QoS (Quality of Service)
Chapter 6
Multimedia Networking

A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in powerpoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

❑ If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)

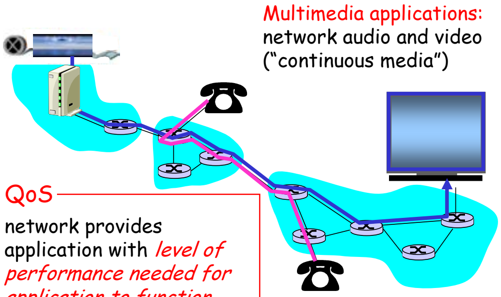
❑ If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK / KWR

Computer Networking: A Top Down Approach Featuring the Internet,
2nd edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2002.

Multimedia, Quality of Service: What is it?

Multimedia applications: network audio and video ("continuous media")



QoS

network provides application with *level of performance needed for application to function.*

MM Networking Applications

Classes of MM applications:

1) Streaming stored audio and video

2) Streaming live audio and video

3) Real-time interactive audio and video

Fundamental characteristics:

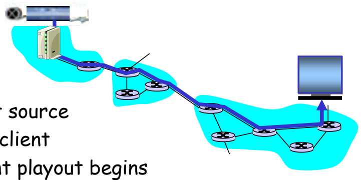
❑ Typically **delay sensitive**
  - ❍ end-to-end delay
  - ❍ delay jitter

❑ But **loss tolerant**: infrequent losses cause minor glitches

❑ Antithesis of data, which are loss intolerant but delay tolerant.

Jitter is the variability of packet delays within the same packet stream

Streaming Stored Multimedia



Streaming:

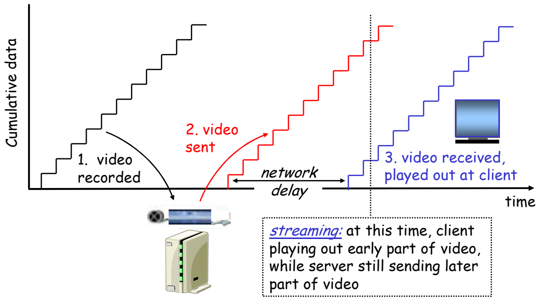
❑ media stored at source

❑ transmitted to client

❑ streaming: client playout begins *before* all data has arrived

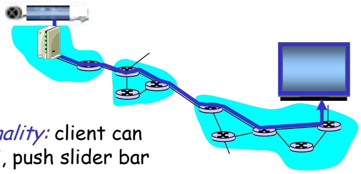
  ❑ timing constraint for still-to-be transmitted data: in time for playout

Streaming Stored Multimedia: What is it?



streaming: at this time, client playing out early part of video, while server still sending later part of video

Streaming Stored Multimedia: Interactivity



❑ *VCR-like functionality:* client can pause, rewind, FF, push slider bar
  - ❍ 10 sec initial delay OK
  - ❍ 1-2 sec until command effect OK
  - ❍ RTSP often used (more later)

❑ timing constraint for still-to-be transmitted data: in time for playout

Streaming Live Multimedia

Examples:

- Internet radio talk show
- Live sporting event

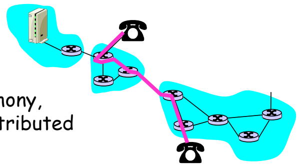
Streaming

- playback buffer
- playback can lag tens of seconds after transmission
- still have timing constraint

Interactivity

- fast forward impossible
- rewind, pause possible!

Interactive, Real-Time Multimedia



- applications: IP telephony, video conference, distributed interactive worlds
- end-end delay requirements:
  - audio: < 150 msec good, < 400 msec OK
    - includes application-level (packetization) and network delays
    - higher delays noticeable, impair interactivity
- session initialization
  - how does callee advertise its IP address, port number, encoding algorithms?

Multimedia Over Today's Internet

TCP/UDP/IP: "best-effort service"

- *no* guarantees on delay, loss

? ? ? ? ? ? But you said multimedia apps requires ? QoS and level of performance to be ? ? effective! ? ?

Today's Internet multimedia applications use application-level techniques to mitigate (as best possible) effects of delay, loss

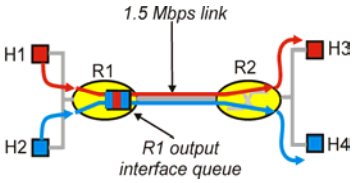
Improving QOS in IP Networks

Thus far: "making the best of best effort"

Future: next generation Internet with QoS guarantees

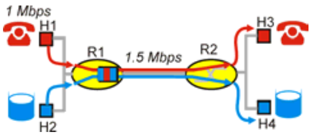
- RSVP: signaling for resource reservations
- Differentiated Services: differential guarantees
- Integrated Services: firm guarantees

- simple model for sharing and congestion studies:



Principles for QOS Guarantees

- Example: 1MbpsI P phone, FTP share 1.5 Mbps link.
  - bursts of FTP can congest router, cause audio loss
  - want to give priority to audio over FTP

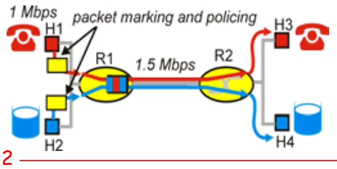


Principle 1

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 (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:
  - similar to ATM UNI (User Network Interface)

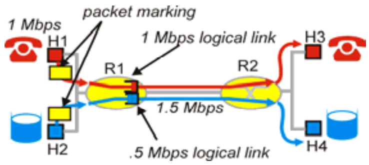


Principle 2

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

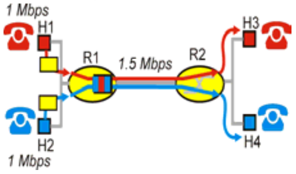


Principle 3

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

Principles for QOS Guarantees (more)

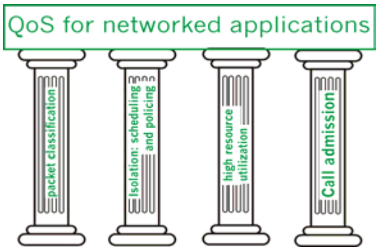
- *Basic fact of life:* can not support traffic demands beyond link capacity



Principle 4

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

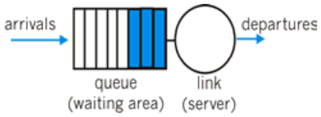
Summary of QoS Principles


QoS for networked applications
packet classification
Isolation: scheduling and policing
high resource utilization
Call admission

Let's next look at mechanisms for achieving this ….

Scheduling And Policing Mechanisms

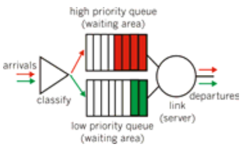
- scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - real-world example?
  - discard policy: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - priority: drop/remove on priority basis
    - random: drop/remove randomly

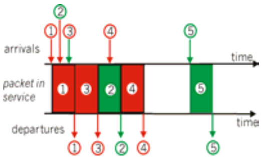


Scheduling Policies: more

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?

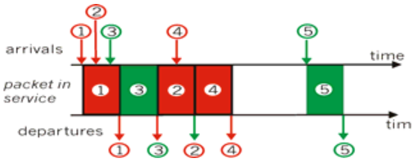




Scheduling Policies: still more

round robin scheduling:

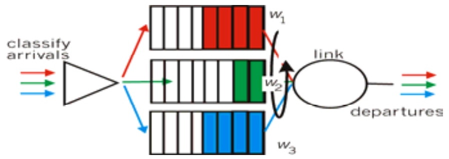
- multiple classes
- cyclically scan class queues, serving one from each class (if available)
- real world example?



Scheduling Policies: still more

Weighted Fair Queuing:

- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example?



Policing Mechanisms

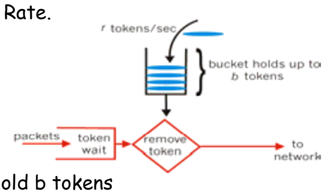
Goal: limit traffic to not exceed declared parameters

Three common-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)

Policing Mechanisms

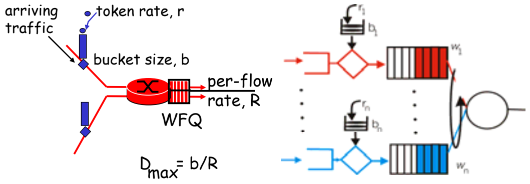
Token Bucket: limit input to specified Burst Size and Average Rate.



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

Policing Mechanisms (more)

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



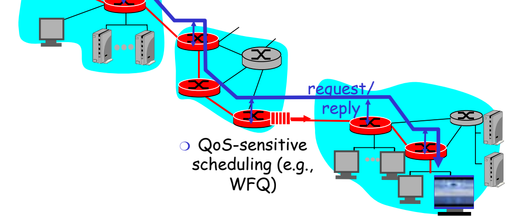
IETF Integrated Services

- architecture for providing QOS guarantees in IP networks for individual application sessions
- resource reservation: routers maintain state info (a la VC) of allocated resources, QoS req's
- admit/deny new call setup requests:

Question: can newly arriving flow be admitted with performance guarantees while not violated QoS guarantees made to already admitted flows?

Intserv: QoS guarantee scenario

- Resource reservation
  - call setup, signaling (RSVP)
  - traffic, QoS declaration
  - per-element admission control



- QoS-sensitive scheduling (e.g., WFQ)

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

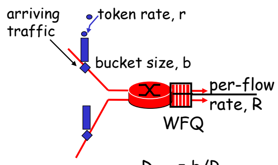
Intserv QoS: Service models [rfc2211, rfc 2212]

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."



$D_{max} = b/R$

IETF Differentiated Services

Concerns with Intserv:

- Scalability: signaling, maintaining per-flow router state difficult with large number of flows
- Flexible Service Models: Intserv has only two classes. Also want "qualitative" service classes
  - "behaves like a wire"
  - relative service distinction: Platinum, Gold, Silver

Diffserv approach:

- simple functions in network core, relatively complex functions at edge routers (or hosts)
- Do't define 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

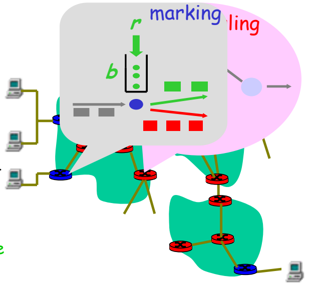
Core router:

- per class traffic management

- buffering and scheduling based on marking at edge

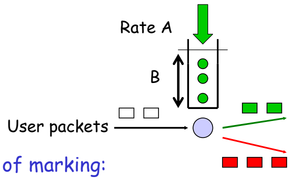
- preference given to in-profile packets

- Assured Forwarding



Edge-router Packet Marking

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

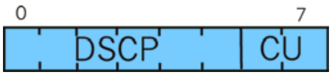


Possible usage of marking:

- 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

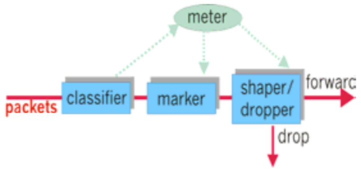
- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits are currently unused



Classification and Conditioning

may be desirable to limit traffic injection rate of some class:

- user declares traffic profile (eg, rate, burst size)
- traffic metered, shaped if non-conforming



MPLS (MultiProtocol Label Switching)

- Defined in RFC 3031
- Different forwarding method
- Similar to VCs (ATM, FR, etc.)
- Called also label switching, tag switching
- New message header (a.k.a tag):
  - 20 bits label
  - 3 bits QoS
  - 1 bit S (stacking)
  - 8 bits TTL
- Supports aggregation of tags called FEC (forwarding equivalence class)

MPLS – VC construction

- Data driven approach: VC is initialized on demand in a recursive way
  - Possible problem of loops
- Control driven approach: when router boots it creates labels for the destination in its routing tables

Signaling in the Internet

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- New requirement: reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- RSVP: Resource Reservation Protocol [RFC 2205]
  - " … allow users to communicate requirements to network in robust and efficient way." i.e., signaling !
- earlier Internet Signaling protocol: ST-II [RFC 1819]

RSVP Design Goals

1. accommodate heterogeneous receivers (different bandwidth along paths)
2. accommodate different applications with different resource requirements
3. make multicast a first class service, with adaptation to multicast group membership
4. leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. control protocol overhead to grow (at worst) linear in # receivers
6. modular design for heterogeneous underlying technologies

RSVP: does not…

- specify how resources are to be reserved
  - rather: a mechanism for communicating needs
- determine routes packets will take
  - that's the job of routing protocols
  - signaling decoupled from routing
- interact with forwarding of packets
  - separation of control (signaling) and data (forwarding) planes

RSVP: overview of operation

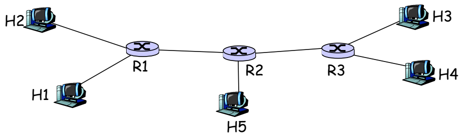
- senders, receiver join a multicast group
  - done outside of RSVP
  - senders need not join group
- sender-to-network signaling
  - *path message:* make sender presence known to routers
  - path teardown: delete sender's path state from routers
- receiver-to-network signaling
  - *reservation message:* reserve resources from sender(s) to receiver
  - reservation teardown: remove receiver reservations
- network-to-end-system signaling
  - path error
  - reservation error

Path msgs: RSVP *sender-to-network* signaling

- path message contents:
  - *address:* unicast destination, or multicast group
  - *flowspec:* bandwidth requirements spec.
  - *filter flag:* if yes, record identities of upstream senders (to allow packets filtering by source)
  - *previous hop:* upstream router/host ID
  - *refresh time:* time until this info times out
- path message: communicates sender info, and reverse-path-to-sender routing info
  - later upstream forwarding of receiver reservations

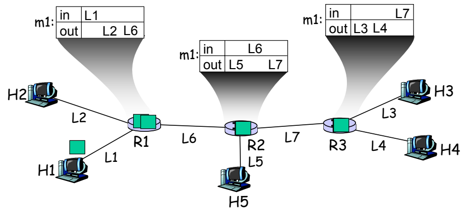
RSVP: simple audio conference

- H1, H2, H3, H4, H5 both senders and receivers
- multicast group m1
- no filtering: packets from any sender forwarded
- audio rate: *b*
- only one multicast routing tree possible



RSVP: building up path state

- H1, …, H5 all send path messages on *m1:*
  (address=*m1*, Tspec=*b*, filter-spec=no-filter,refresh=100)
- Suppose H1 sends first path message



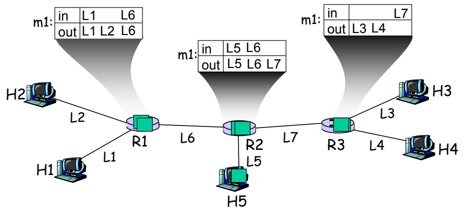
m1:	
in	L1
out	L2 L6

m1:	
in	L6
out	L5 L7

m1:	
in	L7
out	L3 L4

RSVP: building up path state

- next, H5 sends path message, creating more state in routers



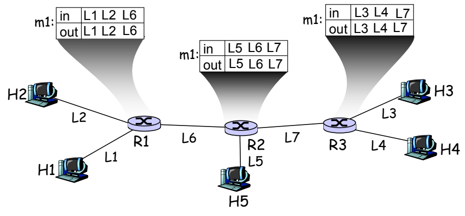
m1:	
in	L1 L6
out	L1 L2 L6

m1:	
in	L5 L6
out	L5 L6 L7

m1:	
in	L7
out	L3 L4

RSVP: building up path state

- H2, H3, H5 send path msgs, completing path state tables



m1:	
in	L1 L2 L6
out	L1 L2 L6

m1:	
in	L5 L6 L7
out	L5 L6 L7

m1:	
in	L3 L4 L7
out	L3 L4 L7

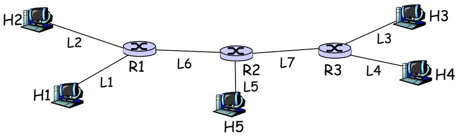
reservation msgs: *receiver-to-network* signaling

- reservation message contents:
  - *desired bandwidth:*
  - *filter type:*
    - no filter: any packets address to multicast group can use reservation
    - fixed filter: only packets from specific set of senders can use reservation
    - dynamic filter: senders who's packets can be forwarded across link will change (by receiver choice) over time.
  - *filter spec*
- reservations flow upstream from receiver-to-senders, reserving resources, creating additional, *receiver-related* state at routers

RSVP: *receiver* reservation example 1

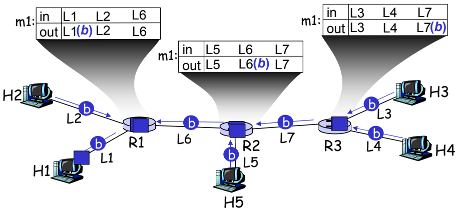
H1 wants to receive audio from all other senders

- H1 reservation msg flows uptree to sources
- H1 only reserves enough bandwidth for 1 audio stream
- reservation is of type "no filter" – any sender can use reserved bandwidth



RSVP: *receiver* reservation example 1

- H1 reservation msgs flows uptree to sources
- routers, hosts reserve bandwidth b needed on downstream links towards H1



m1:

in	L1	L2	L6
out	L1(*b*)	L2	L6

m1:

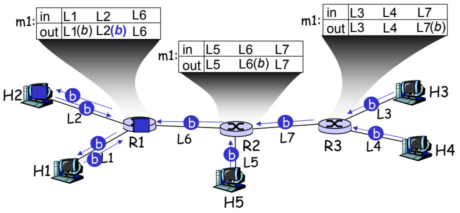
in	L5	L6	L7
out	L5	L6(*b*)	L7

m1:

in	L3	L4	L7
out	L3	L4	L7(*b*)

RSVP: *receiver* reservation example 1 (more)

- next, H2 makes no-filter reservation for bandwidth *b*
- H2 forwards to R1, R1 forwards to H1 and R2 (?)
- R2 takes no action, since *b* already reserved on L6



m1:

in	L1	L2	L6
out	L1(*b*)	L2(*b*)	L6

m1:

in	L5	L6	L7
out	L5	L6(*b*)	L7

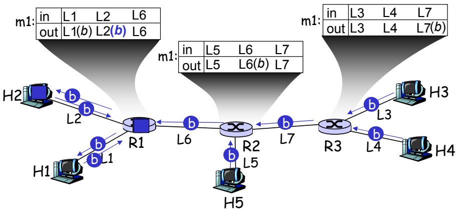
m1:

in	L3	L4	L7
out	L3	L4	L7(*b*)

RSVP: *receiver* reservation: issues

What if multiple senders (e.g., H3, H4, H5) over link (e.g., L6)?

- arbitrary interleaving of packets
- L6 flow policed by leaky bucket: if H3+H4+H5 sending rate exceeds b, packet loss will occur



m1:

in	L1	L2	L6
out	L1(*b*)	L2(*b*)	L6

m1:

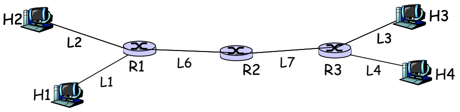
in	L5	L6	L7
out	L5	L6(*b*)	L7

m1:

in	L3	L4	L7
out	L3	L4	L7(*b*)

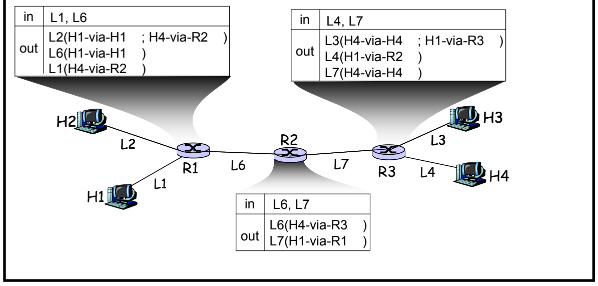
RSVP: example 2

- H1, H4 are only senders
  - send *path messages* as before, indicating filtered reservation
  - Routers store upstream senders for each upstream link
- H2 will want to receive from H4 (only)



RSVP: example 2

- H1, H4 are only senders
  - send *path messages* as before, indicating filtered reservation



in	L1, L6

out	L2(H1-via-H1 ; H4-via-R2)
	L6(H1-via-H1)
	L1(H4-via-R2)

in	L4, L7

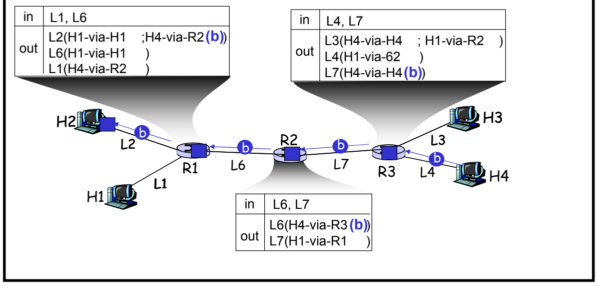
out	L3(H4-via-H4 ; H1-via-R3)
	L4(H1-via-R2)
	L7(H4-via-H4)

in	L6, L7

out	L6(H4-via-R3)
	L7(H1-via-R1)

RSVP: example 2

- receiver H2 sends reservation message for source H4 at bandwidth *b*
  - propagated upstream towards H4, reserving *b*



in	L1, L6

out	L2(H1-via-H1 ;H4-via-R2 (b))
	L6(H1-via-H1)
	L1(H4-via-R2)

in	L4, L7

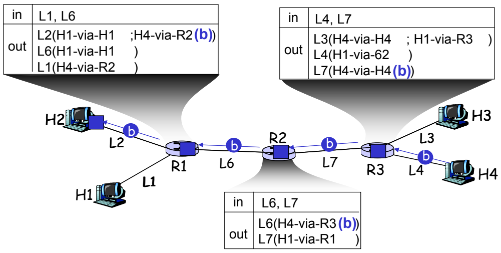
out	L3(H4-via-H4 ; H1-via-R2)
	L4(H1-via-62)
	L7(H4-via-H4 (b))

in	L6, L7

out	L6(H4-via-R3 (b))
	L7(H1-via-R1)

RSVP: *soft-state*

- senders periodically resend path msgs to refresh (maintain) state
- receivers periodically resend resv msgs to refresh (maintain) state
- path and resv msgs have TTL field, specifying refresh interval



in	L1, L6

out	L2(H1-via-H1 ;H4-via-R2 (b))
	L6(H1-via-H1)
	L1(H4-via-R2)

in	L4, L7

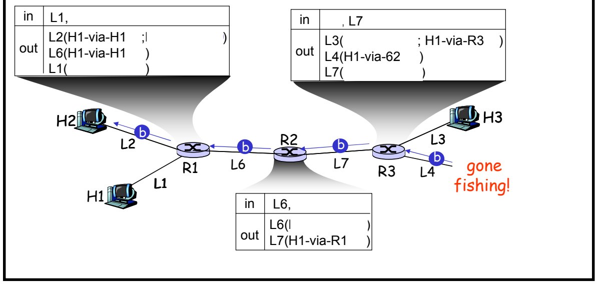
out	L3(H4-via-H4 ; H1-via-R3)
	L4(H1-via-62)
	L7(H4-via-H4 (b))

in	L6, L7

out	L6(H4-via-R3 (b))
	L7(H1-via-R1)

RSVP: *soft-state*

- suppose H4 (sender) leaves without performing teardown
- eventually state in routers will timeout and disappear!



in	L1,

out	L2(H1-via-H1 ;I)
	L6(H1-via-H1)
	L1( )

in	, L7

out	L3(; H1-via-R3)
	L4(H1-via-62)
	L7( )

in	L6,

out	L6(I)
	L7(H1-via-R1)

The many uses of reservation/path refresh

- recover from an earlier lost refresh message
  - expected time until refresh received must be longer than timeout interval! (short timer interval desired)
- Handle receiver/sender that goes away without teardown
  - Sender/receiver state will timeout and disappear
- Reservation refreshes will cause new reservations to be made to a receiver from a sender who has joined since receivers last reservation refresh
  - E.g., in previous example, H1 is only receiver, H3 only sender. Path/reservation messages complete, data flows
  - H4 joins as sender, nothing happens until H3 refreshes reservation, causing R3 to forward reservation to H4, which allocates bandwidth

RSVP: reflections

- multicast as a “first class” service
- receiver-oriented reservations
- use of soft-state