CS 552
Wireless TCP

slides by B. Nath

Wireless TCP

- Packet loss in wireless networks may be due to
  - Bit errors
  - Handoffs
  - Congestion (rarely)
  - Reordering (rarely, except in ad-hoc networks (mobile))
- TCP assumes packet loss is due to
  - Congestion
  - Reordering (rarely)
- TCP's congestion responses are triggered by wireless packet loss but interact poorly with wireless nets

Impact of loss on TCP

- Random losses result in lower throughput
- Wireless loss is not due to congestion
- TCP cannot distinguish between link loss and congestion loss
- Wireless TCP needs to differentiate between the two
- Loss on wireless link means try harder, loss on wired means backoff
- How to reconcile between the two in an end-to-end transport mechanism
- A number of approaches
  - Link level, modified link level or link aware, transport level, explicit loss notification(ELN)

TCP congestion detection

- TCP assumes timeouts and duplicate acks indicate congestion or packet reordering (alternate paths)
- Timeout indicates packet or ack was lost
- Duplicate acks may indicate packet reordering
  - Receipt of duplicate acks means some data is still flowing
- Aggressive congestion control on loss but less aggressive on dup acks

Responses to congestion

- Basic timeout and retransmission
  - If sender receives no ack for data sent, timeout
  - Timeout value is sum of smoothed RTT delay and 4 X mean deviation
  - Exponential back-off
  - Timeout value based on mean and variance of RTT
- Congestion “avoidance” (really congestion control)
  - Uses congestion window (cwnd) for more flow control
  - Cwnd set to 1/2 of its value when congestion loss occurred
  - Sender can send up to minimum of advertised window and cwnd
  - Use additive increase of cwnd (at most 1 segment each RT)
  - Approach limit of the network capacity slowly
- Slow start, fast retransmit

Other problems in a wireless environment

- Burst errors due to poor signal strength or mobility (handoff)
  - More than one packet lost in TCP window
- Delay is often very high
  - RTT quite long
  - Tunneling, satellite
  - True in telephone networks providing data services that deploy fixed gateways (non-optimal routes)

Poor interaction with TCP

- Cumulative ack scheme not good with bursty losses
  - Missing data detected one segment at a time
  - Duplicate acks take a while to cause retransmission
  - TCP Reno may suffer coarse time-out and enter slow start!
  - TCP New Reno still only retransmits one packet per RTT
- Packet loss due to noise or hand-offs indicated by dup acks
  - Enter congestion control
  - Slow increase of cwnd
- Bursts of packet loss and hand-offs indicated by timeouts
  - Timeout
  - Enter slow start (start from cwnd = 1)

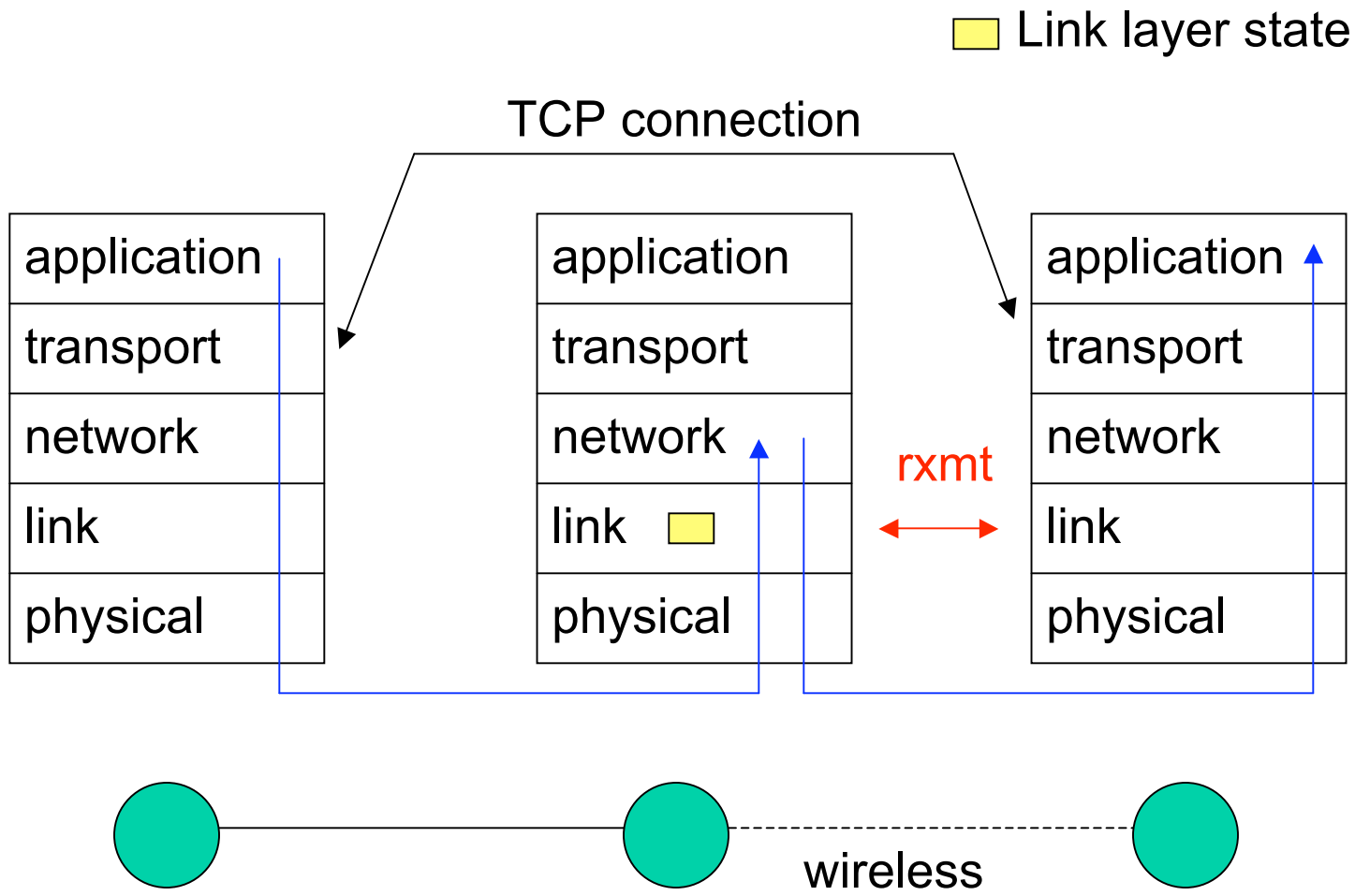
Multiple losses in window

- Assume cwnd of 8
- 1st and 4th packets lost
- 3rd duplicate ack causes retransmit of 1st packet
- Also sets cwnd to 4 + 3 = 7, ssthresh= 4
- Further duplicate acks increment cwnd by 1
- Ack for retransmit of packet 1 is a partial ack since packet 4 is also lost
- In TCP Reno this results in an exit out of fast retransmit
- reset congestion window to 4 but 8 packets were already sent

Approaches

- Link layer enhancements (FEC, retransmissions)
  - Interacts with RTT, higher variance may still lead to timeouts
  - Not a problem with coarse grain timeouts
  - But a problem in slow wireless links, as RTO estimates may be high
  - Interested see (Reiner Ludwig's paper at Infocom)
- Transport layer I-TCP [BakreBadri95]
- TCP aware Link layer aware (Snoop)[Hari et al 96]
- Explicit Loss Notification schemes

Link Level Retransmissions



Link Level Retransmissions Issues

- How many times to retransmit at the link level before giving up?
  - Finite bound -- semi-reliable link layer
  - No bound -- reliable link layer
- What triggers link level retransmissions?
  - Link layer timeout mechanism
  - Link level acks (negative acks, dupacks, sacks)
- How much time is required for a link layer retransmission?
  - Small fraction of end-to-end TCP RTT
  - Large fraction/multiple of end-to-end TCP RTT
- Should the link layer deliver packets as they arrive, or deliver them in-order?
  - Link layer may need to buffer packets and reorder if necessary so as to deliver packets in-order

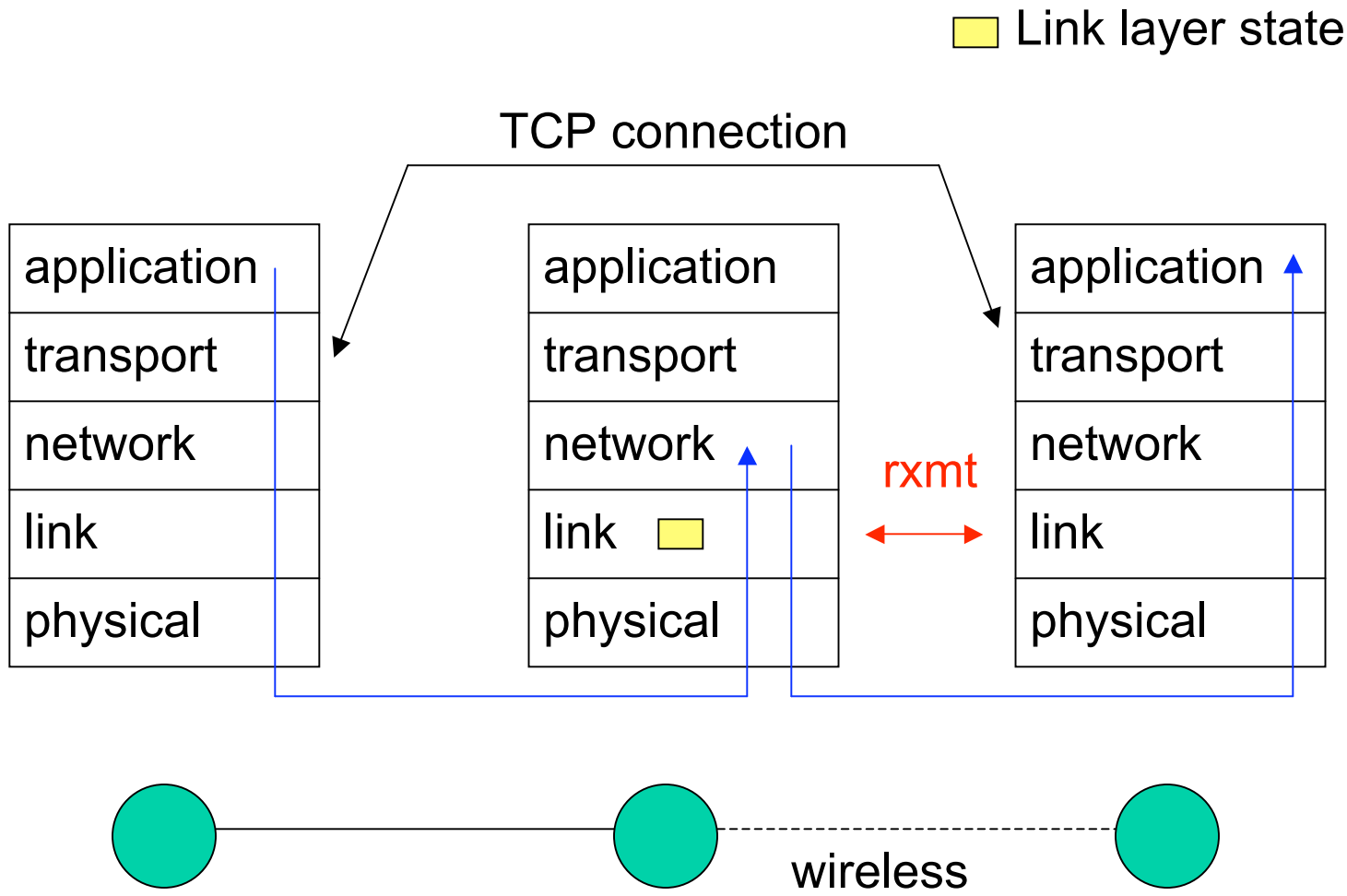
Link Layer Schemes applicability

- When is a reliable link layer beneficial to TCP performance?
- if it provides almost in-order delivery and
- TCP retransmission timeout large enough to tolerate additional delays due to link level retransmits
- Another headache, link layer packets may be smaller than MSS of TCP packets
- GSM protocol an example

Link Layer Schemes: Classification

- Hide wireless losses from TCP sender
- Link layer modifications needed at both ends of wireless link
  - TCP need not be modified

Link Level Retransmissions



Link Level Retransmissions
Issues

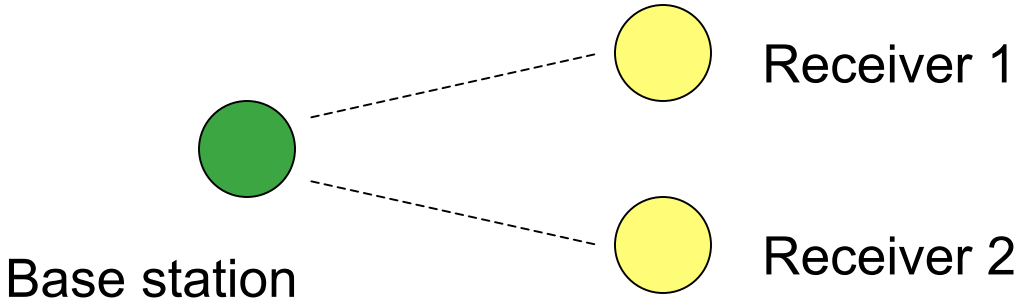
- How many times to retransmit at the link level before giving up?
  - Finite bound -- semi-reliable link layer
  - No bound -- reliable link layer
- What triggers link level retransmissions?
  - Link layer timeout mechanism
  - Link level acks (negative acks, dupacks, …)
  - Other mechanisms (e.g., Snoop, as discussed later)
- How much time is required for a link layer retransmission?
  - Small fraction of end-to-end TCP RTT
  - Large fraction/multiple of end-to-end TCP RTT

Link Level Retransmissions Issues

- Should the link layer deliver packets as they arrive, or deliver them in-order?
  - Link layer may need to buffer packets and reorder if necessary so as to deliver packets in-order

Link Level Retransmissions Issues

- Retransmissions can cause congestion losses



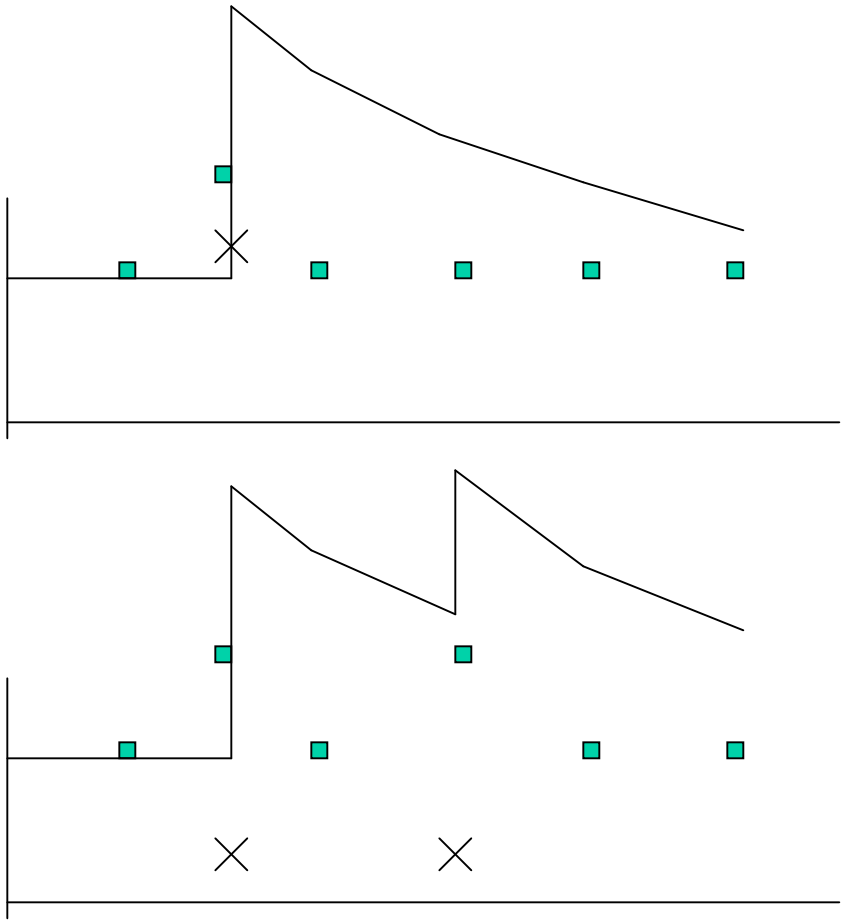
- Attempting to retransmit a packet at the front of the queue, effectively reduces the available bandwidth, potentially making the queue at base station longer
- If the queue gets full, packets may be lost, indicating congestion to the sender
- Is this desirable or not ?

Link Level Retransmissions
An Early Study [DeSimone93]

- The sender's Retransmission Timeout (RTO) is a function of measured RTT (round-trip times)
  - **Link level retransmits increase RTT, therefore, RTO**

- **If errors not frequent**, RTO will **not** account for RTT variations due to link level retransmissions
  - When errors occur, the sender may timeout & retransmit before link level retransmission is successful
  - Sender and link layer both retransmit
  - Duplicate retransmissions (interference) waste wireless bandwidth
  - Timeouts also result in reduced congestion window

RTO Variations



Wireless Packet loss

RTT sample

RTO

A More Accurate Picture

- Analysis in [DeSimone93] does not accurately model real TCP stacks

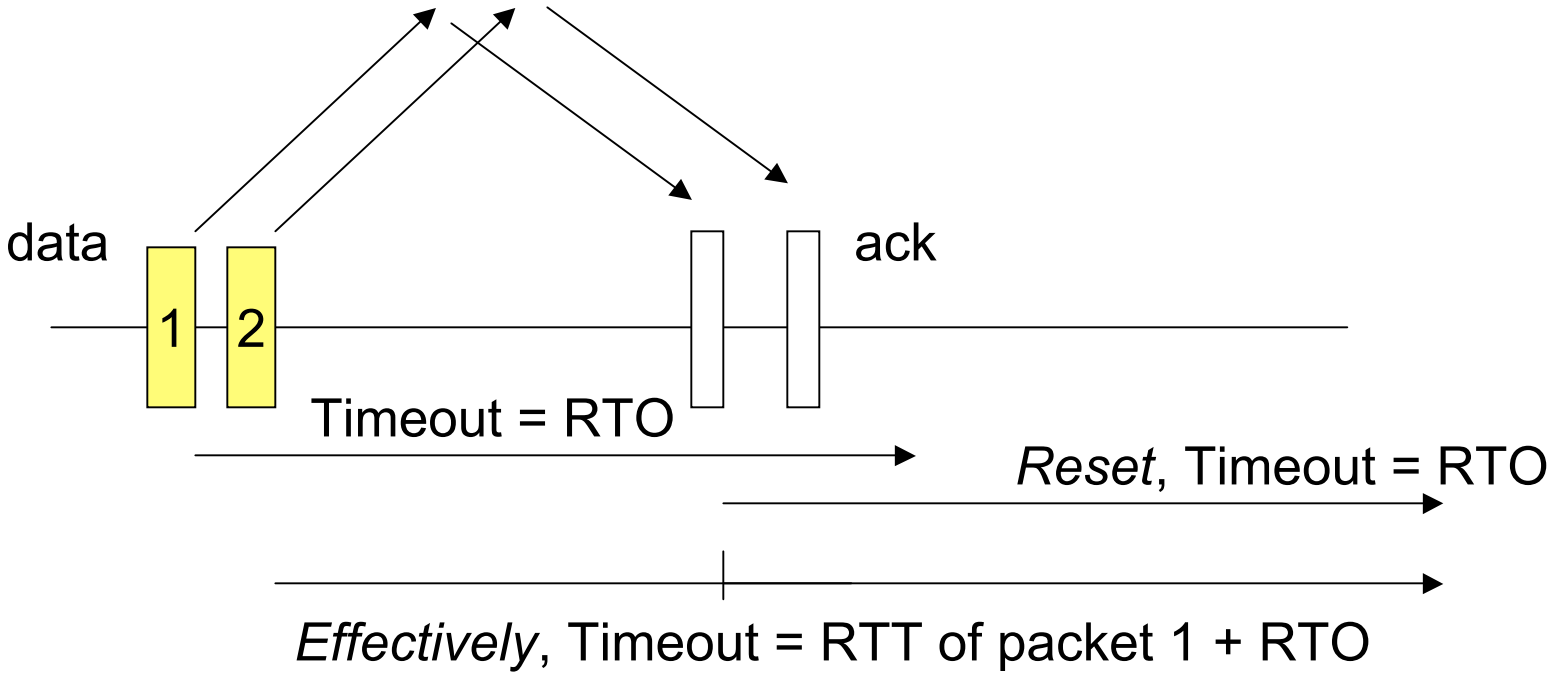
- With large **RTO granularity**, interference is unlikely, if time required for link-level retransmission is small compared to TCP RTO [Balakrishnan96Sigcomm]
  - Standard TCP RTO granularity is often large
  - Minimum RTO (2*granularity) is large enough to allow a small number of link level retransmissions, if link level RTT is relatively small
  - Interference due to timeout not a significant issue when wireless RTT small, and RTO granularity large [Eckhardt98]

Link Level Retransmissions
A More Accurate Picture

- **Frequent errors** increase RTO significantly on slow wireless links
  - RTT on slow links large, retransmissions result in large variance, pushing RTO up
  - Likelihood of interference between link layer and TCP retransmissions smaller
  - But congestion response will be delayed due to larger RTO
  - When wireless losses do cause timeout, much time wasted

Link-Layer Retransmissions
A More Accurate Picture [Ludwig98]

- Timeout interval may actually be larger than RTO
  - Retransmission timer reset on an ack
  - If the ack'd packet and next packet were transmitted in a burst, next packet gets an additional RTT before the timer will go off



Large TCP Retransmission Timeout Intervals

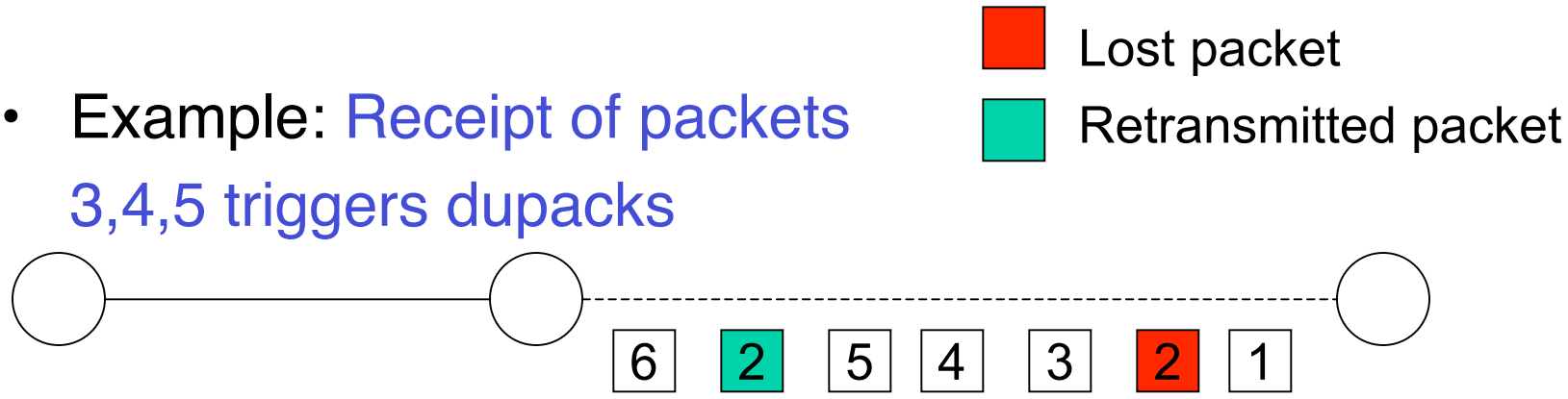
- Good for reducing interference with link level retransmits
- Bad for recovery from congestion losses
- Need a timeout mechanism that responds appropriately for both types of losses
  - Open problem

Link Level Retransmissions

- Selective repeat protocols can deliver packets out of order

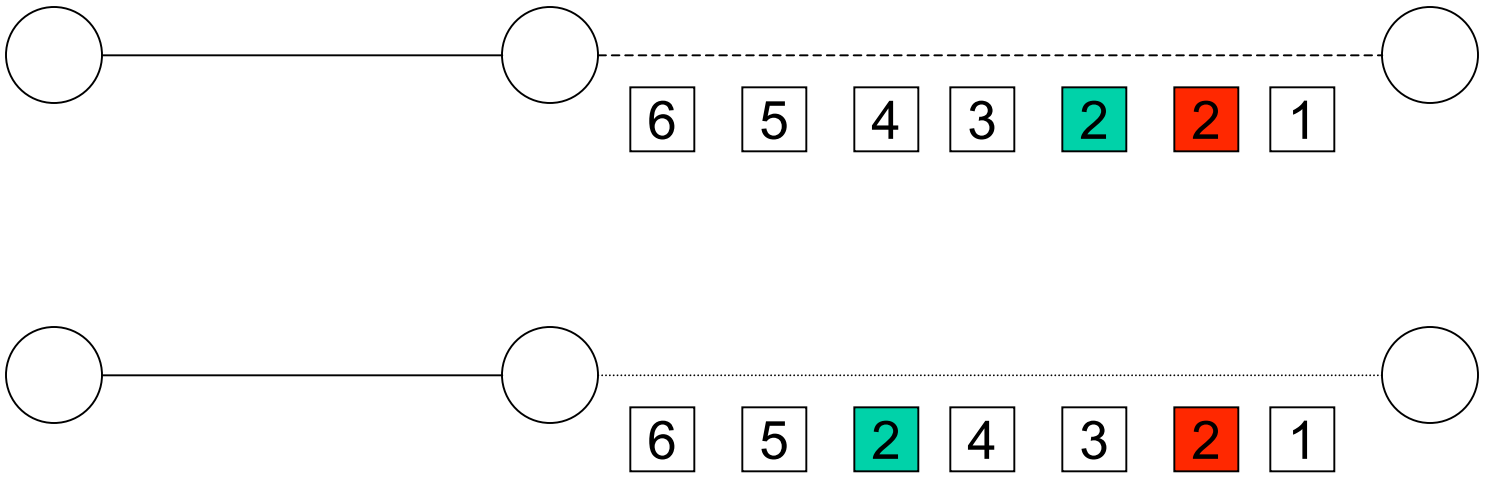
- Significantly out-of-order delivery can trigger TCP fast retransmit
  - Redundant retransmission from TCP sender
  - Reduction in congestion window

- Example: Receipt of packets 3,4,5 triggers dupacks



Link Level Retransmissions
In-order delivery

- To avoid unnecessary fast retransmit, link layer using retransmission should attempt to deliver packets “almost in-order”



Link Level Retransmissions
In-order delivery

- Not all connections benefit from retransmissions or ordered delivery
  - audio
- Need to be able to specify requirements on a per-packet basis
  - Should the packet be retransmitted? How many times?
  - Enforce in-order delivery?

Link Layer Schemes: Summary

When is a reliable link layer beneficial to TCP performance?

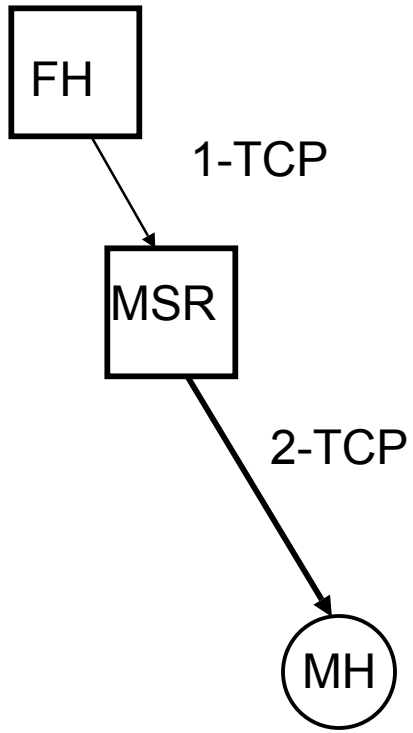
- if it provides almost in-order delivery

and

- TCP retransmission timeout large enough to tolerate additional delays due to link level retransmits

I-TCP

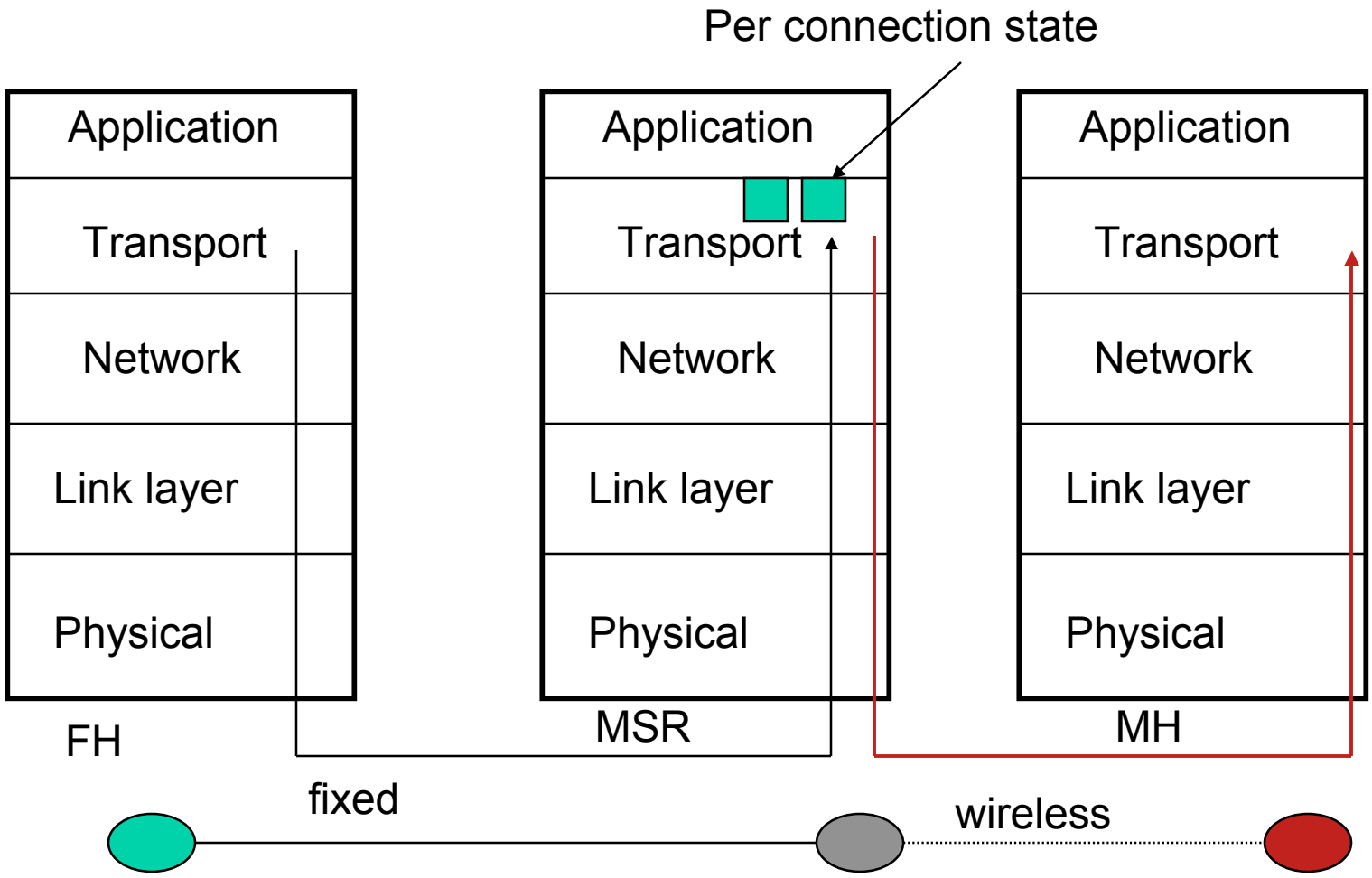
- Uses a split connection
  - End-to-end connection is broken into one connection for the wired part and another connection for the wireless part
  - Wireless part of the TCP can be optimized for wireless
  - TCP optimization close to where it is needed



Split connection approach

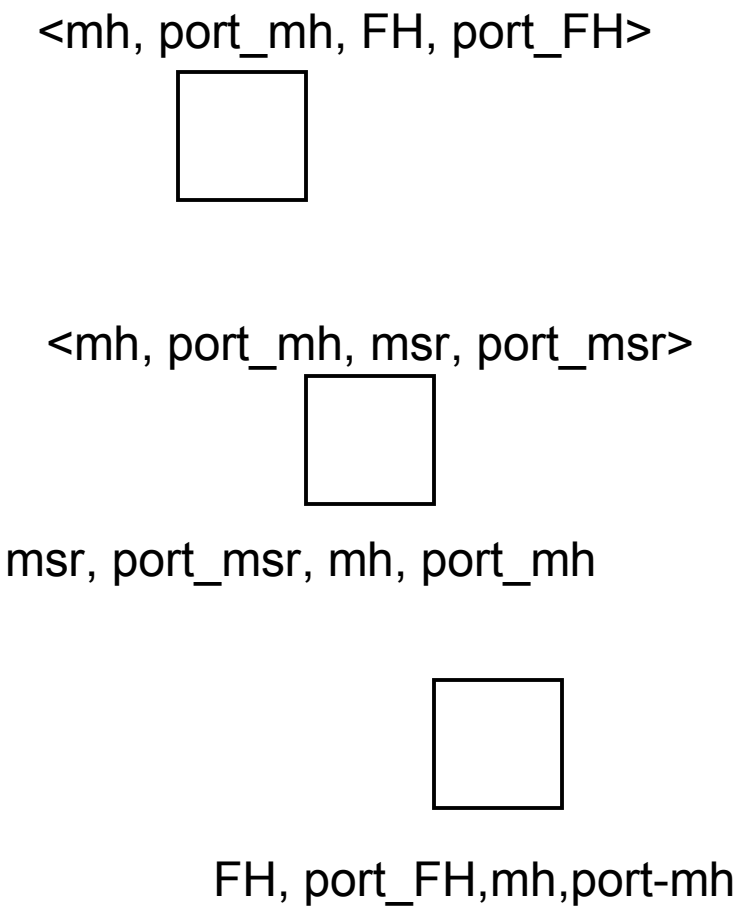
- Split connection results in two independent flows. Hence, independent decision of what do with packet loss
- On wireless, loss ➔try harder
- On fixed, loss ➔ backoff
- Tune TCP stack to get this behavior

Transport level solution



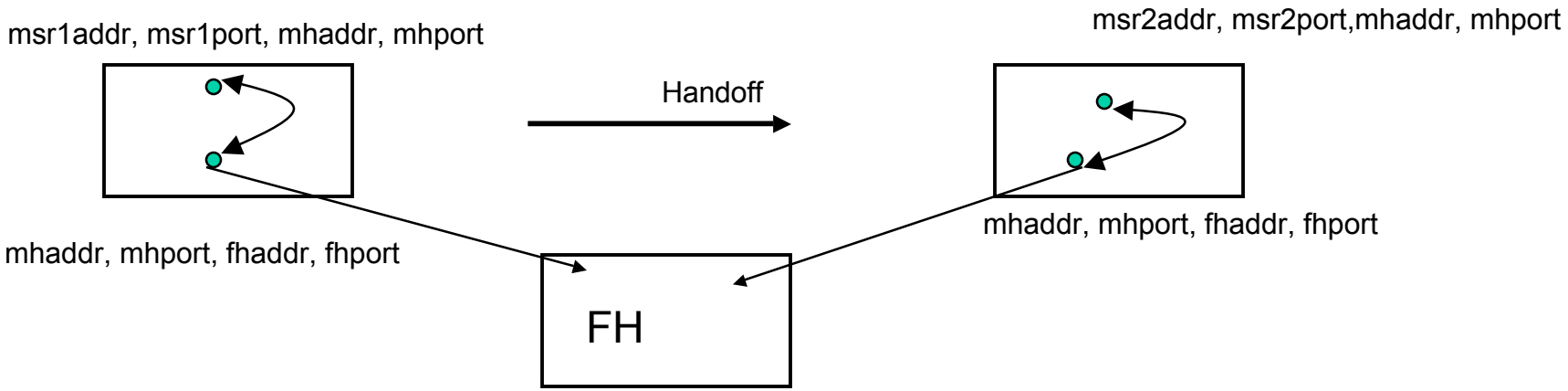
Establishing TCP connections

- FH should see a TCP connection coming from MH and not from MSR
- MH should open a TCP connection to FH and should not be aware that the connection is going to MSR
- MH has a I-TCP library that intercepts connection requests and opens a connection to MSR
- MSR opens a connection to FH but with the <address of MH and port #> sent by FH



I-TCP handoff

- When a MH moves to a new location, it establishes a connection with the new MSR
- The new MSR get the TCP state from the old MSR and continues the TCP connection



I-TCP features

- Hides packet loss due to wireless from sender
- Wireless TCP can be independently optimized
- Good performance in case of wide-area networks
- Retransmission occurs only on the bad link
- Faster recovery due to relatively short RTT for wireless link
- Handoff requires state transfer
- Buffer space needed, extra copying at MSR
- End-to-end semantics violation needs to be augmented by application level actions
- Base station (MSR) failure may cause loss of TCP state

I-TCP : Advantages

- BS-MH connection can be optimized independent of FH-BS connection
  - Different flow / error control on the two connections
- Local recovery of errors
  - Faster recovery due to relatively shorter RTT on wireless link
- Good performance achievable using appropriate BS-MH protocol
  - Standard TCP on BS-MH performs poorly when multiple packet losses occur per window (timeouts can occur on the BS-MH connection, stalling during the timeout interval)
  - Selective acks improve performance for such cases

I-TCP disadvantages

- End-to-end semantics violated
  - ack may be delivered to sender, before data delivered to the receiver
  - May not be a problem for applications that do not rely on TCP for the end-to-end semantics
- BS retains hard state BS failure can result in loss of data (unreliability)
  - Acked packets from BS, sender assumes that packet actually reached the receiver

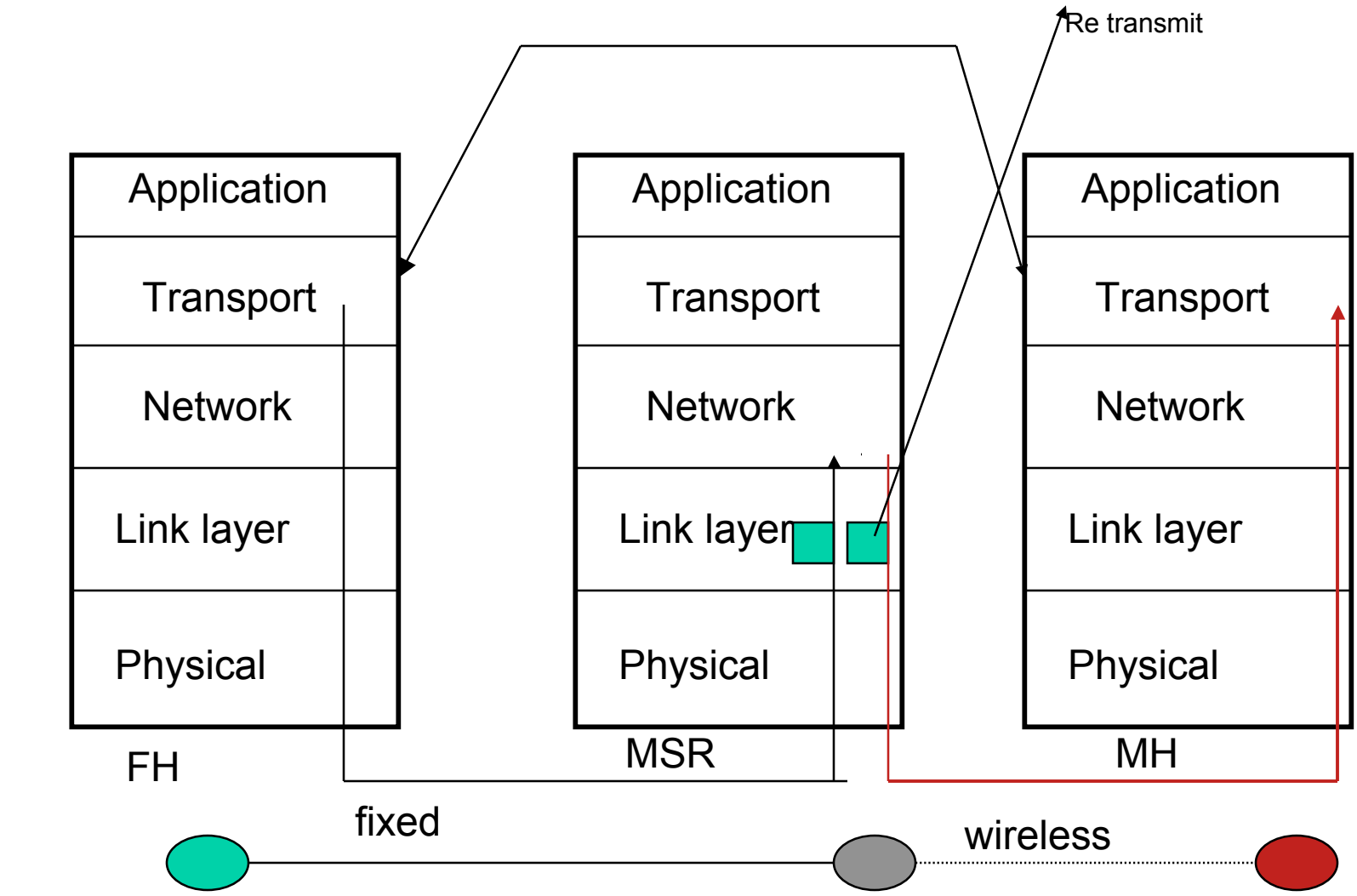
I-TCP : Disadvantages

- BS retains per-connection state
- Bufferred packets at BS must be transferred to new BS
- Hand-off latency increases due to state transfer
- Buffer space needed at BS for each TCP connection
  - BS buffers tend to get full, when wireless link slower (one window worth of data on wired connection could be stored at the base station, for each split connection)
- Extra copying of data at BS
  - copying from FH-BS socket buffer to BS-MH socket buffer
  - increases end-to-end latency
- May not be useful if data and acks traverse different paths (both do not go through the base station)
  - Example: data on a satellite wireless links

Snoop Protocol

- Uses the same idea of local recovery (in I-TCP)
- Link aware TCP at base station (MSR)
- Snoop on TCP flows, buffer and retransmit on packet loss
- End-to-end semantics retained
- Soft state at base station, instead of hard state

Snoop protocol



Snoop protocol

- Snoop module monitors every packet that passes through
- Also, buffers packets from FH to MH as yet unacknowledged
- A new packet is cached in the buffer, flushed when an ack covering the packet is received
- When dup ack is received, retransmit from buffer

Snoop features

- Snoop prevents fast retransmit from sender despite wireless loss
- Hides wireless loss from sender
- Sender can still timeout
- Snoop state is soft state
- On a move a new snoop state is built at the new MSR
- Can a TCP-unaware link level scheme achieve the same effect

Snoop features

- Unlike I-TCP, end-to-end semantics retained
- High throughput at medium error rates
- Not useful if TCP headers are encrypted
- Cannot be used on asymmetric links

Snoop Protocol-advantages

- Snoop prevents fast retransmit from sender despite transmission errors, and out-of-order delivery on the wireless link
- What about for small window sizes
- TCP_new reno scheme
- Snoop should help as well

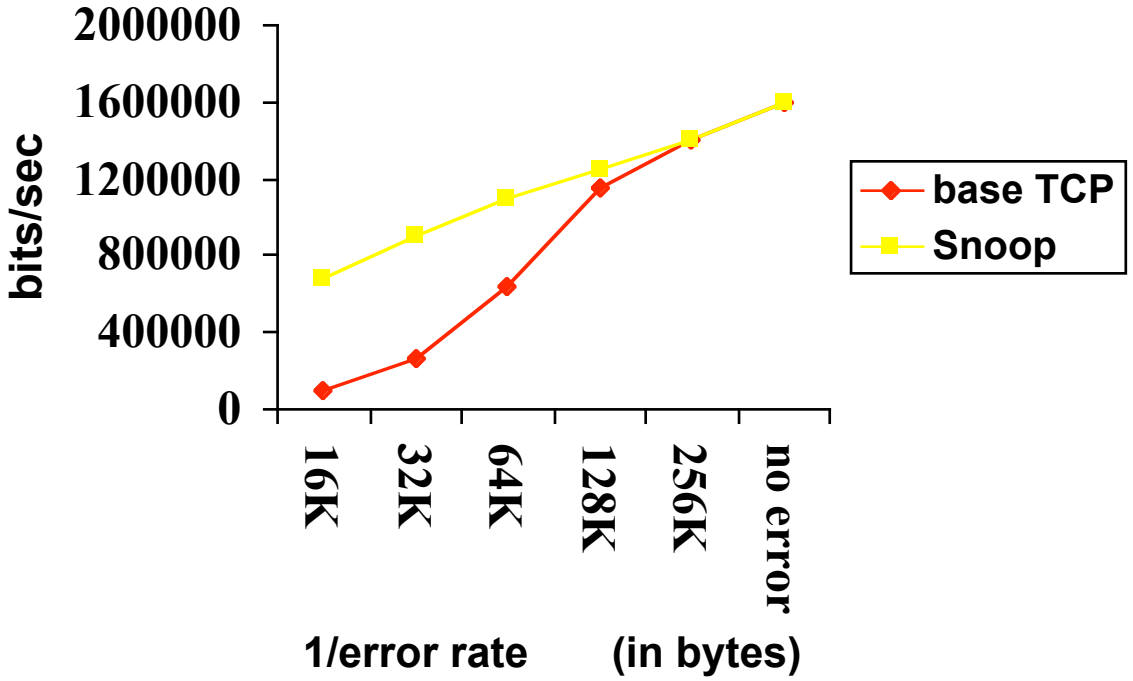
Snoop Protocol : Advantages

- High throughput can be achieved
  - performance further improved using selective acks
- Local recovery from wireless losses
- Fast retransmit not triggered at sender despite out-of-order link layer delivery
- End-to-end semantics retained
- Soft state at base station
  - loss of the soft state affects performance, but not correctness

Snoop Protocol disadvantages

- Link layer at base station needs to be TCP-aware
- Not useful if TCP headers are encrypted (IPsec)
- Cannot be used if TCP data and TCP acks traverse different paths (both do not go through the base station)

Snoop- performance



2 Mbps Wireless link

Snoop : Basic Idea

- Data from FH -> MH
  - Cache unacknowledged TCP data
  - Perform local retransmissions
- Data from MH -> FH
  - Detect missing packets
  - Perform negative acknowledgements

FH -> MH : Snoop_data() – case 1 and 2

- New Packet in normal TCP sequence

  Normal case

  Add to snoop cache

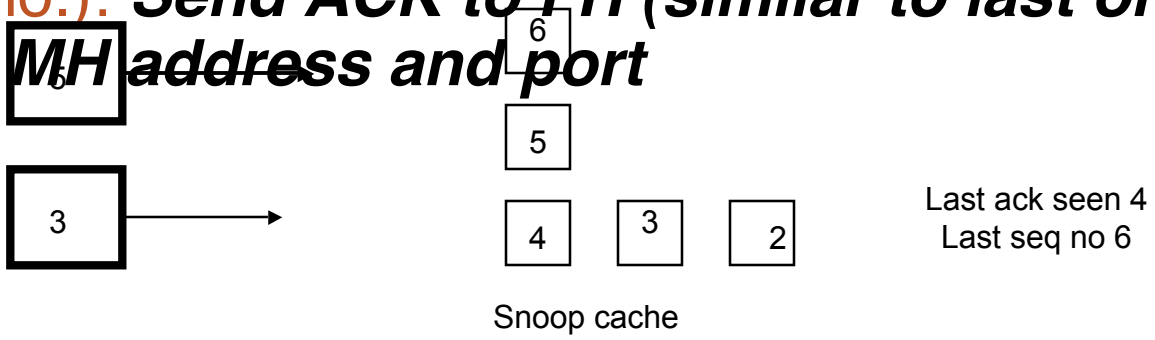
  Forward to MH

- Out of sequence packet cached earlier

  Fast Retransmission/timeout at sender due to

  A)Loss in wireless link (if last ACK is < current seq.no.): ***Forward to MH***

  B) Loss of previous ACK (if last ACK > current seq.no.): ***Send ACK to FH (similar to last one seen) with MH address and port***



FH -> MH: Snoop_data() – case 3

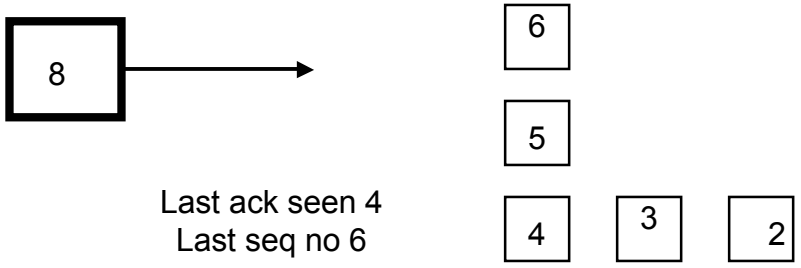
- Out of sequence packet not cached earlier

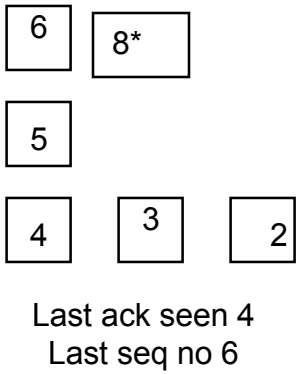
  A) Congestion in fixed n/w (if seq. no is more than 1 or 2 packets away from last one seen):

  Forward to MH

  Mark it as retransmitted by sender

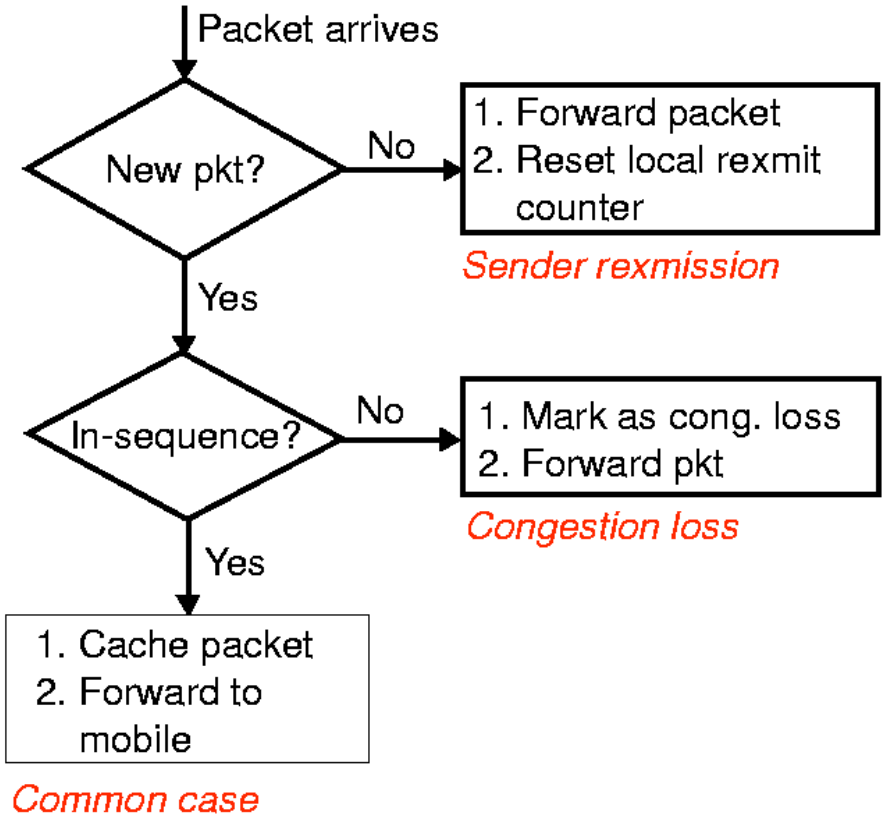
  B) Out Of Order Delivery





Snoop: FH -> MH

Data Processing



FH -> MH: Snoop_ack() - 1

- New ACK

  Common case

  Cleaning of snoop cache

  Update round trip estimate

  Forward ACK to FH

- Spurious ACK

  Discard it

FH -> MH: Snoop_ack() - 2

- Duplicate ACK (DUPACK) – Identical to last received highest cumulative ACK, MH generates DUPACK for every packet received out-of-sequence

  A) Packet not in snoop cache

  Lost in fixed n/w

  Forward to FH

  B) Packet marked as sender retransmitted

  Forward to FH – TCP keeps track of no. of dupacks received when it retransmits

FH -> MH: Snoop_ack() - 3

C) Unexpected DUPACK – first DUPACK after a packet loss

Lost packet on wireless link

Retransmit at higher priority (reduces no. of DUPACKS, improves throughput)

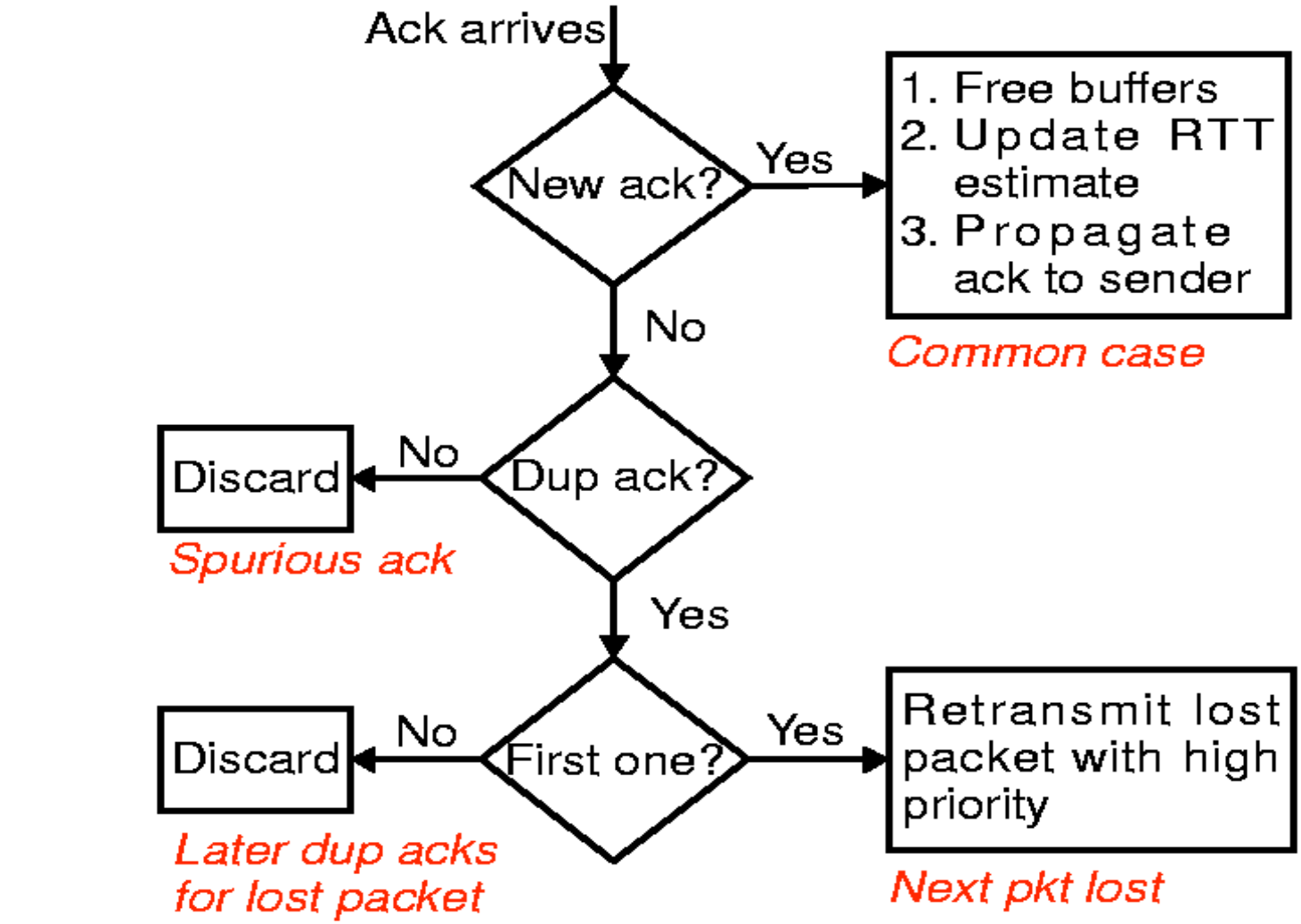
Estimate max. of DUPACKS

D) Expect DUPACK

Subsequent packets after the lost one reaching MH

Discard it

Snoop: ACK Processing



Data Transfer from MH -> FH

Why? MH timeouts for packets lost in first link will happen much later than they should.

NACKs* sent from BS to MH when

A) threshold no. of packets from a single window have reached

B) No new packets from MH for certain time

*- Based on TCP SACK.