



CS519: Computer Networks

Lecture 5, Part 5: Mar 31, 2004
Queuing and QoS



Ways to deal with congestion

CS519

- Host-centric versus router-centric
- Reservation-based versus feedback-based
- Window-based versus rate-based
- The Internet is: host-centric, feedback-based, and window-based
 - Because that's what TCP is
 - But this is to some extent an “accident” of TCP's history



Alternative approaches

CS519

- In the mid-90's, there was a concerted effort to make Internet QoS more router-centric, reservation-based, and rate-based
 - An architecture called Integrated-Services (“intserv”)
 - And a resource reservation protocol called RSVP
 - This didn't take off, but its interesting to look at, and to see where things stand now



Queuing disciplines



CS519

- We talked a bit about RED
- But in fact, most queuing in the internet is FIFO with tail-drop
 - FIFO means First-In-First-Out, like the queue in a bank
 - This is a *scheduling discipline*
 - Tail drop means that, if the queue overflows, you drop the last packet received
 - This is a *drop policy*



Limitations of FIFO . . .

CS519

- The problem with FIFO is that aggressive flows can squeeze out conservative flows
 - A TCP that doesn't follow AIMD rules can grab all the bandwidth
 - Non-TCP connections (voice) can grab all the bandwidth
- It just isn't fair!



Fair Queuing

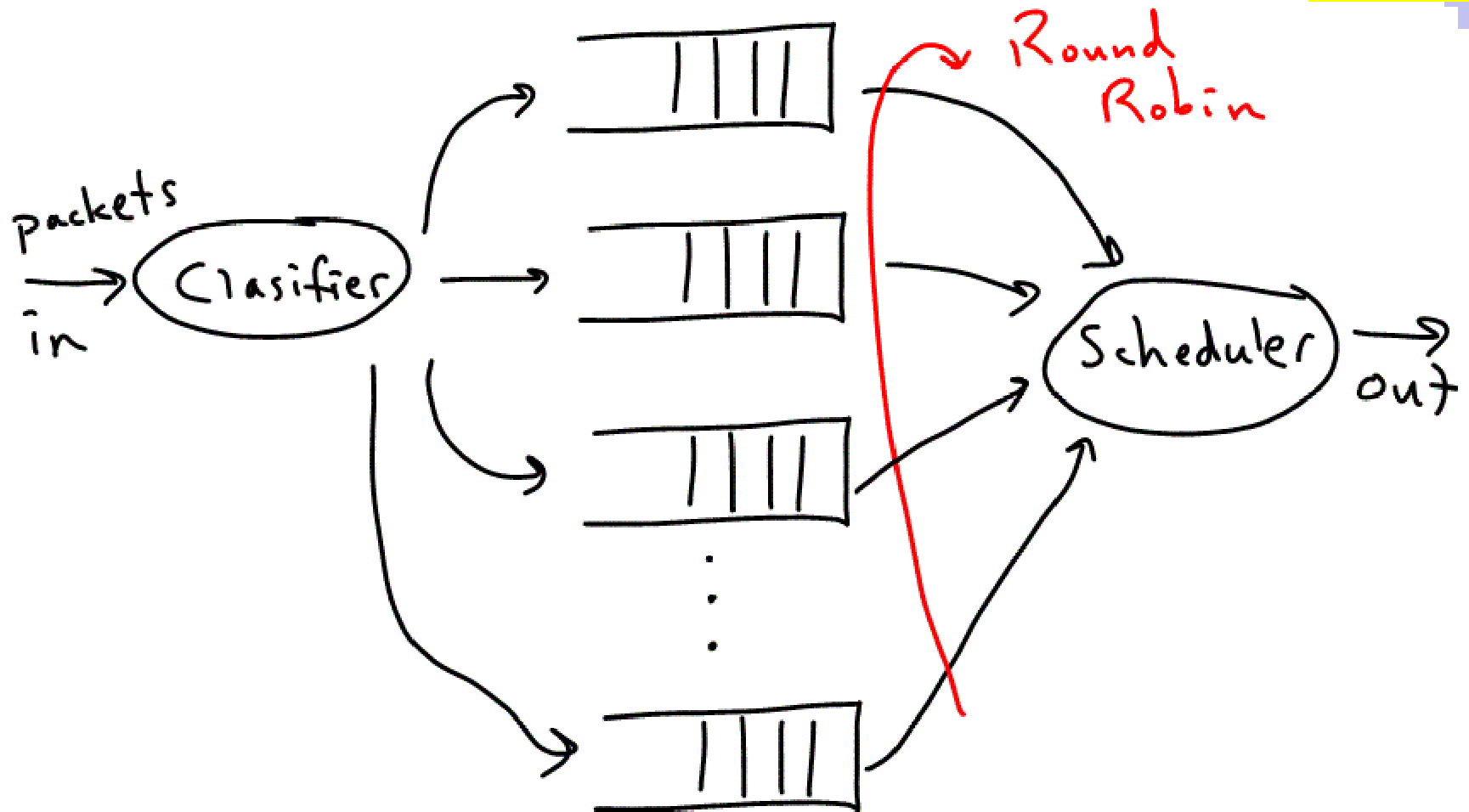


CS519

- Use multiple FIFO queues instead of just one
- Assign traffic to queue according to some policy
 - Such as type of traffic (voice versus TCP)
 - Or by TCP flow
- Service each queue in turn

Fair Queuing

CS519





Fair Queuing issues



CS519

- Scheduling must be (conceptually) per bit, not per packet
 - Else large packet flows get more bandwidth
- Often you want to schedule a short packet from Q1 that arrived after a long packet in Q2
 - Unless of course the long packet is already in transit . . .



Fair Queuing issues

CS519

- Perfect “per bit” scheduling of fair queues a bit expensive
 - Book defines giving a sending timestamp to each packet, and then sending in order, but now you have an ordering job
- I’ll ask you to build simple “reasonable approximations” in project 4!



Weighted Fair Queuing



CS519

- If we want to give higher priority to some queues over other, we can schedule bits from some queues more often than others
 - “Q1 gets 2 bits for every 1 bit from Q2”
- Why do this rather than a strict priority queuing scheme???
- Service Q2 only if Q1 empty, service Q3 only if Q2 empty, etc...

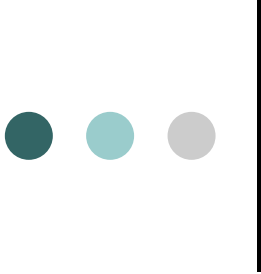


Fair Queuing is “work conserving”



CS519

- “Work conserving” means:
 - If there is work to be done, it will be done
- With fair queuing, if any queue has something to send, it will be sent
- Note that this is not the case with pure circuit switches, where BW has been reserved whether it is used or not!



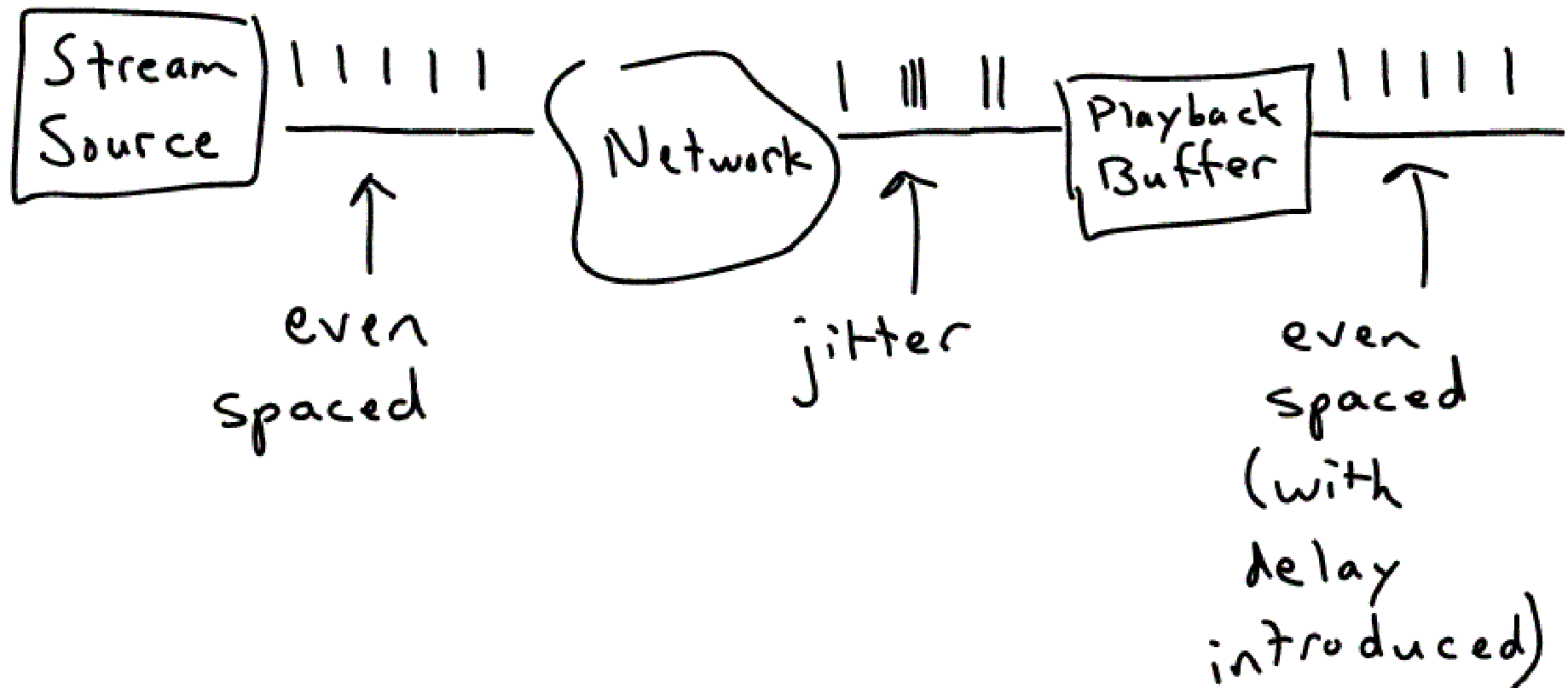
What is a “real-time” application?

CS519

- One where the time at which a packet is “played out” is important
 - Voice or video . . .
- But real-time applications can have extremely different network requirements
 - Voice conversation is very bad if delay > 200ms or so
 - Streaming media can be delayed for many seconds
 - Telnet has much stricter delay requirements!

Play-out (or playback) buffer

CS519



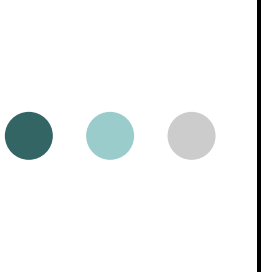


Real-time applications



CS519

- Some video applications can adapt bandwidth requirements over a large range
 - High-fidelity versus low-fidelity bits
- And can therefore tolerate wide BW variance
- Others won't or can't do this...



IETF Intserv (Integrated Services)

CS519

- IETF attempt at fine-grained (per-flow) QoS
 - Resource reservation with admission control
- Settled on two types of service:
- Guaranteed
 - i.e. conversational voice
- Controlled Load
 - More tolerant/adaptive realtime applications
- (In addition to existing “best effort” service)



Guaranteed versus Controlled Load

CS519

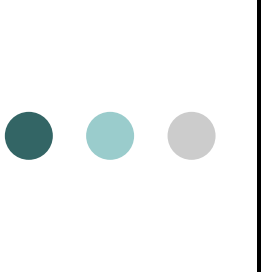
- Guaranteed really requires reserved resources, careful packet scheduling
- Controlled Load is based on the notion that most realtime apps work well as long as the network is lightly loaded
 - Simply give this class adequate bandwidth (WFQ), but otherwise treat FIFO



Flowspec

CS519

- Recall that the more bursty traffic is, the quicker queues build up
- To make admission control decisions, the network needs to know how bursty a given flow is going to be
- And, it needs to guarantee that the flow is no more bursty than it claimed
- A flowspec is what describes the traffic (TSpec) and the network requirements (RSpec)



Token bucket (aka Leaky bucket)

CS519

- A simple and common way to describe traffic is with a token bucket
- Two parameters:
 - Rate r (bits per second)
 - The size of the hole in the bucket
 - (average throughput)
 - and bucket Depth B (bits)
 - The size of the bucket itself
 - (max burst size)



Token bucket policing



CS519

- A token bucket flowspec (r, B) can be enforced with a queue of size B that is serviced at a rate of r
- The network can therefore enforce compliance
- The network will tag a non-compliant packet as “out of spec” rather than drop it
 - And then drop with higher priority should there actually be congestion



Resource Reservation Protocol (RSVP)

CS519

- IETF's version of a “call setup” protocol
- Different from a virtual circuit network in several interesting ways
- VCs couple routing and resource reservation (RR), whereas Internet already has routing (decoupled from RR)
- IETF wanted to allow router failure and not lose the “call”
- IETF wanted to accommodate multicast

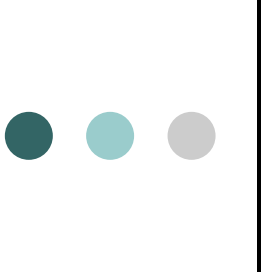


RSVP



CS519

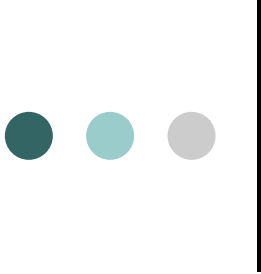
- Recipient makes the actual resource reservation
 - But initiator gives the recipient the path to use
 - Resource reservation is on reverse path
- Reservation is “soft-state”
 - Network will “forget” the reservation if recipient doesn’t refresh it
 - Essentially, the recipient refreshes the reservation every minute or so!



An aside: soft-state

CS519

- Internet community was (still is???) big on the notion of soft state
- Idea is to allow control state to “age” (timeout) rather than require explicit deletion of state
- More robust, because if state creator crashes, state goes away naturally
- Simpler, because only need state create commands



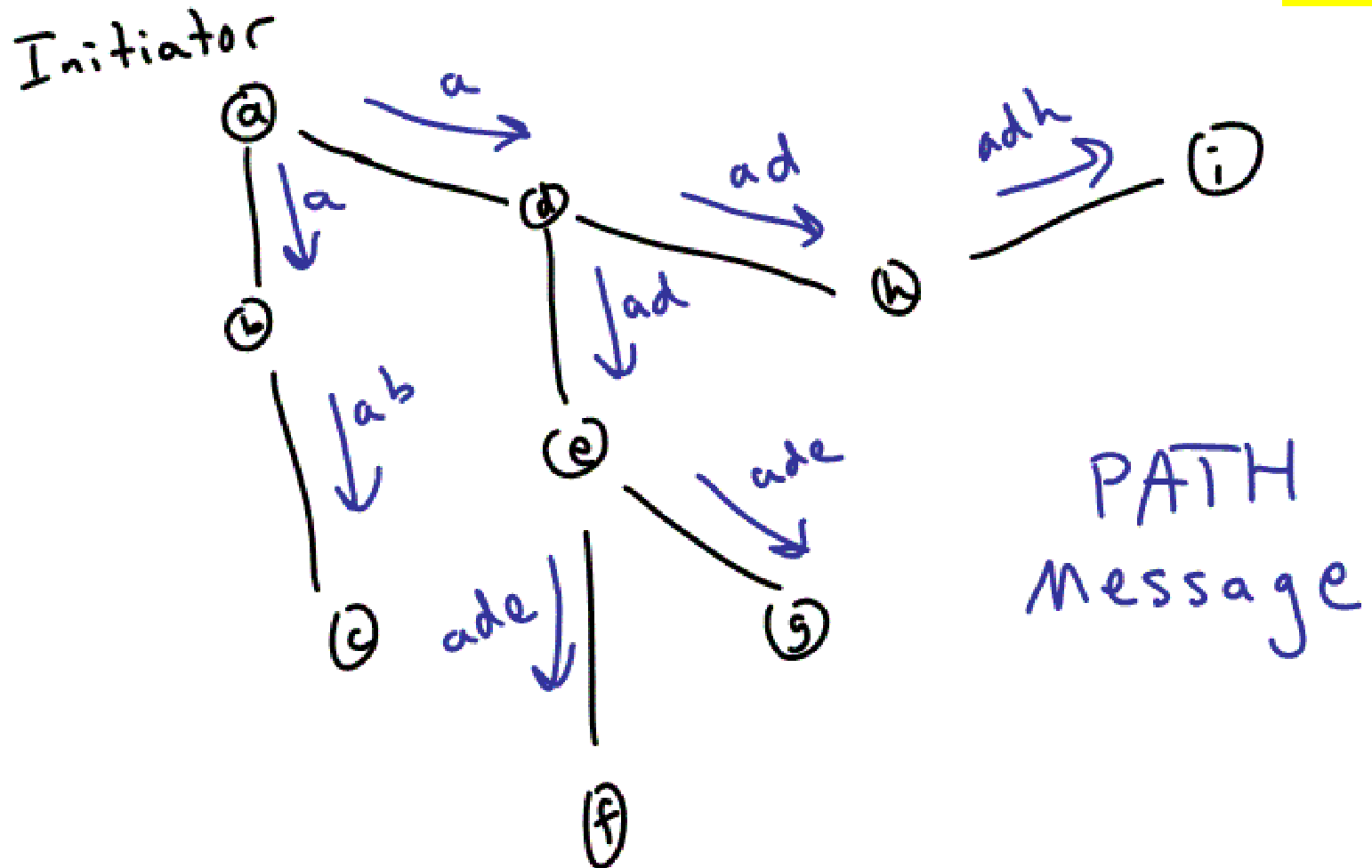
An aside: soft-state

CS519

- Can be a nice principle if functions degrade gracefully rather than stop working when state disappears
 - RSVP: packet still forwarded, but just without requested QoS
- Or if actual usage (user data packets) is what refreshes the state
 - LRU caching is a form of soft state
- A nice design principle to keep in mind, but don't be religious about it...

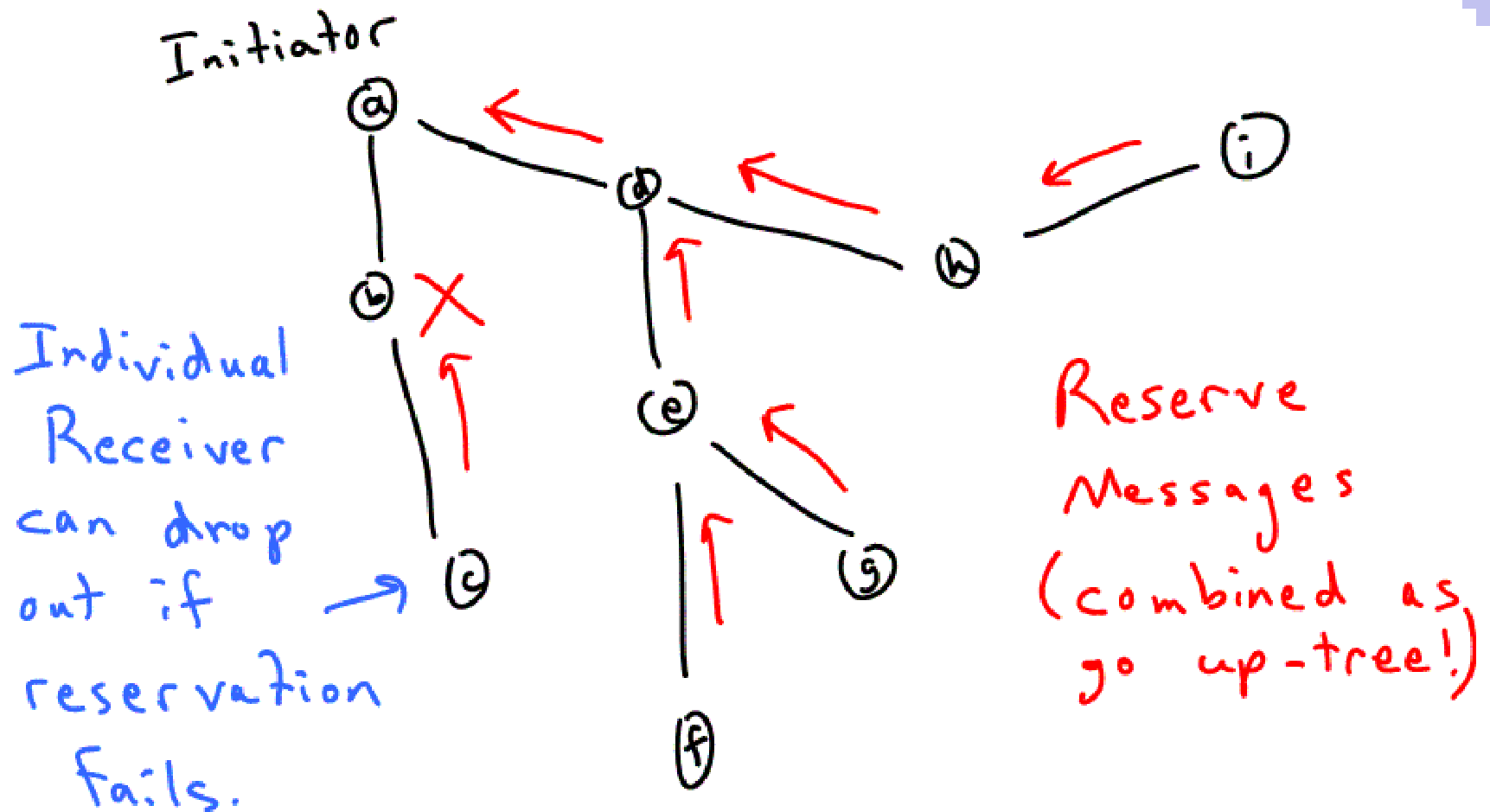
RSVP with multicast

CS519



RSVP with multicast

CS519





Intserv failed in the commercial marketplace

CS519

- Scaling issues
 - Core routers can't handle so much flow state
 - Rather spend energy on high speed (rightfully)
- Lack of business model?
- Requires buy-in from too many communities
 - ISPs, OS vendors, application developers



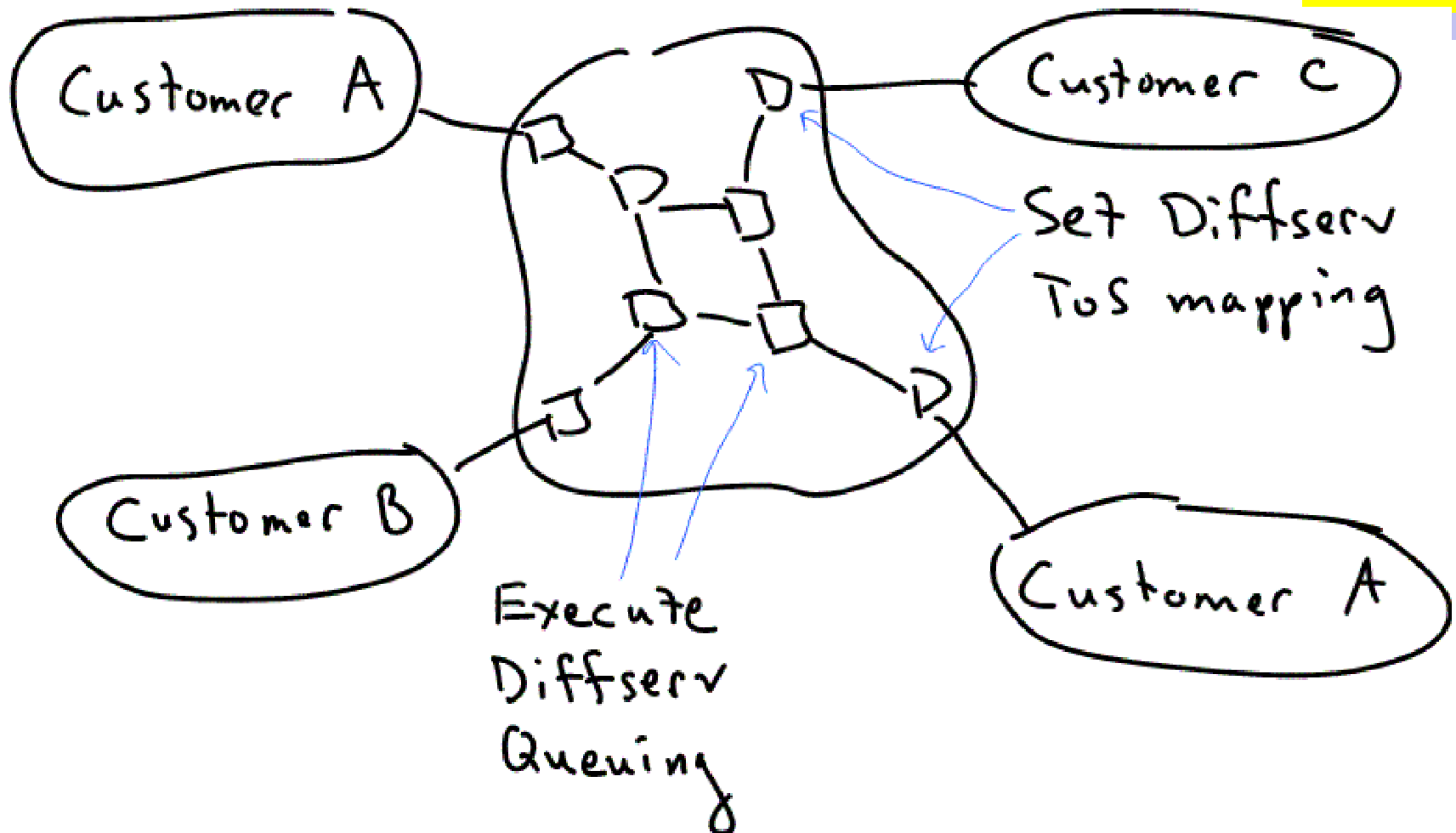
Differentiated Services (Diffserv)

CS519

- A more modest (and realistic) proposal from IETF
- No resources reservations
- No per-flow handling
- Simply define a smallish number of service classes, encoded in IP's ToS bits
- These bits can be set by ISP edge routers, handled by internal routers according to ISP policies

Example Diffserv deployment model

CS519





Service classes reflect those of Intserv

CS519

- EF (Expedited Forwarding)
 - For highly delay sensitive and intolerant apps
- AF (Assured Forwarding)
 - To give “high priority” traffic the effect of a lightly loaded network
 - 12 classes of this



Several approaches to AF

CS519

- Weighted RED (or RIO: Red with In and Out)
 - Different drop thresholds for different classes
- WFQ
- Combinations of these

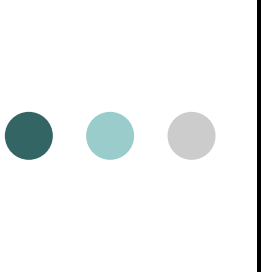


Status of Diffserv



CS519

- I've seen it defined for cellular wireless data networks
 - Where there is a clear bottleneck and need for differentiated services
- Certainly people believe that this is the best usage of the IP ToS bits
- Not aware that this has taken off for backbone services



TCP Friendly Rate Control (TFRC, RFC3448)

CS519

- What if you don't need TCP's reliability/sequencing, but want to be TCP friendly?
 - A BW-flexible realtime video that can tolerate some packet loss
- TCP behavior can be described by an equation
 - Some time called “equation-based congestion control”



Approaches for TCP-friendly congestion control

**CS519**

- Round-trip delay R
- Packet size s
- Loss event rate p (receiver feedback every RTT)
- Retransmission timeout $t_{\text{RTO}} \sim 4R$

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{\text{RTO}}(3\sqrt{\frac{3p}{8}})p(1 + 32p^2)}$$



Simple TCP model

CS519

- Bandwidth as function of packet loss:

$$B(p) = \frac{1}{RTT} \sqrt{\frac{3}{2bp}} + o(1/\sqrt{p})$$

- Assumes triple-duplicate-ACK triggering retransmission
- Does not take timeout into account
- Model: single saturated TCP pumping data into bottleneck
 - other flows only modeled through packet loss



TFRC



CS519

- Defines an algorithm for
 - Measuring loss at receiver
 - Feeding back that info to sender
 - Measuring RTT at sender
 - Adjusting send rate accordingly