.
  - HTTP bursts can congest router, cause audio loss
  - want to give priority to audio over HTTP

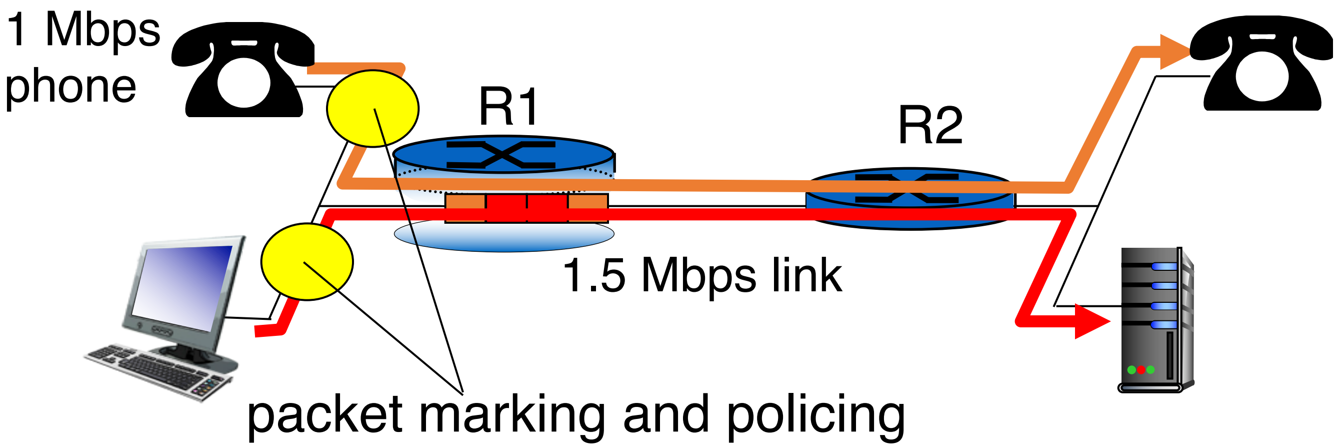


Principle 1

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 (VoIP sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- *marking*, *policing* at network edge

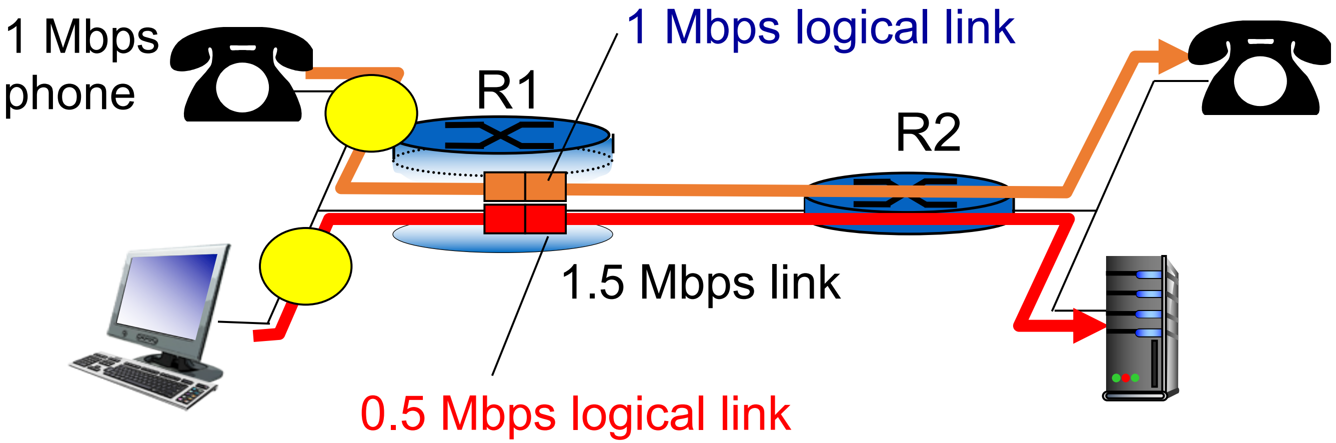


Principle 2

provide protection (isolation) for one class from others

Principles for QOS guarantees (more)

- allocating *fixed* (non-sharable) bandwidth to flow: *inefficient* use of bandwidth if flows doesn't use its allocation



Principle 3

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

Review: Queues on routers

- Where does contention between the two connections happen in the network earlier?
  - Contention == increased queueing (and queuing delays), possibility of loss

- Where are the queues located on the routers?

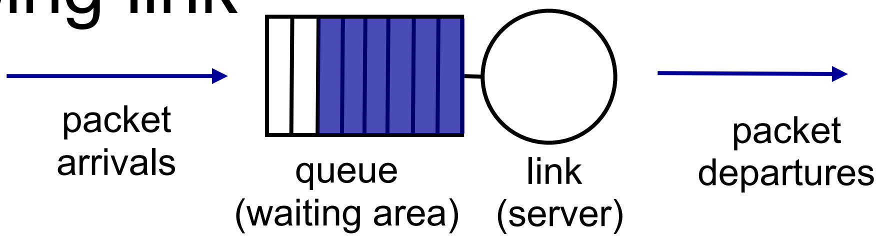
- Principle: It is useful to provide quality of service by managing how packets traverse router queues, ie: packet scheduling

Packet scheduling for QoS

Shaping and policing

Review: Packet scheduling for QoS

- *packet scheduling:* choose next queued packet to send on outgoing link



- previously covered under the networking layer
  - FCFS: first come first served
  - simply multi-class priority
  - round robin
  - weighted fair queueing (WFQ)

Packet scheduling mechanisms

Goal: provide isolation between different kinds of traffic by limiting traffic not to 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 the interval length: 100 packets per sec or 6000 packets per min have same average!
- *peak rate:* e.g., 6000 pkts per min (ppm) avg.; 1500 ppm peak rate
- *(max.) burst size:* max number of pkts sent consecutively (with no intervening idle)

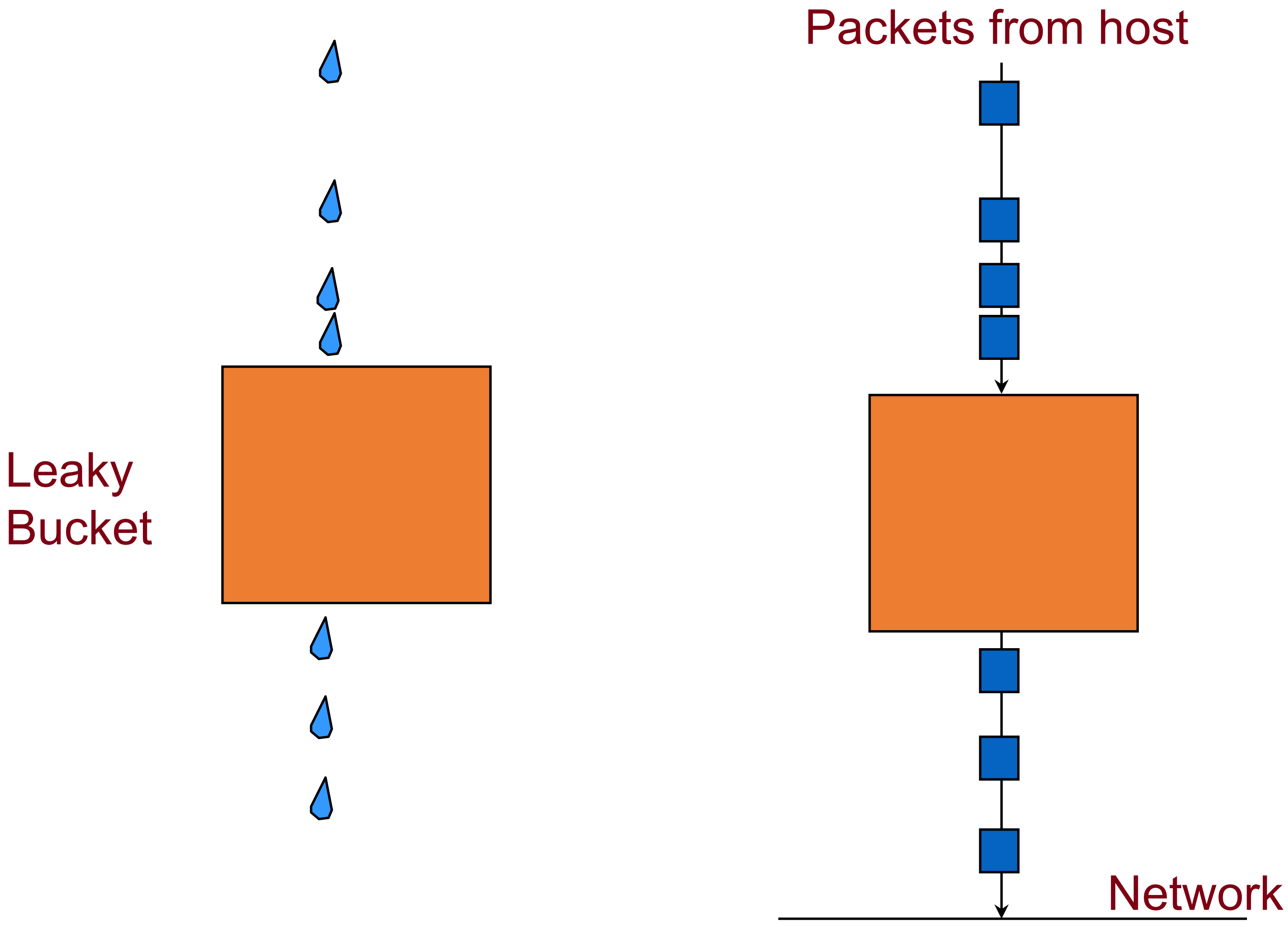
QoS mechanism (1): Leaky Bucket

- Used in conjunction with resource reservation to police the host's reservation

- At the host-network interface, allow packets into the network at a constant rate

- Packets may be generated in a bursty manner, but after they pass through the leaky bucket, they enter the network evenly spaced

Leaky Bucket: Analogy



Shaping traffic with leaky buckets

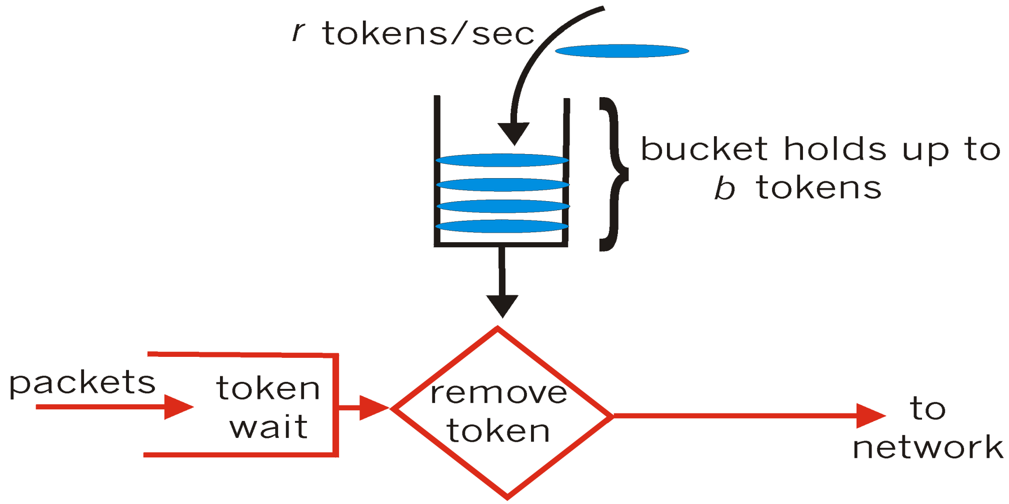
- The leaky bucket is a traffic shaper: It changes the characteristics of packet stream
- Traffic shaping makes traffic more manageable and more predictable
- Usually, a system/network administrator would set the rate at which packets may be sent through the leaky bucket
- Administrator also sets up policies to map any connection that started up to a leaky bucket (and rate) of its own

Issues with a leaky bucket

- In some cases, we may want to allow short bursts of packets to enter the network without smoothing them out
- For a leaky bucket, average rate == peak rate
- But sometimes, we can allow the peak rate to be higher
  - Ex: a short transfer that only has a few packets
- For this purpose we use a token bucket, which is an enhanced leaky bucket

QoS mechanism (2): Token Bucket

token bucket: limit input to specified *burst size* and *average rate*



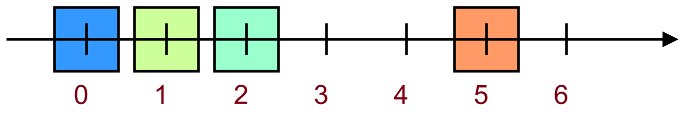
- bucket can hold b tokens
- tokens generated at rate *r tokens/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to $(r * t + b)$*

Token Bucket: Notes

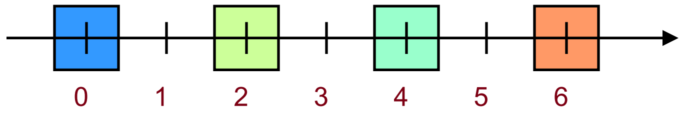
- The bucket holds tokens instead of packets
- Tokens are generated and placed into the token bucket at a constant rate
- When a packet arrives at the token bucket, it is transmitted if there is a token available.
- Otherwise it may be buffered until a token becomes available
  - Or even dropped: in which case we call it a policer
  - The Internet is full of traffic policers
- The token bucket has a fixed size, so when it becomes full, subsequently generated tokens are discarded

Token Bucket vs. Leaky Bucket

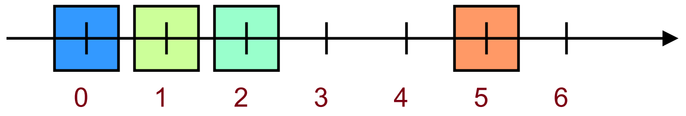
Case 1: Short burst arrivals



Arrival time at bucket



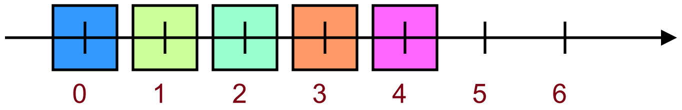
Departure time from a leaky bucket
Leaky bucket rate = 1 packet / 2 time units
Leaky bucket size = 4 packets



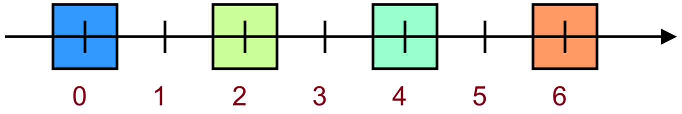
Departure time from a token bucket
Token bucket rate = 1 tokens / 2 time units
Token bucket size = 2 tokens

Token Bucket vs. Leaky Bucket

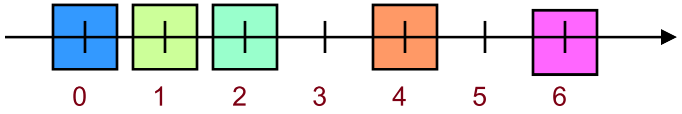
Case 2: Large burst arrivals



Arrival time at bucket



Departure time from a leaky bucket
Leaky bucket rate = 1 packet / 2 time units
Leaky bucket size = 2 packets



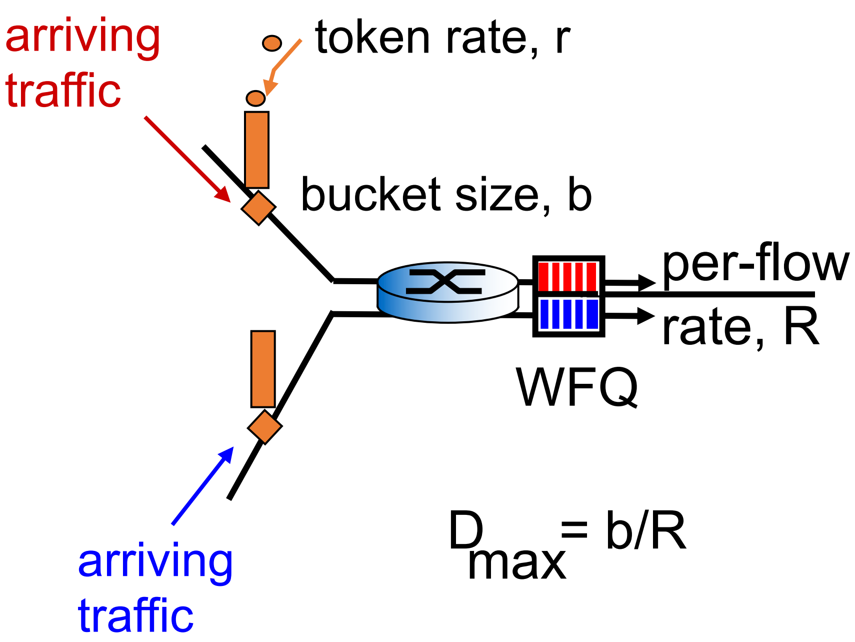
Departure time from a token bucket
Token bucket rate = 1 token / 2 time units
Token bucket size = 2 tokens

??

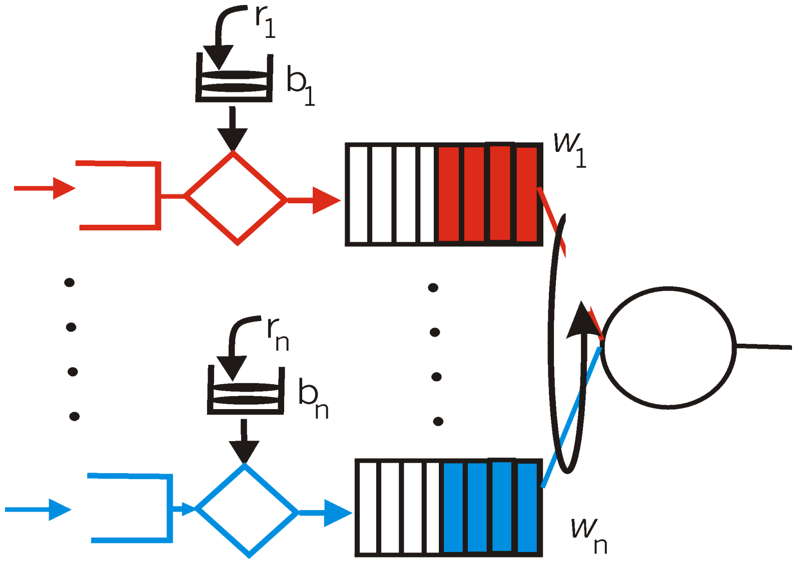
Departure time from a token bucket policer
Token bucket rate = 1 token / 2 time units
Token bucket size = 2 tokens

QoS guarantees for delays too!

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



$D_{max} = b/R$



Differentiated Services

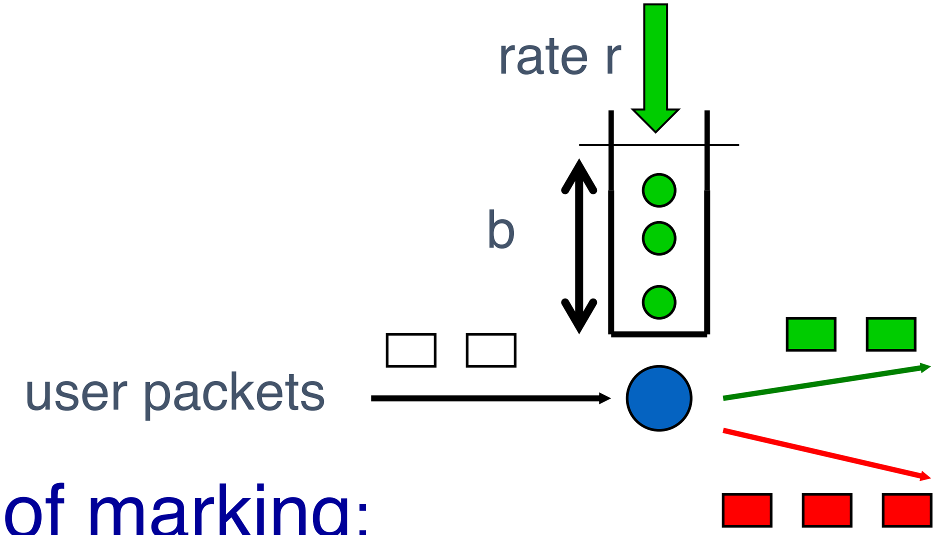
Using marking and packet scheduling to provide QoS

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 define service classes, provide functional components to build service classes

Edge-router packet marking

- profile: pre-negotiated rate r, bucket size b
- packet marking at edge based on per-flow profile



possible use of marking:

- class-based marking: packets of different classes marked differently
- intra-class marking: conforming portion of flow marked differently than non-conforming one

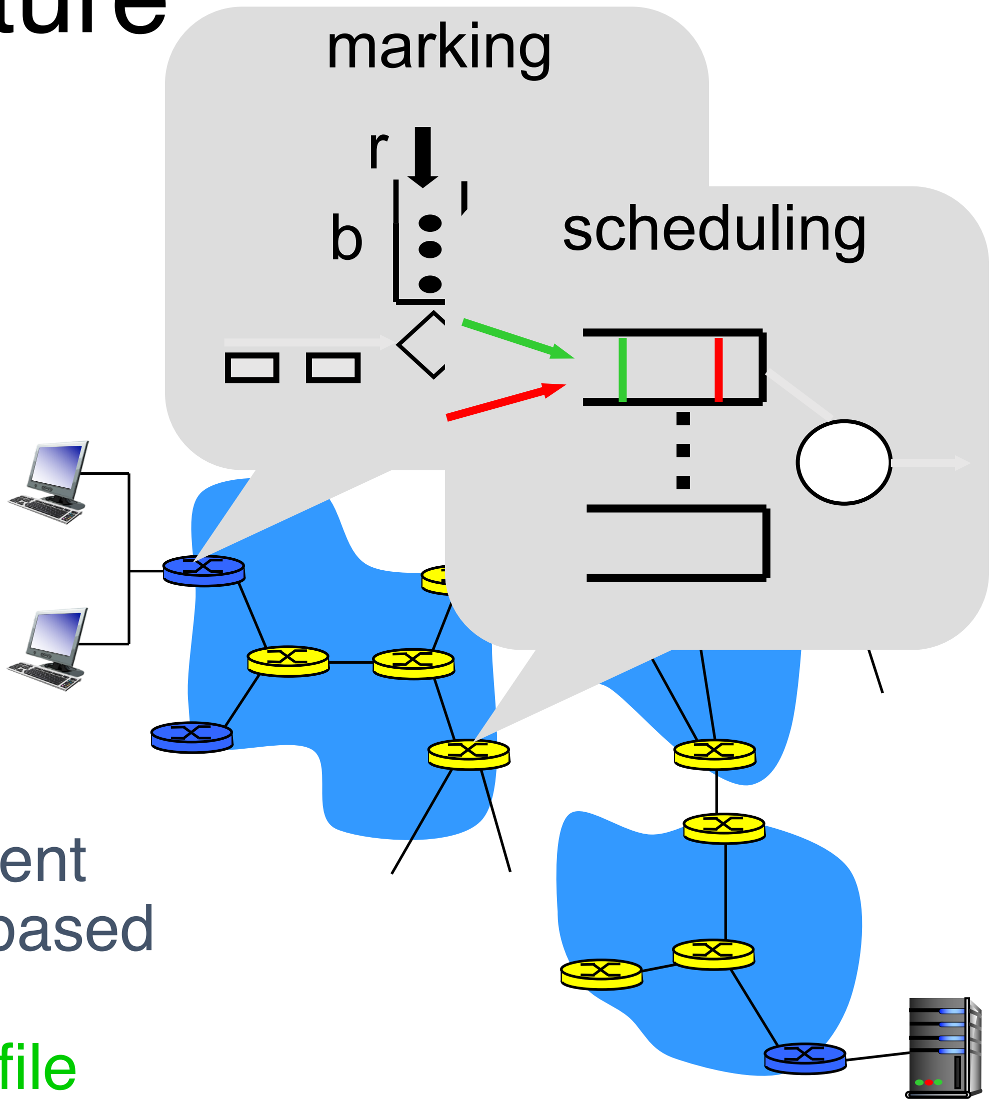
Diffserv architecture

edge router:

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

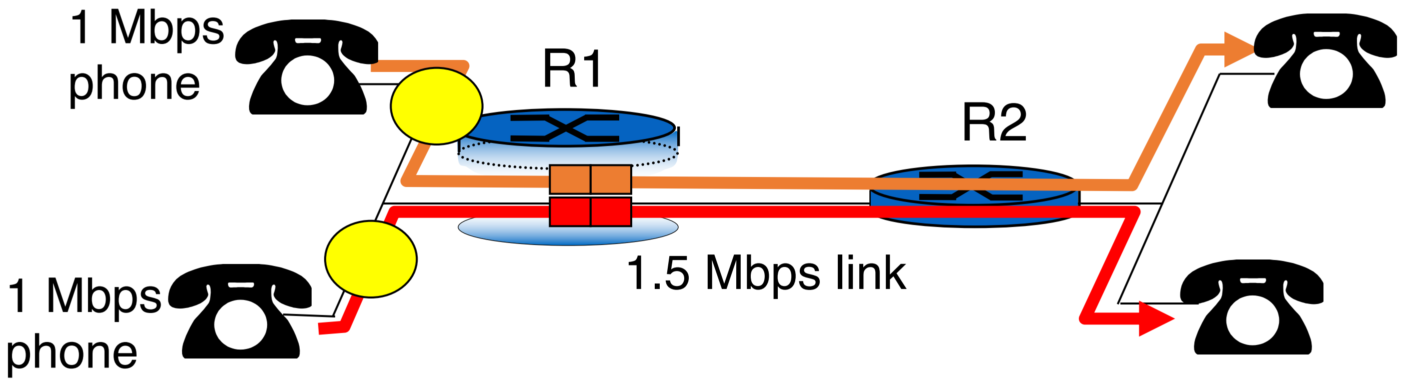
core router:

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



Per-connection QoS guarantees

- *basic fact of life:* can not support traffic demands beyond link capacity

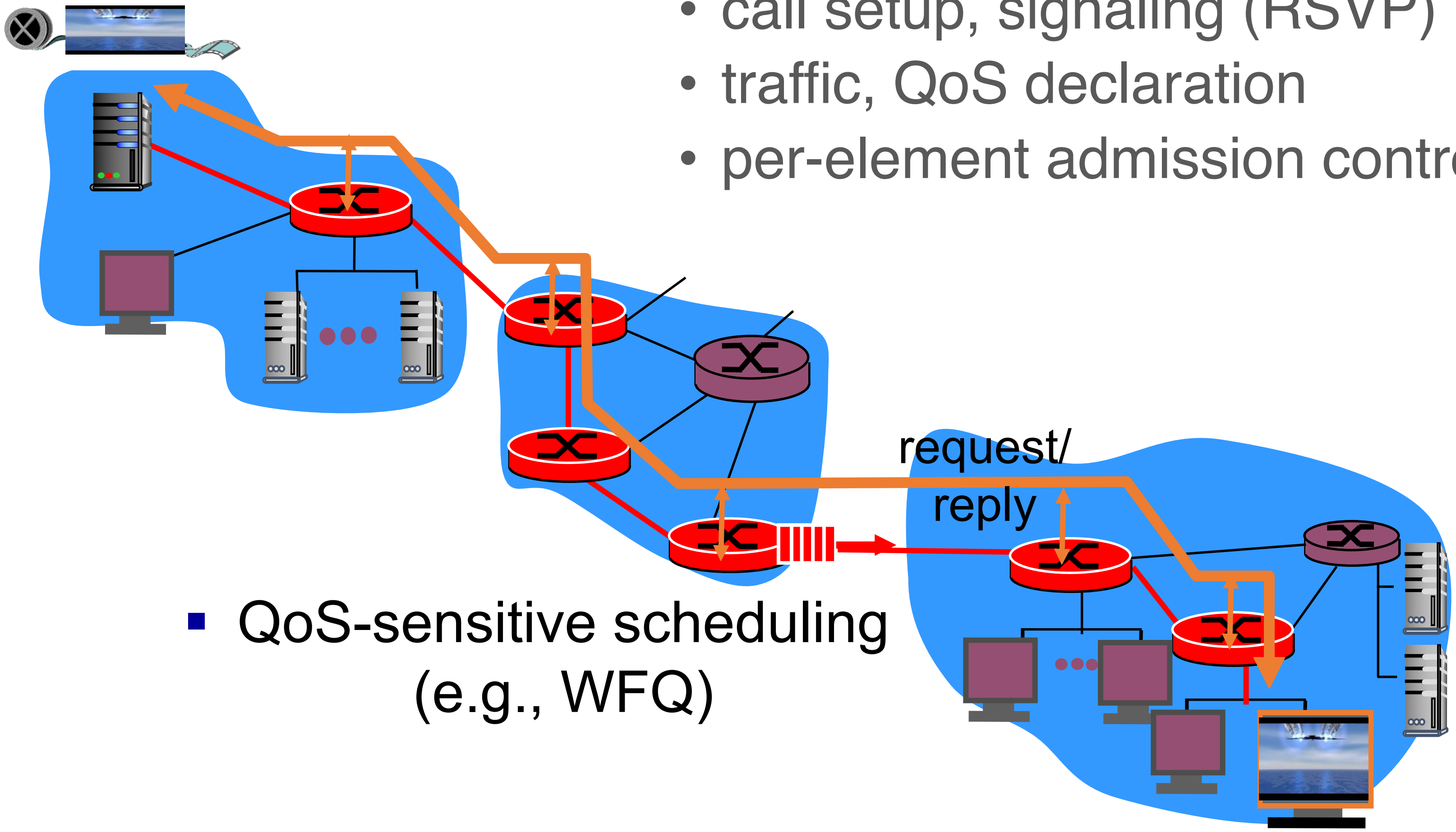


Principle 4

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)

Summary of network support for QoS

- Need ways to distinguish traffic (marking) and handle traffic contention (scheduling)
- Packet scheduling: a great place to provide QoS by isolating traffic from each other
- Abstractions:
  - Leaky bucket: shape traffic
  - Token bucket: shape and police traffic
- The Internet DiffServ architecture builds on these mechanisms
- Resource reservation and call admission required to ensure QoS when demand exceeds capacity
  - However, not very widely deployed on the Internet today