IntServ / DiffServ

- Integrated Services (IntServ)
  - Resource Reservation Protocol: RSVP
- Differentiated Services (DiffServ)
  - Assured Forwarding
  - Expedited Forwarding
  - Comparison: AF vs. EF

- Reading:Kurose-Ross Chapter 6.7-6.9

QoS (Quality of Service) in the Internet

- The Internet Protocol (IP) does not guarantee QoS to applications
- Idea: Re-engineer IP to provide quality of service
  - Let routers distinguish classes of flows
- Q: What is the model for a class?
- Solution must consider:
  - the needs of (some/most/all) applications
  - the add'l state that routers must maintain
  - the add'l communication overhead (add'l packets or bits)
  - # of flows or classes a router must handle

IntServ vs. DiffServ

IntServ (1992)

- per-flow reservations
  - i.e., needs RSVP
- flows provide traffic characterization
- "heavy" state: per-flow
- "strong" guarantees, e.g.,
  - conformance to leaky-bucket characterization [RFC 2215]
  - bound on max e2e delay [RFC 2212]

DiffServ (1997)

- packet classification
- edge & core routers
  - edge: "heavy" state
  - core: "light" state
- "weak" guarantees, e.g.,
  - Flow A gets better service than Flow B

Integrated Services (1992)

As described in [RFC 1633] from 1994:

- Philosophy: "guarantees cannot be achieved without reservations"
- Four components to IntServ architecture:
  - packet scheduler
  - classifier
  - admission control routine
  - reservation setup protocol

IntServ Components

All components implemented at all routers!

- Packet Scheduler
  - Manages forwarding of different streams
  - Required resources: sets of queues, timers
  - Example: Implementation of Weighted Fair Queuing (WFQ)
- Classifier
  - Maps packets to a class
  - Packets in same class treated similarly
  - Examples:
    - per-flow class
    - video-packet class

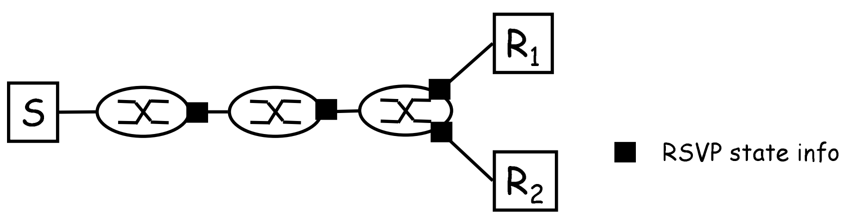
IntServ Components (cont'd)

- Admission Controller
  - Determines whether or not to admit a new flow
  - Q: why would a flow be rejected?
  - Requirements:
    - Knowledge of available resources at router
    - (conservative) model of flow's resource consumption
      - e.g., leaky bucket
  - The hard part: getting apps to characterize their flows
- Reservation Setup Protocol
  - Sets up and maintains (distributed) flows' network resource usage
    - "negotiates" between admission controllers at routers
    - establishes active classifiers at routers
  - e.g., RSVP protocol

RSVP protocol

- The commonly suggested reservation setup protocol
- Designed for multicast sessions (unicast is a special case)
- Receiver-oriented: receivers initiate requests
    - allows for receiver heterogeneity
- Reservation styles allow merging of reservations (i.e., use the style that's appropriate for the app)
- Uses soft-state: reservations need to be refreshed or they expire. Why soft-state?
- Dynamic: able to reconfigure reservation rather than perform complete teardown / setup

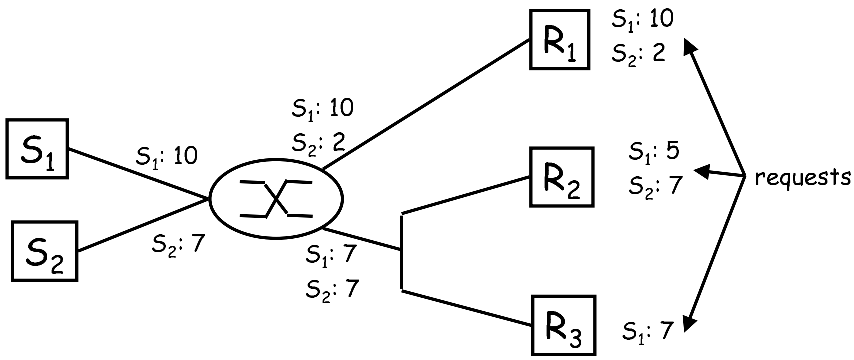
RSVP messaging



- Rcvrs make requests for reservations
- Sufficient resources: Router reserves per outgoing interface (i.e., link) and forwards request upstream
- Insufficient: send ResvError message downstream
- Path messages: from sender toward rcvr so that routers know where to forward receiver requests.
    - Why not just head toward sender using Internet routing tables?

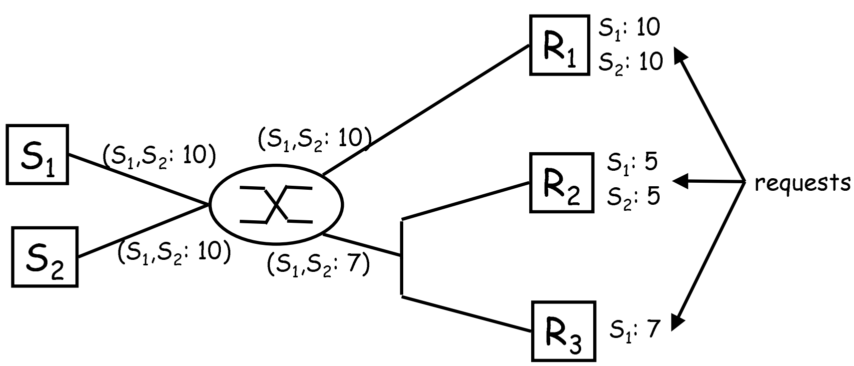
RSVP Reservation Styles

- Fixed-Filter: Allocation per sender indicated
  - Sample application: multimedia (e.g., send audio (S_1) and video (S_2) at same time)



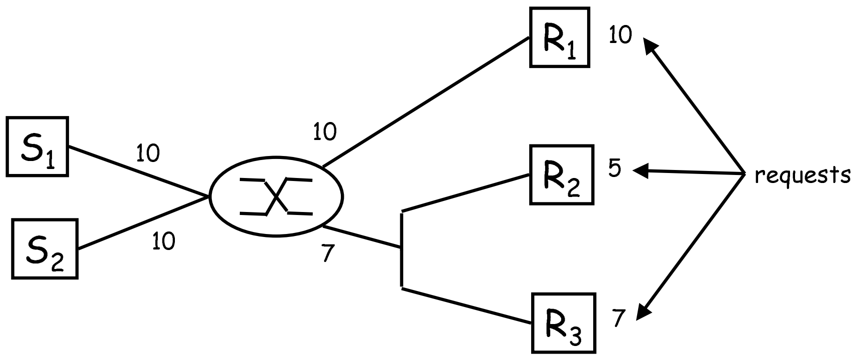
RSVP Reservation Styles

- Shared-Explicit: Allocation shared by list of senders
  - Sample application: multimedia (e.g., debate w/ 2 speakers)



RSVP Reservation Styles

- Wildcard-Filter: Allocation shared by all senders
  - Sample application: town meeting (one sender, but not clear who the speakers might be)



Style Summary

- Fixed-Filter: reservation per sender
  - Senders don't "share" bandwidth
  - Dynamic event: rcvr wants to change a sender allocation
- Shared-Explicit: reservation per list-of-senders
  - Fixed set of senders "share" bandwidth
  - Dynamic event: rcvr wants to add/remove sender or change group allocation
- Wildcard-Filter: no sender specified w/ reservation
  - Any sender can "share" bandwidth
  - Dynamic event: new sender begins transmitting, rcvr wants to increase its receiving allocation

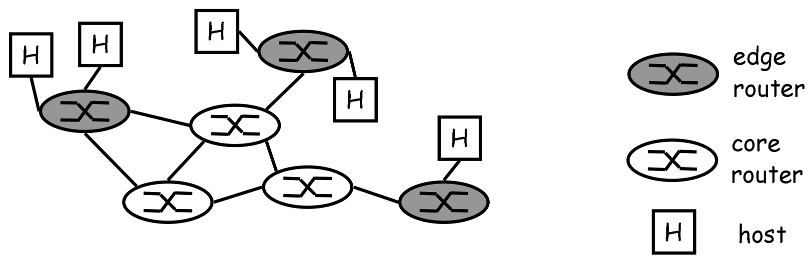
IntServ: Problems

- Reservation protocols and structure complicated
  - lots of message passing
  - coordination problems
- All routers maintain state
  - state maintenance requires additional processing / memory resources
  - Lots of flows traverse core (backbone) routers
    - Lots of state: need more memory
    - Lots of RSVP msgs to process: slows transfer speeds
    - Scheduler and Classifier have too much to deal with

DiffServ

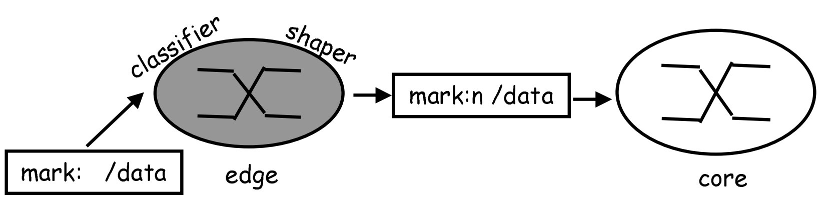
- Q: What if IntServ is too complex/costly to deploy?
- A: Build a simpler scheme that takes into account
  - many apps have simple requirements (e.g., need fixed bandwidth, low jitter)
  - App can't/doesn't always conform to/provide "strict" model of resource usage
  - different levels of functionality can be placed at different "types" of routers
    - network edge
    - network core

Differentiated Services



- Idea: keep the architecture simple within the core.
  - higher complexity permitted at edge routers
- Just provide service differences, no explicit guarantees
  - i.e., high and low priority classes (extra $$$ for high)

DiffServ Architecture



- Edge router
  - classify packet and mark packet
  - shape flow (control entry rate into core, drop pkts, change mark, etc.)
- Core router
  - handle packet based on its mark
  - possibly remark at peering points
- Maintain **Per-Hops Behavior** (PHB): the desired service (e.g. rate) provided to a class at a given hop (router)

2 Competing PHBs

- Expedited Forwarding (EF) [RFC 2598]
  - Router must support classes' configured rates
  - EF class allocated fixed portion of router processing per unit time, e.g.,
    - Class-based queueing (CBQ) w/ priority to EF queue
    - Weighted Fair Queuing
- Assured Forwarding (AF) [RFC 2597]
  - N classes (current standard: N=4)
  - M possible drop preferences w/in class (current standard: M=3)
  - Each classes' traffic handled separately
  - Packet drop "likelihood" increases w/ drop preference

PHB Specs Omit…

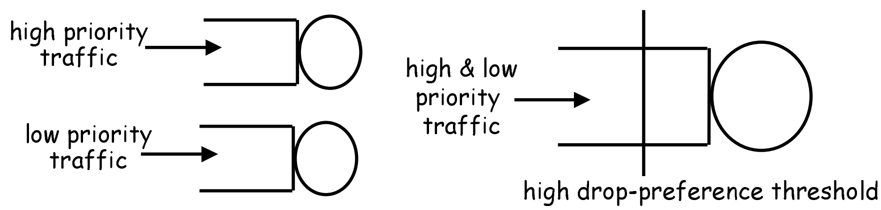
- EF and AF PHBs do not specify mechanism, e.g., not specified:
  - edge classification, shaping or marking policy
  - core router queuing mechansim
  - ranges of rates, relative class/preference service ratios, etc.
- Why are these details omitted?
  - Allow flexibility - as long as specified requirements are met.
  - DiffServ is a new idea - still unclear on which mechanism is best - so standardize later
- Which is better, EF or AF?

Comparing PHB Models [Sahu]

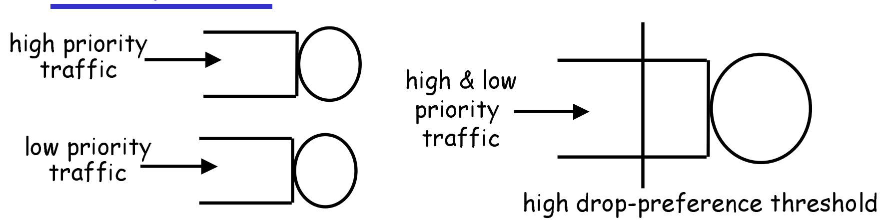
- How does isolating traffic (EF) compare with preferential treatment (AF w/ preferences)?

- Measures:
  - expected loss rate
  - expected delays
- Reqm't: overall buffer / bandwidth fixed

- Queue models:
  - EF: separate queues per class. High priority queue always serviced first (when non-empty)
  - AF: one queue w/ threshold for accepting high drop-preference pkts



Intuition



- Which is
  - better for reducing high-priority delay?
  - for high-priority loss?

- How should buffer be allocated in the 2 models to make a fair comparison?
  - EF: low priority gets its own buffer
  - AF: low priority must share its buffer with high

EF vs. AF Comparison

- Choose buffer partitions and threshold such that low-priority traffic sees similar loss rates in two systems
- Examine impact on high priority traffic
- Main Results for high priority traffic:
  - AF router needs to process 30-70% faster than an EF router to maintain same delays (function of partition point and threshold location)
  - EF router needs only 15% add'l buffer to yield same loss rates to low priority traffic as AF

DiffServ Open Issues

- How to decide "how much" to reserve
- How to do DiffServ for multicast
  - Much more complicated
  - Multicast reservation issues significantly complicated IntServ. What about DiffServ?

Summary: Internet Multimedia

- Internet design:
  - flexible
  - easy to extend
  - difficult to support time-bounded applications
- Approach #1: Build on a best-effort network
  - adaptive applications (quality vs. available bandwidth)
  - deal with loss and jitter (e.g., RTP/RTCP)
- Approach #2: Modify (extend) IP design
  - IntServ: guarantee QoS, but takes lots of state
  - DiffServ: create high and low priority customers - give more to high