

RSVP: does not...

- ❑ specify how resources are to be reserved
 - ❑ does not provide a flowspec or admission control
 - ❑ 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 (e.g., packet scheduling)

Part 1.1 18

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

Part 1.1 19

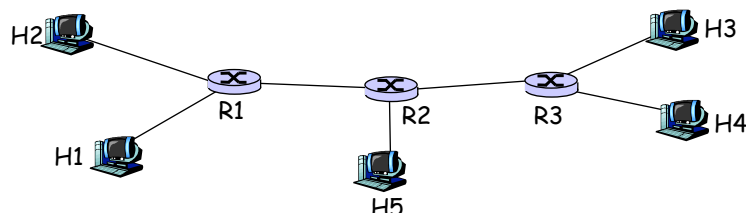
Path msgs: RSVP *sender-to-network* signaling

- **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

Part 1.1 20

RSVP: simple audio conference

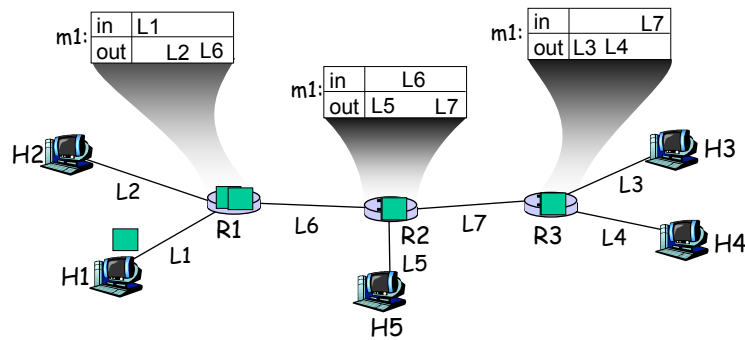
- H1, H2, H3, H4, H5 both senders and receivers
- multicast group m1
- no filtering: packets from any sender forwarded
- audio rate: b
- only one multicast routing tree possible



Part 1.1 21

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



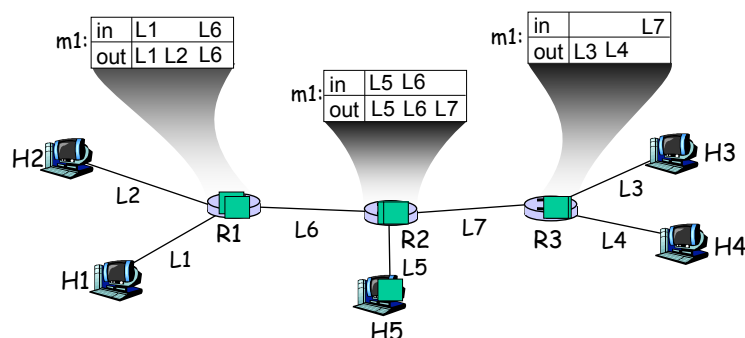
In: senders are downstream; Out: receivers are downstream

Part 1.1

22

RSVP: building up path state

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



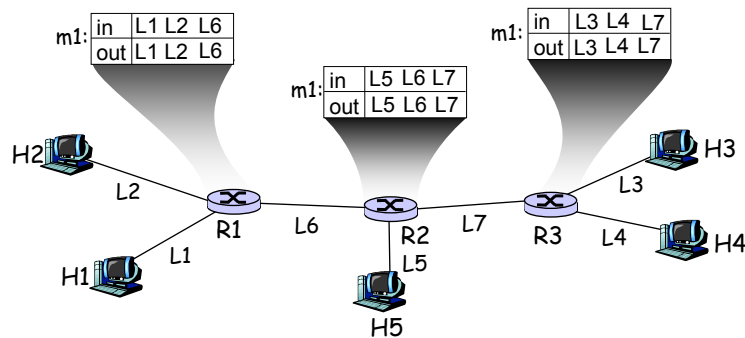
In: senders are downstream; Out: receivers are downstream

Part 1.1

23

RSVP: building up path state

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



In: senders are downstream; Out: receivers are downstream

Part 1.1 24

reservation msgs: receiver-to-network signaling

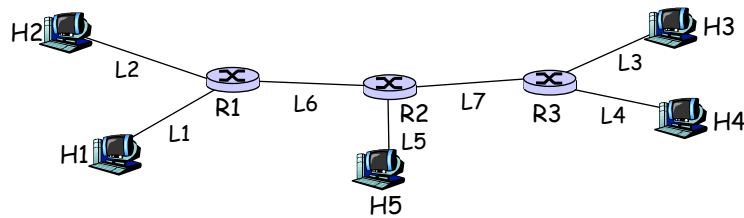
- reservation message contents:
 - *desired bandwidth:*
 - *filter type:*
 - no filter: any packets address to multicast group can use reservation
 - fixed filter: only packets from specific set of senders can use reservation
 - dynamic filter: senders who's packets can be forwarded across link will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

Part 1.1 25

RSVP: *receiver* reservation example 1

H1 wants to receive audio from all other senders

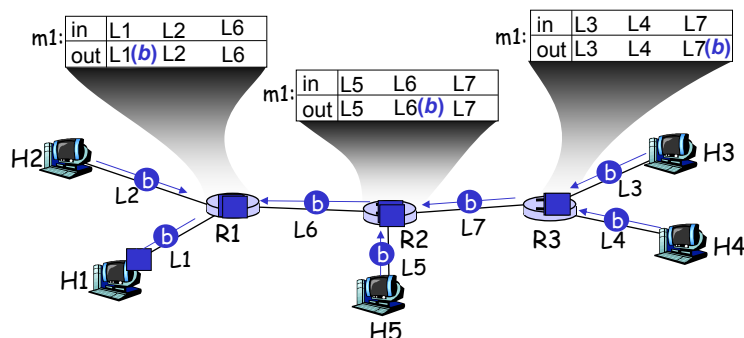
- H1 reservation msg flows up-tree to sources
- H1 only reserves enough bandwidth for 1 audio stream
- reservation is of type "no filter" - any sender can use reserved bandwidth



Part 1.1 26

RSVP: *receiver* reservation example 1

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

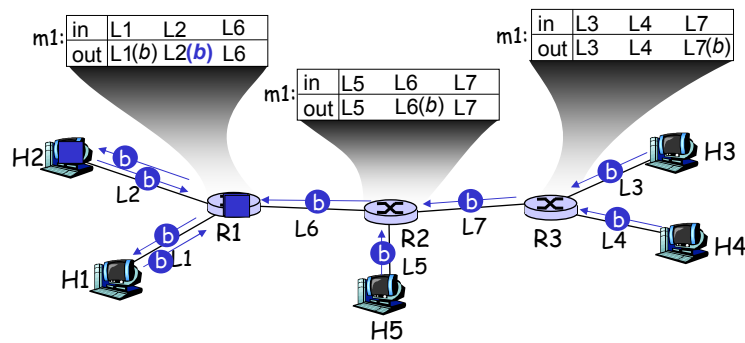


In: senders are downstream; Out: receivers are downstream

Part 1.1 27

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 (?)
- **Q**: What action does R2 take?
- **A**: Nothing, since b already reserved on L6

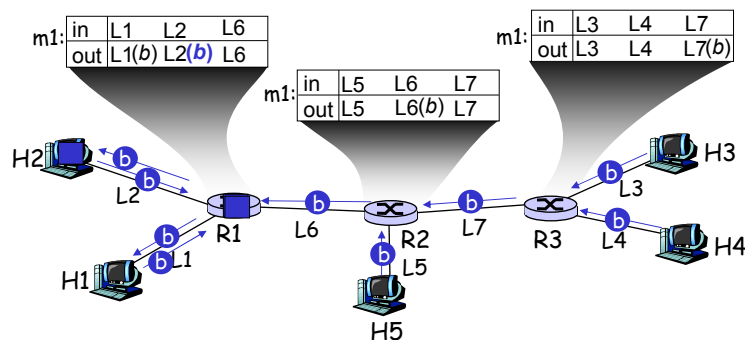


In: senders are downstream; Out: receivers are downstream

Part 1.1 28

RSVP: *receiver* reservation: issues

- Q**: 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

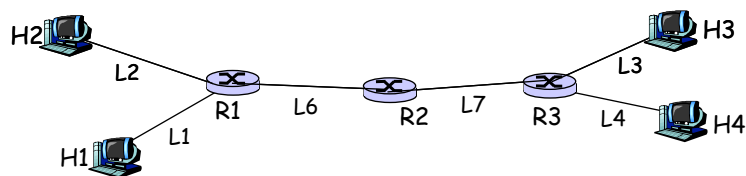


In: senders are downstream; Out: receivers are downstream

Part 1.1 29

RSVP: example 2

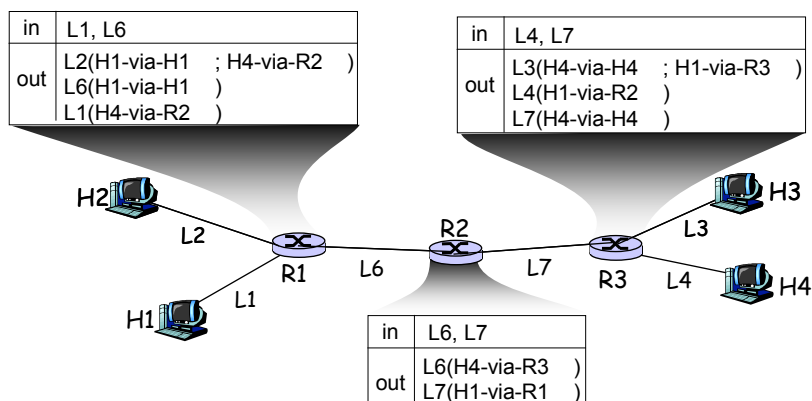
- H1, H4 are only senders
 - send *path messages* as before, indicating filtered reservation
 - Routers store upstream senders for each upstream link
- H2 will want to receive from H4 (only)



Part 1.1 30

RSVP: example 2

- H1, H4 are only senders
 - send *path messages* with filtered reservation

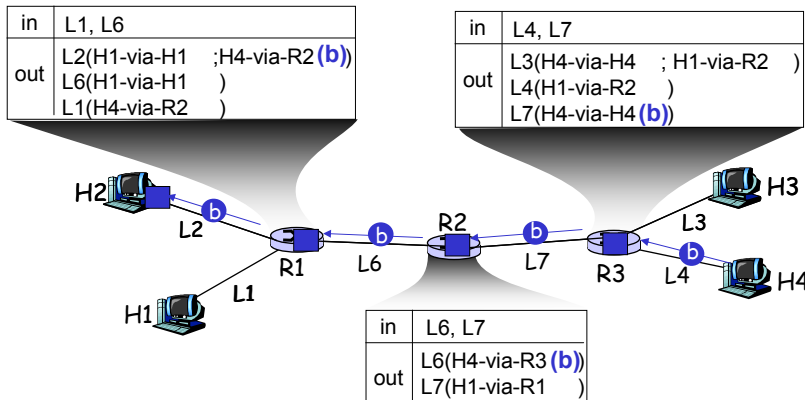


In: senders downstream; Out: receivers downstream; Via: route to senders

Part 1.1 31

RSVP: example 2

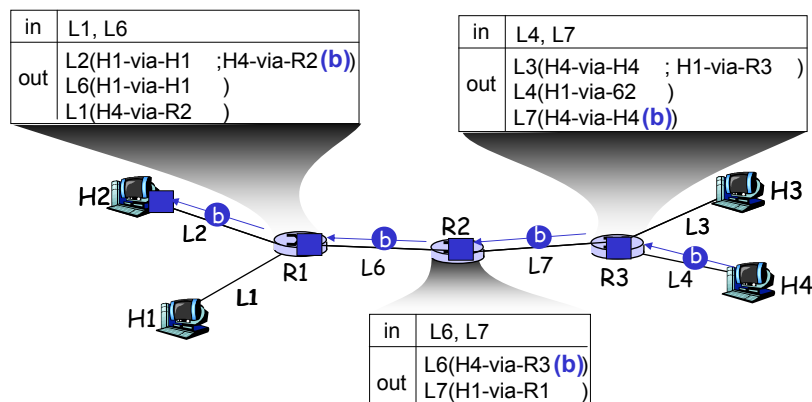
- receiver H2 sends reservation message for source H4 at bandwidth b
 - propagated upstream towards H4, reserving b



Part 1.1 32

RSVP: soft-state

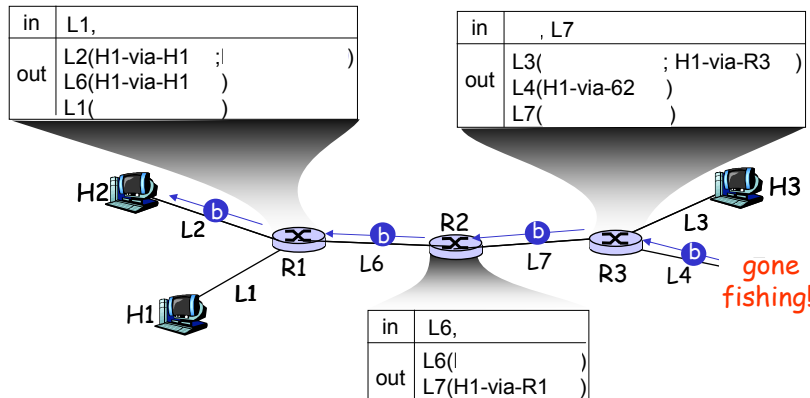
- senders periodically resend path msgs to refresh (maintain) state
- receivers periodically resend resv msgs to refresh (maintain) state
- path and resv msgs have TTL field, specifying refresh interval



Part 1.1 33

RSVP: *soft-state*

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



Part 1.1 34

The many uses of reservation/path refresh

- Recover from an earlier lost refresh message
- 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 H1 refreshes reservation, causing R3 to forward reservation to H4, which allocates bandwidth

Part 1.1 35

RSVP: reflections

- ❑ multicast as a "first class" service
- ❑ receiver-oriented reservations
- ❑ use of soft-state

Part 1.1 36

Signaling Discussion: SS7 vs RSVP

- ❑ Similarities
 - path can't change without a lot of hassle since end-end path state stored in each router/switch
 - all need (*are*) setup signaling protocols
- ❑ Differences
 - RSVP: bandwidth sharing among receivers/senders
 - Definitely ss7 is out of band, RSVP signaling separate from data flow, but all in IP packets
 - switch dies: what do other switches do?
 - SS7: cascades signaling removes state. Without signaling/explicit coordination, orphaned state will disappear via timeout
 - RSVP: end systems re-establish path state (in simplest case). Path re-establishment is "free" with RSVP as part of refresh mechanisms. Need explicit re-signaling in SS7
 - Multicast is "first class object" in RSVP
 - Receiver orientation in RSVP

Part 1.1 37

1: Separation of control and data

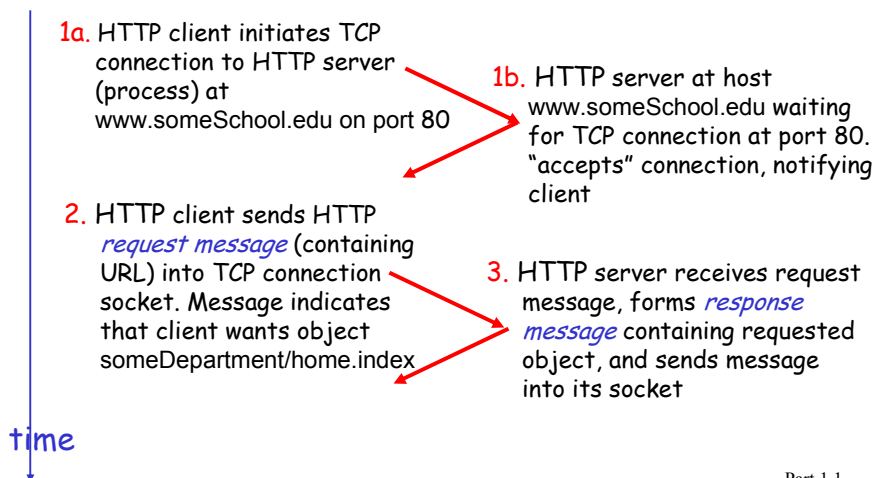
- **PSTN (public switched telephone network):**
 - SS7 (packets-switched control network) separate from (circuit-switched) call trunk lines
 - earlier tone-based (in-band signaling)
- **Internet:**
 - RSVP (signaling) separate from routing, forwarding.
 - http: in-band signaling; ftp: out-of-band signaling

Part 1.1 38

Internet: HTTP - inband signaling

Suppose user enters URL

`www.someSchool.edu/someDepartment/home.index`



Part 1.1 39

Nonpersistent HTTP (cont.)

- time ↓
4. HTTP server closes TCP connection.
 5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects
 6. Steps 1-5 repeated for each of 10 jpeg objects

Part 1.1 40

HTTP request message

request line
(GET, POST,
HEAD commands) → GET /somedir/page.html HTTP/1.0

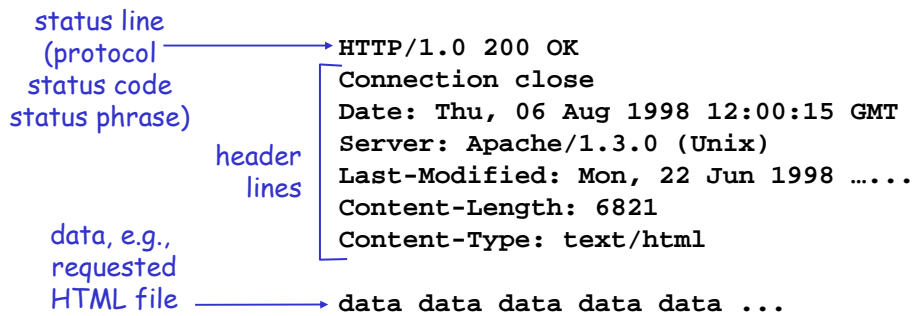
header lines → Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr

Carriage return,
line feed → (extra carriage return, line feed)
indicates end
of message

Note: request msg just a signaling msg (no data)

Part 1.1 41

HTTP response message

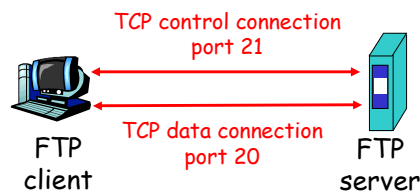


- Note:** response msg mixes signaling and data
- request, response msgs exchanged over *single* TCP connection

Part 1.1 42

FTP: separate control, data connections

- FTP client contacts FTP server at port 21
- client obtains authorization over control connection
- client browses remote directory via commands sent over control connection.
- when server receives file transfer command server opens new TCP data connection to client
- after transferring one file, server closes connection.



- server opens 2nd TCP data connection to transfer another file.
- control connection: "out of band" signaling
- FTP server maintains "state": current directory, earlier authentication

Part 1.1 43

Separate control, data: why (or why not)?

Why?

- ❑ allows concurrent control + data
- ❑ allows perform authentication at control level
- ❑ simplifies processing of data/control streams- higher throughput
- ❑ provide QoS appropriate for control/data streams

Why not?

- ❑ separate channels complicate management, increases resource requirements
- ❑ can increase latency, e.g., http - two tcp connections vs one.

Part 1.1 44

Discussion

- ❑ How to use postal network to efficiently implement video-on-demand?
- ❑ What we want: high bandwidth, low latency
- ❑ What we have:
 - Data plane: CDs mailed through postal networks
 - High bandwidth, high latency
 - Control plane: ??

Part 1.1 45

Maintaining network state

state: information *stored* in network nodes by network protocols

- ❑ updated when network "conditions" change
- ❑ stored in multiple nodes
- ❑ often associated with end-system generated call or session
- ❑ examples:
 - RSVP router maintain lists of upstream sender IDs, downstream receiver reservations
 - ATM switch maintains list of VCs: bandwidth allocation, VPI/VCI input-output mapping
 - VPI/VCI: Virtual Path/Circuit Identifier,
 - TCP: sequence numbers, timer values, RTT estimates

Part 1.1 46

State: senders, receivers

- ❑ **sender:** network node that (re)generates signaling (control) msgs to install, keep-alive, remove state from other nodes
- ❑ **receiver:** node that creates, maintains, removes state based on signaling msgs received from sender

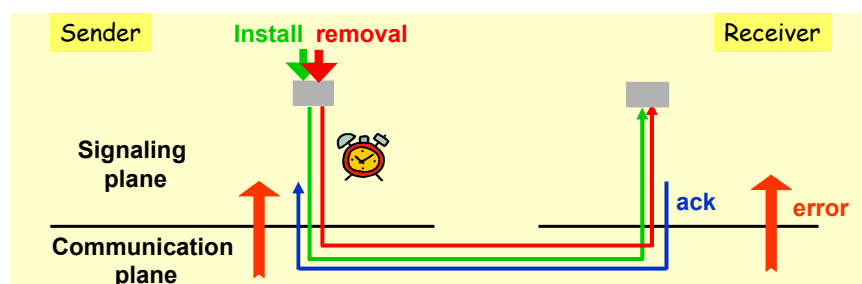
Part 1.1 47

Hard-state

- state *installed* by receiver on receipt of *setup msg* from sender
- state *removed* by receiver on receipt of *teardown msg* from sender
- *default assumption*: state valid unless told otherwise
 - in practice: failsafe-mechanisms (to remove orphaned state) in case of sender failure e.g., receiver-to-sender "heartbeat": is this state still valid ?
- examples:
 - TCP [Q: retransmission of data or ACK if no activity?]
 - Q.2931 (ATM Signaling)
 - ST-II (Internet hard-state signaling)

Part 1.1 48

Hard-state signaling



- reliable signaling
- state removal by request
- requires additional error handling
 - e.g., sender failure

Part 1.1 49

Soft-state

- state *installed* by receiver on receipt of *setup (trigger) msg* from sender (typically, an endpoint)
 - sender also sends periodic *refresh msg*: indicating receiver should continue to maintain state
- state *removed* by receiver via timeout, in absence of refresh msg from sender
- default assumption: state becomes invalid unless refreshed
 - in practice: explicit state removal (*teardown*) msgs also used
- examples:
 - RSVP, RTP, IGMP

Part 1.1 50

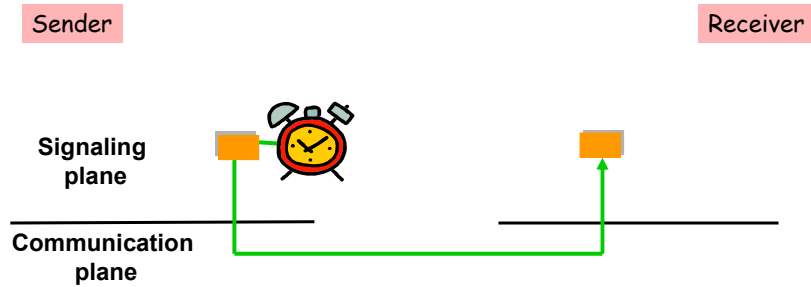
Soft-state signaling



- best effort signaling

Part 1.1 51

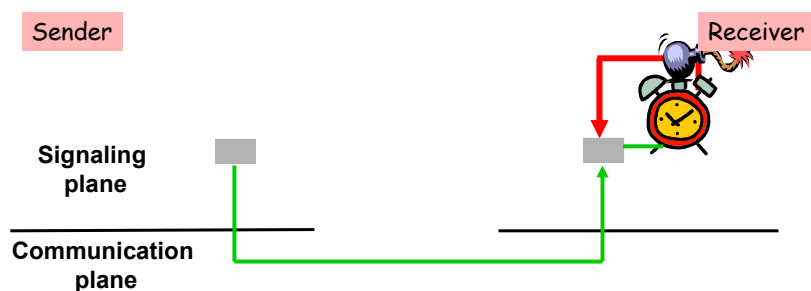
Soft-state signaling



- ❑ best effort signaling
- ❑ refresh timer, periodic refresh

Part 1.1 52

Soft-state signaling



- ❑ best effort signaling
- ❑ refresh timer, periodic refresh
- ❑ state time-out timer, state removal only by time-out

Part 1.1 53

Soft-state: claims

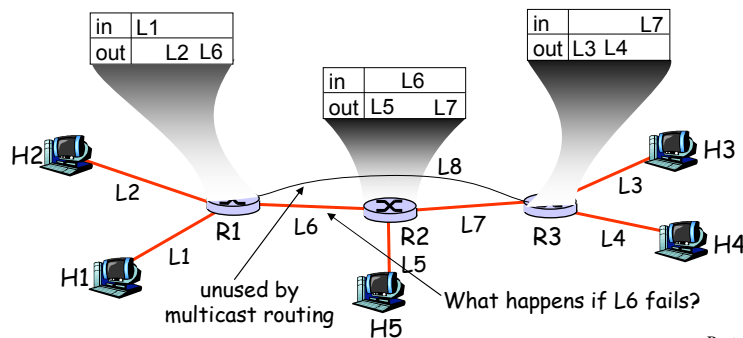
- "Systems built on soft-state are robust" [Raman 99]
- "Soft-state protocols provide .. greater robustness to changes in the underlying network conditions..." [Sharma 97]
- "obviates the need for complex error handling software" [Balakrishnan 99]

What does this mean?

Part 1.1 54

Soft-state: "easy" handling of changes

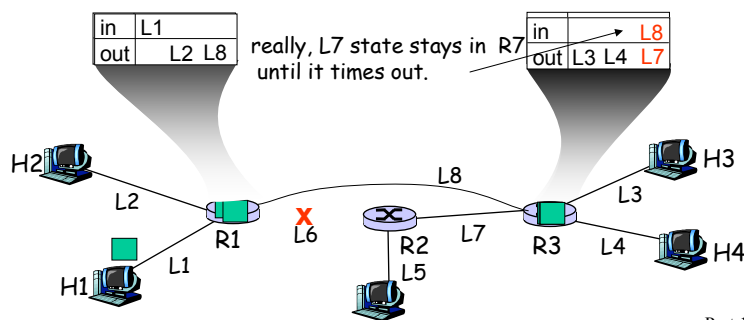
- **Periodic refresh:** if network "conditions" change, refresh will re-establish state under new conditions
- example: RSVP/routing interaction: if routes change (nodes fail) RSVP PATH refresh will *re-establish* state along new path



Part 1.1 55

Soft-state: "easy" handling of changes

- L6 goes down, multicast routing reconfigures but...
- H1 data no longer reaches H3, H4, H5 (no sender or receiver state for L8)
- H1 refreshes PATH, establishes *new* state for L8 in R1, R3
- H4 refreshes RESV, propagates upstream to H1, establishes new receiver state for H4 in R1, R3



Part 1.1 56

Soft-state: "easy" handling of changes

- "recovery" performed transparently to end-system by normal refresh procedures
- no need for network to signal failure/change to end system, or end system to respond to specific error
- less signaling (volume, types of messages) than hard-state from network to end-system but...
- more signaling (volume) than hard-state from end-system to network for refreshes

Part 1.1 57

Soft-state: refreshes

- refresh msgs serve many purposes:
 - **trigger**: first time state-installation
 - **refresh**: refresh state known to exist ("I am still here")
 - <lack of refresh>: remove state ("I am gone")
- challenge: all refresh msgs unreliable
 - would like triggers to result in state-installation asap
 - enhancement: add receiver-to-sender refresh_ACK for triggers
 - e.g., see "Staged Refresh Timers for RSVP"