

Design Principles

Goals:

- ❑ Identify, study common architectural components, protocol mechanisms, approaches do we find in network architectures?
- ❑ *Synthesis*: big picture

7 design principles:

- ❑ Separation of data, control
- ❑ Hard state versus soft state
- ❑ Randomization
- ❑ Indirection
- ❑ Network virtualization / overlays
- ❑ Resource sharing
- ❑ Design for scale

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

Internet: HTTP – inband signaling

Suppose user enters URL

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

1a. HTTP client initiates TCP connection to HTTP server (process) at `www.someSchool.edu` on port 80

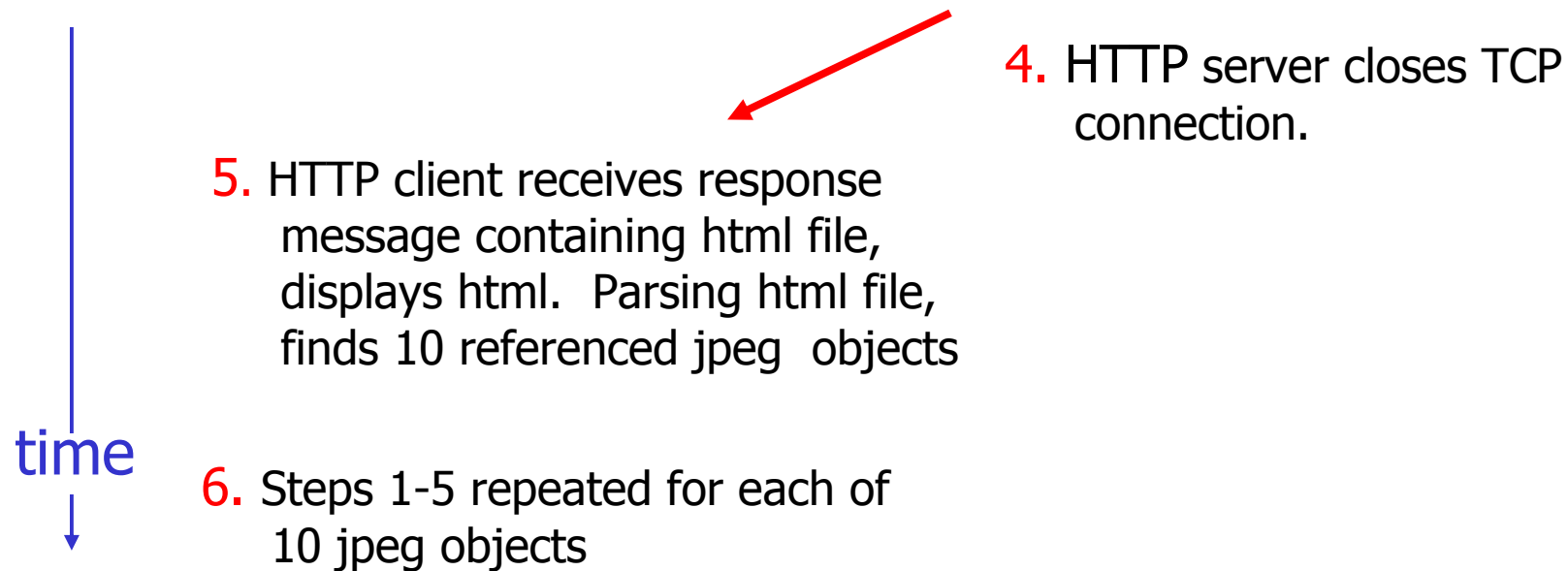
1b. HTTP server at host `www.someSchool.edu` waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object `someDepartment/home.index`

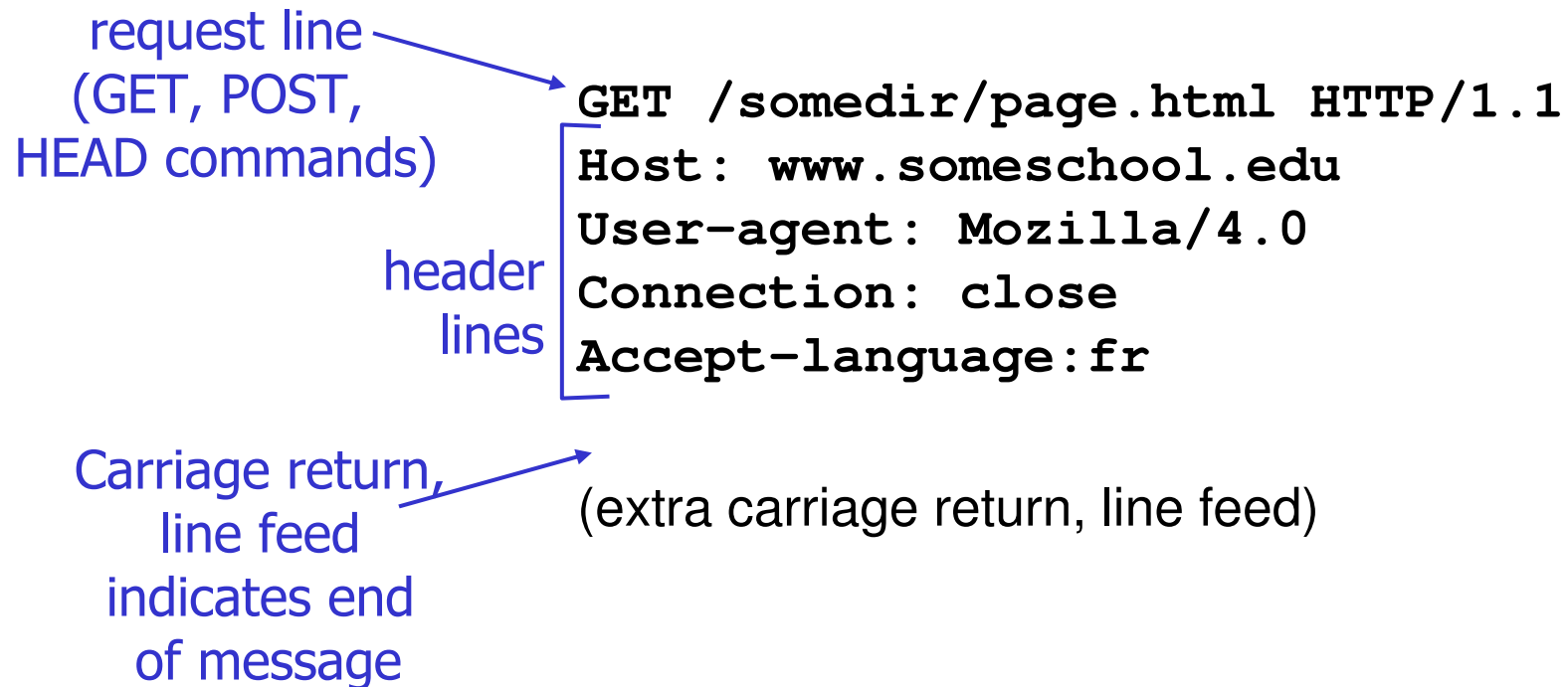
3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time
↓

Internet: HTTP – inband signaling (2)

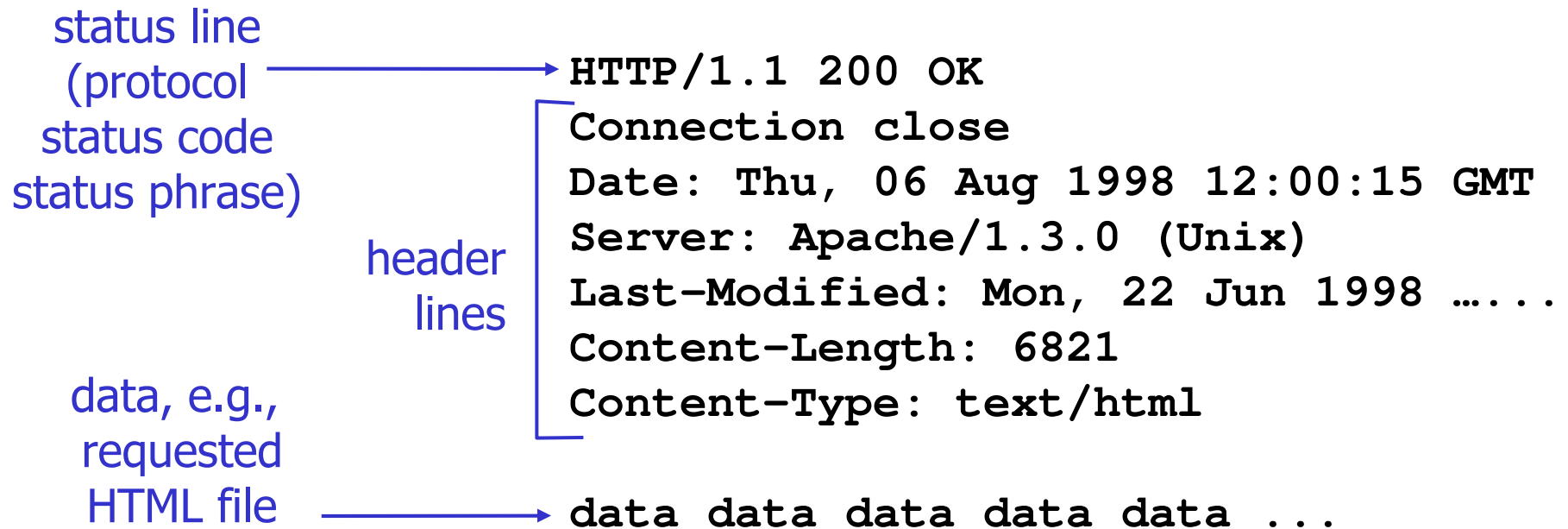


HTTP request message



Note: Request msg typically just a signaling msg (no data)

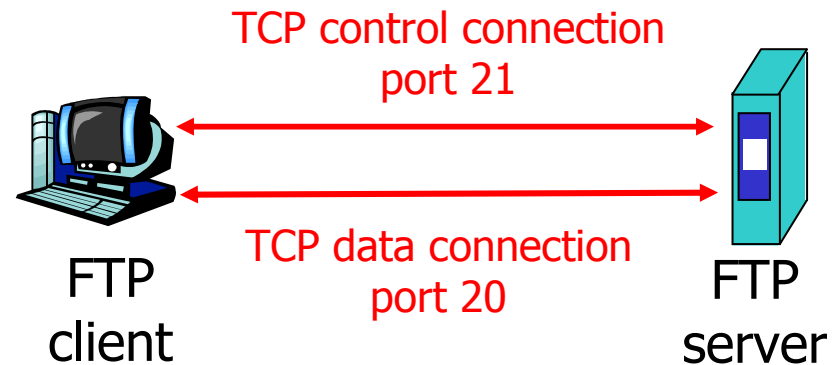
HTTP response message



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

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

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 top connections vs. one

2: 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
 - TCP: Sequence numbers, timer values, RTT estimates

State: Sender, receiver

- ❑ **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

Hard-state

- ❑ State *installed* by receiver on receipt of *setup msg* from sender
- ❑ State *removed* by receiver on receipt of *teardown msg* from sender

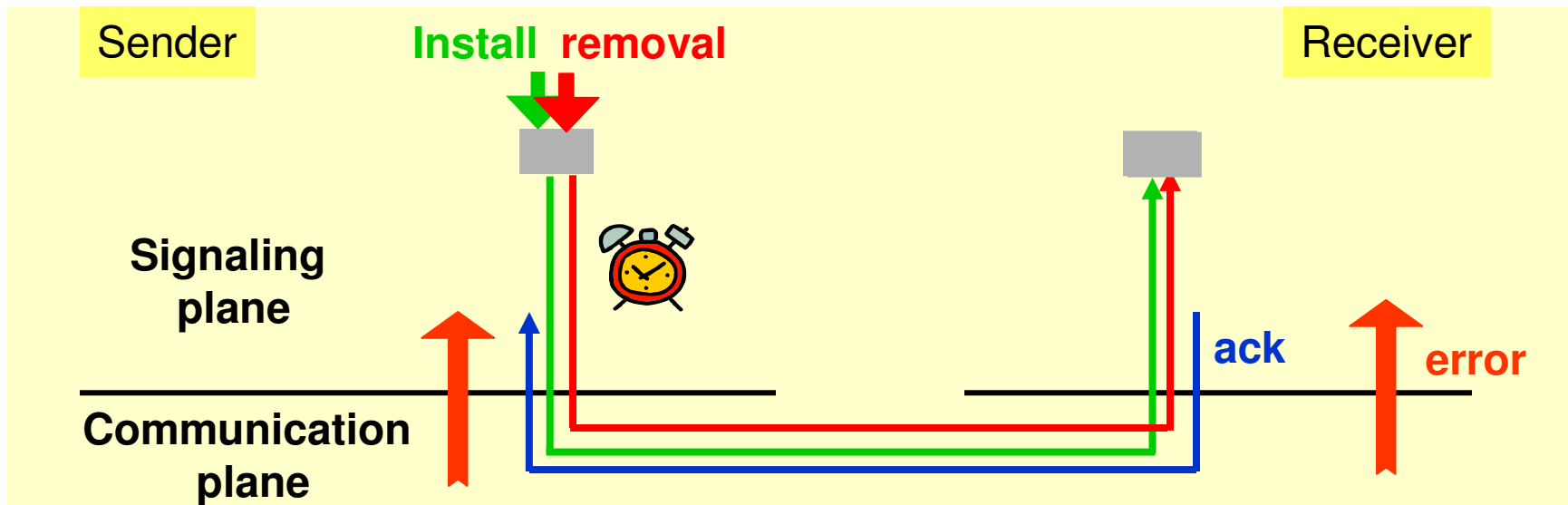
- ❑ *Default assumption*: state valid unless told otherwise
 - In practice: Failsafe-mechanisms (to remove orphaned state) in case of sender: e.g., receiver-to-sender “heartbeat”: Is this state still valid ?

- ❑ Examples:
 - TCP

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

Hard-state signaling

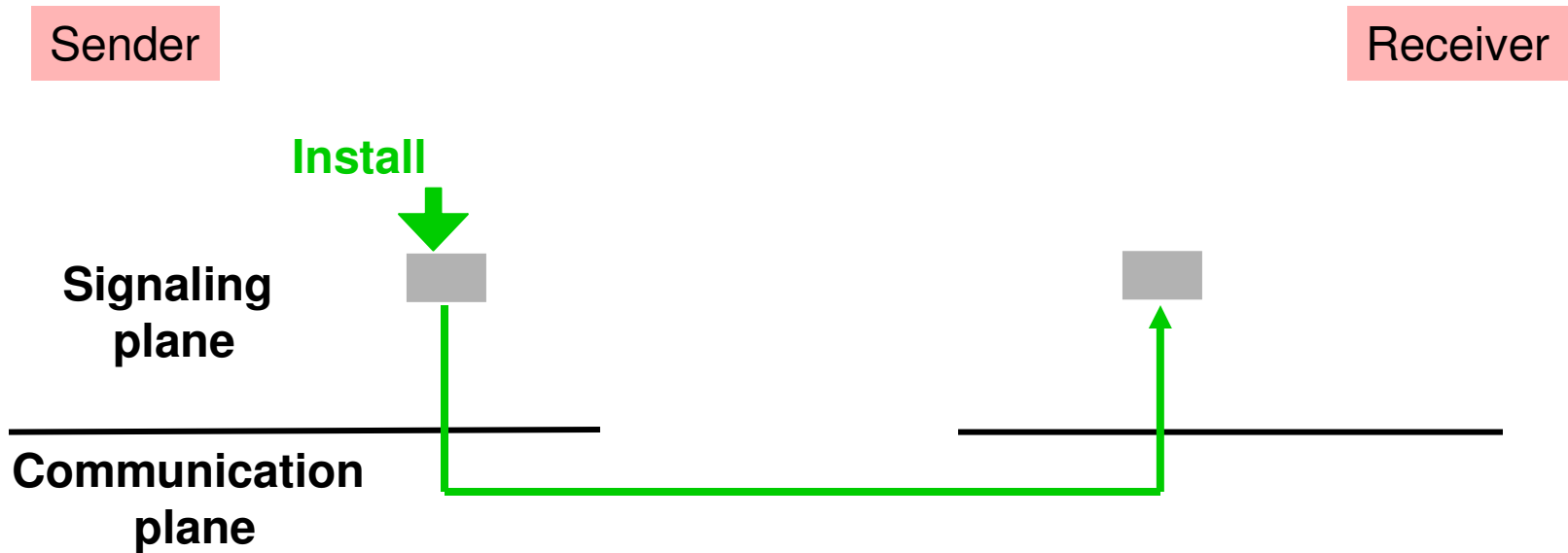


- Reliable signaling
- State removal by request
- Requires additional error handling
 - E.g., sender failure

Soft-state

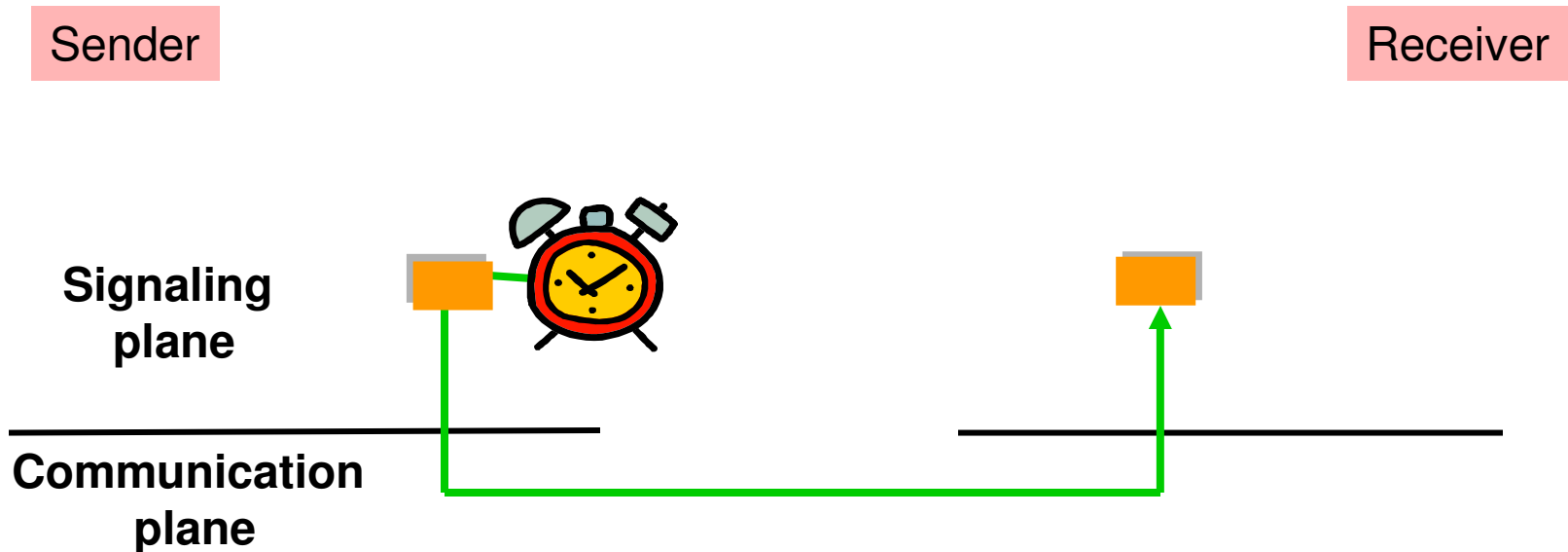
- ❑ State *installed* by receiver on receipt of *setup (trigger) msg* from sender (typically, an endpoint)
Sender also sends periodic *refresh msg* indicating receiver should continue to maintain state
- ❑ State *removed* by receiver via timeout, in absence of refresh msg from sender
- ❑ *Default assumption:*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

Soft-state signaling



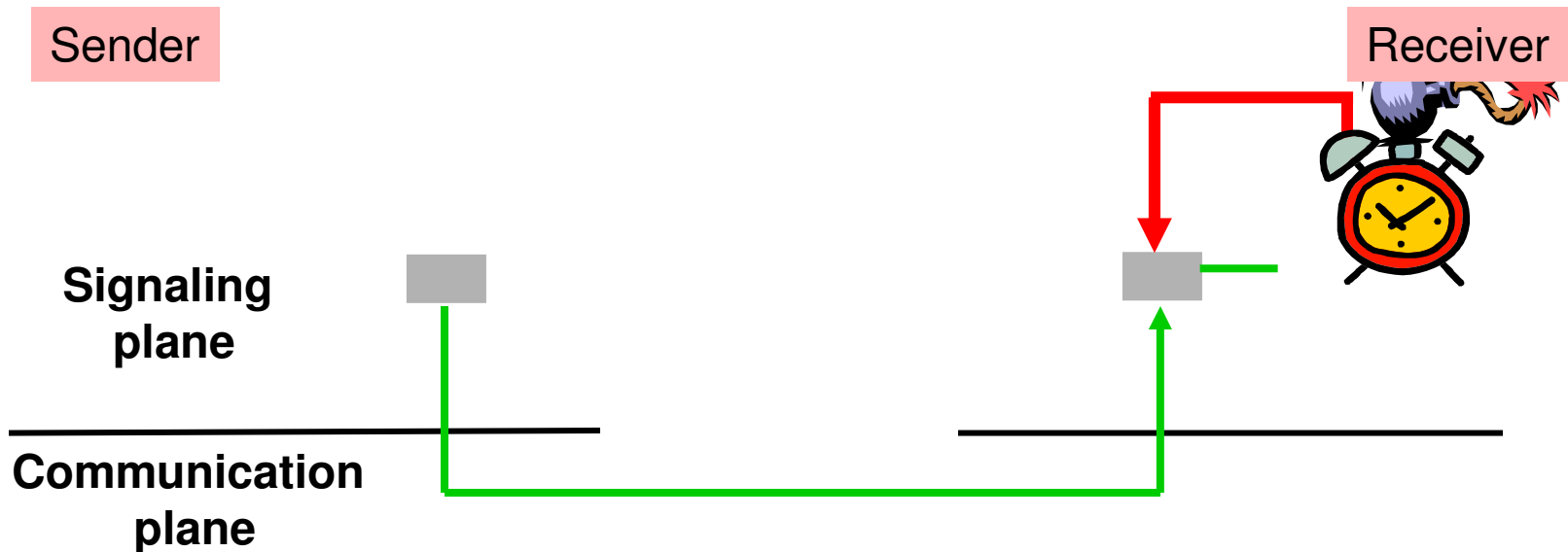
- Best effort signaling

Soft-state signaling



- Best effort signaling
- Refresh timer, periodic refresh

Soft-state signaling



- ❑ Best effort signaling
- ❑ Refresh timer, periodic refresh
- ❑ State time-out timer, state removal only by time-out

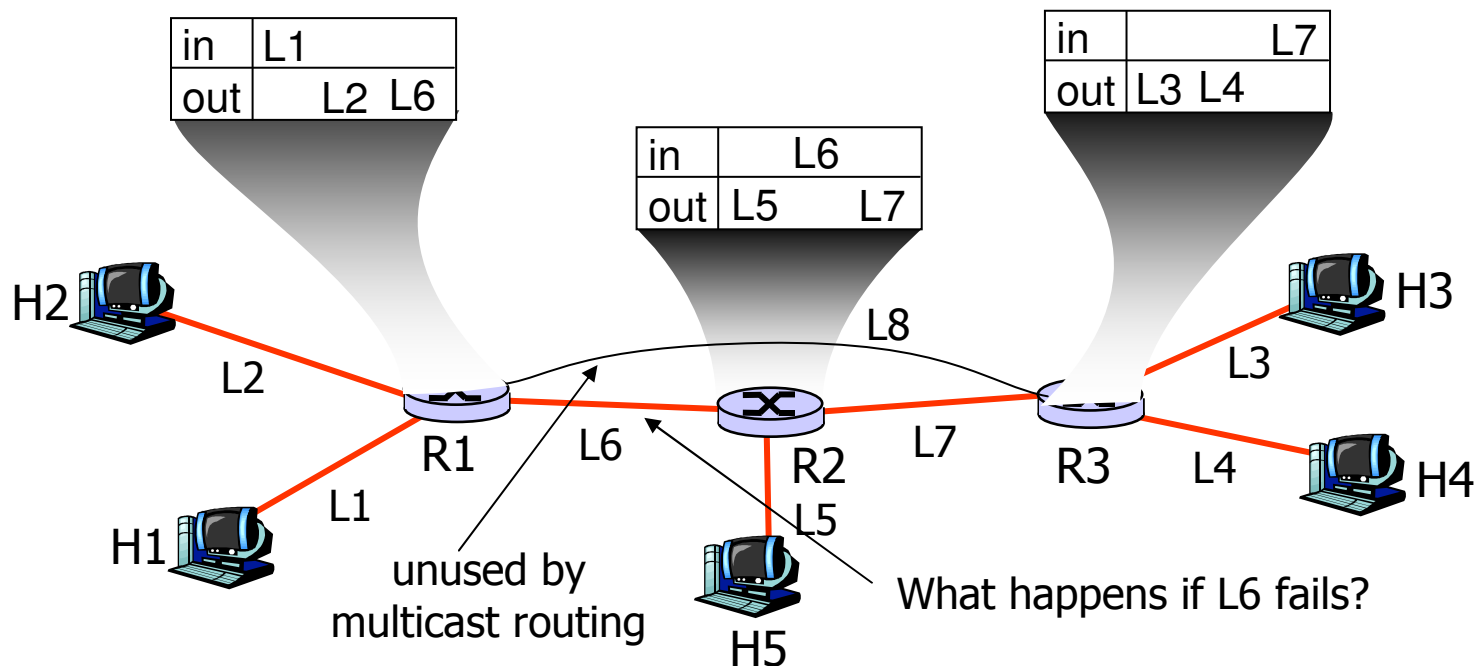
Soft-state: Claims

- ❑ “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?

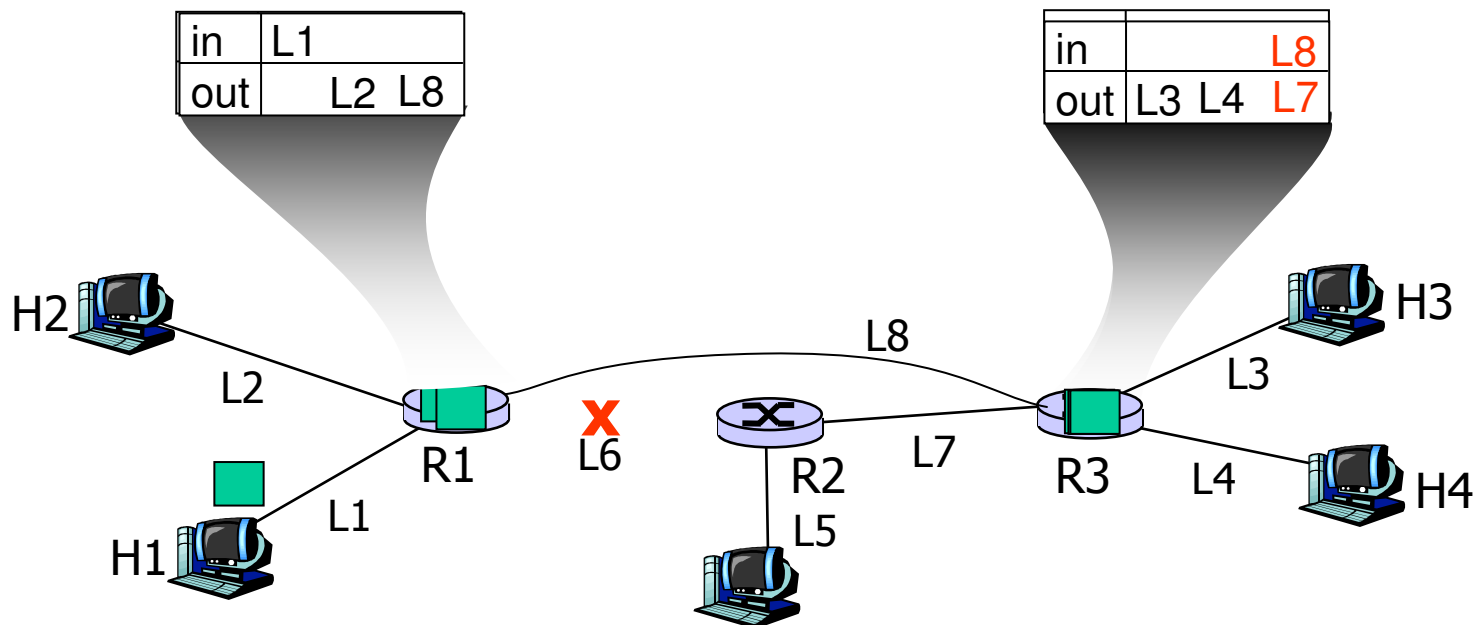
Soft-state: "Easy" handling of changes

- ❑ **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



Soft-state: "Easy" handling of changes

- ❑ L6 goes down, multicast routing reconfigures but...
- ❑ H1 data no longer reaches H3, H3, 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



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

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"

Soft-state: Setting timer values

Q: How to set refresh/timeout timers

- ❑ State-timeout interval = $n * \text{refresh-interval}$ -timeout
 - What value of n to choose?
- ❑ Will determine amount of signaling traffic, responsiveness to change
 - Small timers: fast response to changes, more signaling
 - Long timers: slow response to changes, less signaling
- ❑ Ultimately: Consequence of slow/fast response, msg loss probability will dictate appropriate timer values

How do we model/evaluate?

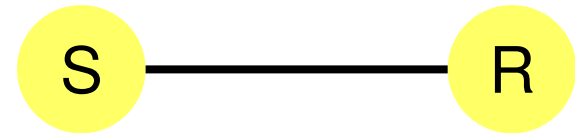
Metrics

- ❑ **Inconsistency ratio** – fraction time participating nodes disagree
- ❑ **Signaling overhead** – average # of messages during session lifetime

- ❑ Measures of robustness?
 - Convergence time
- ❑ Complexity?

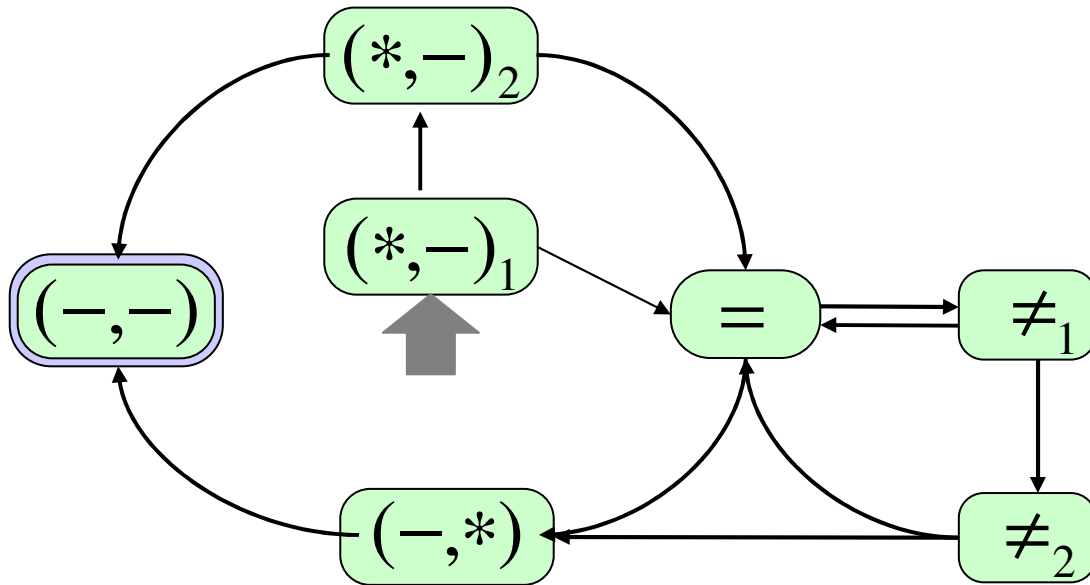
Single hop model

- ❑ Sender, receiver
- ❑ Single state variable
- ❑ State lifetime – $\exp(\mu)$
- ❑ Updates – Poisson(λ)
- ❑ Timers – exponentially distributed
 - Refresh - $1/T$
 - State expiration – $1/X$
- ❑ Link: Delay $\exp(1/D)$, loss prob. p



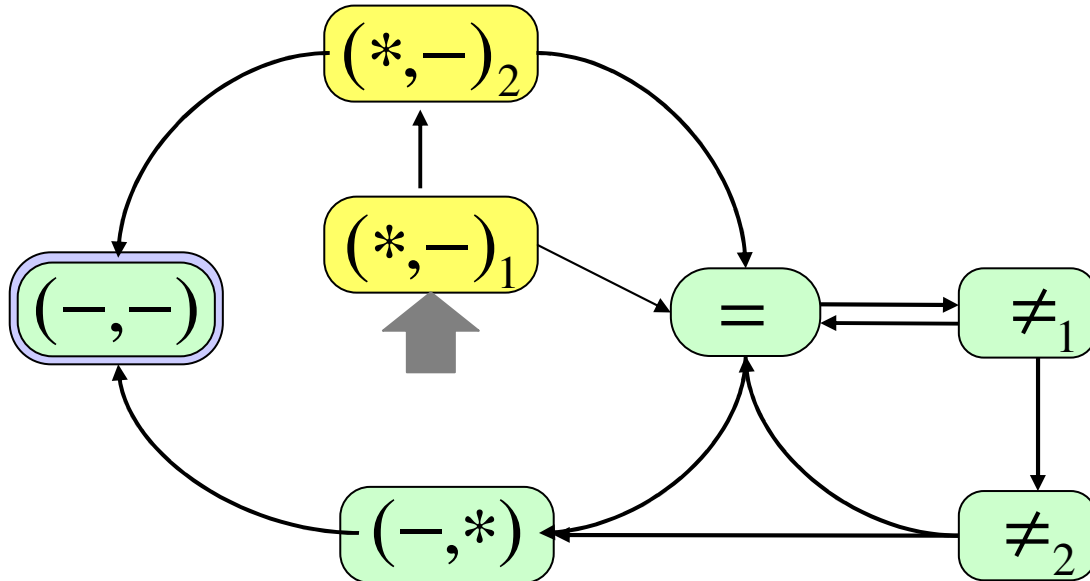
Transient Markov model

Model for SS (Ji03)



Model for SS

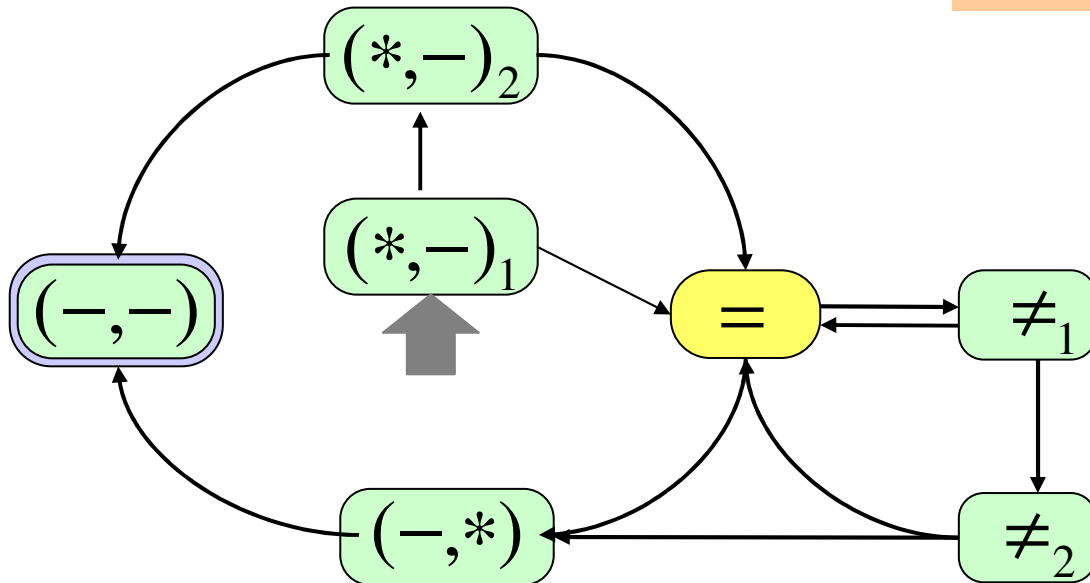
$(*, -)$: signaling state generated at sdr, not installed at rcvr



Model for SS

$(*, -)$: signaling state generated at sdr, not installed at rcvr

$=$: signaling state consistent at sdr/rcvr

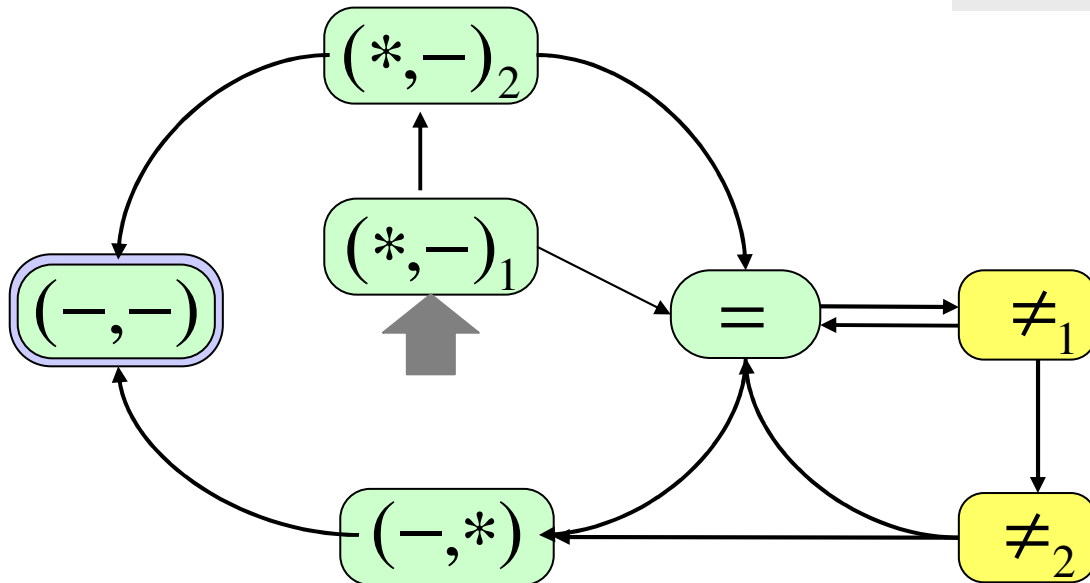


Model for SS

$(*, -)$: signaling state generated at sdr, not installed at rcvr

$=$: signaling state consistent at sdr/rcvr

\neq : signaling state inconsistent at sdr/rcvr

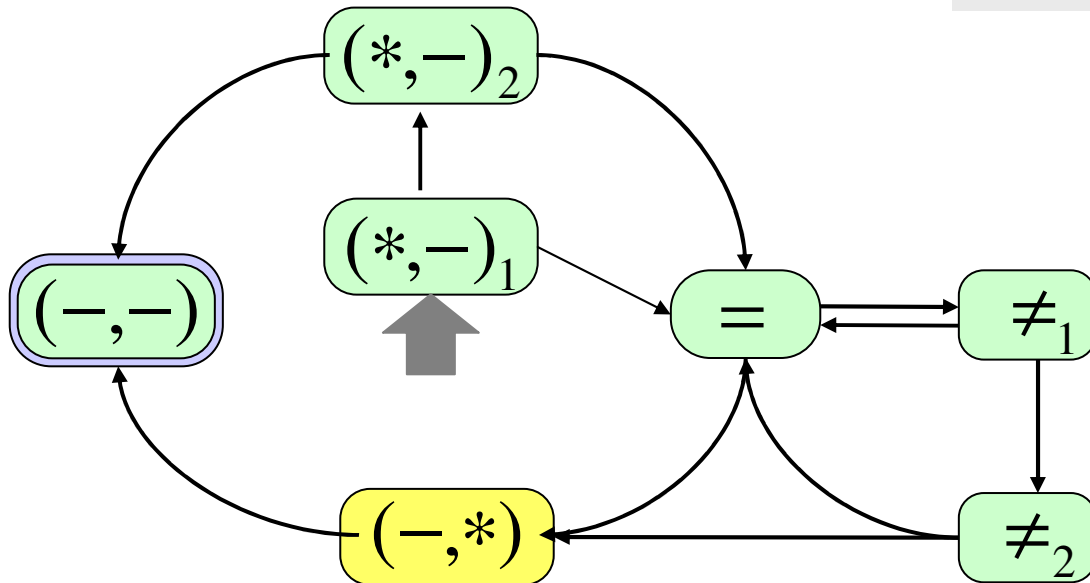


Model for SS

$(*, -)$: signaling state generated at sdr, not installed at rcvr

$=$: signaling state consistent at sdr/rcvr

\neq : signaling state inconsistent at sdr/rcvr



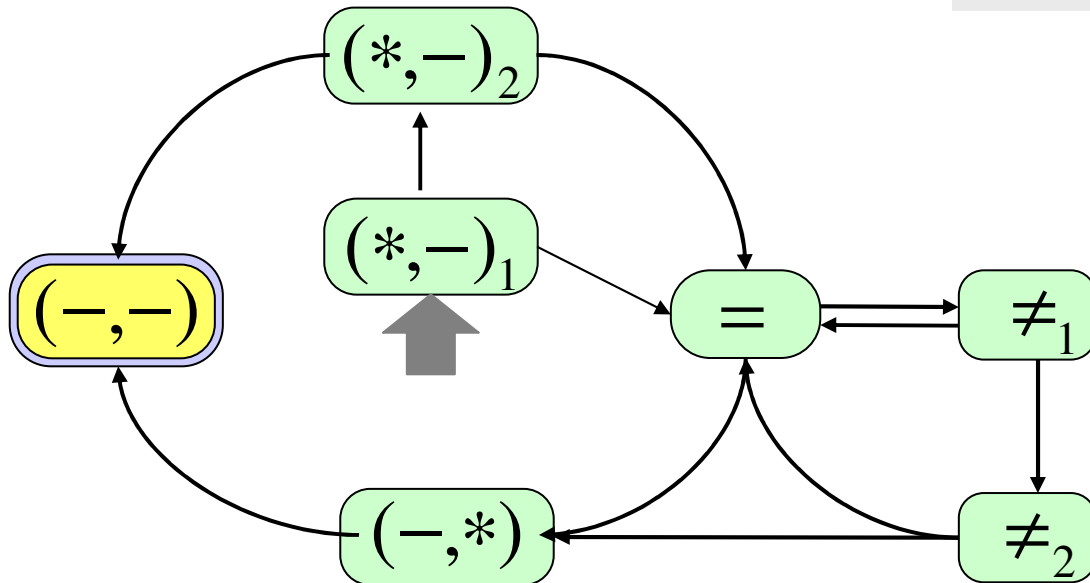
$(-, *)$: signaling state removed at sender, present at receiver

Model for SS

$(*, -)$: signaling state generated at sdr, not installed at rcvr

$=$: signaling state consistent at sdr/rcvr

\neq : signaling state inconsistent at sdr/rcvr



$(-, -)$: signaling state removed at sdr/rcvr

$(-, *)$: signaling state removed at sender, present at receiver

Model for SS

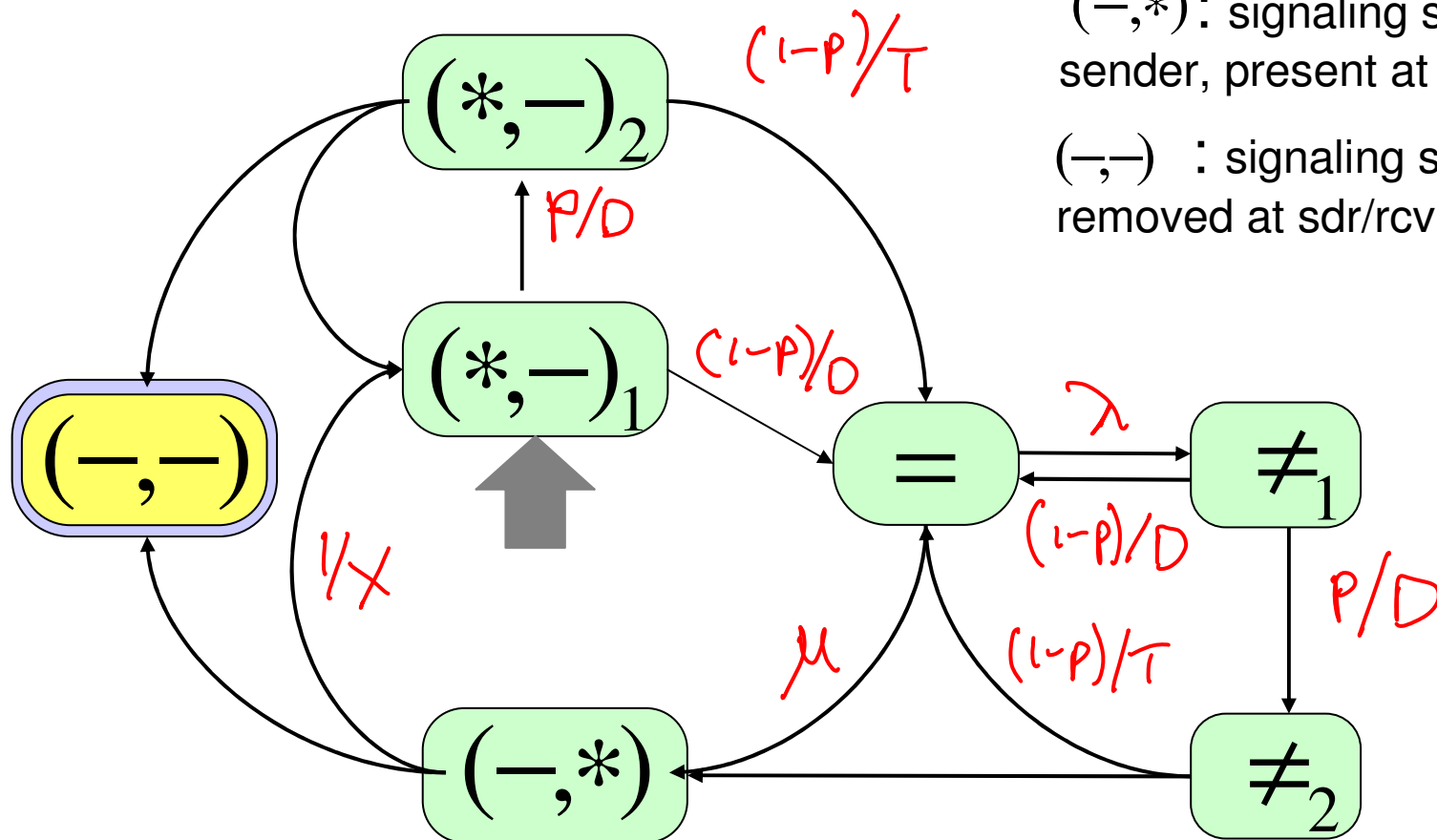
$(*, -)$: signaling state generated at sdr, not installed at rcvr

$=$: signaling state consistent at sdr/rcvr

\neq : signaling state inconsistent at sdr/rcvr

$(-, *)$: signaling state removed at sender, present at receiver

$(-, -)$: signaling state removed at sdr/rcvr



Performance metrics (SS)

- Inconsistency ratio –

$$\delta = 1 - \pi_{=}$$

- Signaling overhead

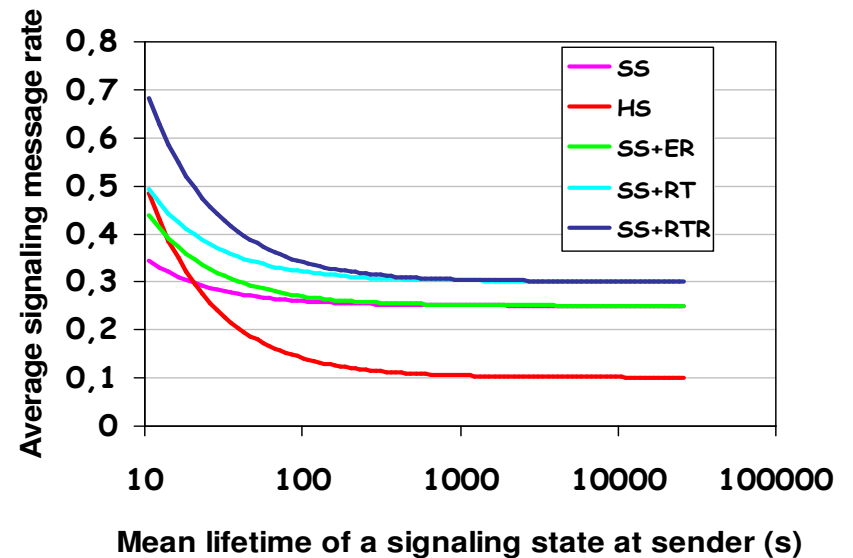
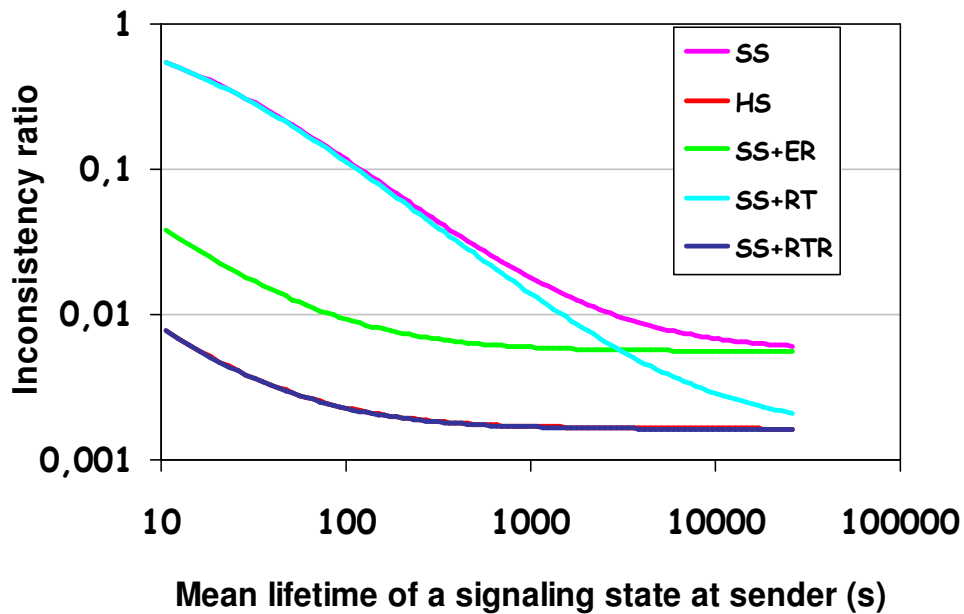
$$\Gamma = (1 + \lambda + 1/T) / \mu$$

Parameter settings

- ❑ Mean lifetime – 30 min.
- ❑ Refresh timer, $T=5\text{sec}$
- ❑ State timer, $X = 15 \text{ sec}$
- ❑ Update rate – $1/20\text{sec}$
- ❑ Signal loss rate – 2%

Motivated by Kazaa

Impact of state lifetime



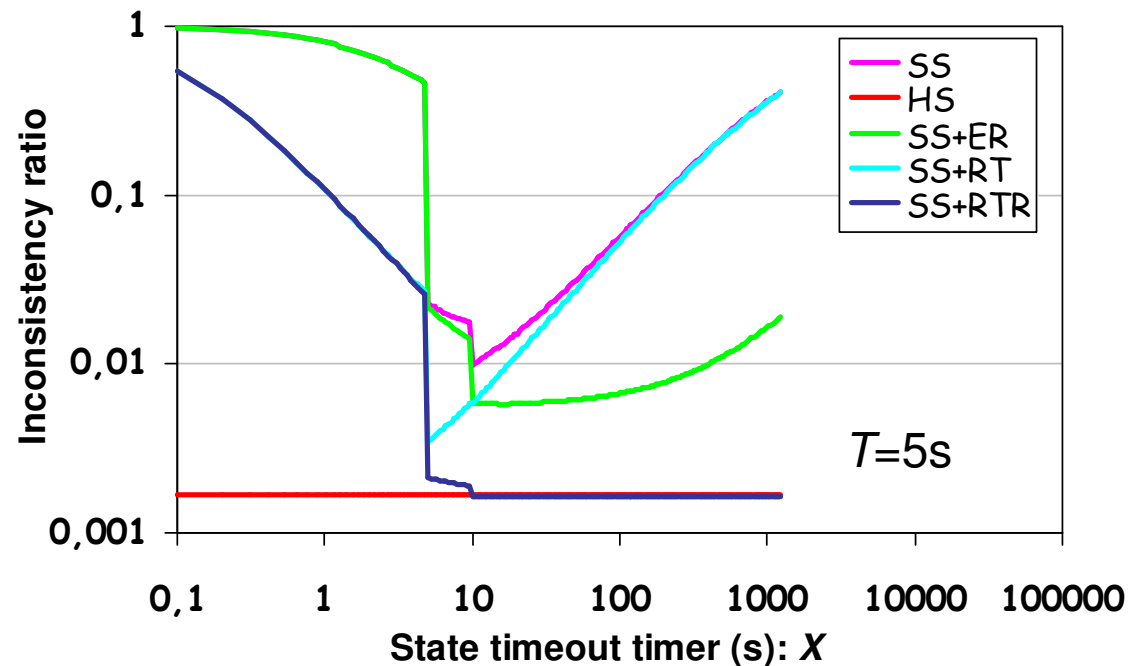
- ❑ Inconsistency, overhead decrease as state life-time increases
- ❑ Explicit removal improves consistency with little additional overhead

Soft-state: Setting timer values

Q: How to set refresh/timeout timers

- ❑ State-timeout interval = $n * \text{refresh-interval}$ -timeout
 - What value of n to choose?
- ❑ Will determine amount of signaling traffic, responsiveness to change
 - Small timers: fast response to changes, more signaling
 - Long timers: slow response to changes, less signaling
- ❑ Ultimately: Consequence of slow/fast response, msg loss probability will dictate appropriate timer values

Impact of state timeout timer



- $X < T$: inconsistency high (premature state removal)
- $X > 2T$: increasing $X \Rightarrow$ increasing inconsistency for SS, SS+ER, SS+RT (due to orphan state)
- $X = 2T$: sweet spot

Hard-state versus soft-state: Discussion

Q: which is preferable and why?

Hard state:

- Better if message OH really high
- Potentially greater consistency
- System wide coupling -> difficult to analyze

Soft state:

- Robustness, shorter convergence times
- Implicit reliability
- Easier error recovery
- Easily decomposed -> simpler analysis