CS244a: Introduction to Computer Networks

Handout 9: Quality of Service, fairness, and delay guarantees

Nick McKeown
Professor of Electrical Engineering and Computer Science, Stanford University
nickm@stanford.edu
http://www.stanford.edu/~nickm

The problems caused by FIFO queues in routers

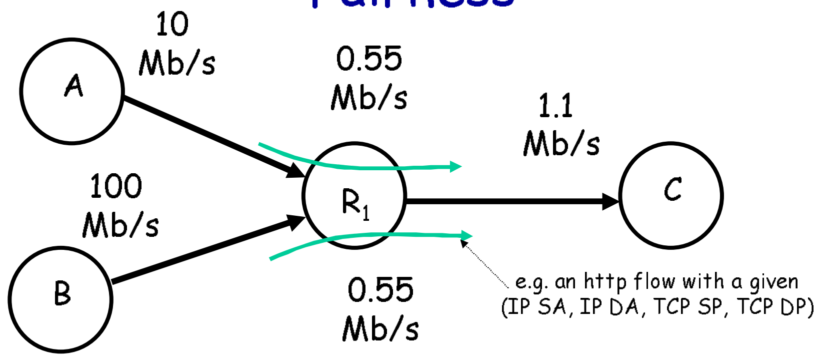
Fairness

1. In order to maximize its chances of success, a source has an incentive to maximize the rate at which it transmits.
2. (Related to #1) When many flows pass through it, a FIFO queue is "unfair" – it favors the most greedy flow.

Delay Guarantees

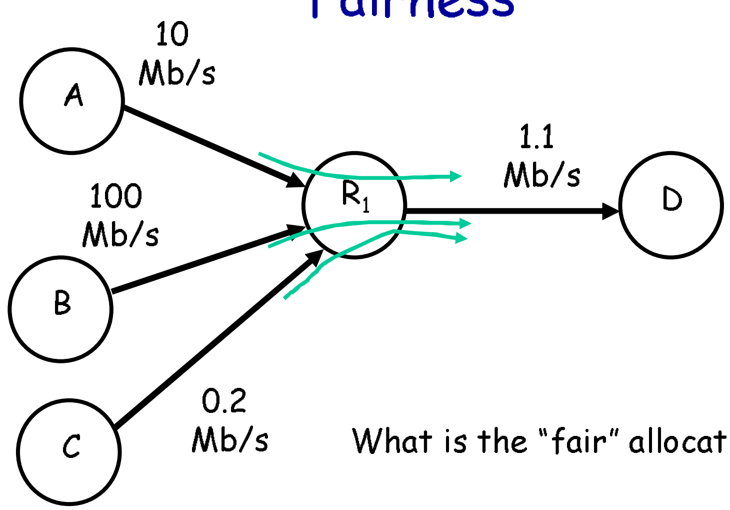
3. It is hard to control the delay of packets through a network of FIFO queues.

Fairness



What is the "fair" allocation:
(0.55Mb/s, 0.55Mb/s) or (0.1Mb/s, 1Mb/s)?

Fairness



What is the "fair" allocation?

Max-Min Fairness

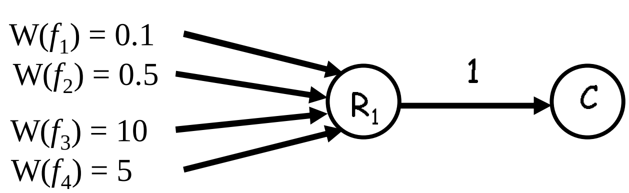
A common way to allocate flows

N flows share a link of rate C. Flow f wishes to send at rate $W(f)$, and is allocated rate $R(f)$.

1. Pick the flow, f, with the smallest requested rate.
2. If $W(f) < C/N$, then set $R(f) = W(f)$.
3. If $W(f) > C/N$, then set $R(f) = C/N$.
4. Set $N = N - 1$. $C = C - R(f)$.
5. If $N > 0$ goto 1.

Max-Min Fairness

An example



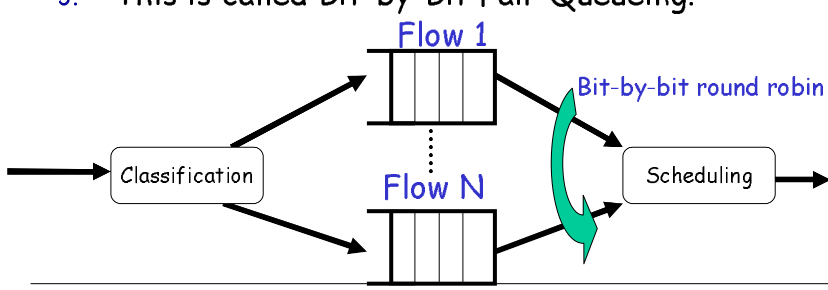
Round 1: Set $R(f_1) = 0.1$
Round 2: Set $R(f_2) = 0.9/3 = 0.3$
Round 3: Set $R(f_4) = 0.6/2 = 0.3$
Round 4: Set $R(f_3) = 0.3/1 = 0.3$

Max-Min Fairness

- How can an Internet router "allocate" different rates to different flows?
- First, let's see how a router can allocate the "same" rate to different flows…

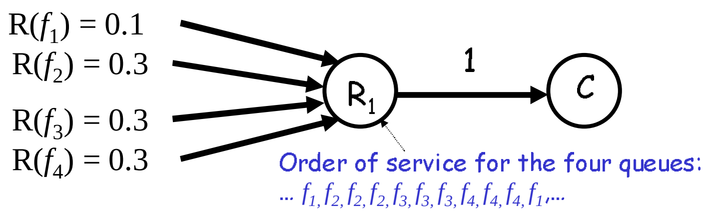
Fair Queueing

1. Packets belonging to a flow are placed in a FIFO. This is called "per-flow queueing".
2. FIFOs are scheduled one bit at a time, in a round-robin fashion.
3. This is called Bit-by-Bit Fair Queueing.



Weighted Bit-by-Bit Fair Queueing

Likewise, flows can be allocated different rates by servicing a different number of bits for each flow during each round.



Also called "Generalized Processor Sharing (GPS)"

Packetized Weighted Fair Queueing (WFQ)

Problem: We need to serve a whole packet at a time.

Solution:

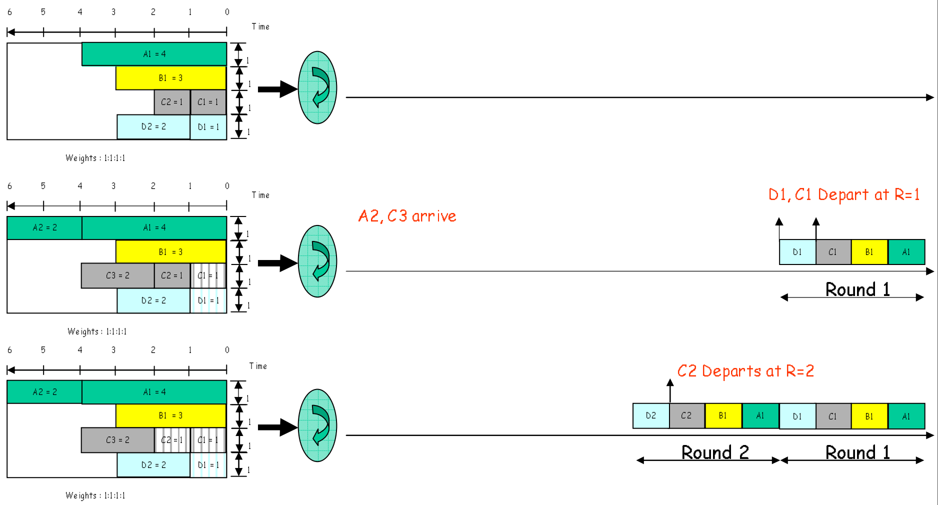
1. Determine what time a packet, p, would complete if we served flows bit-by-bit. Call this the packet's finishing time, F_p.
2. Serve packets in the order of increasing finishing time.

Theorem: Packet p will depart before $F_p + \mathrm{TRANSP}_{max}$

Also called "Packetized Generalized Processor Sharing (PGPS)"

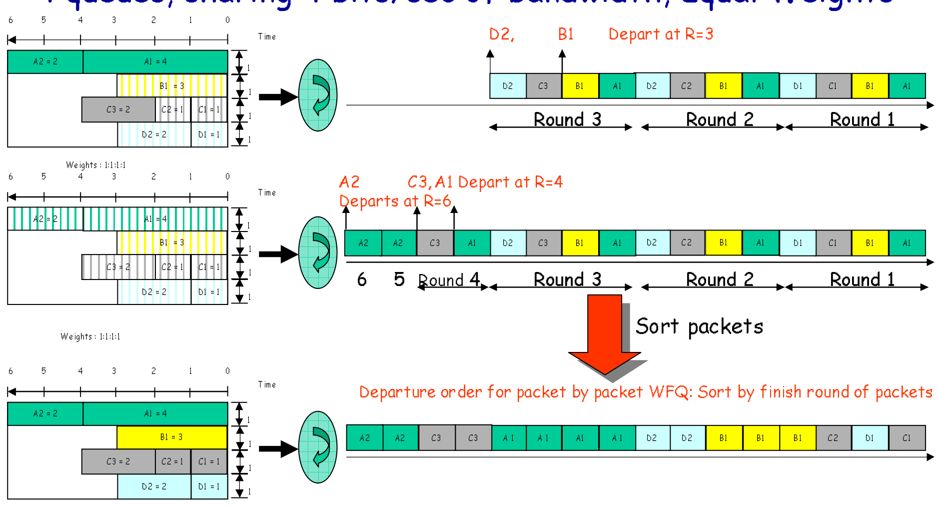
Understanding bit by bit WFQ

4 queues, sharing 4 bits/sec of bandwidth, Equal Weights



Understanding bit by bit WFQ

4 queues, sharing 4 bits/sec of bandwidth, Equal Weights



The use of WFQ for (weighted) fairness

- WFQ can be used to provide different rates to different flows.
- Most routers today implement WFQ and can be used to give different rates to different flows. (Not used much yet).
- Different definitions of a flow are possible: Application flow, all packets to a destination, all packets from a source, all http packets, the CEO's traffic, … etc.

The problems caused by FIFO queues in routers

Fairness

1. In order to maximize its chances of success, a source has an incentive to maximize the rate at which it transmits.
2. (Related to #1) When many flows pass through it, a FIFO queue is "unfair" – it favors the most greedy flow.

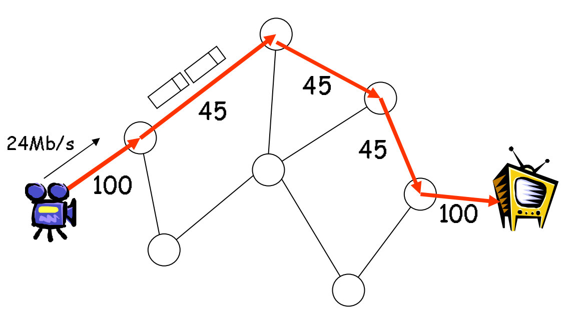
Delay Guarantees

3. It is hard to control the delay of packets through a network of FIFO queues.

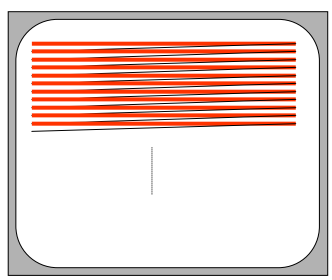
Some applications that would like bounds on packet delay

- Multimedia Applications
  - One-to-many streaming stored audio or video.
  - Interactive streaming audio or video.
- Other delay-sensitive applications
  - Real-time control.
- Other delay-sensitive applications
  - Premium Internet/web access

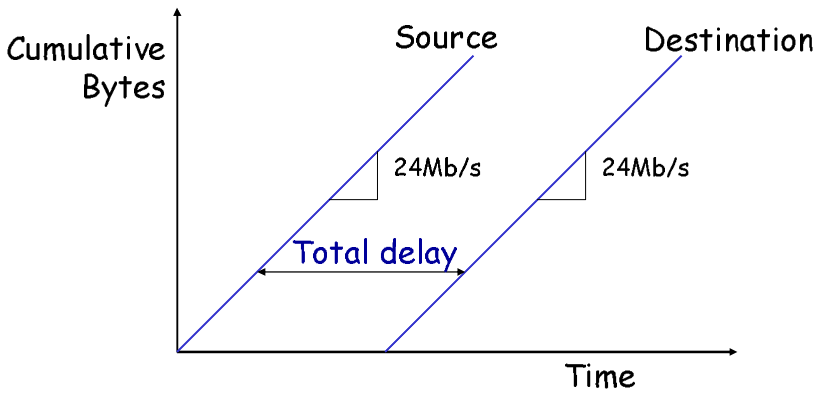
The need for a playback buffer



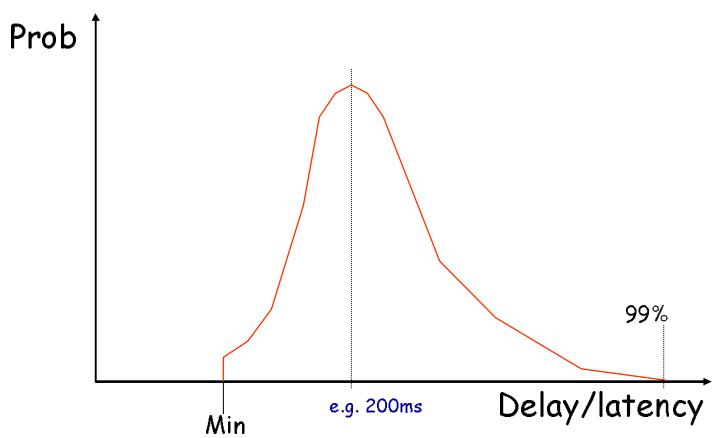
e.g. 10fps, 600x500 bytes per frame ~= 24Mb/s



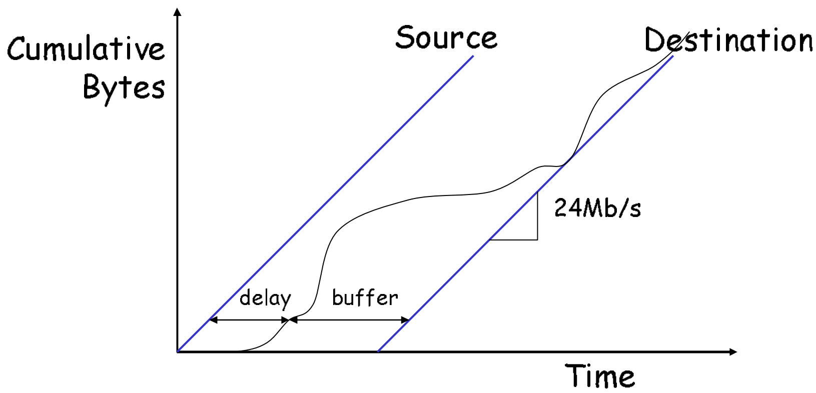
The need for a playback buffer



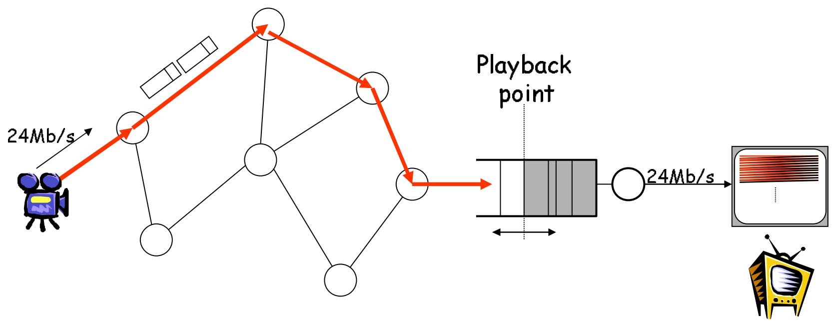
In real life: Delay Variation



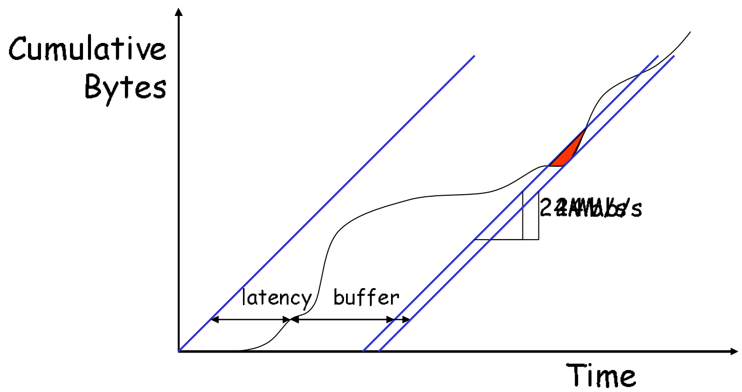
The need for a playback buffer



The need for a playback buffer



The need for a playback buffer



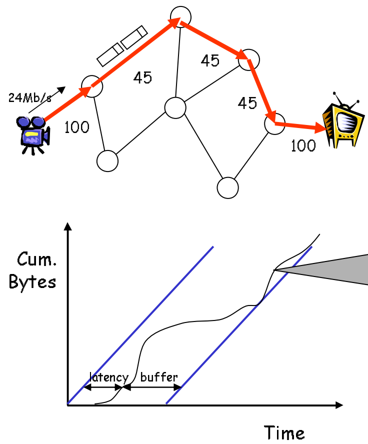
Winter 2008 Handout 9 21

Playback buffer

- How to set the playback point?
  - Why not just set it to be very large?
- How big to make the buffer?
- Can we ever know how big to make the buffer?
  - (We're expert at answering these questions now!)

Winter 2008 Handout 9 22

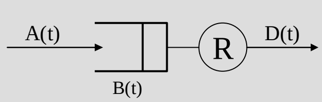
Some observations

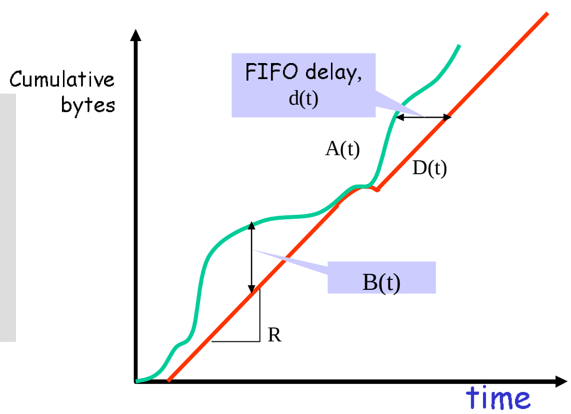


1. Has a maximum instantaneous rate of <= 100Mb/s
2. Has an average rate of <= 24Mb/s
3. Has a short-term average rate of <= 45Mb/s

Deterministic analysis of a router queue

Model of router queue



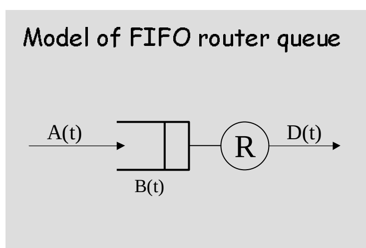


So how can we control the delay of packets?

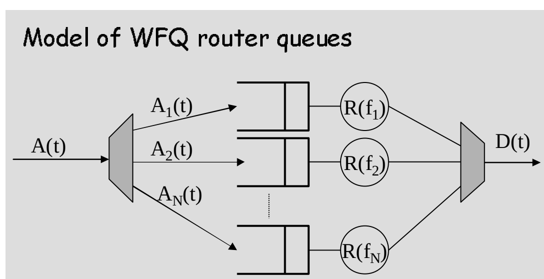
Assume continuous time, bit-by-bit flows for a moment…

1. Let's say we know the arrival process, $A_f(t)$, of flow f to a router.
2. Let's say we know the rate, $R(f)$ that is allocated to flow f.
3. Then, in the usual way, we can determine the delay of packets in f, and the buffer occupancy.

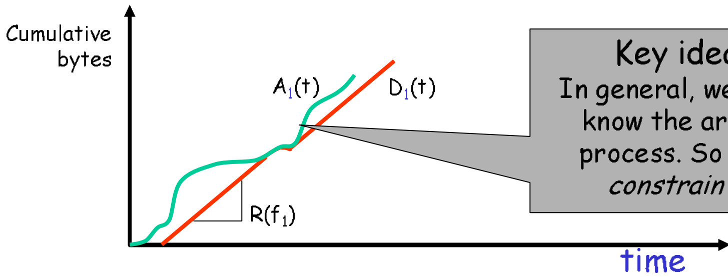
Router Without QoS:



Router With QoS:

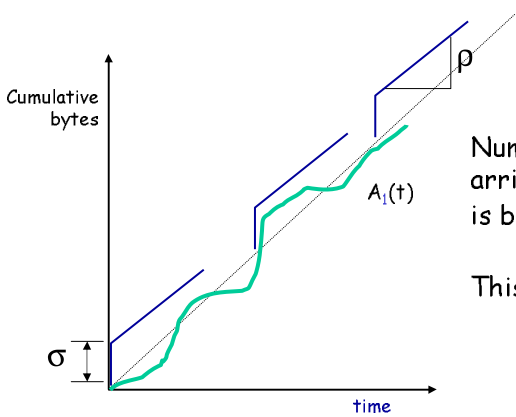


How to bound packet delay?



Key idea:
In general, we don't know the arrival process. So let's *constrain* it.

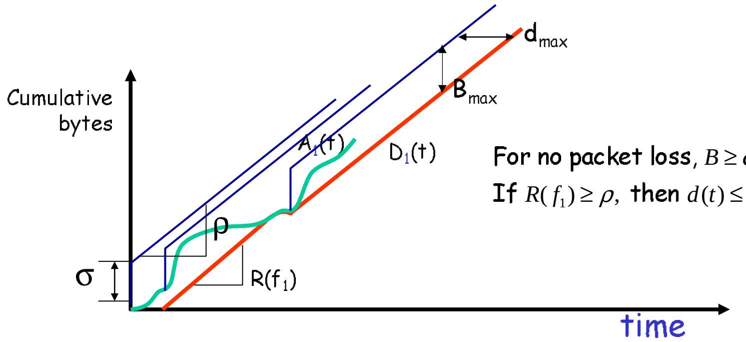
Let's say we can bound the arrival process



Number of bytes that can arrive in any period of length t is bounded by: $\sigma + \rho t$

This is called "(σ,ρ) regulation"

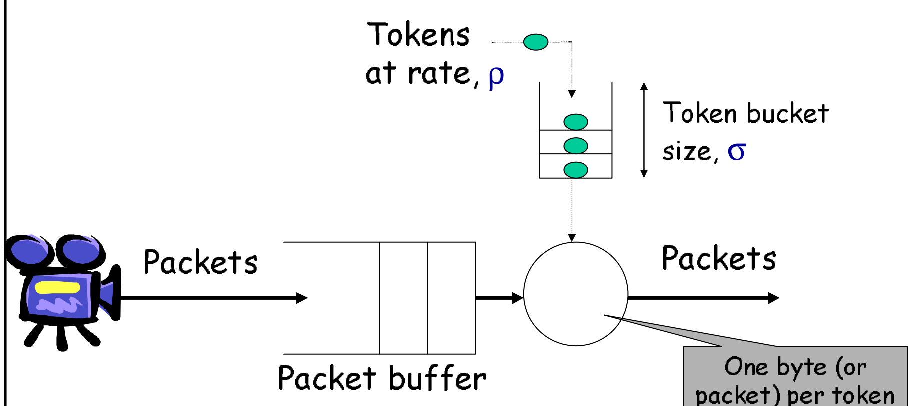
(σ,ρ) Constrained Arrivals and Minimum Service Rate



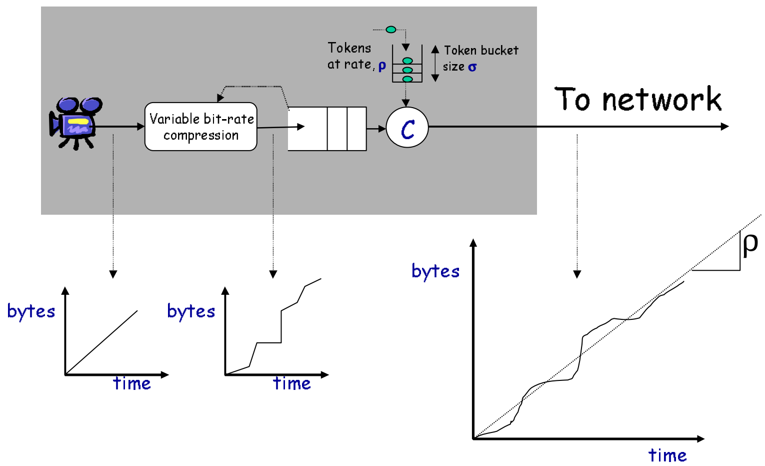
For no packet loss, $B \geq \sigma$.
If $R(f_1) \geq \rho$, then $d(t) \leq \sigma / R(f_1)$.

Theorem [Parekh,Gallager '93]: If flows are leaky-bucket constrained, and routers use WFQ, then end-to-end delay guarantees are possible.

The leaky bucket "(σ,ρ)" regulator



How the user/flow can conform to the (σ,ρ) regulation



Providing Delay Guarantees: Summary

1. Before starting a new flow, source asks network for end-to-end delay guarantee.
2. Source *negotiates* (σ,ρ) values with each router along the path so at to bound delay through every router, and hence bound the end-to-end delay.
3. Routers perform *admission control* to check whether they have sufficient resources (link data rate and buffers).
4. Each router along path *reserves* resources.
5. Flow starts, and source starts sending packets according to agreed (σ,ρ) values.
6. Router determines which flow each arriving packet belongs to, and puts it in the right queue.
7. Router serves queues, using WFQ, so as to bound packet delay through the router.

Steps Involved in Providing Delay Guarantees

- Per-session
  - Call setup, call admission and resource reservation
    "Can the network accept the call and provide the QoS?"
- Per-packet
  - Packet Classification: "What flow does this packet belong to, and when should I send it?"
  - Shaping: "Am I keeping my side of the contract?"
  - Policing: "Did the user keep his/her side of the contract?"
  - Packet Scheduling: "Sending the packet at the right time."

Proposed Techniques for QoS in the Internet

- Resource ReSerVation Protocol (RSVP)
- Integrated Services
- Differentiated Services

Resource Reservation: RSVP

- RSVP is a protocol for establishing a guaranteed QoS path between a sender and receiver(s).
- RSVP establishes end-to-end reservations over a connectionless network.
- RSVP is robust when routers/links fail: traffic is re-routed and new reservations are established.
- RSVP is receiver-initiated and so is designed with multicast in mind.

Resource Reservation: RSVP

The network needs to know the TSpec, the RSpec and the Path followed by packets.

- The TSpec (specification of the transmitted traffic) is only known by the source.
- The Path is only known by the network.
- The Rspec (specification of what the receiver would like to receive).

Resource Reservation: RSVP

So, the sender periodically sends the Tspec to the whole multicast group ("PATH messages").

- The network learns the Path taken by packets in the multicast group.
- The receiver/network learns the TSpec.

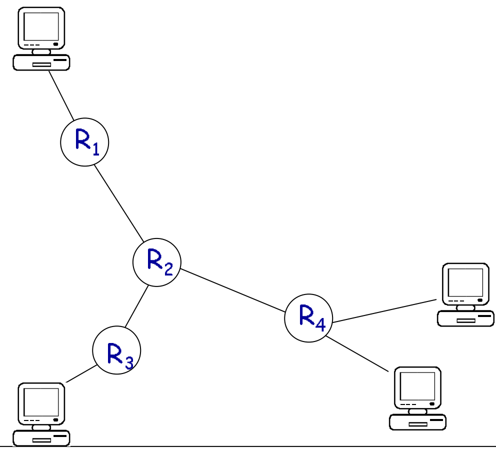
Resource Reservation: RSVP

To initiate a new reservation, a receiver sends messages to reserve resources "up" the multicast tree ("RESV messages").

- The routers forward RESV messages towards the source.
- The routers determine if the reservation can be fulfilled.
- If necessary/possible, the routers merge the requests from different receivers.

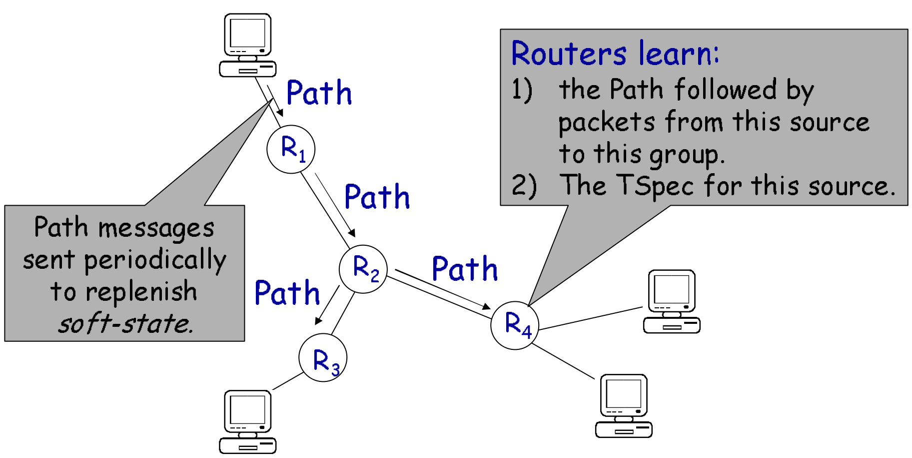
Establishing a reservation

1: The multicast group is established



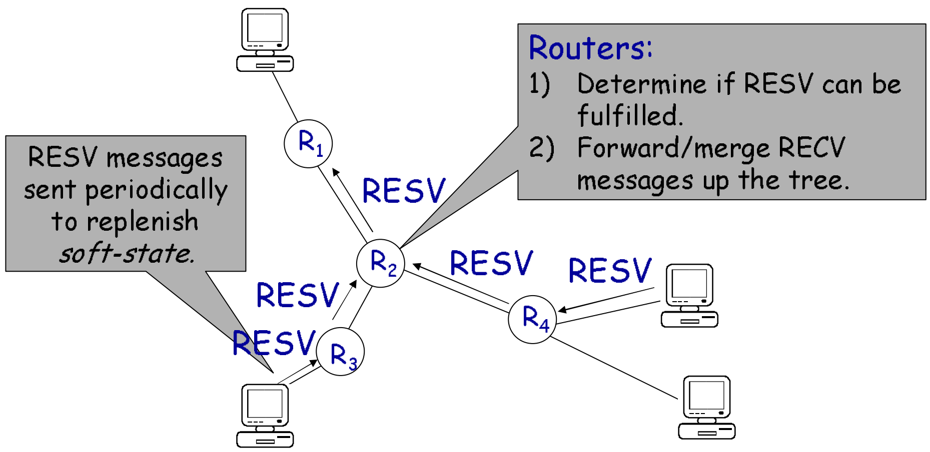
Establishing a reservation

2: RSVP Path messages sent by source(s)



Establishing a reservation

3: RSVP RESV messages sent by receiver(s)



Establishing a reservation

Merging RESV messages

Examples:

1. Router receives RESVs from two receivers, A and B, asking for 100ms delay and 50ms delay respectively. Router passes up request for 50ms.
2. Router receives RESVs for a audio teleconference call with 100 participants requesting 1.5Mb/s each. The tree need support only 1.5Mb/s total data-rate.

RSVP supports many styles of RESV merging.

Outline

- Resource ReSerVation Protocol (RSVP)
- ➔ Integrated Services: Uses RSVP.
- Differentiated Services

Outline

- Resource ReSerVation Protocol (RSVP)
- Integrated Services: Uses RSVP.
- ➔ Differentiated Services: Doesn't!

Differentiated Services

- Instead of *per-flow* queues and rates, uses *per-class* queues and rates.
- Much simpler to implement, introduce and manage.
- But… means that many flows share the same queue, state and capacity. So, it is like a simple approximation to Integrated Services.

Quality of Service in the Internet
Some Interesting Questions

- How can we implement per-flow buffering and scheduling when the number of flows is large?
- How can we encourage/force users to ask for only the resources they need?
- How can we bill users for the service they receive?

Quality of Service in the Internet

The jury is still out as to what scheme makes most sense, if any.

1. Some people believe that fine-grained guarantees are needed.
2. Others believe that simple, coarse-grained guarantees will be sufficient.
3. Still others believe that a simple strict priority mechanism between "premium" and "best-effort" service will be OK.
4. Many people believe we should just provision a lot more capacity and be done with it!