Chair for Network Architectures and Services
Institute for Informatics
TU München – Prof. Carle, Dr. Fuhrmann

Master Kurs Rechnernetze Computer Networks IN2097

Prof. Dr.-Ing. Georg Carle
Dr. Thomas Fuhrmann

Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chair for Network Architectures and Services
Institute for Informatics
TU München – Prof. Carle, Dr. Fuhrmann

Maintaining network state

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 routers maintain lists of upstream sender IDs, downstream receiver reservations
  - ATM switches maintain lists of VCs: bandwidth allocations, VCI/VPI input-output mappings
  - TCP: Sequence numbers, timer values, RTT estimates

Soft-state

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

Hard-state

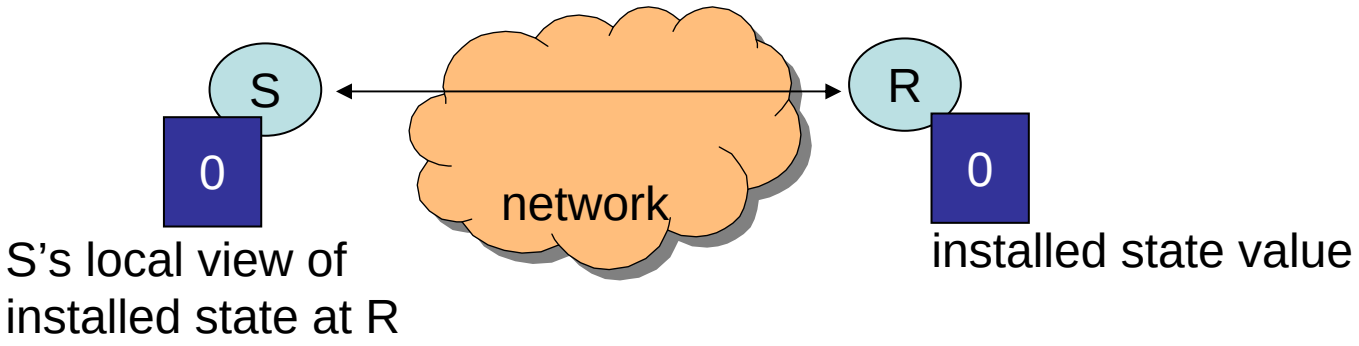
- state *installed* by receiver on receipt of *setup message* from sender
- state *removed* by receiver on receipt of *teardown message* from sender
- *default assumption:* state valid unless told otherwise
  - in practice: failsafe-mechanisms (to remove orphaned state) in case of sender failure e.g., receiver-to-sender “heartbeat”: is this state still valid?
- examples:
  - Q.2931 (ATM Signaling)
  - ST-II (Internet hard-state signaling protocol - outdated)
  - TCP

State: senders, receivers

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

Let's build a signaling protocol

- *S:* state *S*ender (state installer)
- *R:* state *R*eceiver (state holder)
- desired functionality:
  - S: set values in R to 1 when "installed", set to 0 when not installed
  - if other side is down, state is not installed (0)
  - initial condition: state not installed

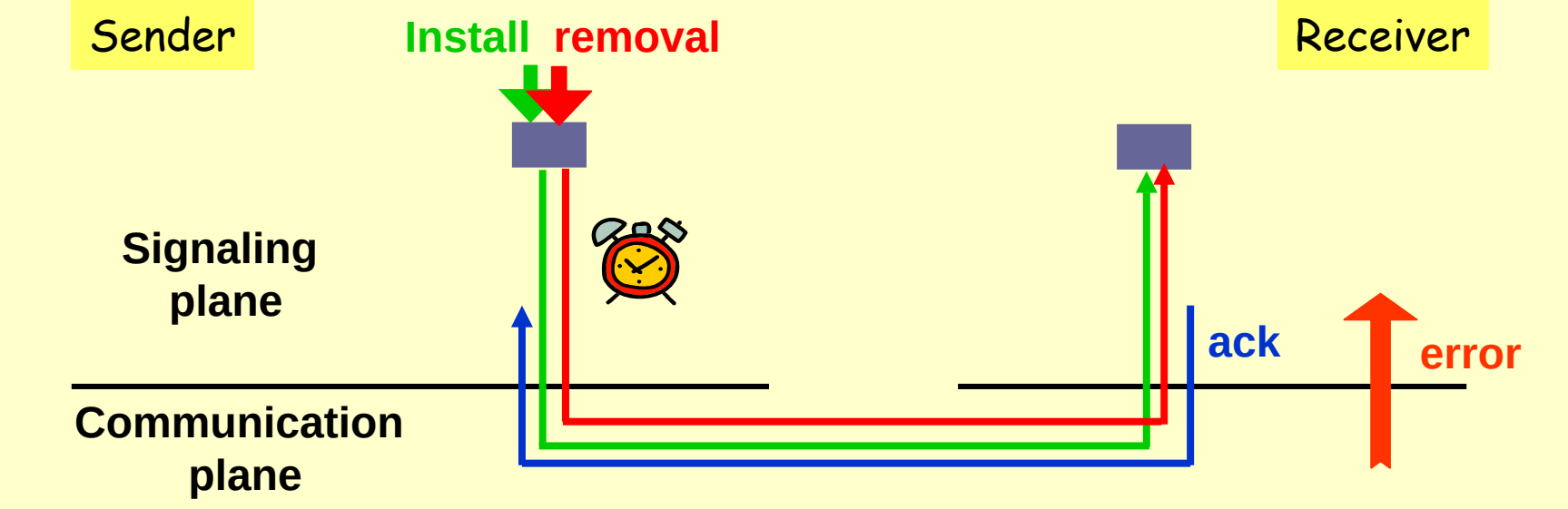


Let’s build signaling protocol

Now: design and specification

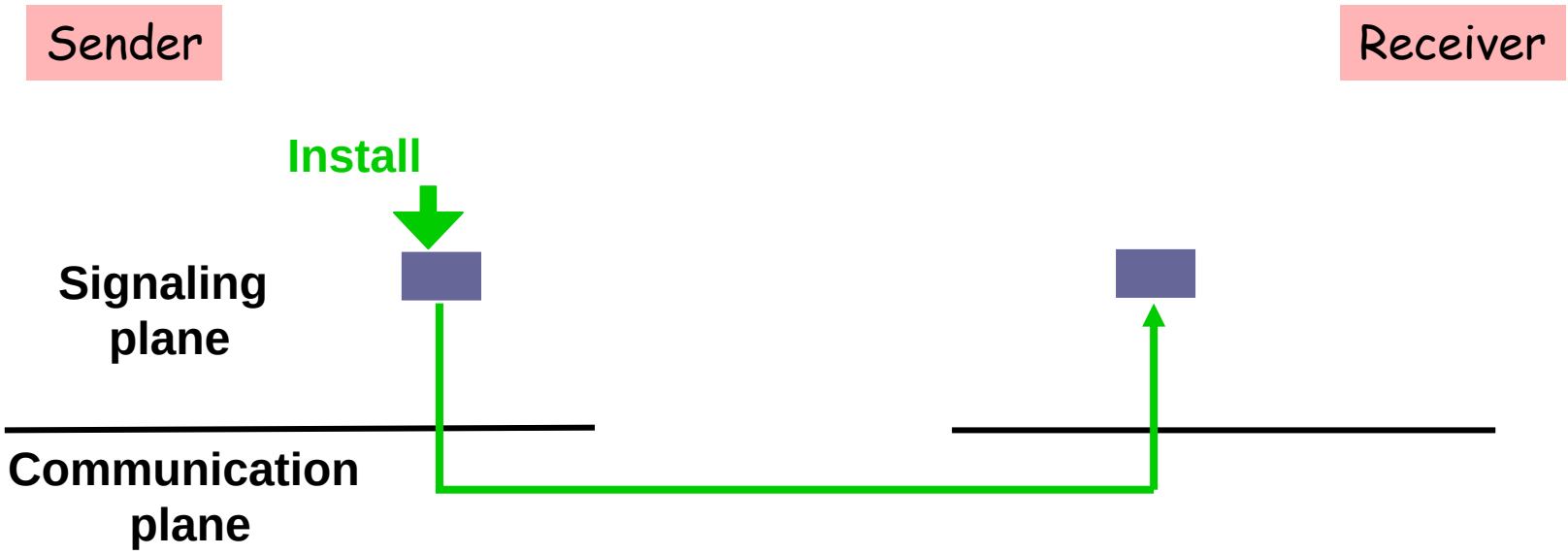
Later: performance model

Hard-state signaling



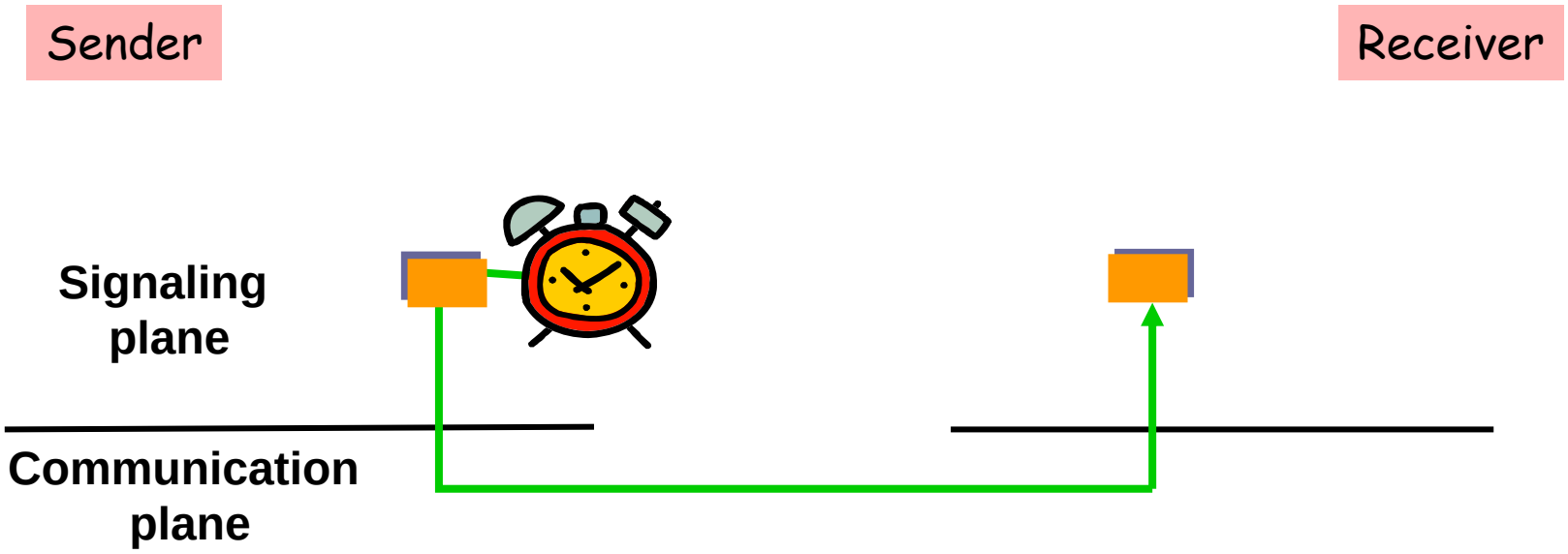
- reliable signaling
- state removal by request
- requires additional error handling
  - e.g., sender failure

Soft-state signaling



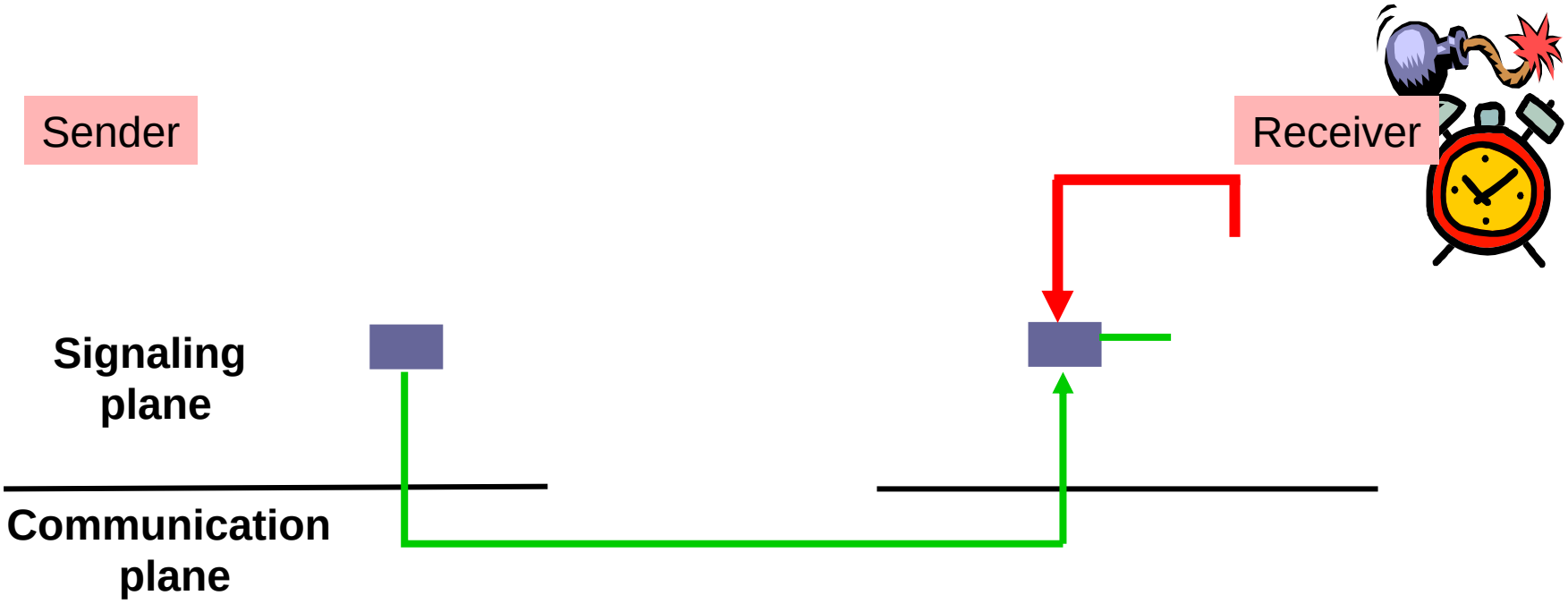
- best effort signaling

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

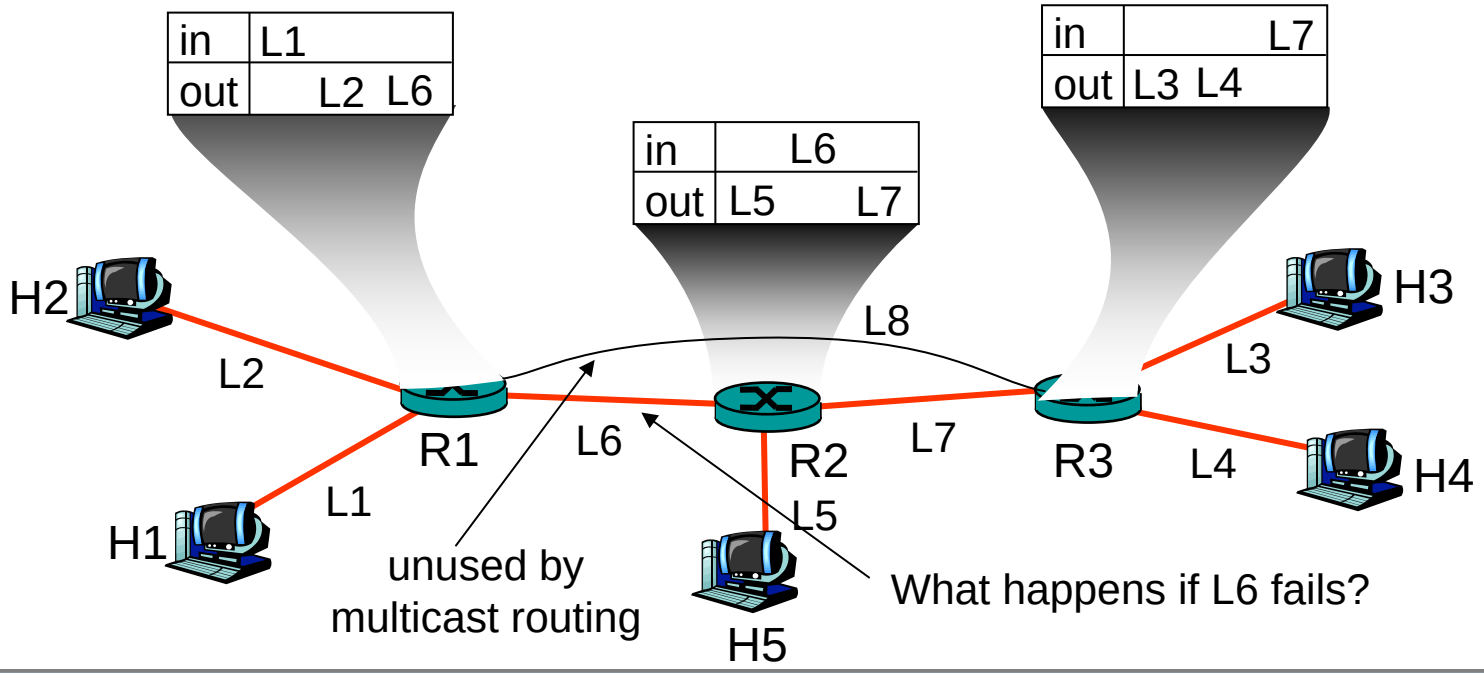
Soft-state: claims

- “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?

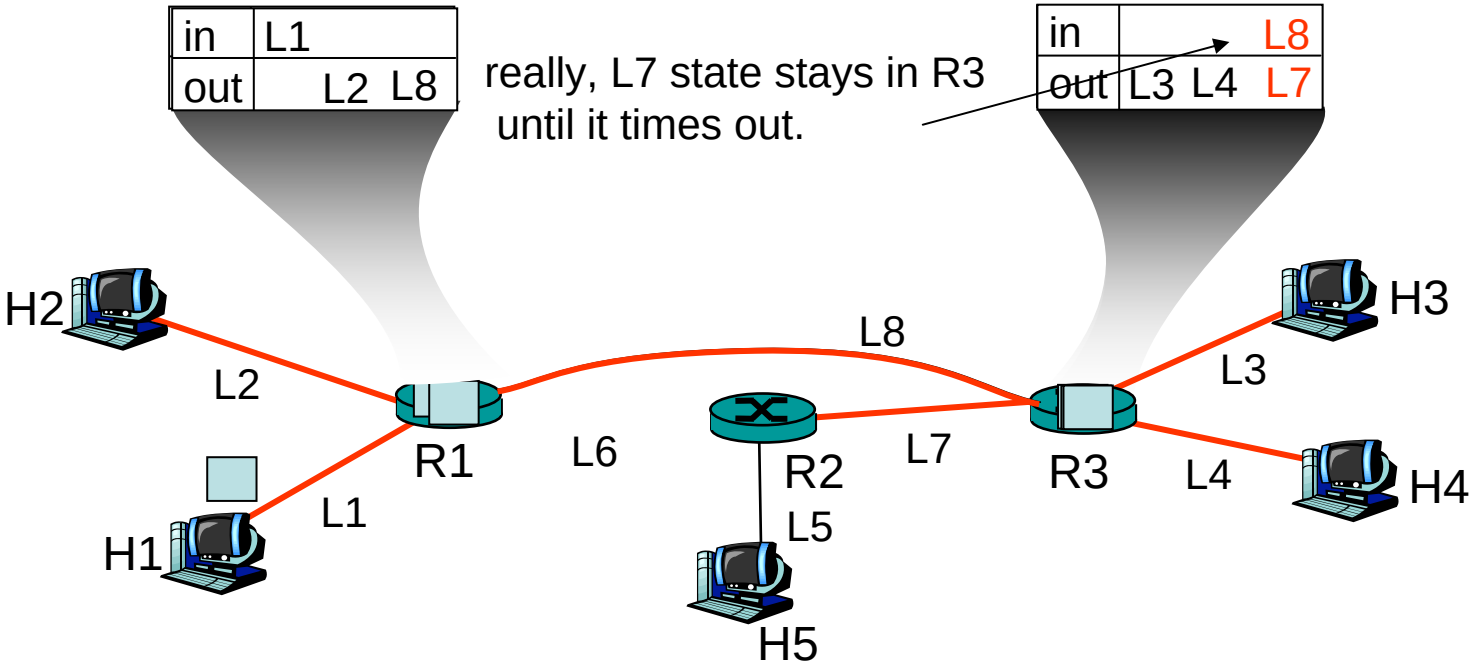
Soft-state: “easy” handling of changes

- 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, 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



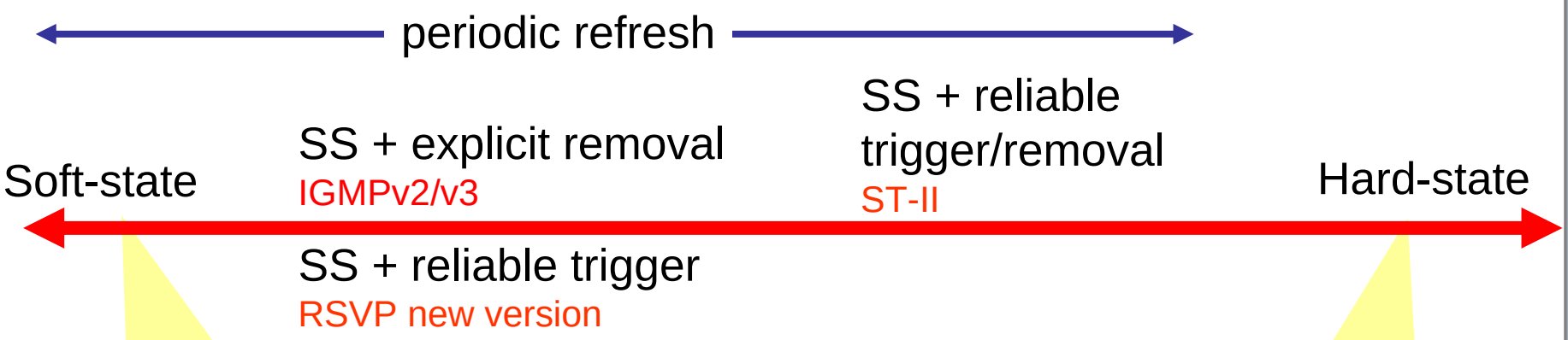
Soft-state: “easy” handling of changes

- “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 messages 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 messages unreliable
  - would like triggers to result in state-installation a.s.a.p.
  - enhancement: add receiver-to-sender refresh_ACK for triggers
  - e.g., see “Staged Refresh Timers for RSVP”

Signaling spectrum



- best effort periodic state installation/refresh
- state removal by time out
- RSVP, IGMPv1

- reliable signaling
- explicit state removal
- requires additional mechanism to remove orphan state
- Q2931b