# TCP in Wireless Networks

- Transmission errors
  - Random errors
  - Burst errors
- Mobility
  - Infrastructure wireless networks
  - Wireless ad hoc networks

## Impacts of Random Errors

# Impacts of Random Errors

- Random errors may cause fast retransmit
  - Fast retransmit results in
    - retransmission of lost packet
    - reduction in congestion window
  - Reducing congestion window in response to errors is unnecessary and reduces the throughput
- Random errors may cause timeout
  - Multiple packet losses in a window can result in timeout when using TCP-Reno (and to a less extent when using SACK)

## Impacts of Burst Errors

## Burst Errors May Cause Timeouts

- If wireless link remains unavailable for extended duration, a window worth of data may be lost
  - passing a truck
  - driving through the tunnel
- Timeout results in
  - Possibly long idle time
  - Slow start, which reduces congestion window to 1 MSS and reduces ssthresh to 1/2
  - Reduction in window and ssthresh in response to errors are unnecessary

## What are your solutions?

# Various Schemes

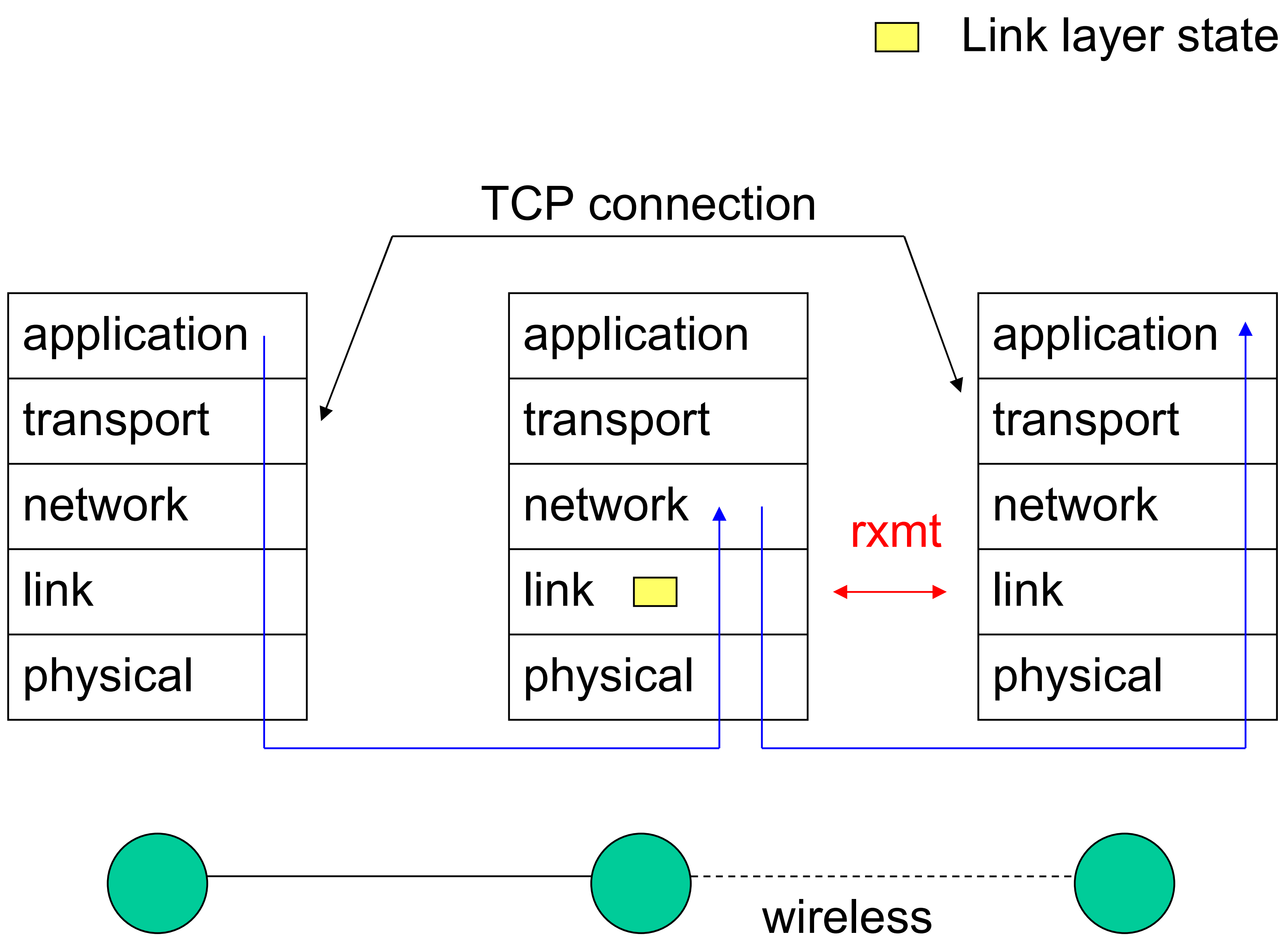
- Link level mechanisms
- Split connection approach
- TCP-Aware link layer
- TCP-Unaware approximation of TCP-aware link layer

Hide losses

- Explicit notification
- Receiver-based discrimination
- Sender-based discrimination

Find reasons for losses

# Link Level Schemes



# Link Layer Schemes (Cont.)

Ideas

- Recover wireless losses using FEC code, retransmission, and/or adapting frame size

Characteristics

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

When is a reliable link layer beneficial to TCP performance?

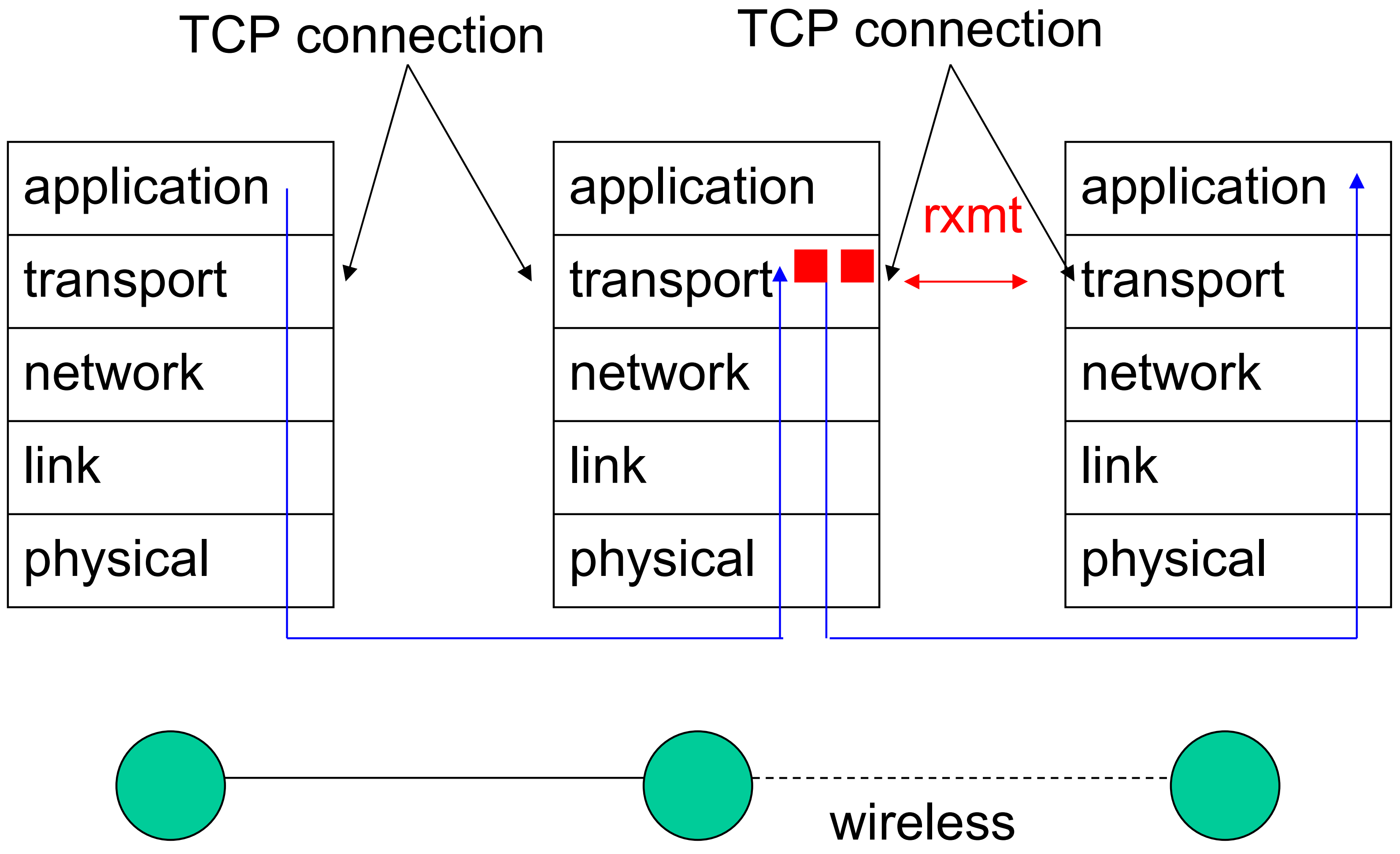
- If it provides almost in-order delivery

and

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

# Split Connection Approach

🟥 Per-TCP connection state



## Split Connection Approach (Cont.)

- Idea
  - End-to-end TCP connection is broken into one connection on the wired part of route and one over wireless part of the route
- Characteristics
  - Hides transmission errors from sender
  - Primary responsibility at base station
  - If specialized transport protocol used on wireless, then wireless host also needs modification
- Pros and cons?

## Split Connection Approach : Advantages

- Local recovery of errors
  - Faster recovery due to relatively shorter RTT on wireless link
- BS-MH connection can be optimized independent of FH-BS connection
  - Different flow / error control on the two connections
- 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

## <u>Split Connection Approach : Disadvantages</u>

# Split Connection Approach : 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
- May not be useful if data and acks traverse different paths (both do not go through the base station)
- Extra copy and storage required at base station

## TCP-Aware Link Layer

## Various Schemes

- Link level mechanisms
- Split connection approach
- TCP-Aware link layer
- TCP-Unaware approximation of TCP-aware link layer

$\Big\}$ Hide losses

- Explicit notification
- Receiver-based discrimination
- Sender-based discrimination

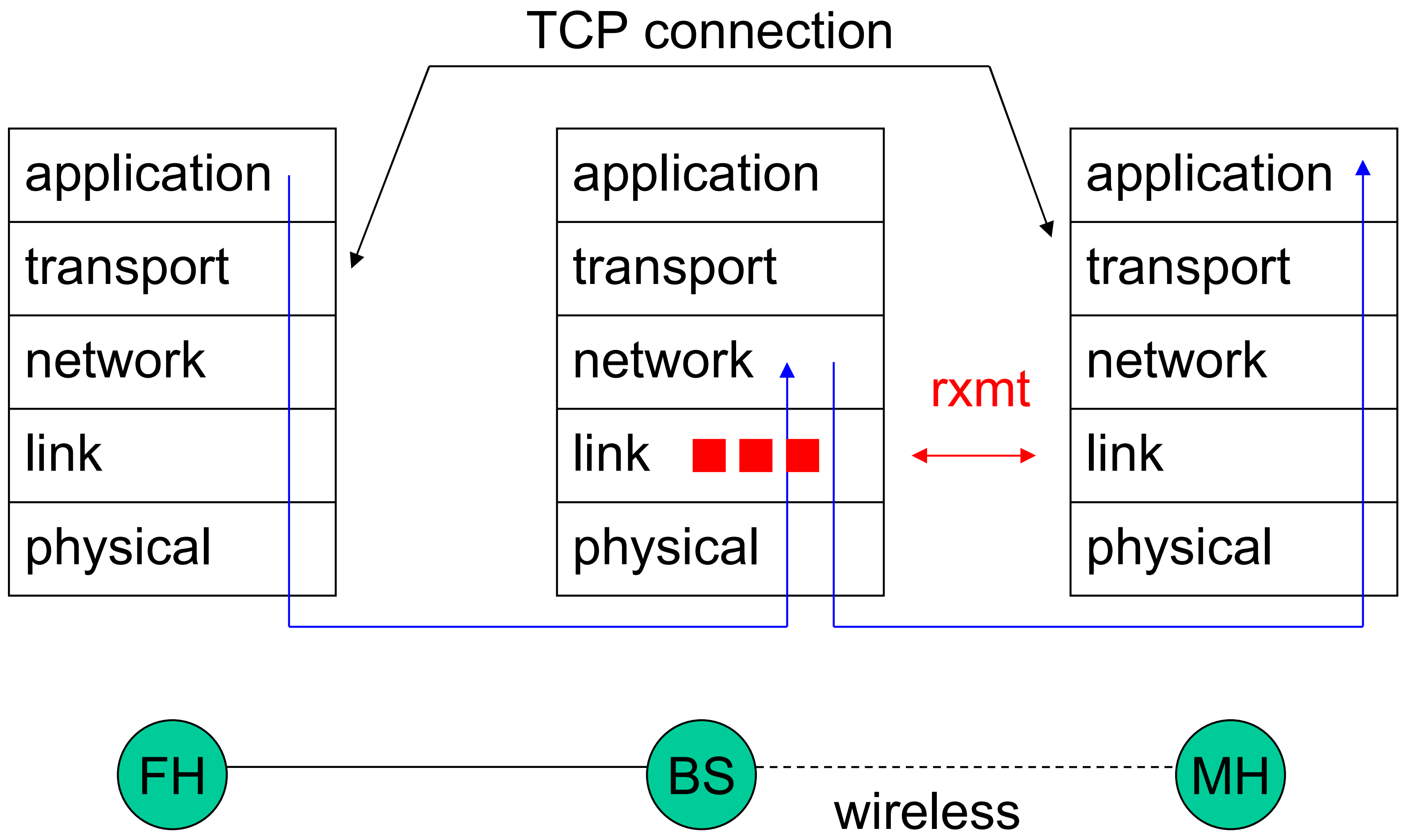
$\Big\}$ Find reasons for losses

# Snoop Protocol [Balakrishnan95acm]

- Retains local recovery of Split Connection approach and link level retransmission schemes
- Improves on split connection
  - end-to-end semantics retained
  - soft state at base station, instead of hard state

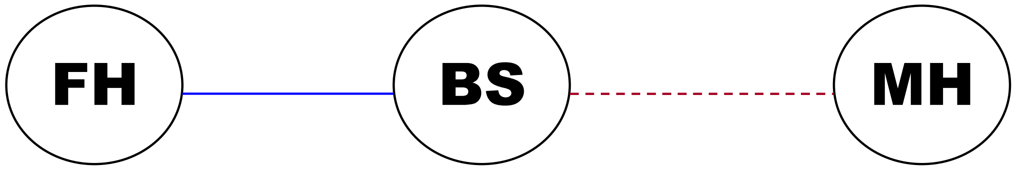
# Snoop Protocol

🟥 Per TCP-connection state

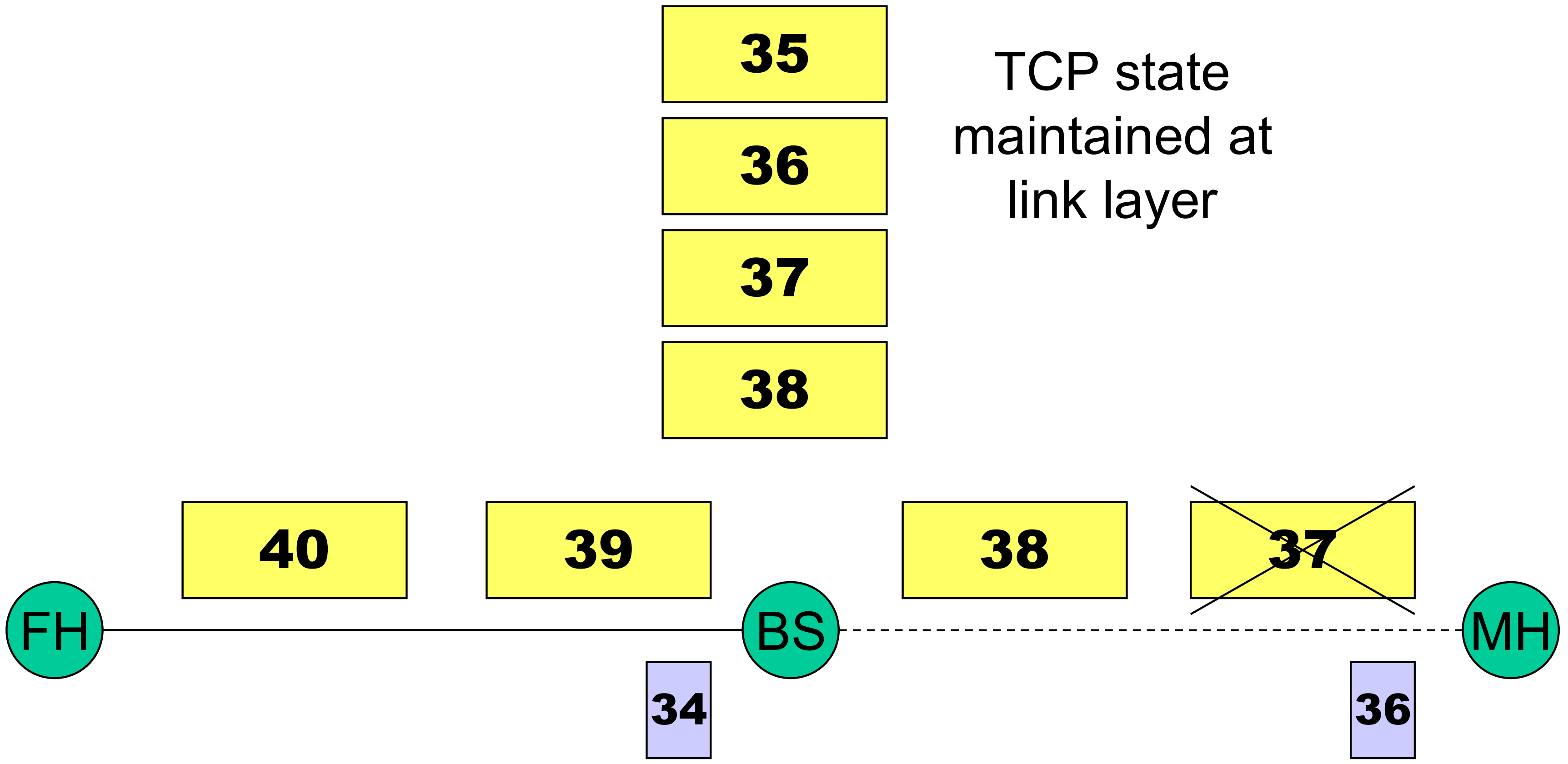


# Snoop Protocol

- Buffers data packets at the base station BS
  - to allow link layer retransmission
- When dupacks received by BS from MH, retransmit on wireless link, if packet present in buffer
- Prevents fast retransmit at TCP sender FH by dropping the dupacks at BS

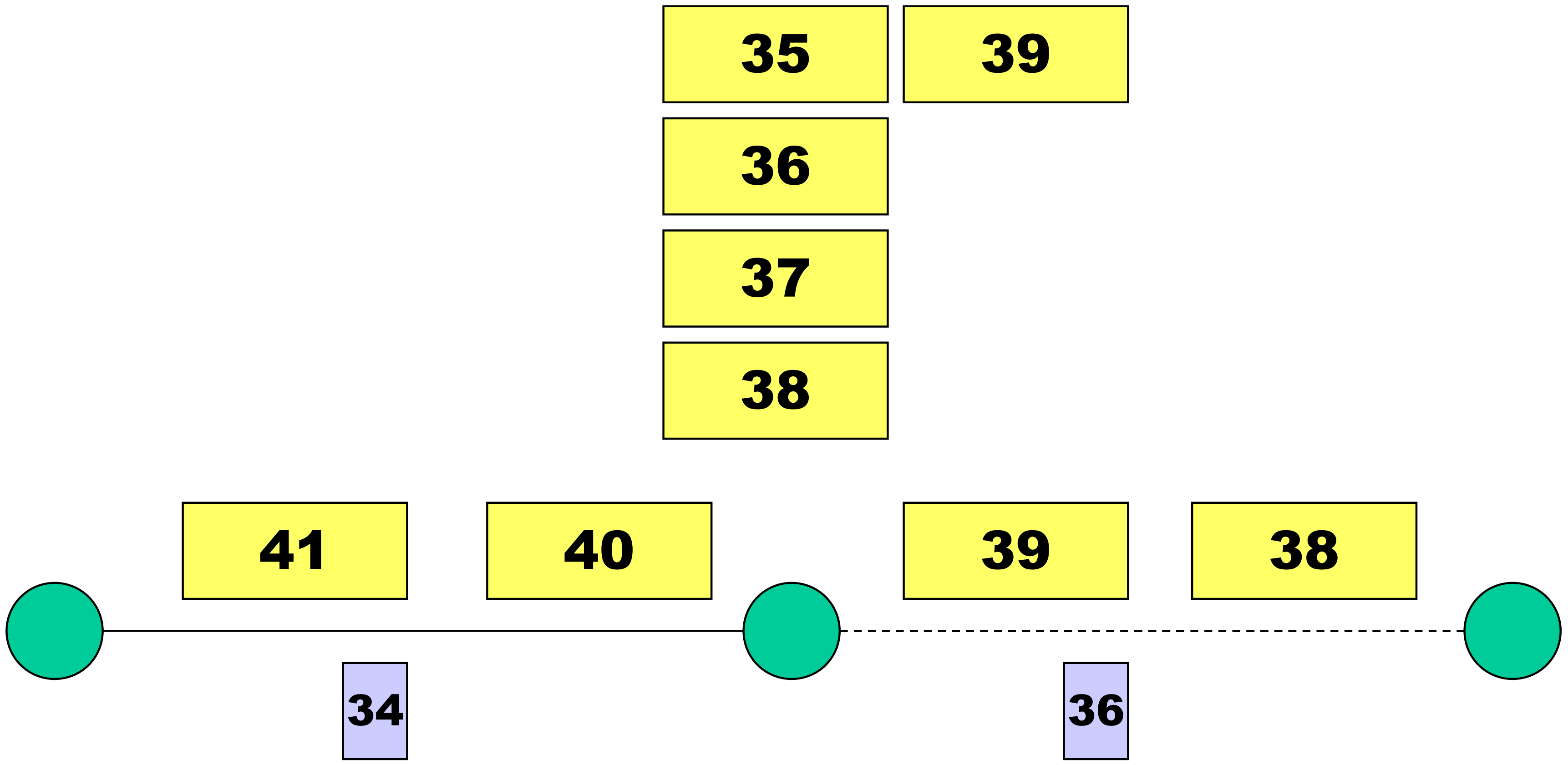


# Snoop : Example

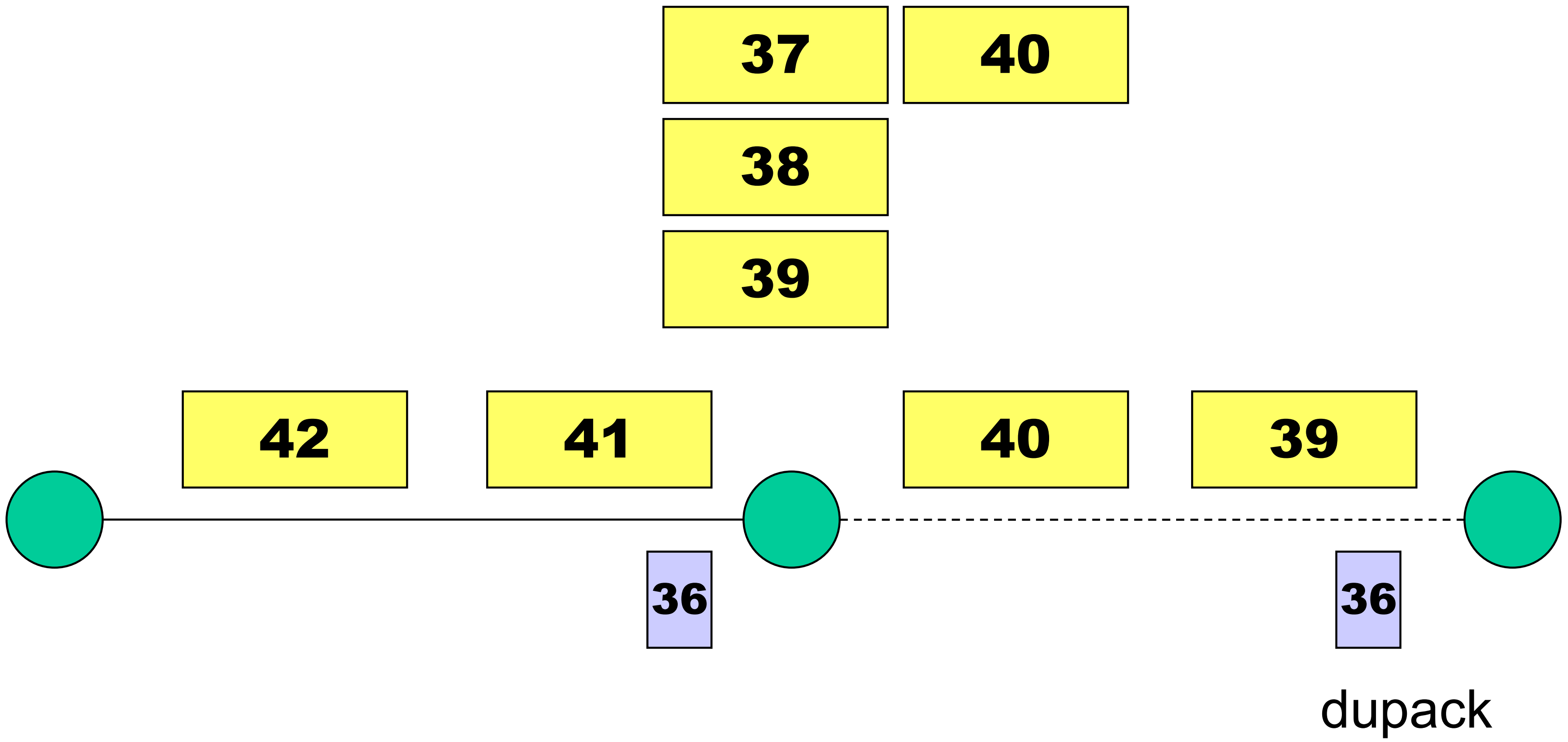


Example assumes delayed ack - every other packet ack'd

# Snoop : Example

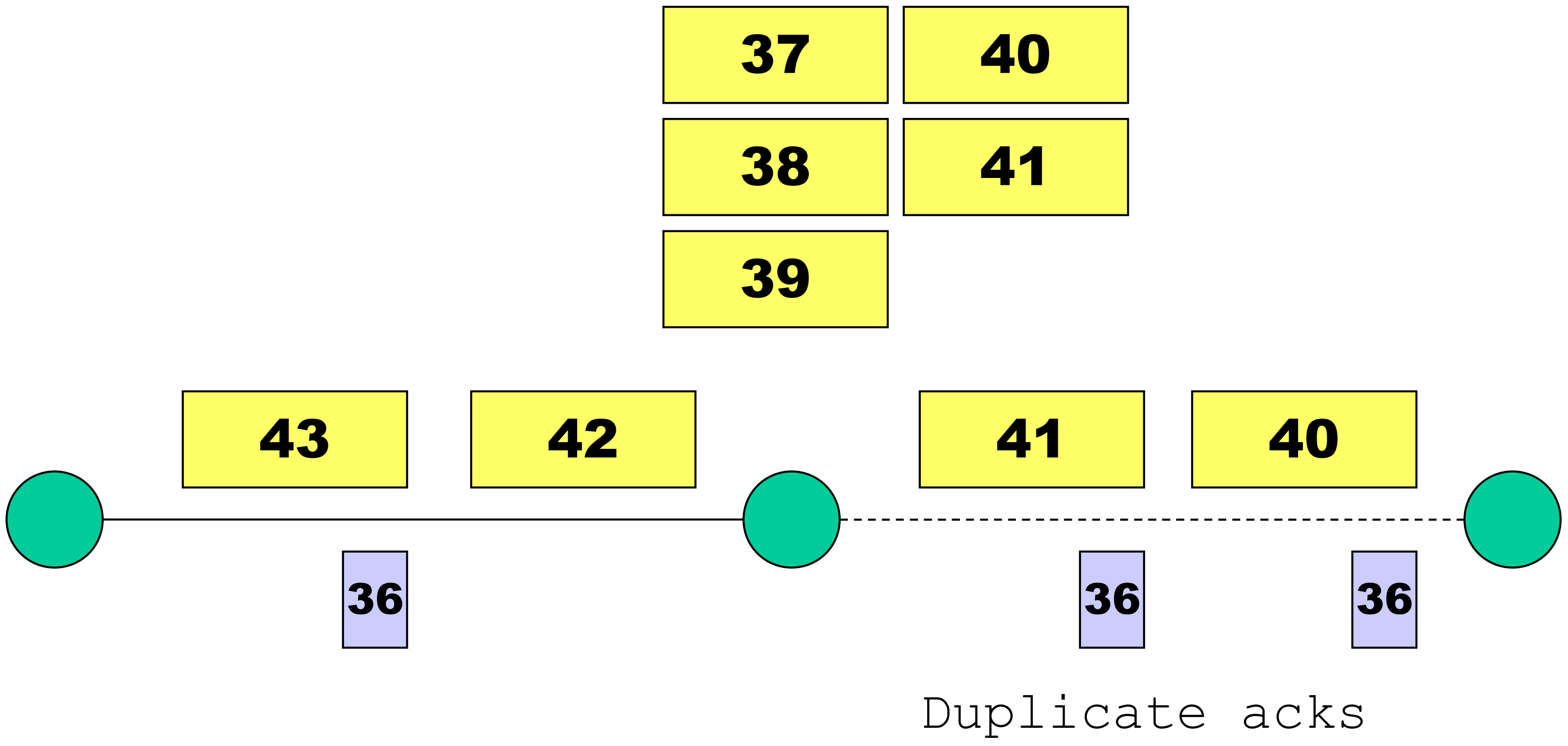


# Snoop : Example

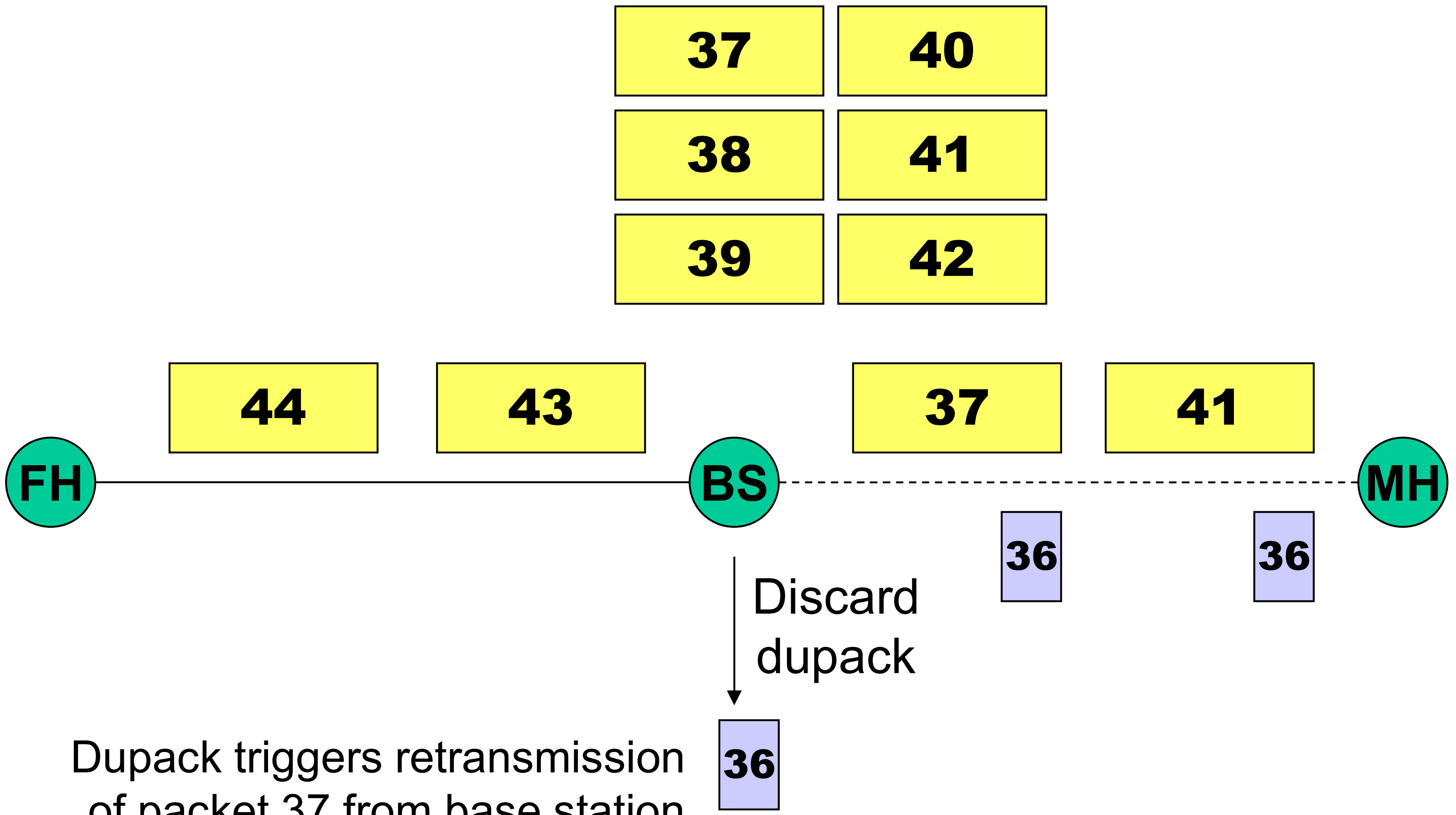


Duplicate acks are not delayed

# Snoop : Example



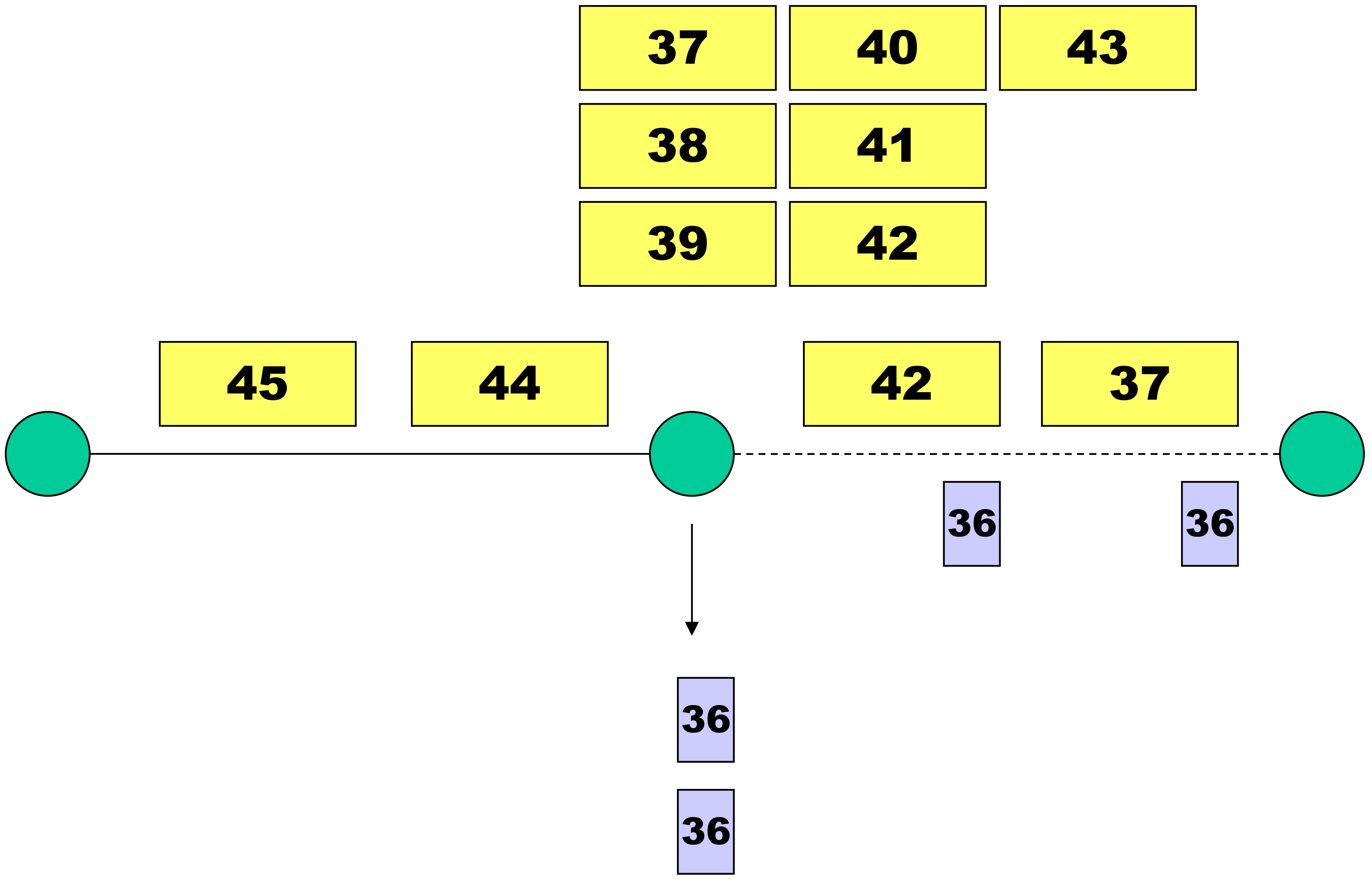
# Snoop : Example



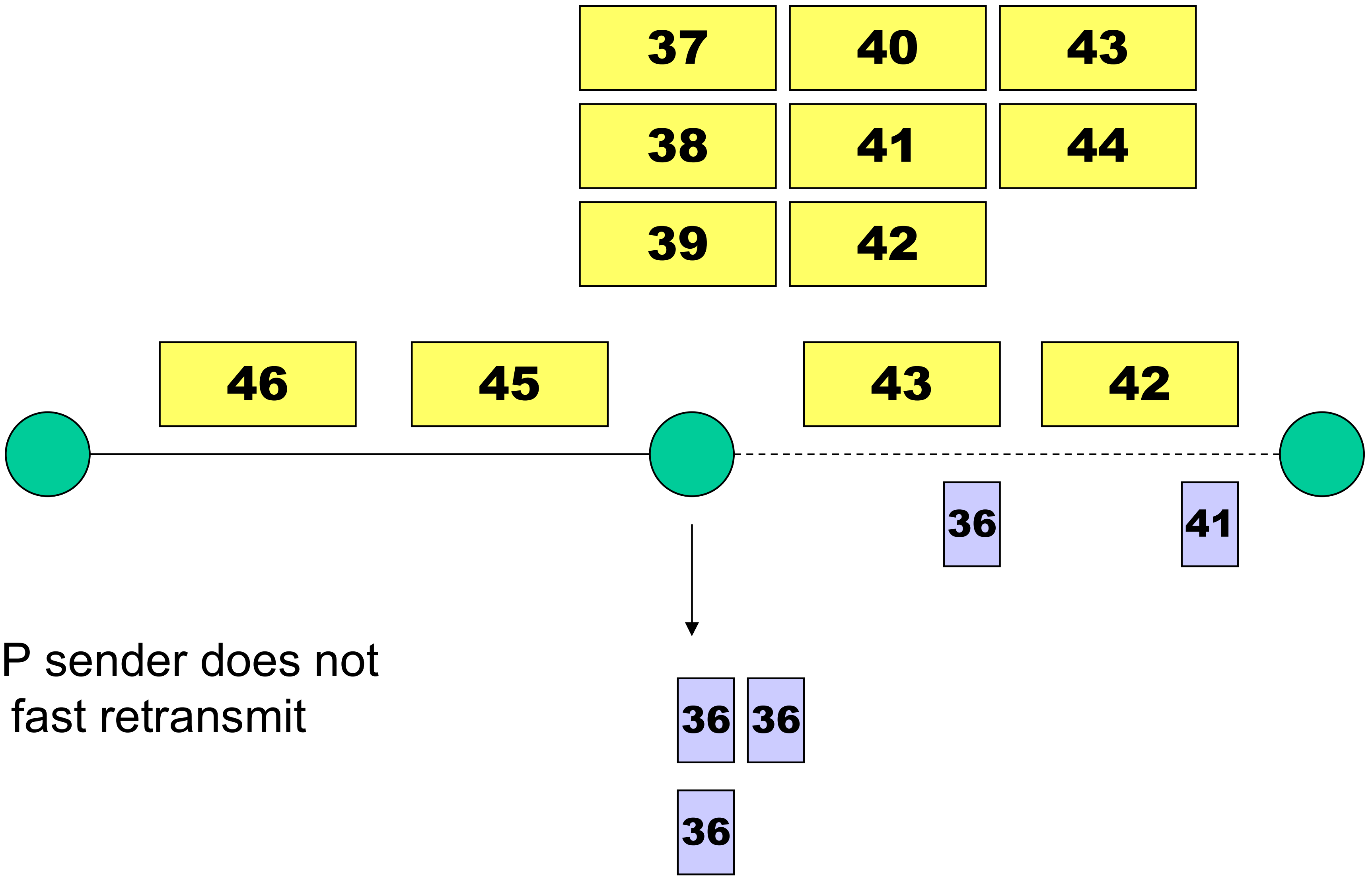
Dupack triggers retransmission of packet 37 from base station

BS needs to be TCP-aware to be able to interpret TCP headers

# Snoop : Example

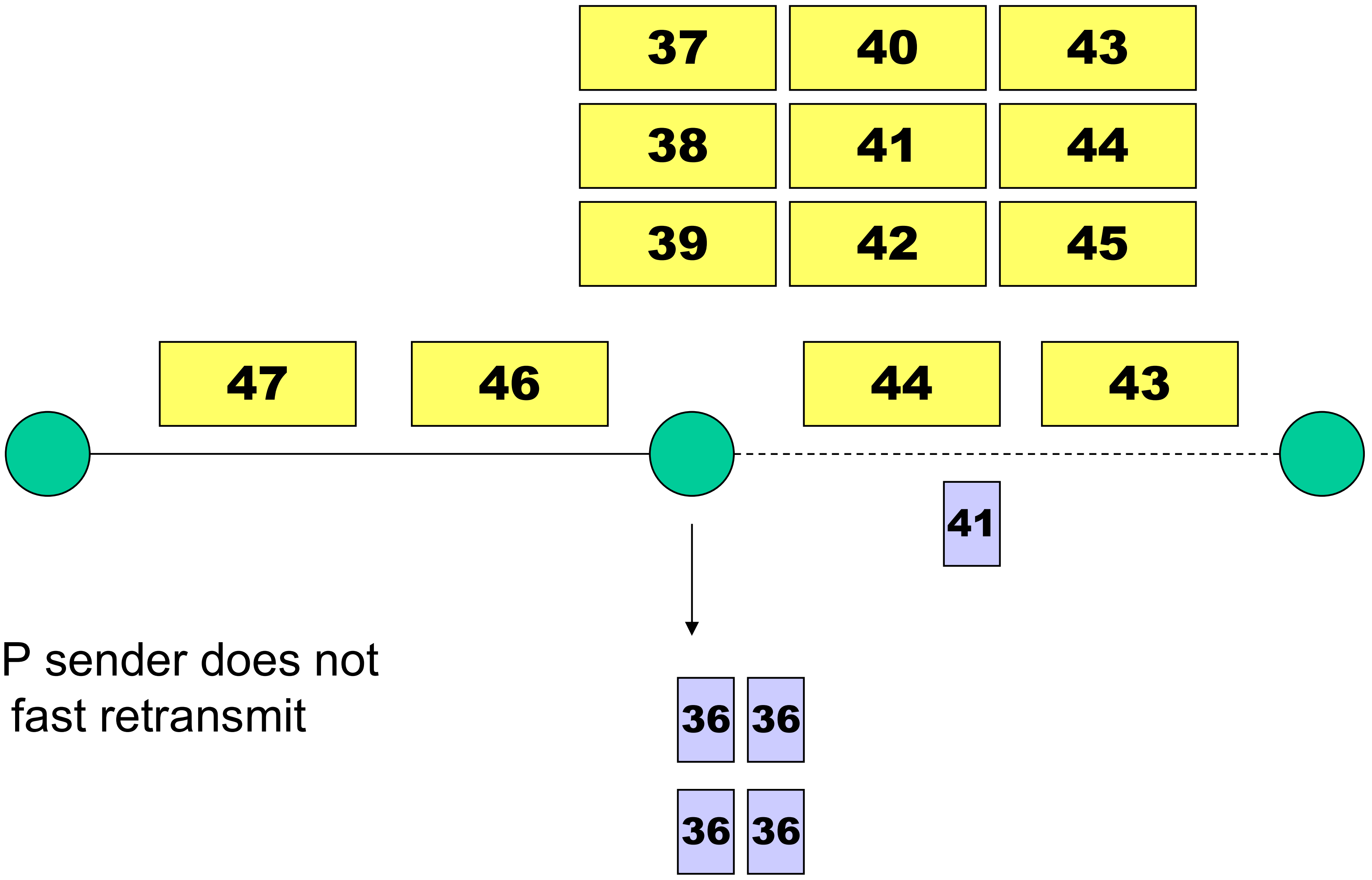


# Snoop : Example



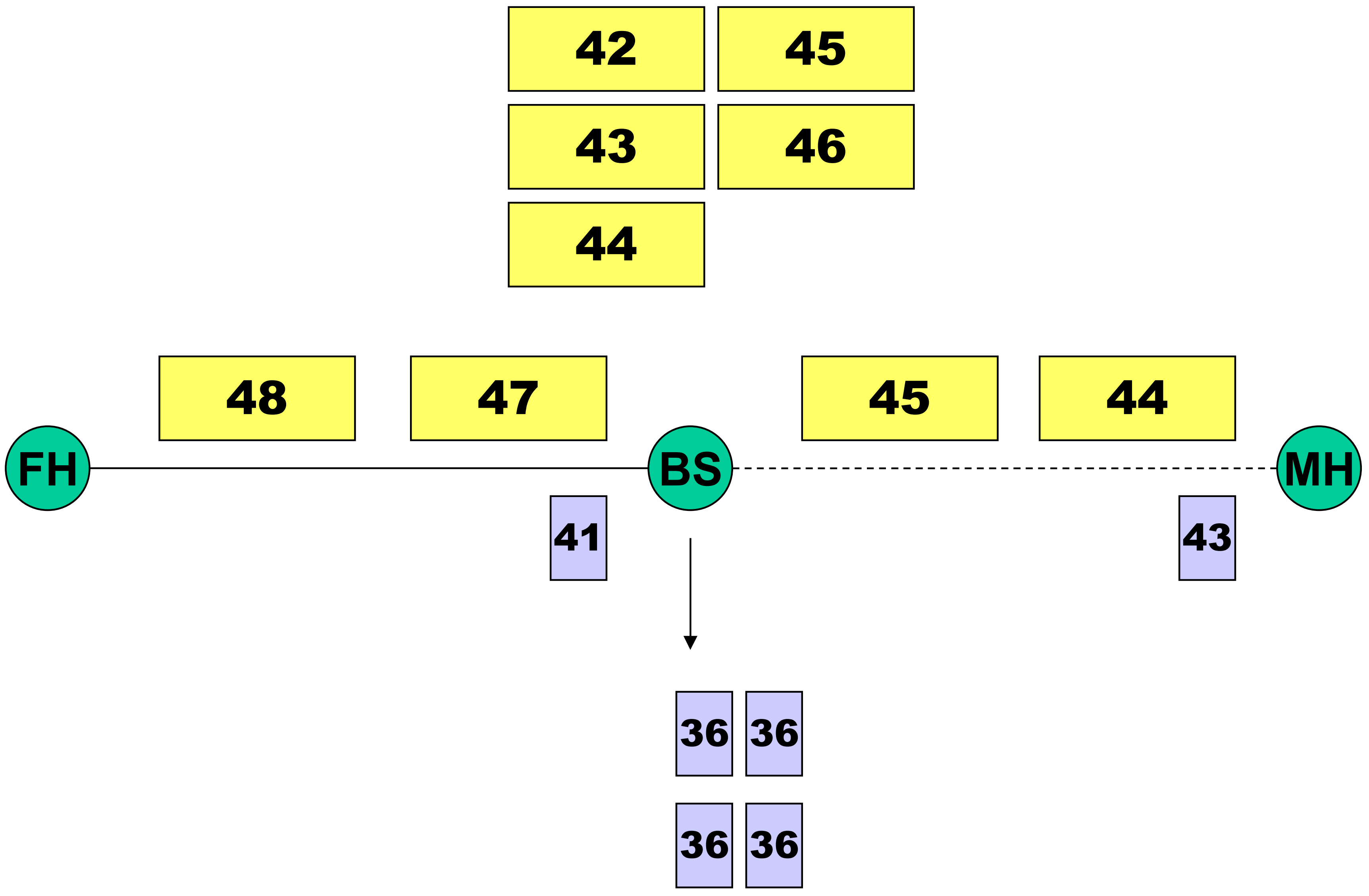
TCP sender does not fast retransmit

## Snoop : Example



TCP sender does not fast retransmit

# Snoop : Example



## Snoop Protocol

- Snoop prevents fast retransmit from sender despite transmission errors and out-of-order delivery on the wireless link

- If wireless link level delay-bandwidth product is less than 4 packets, a simple (TCP-unaware) link level retransmission scheme can suffice
    - Since delay-bandwidth product is small, the retransmission scheme can deliver the lost packet without resulting in 3 dupacks from the TCP receiver

# Snoop Protocol Characteristics

- Hides wireless losses from the sender
- Requires modification to only BS (network-centric approach)
- Pros and cons?

# 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)

## Various Schemes

- Link level mechanisms
- Split connection approach
- TCP-Aware link layer
- TCP-Unaware approximation of TCP-aware link layer
- Explicit notification
- Receiver-based discrimination
- Sender-based discrimination

$\left.\begin{matrix}\\ \\ \\ \\ \end{matrix}\right\}$ Hide losses

$\left.\begin{matrix}\\ \\ \\ \end{matrix}\right\}$ Find reasons for losses

## TCP-Unaware Approximation of TCP-Aware Link Layer

## Delayed Dupacks Protocol [Mehta98,Vaidya99]

- Attempts to imitate Snoop without making the base station TCP-aware
- Snoop implements two features at the base station
  - link layer retransmission
  - reducing interference between TCP and link layer retransmissions (by dropping dupacks)
- Delayed Dupacks implements the same two features
  - at BS: link layer retransmission
  - at MH: reducing interference between TCP and link layer retransmissions (by delaying third and subsequent dupacks)

# Delayed Dupacks Protocol

- TCP receiver delays dupacks (third and subsequent) for interval D, when out-of-order packets received
- Dupack delay intended to give link level time to retransmit
- Pros
  - Delayed dupacks can result in recovery from a transmission loss without triggering a response from the TCP sender
- Cons
  - Recovery from congestion losses delayed

## Various Schemes

- Link-layer retransmissions
- Split connection approach
- TCP-Aware link layer
- TCP-Unaware approximation of TCP-aware link layer
- Explicit notification
  - ECN: Router tags packet if it experiences congestion
  - ELN: Base station tags dup-ack with ELN if it's wireless related loss

- Receiver-based discrimination
  - Receiver attempts to guess cause of packet loss
  - When receiver believes that packet loss is due to errors, it sends a notification to the TCP sender
  - TCP sender, on receiving the notification, retransmits the lost packet without reducing congestion window

- Sender-based discrimination
  - Sender can attempt to determine cause of a packet loss
  - If packet loss determined to be due to errors, do not reduce congestion window

## Summary

| Schemes | Idea | Who | Characteristics |
|---|---|---|---|
| Link layer | Link layer retx | Wireless end points | Hide wireless error |
| Split connection | Local retx + independent optimization of wireless conn. | Base station | Hide wireless error |
| Snoop | Link layer retx + drop dup ACK at base station | Base station | Hide wireless error + avoid unnecessary cwnd reduction |
| Delayed dup ACK | Link layer retx + delay dup ACK at wireless host | Base station Wireless host | Hide wireless error + avoid unnecessary cwnd reduction |

# Summary (Cont.)

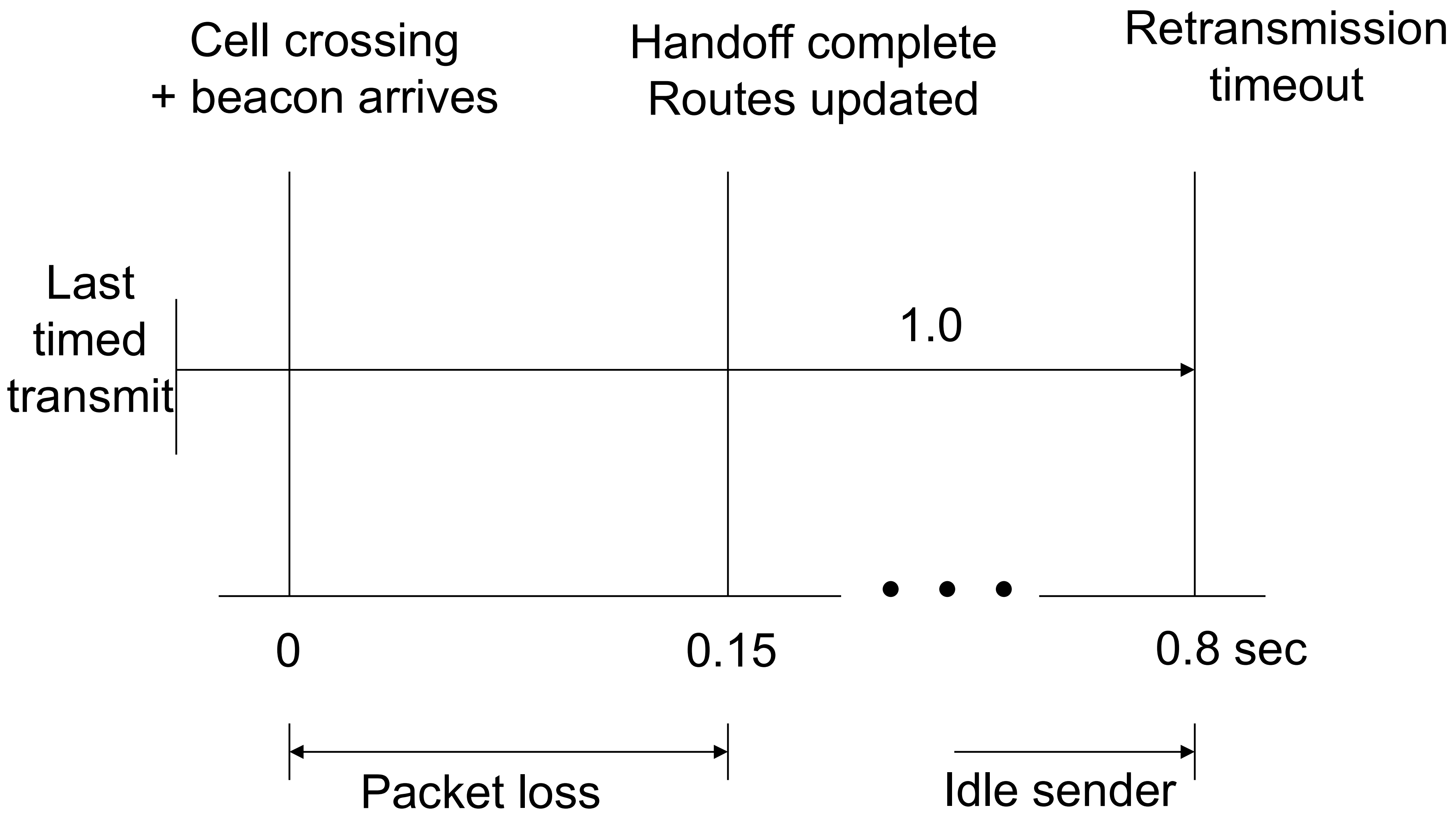
| Schemes | Idea | Who | Characteristics |
|---|---|---|---|
| ELN | Tag dup ack with ELN if loss occurs at wireless link | Base station | Avoid unnecessary cwnd reduction |
| Receiver | When receiver believes that packet loss is due to errors, it sends a notification to the TCP sender | Receiver Sender | Avoid unnecessary cwnd reduction |
| Sender | When sender believes that packet loss is due to errors, it does not reduce cwnd. | Sender | Avoid unnecessary cwnd reduction |

## Techniques to Improve TCP Performance in Presence of Mobility

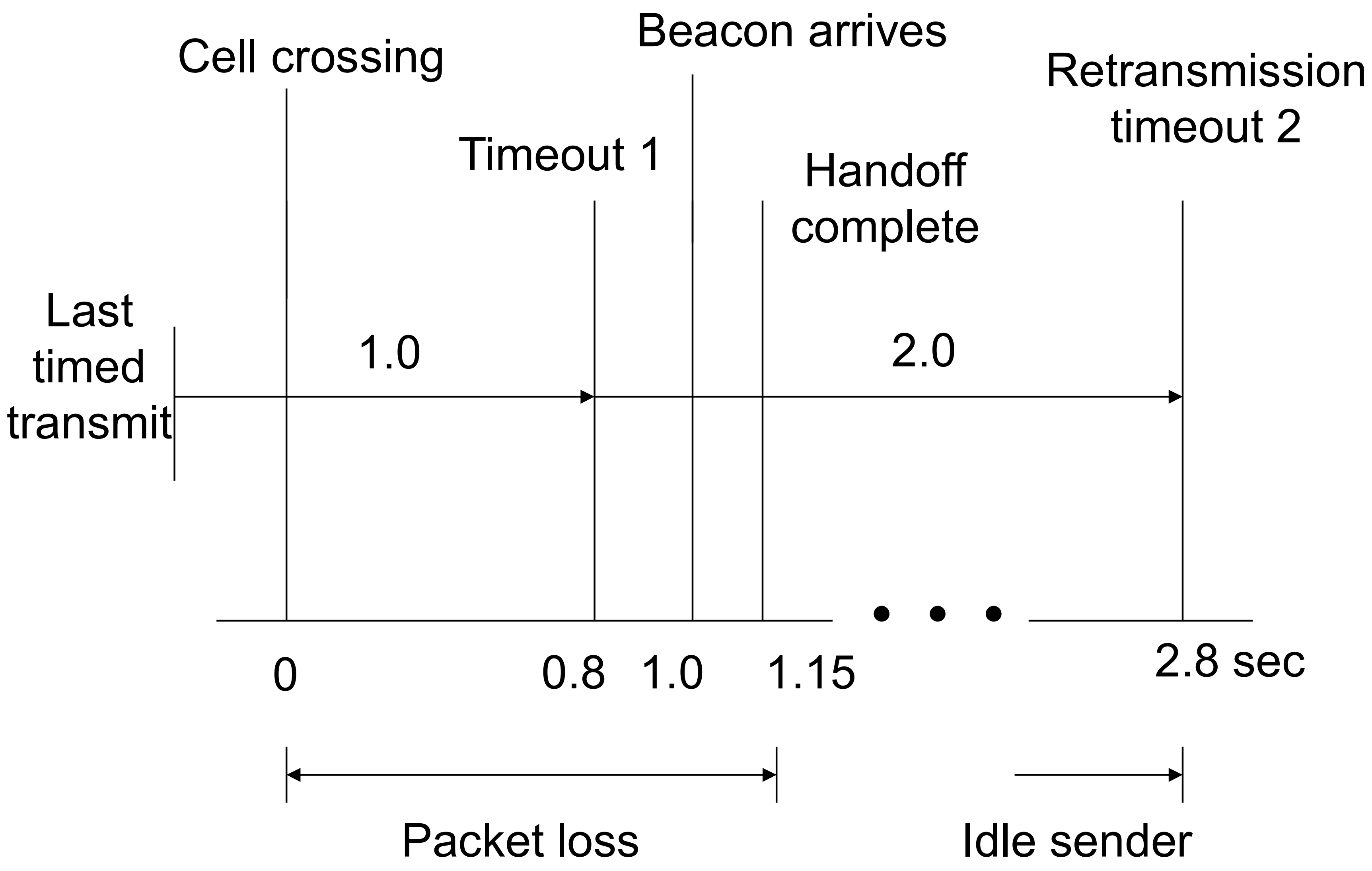
## Classification

- Hide mobility from the TCP sender
- Make TCP adaptive to mobility

## 0-second Rendezvous Delay : Beacon arrives as soon as cell boundary crossed



# 1-second Rendezvous Delay : Beacon arrives 1 second after cell boundary crossed



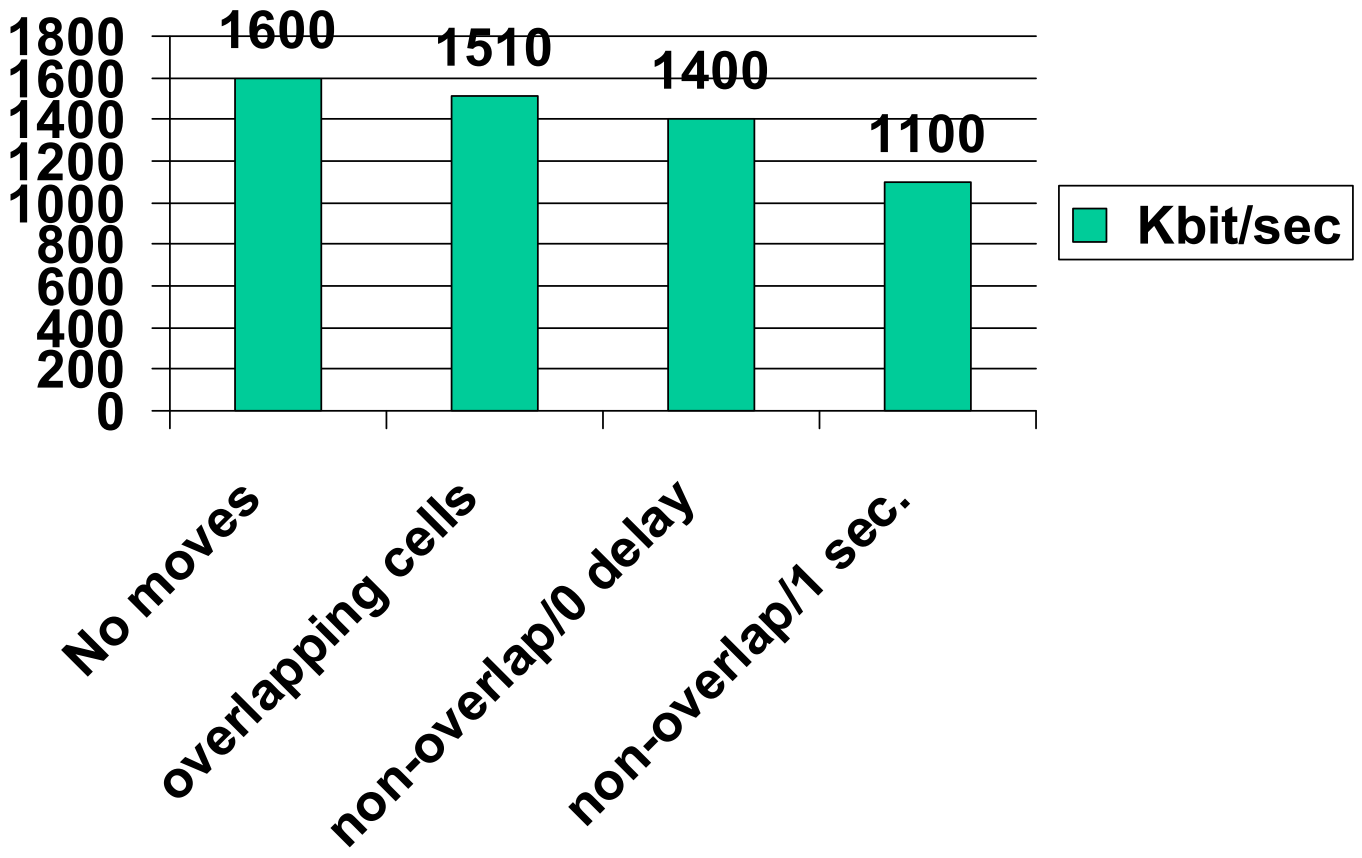
## Performance [Caceres95]

Four environments

1. No moves
2. Moves (once per 8 sec) between overlapping cells
3. Moves between non-overlapping cells, 0 sec delay
4. Moves between non-overlapping cells, 1 sec delay

Experiments using 2 Mbps WaveLan

# TCP Performance



## TCP Performance

- Degradation in case 2 (overlapping cells) is due to *encapsulation and forwarding delay* during handoff

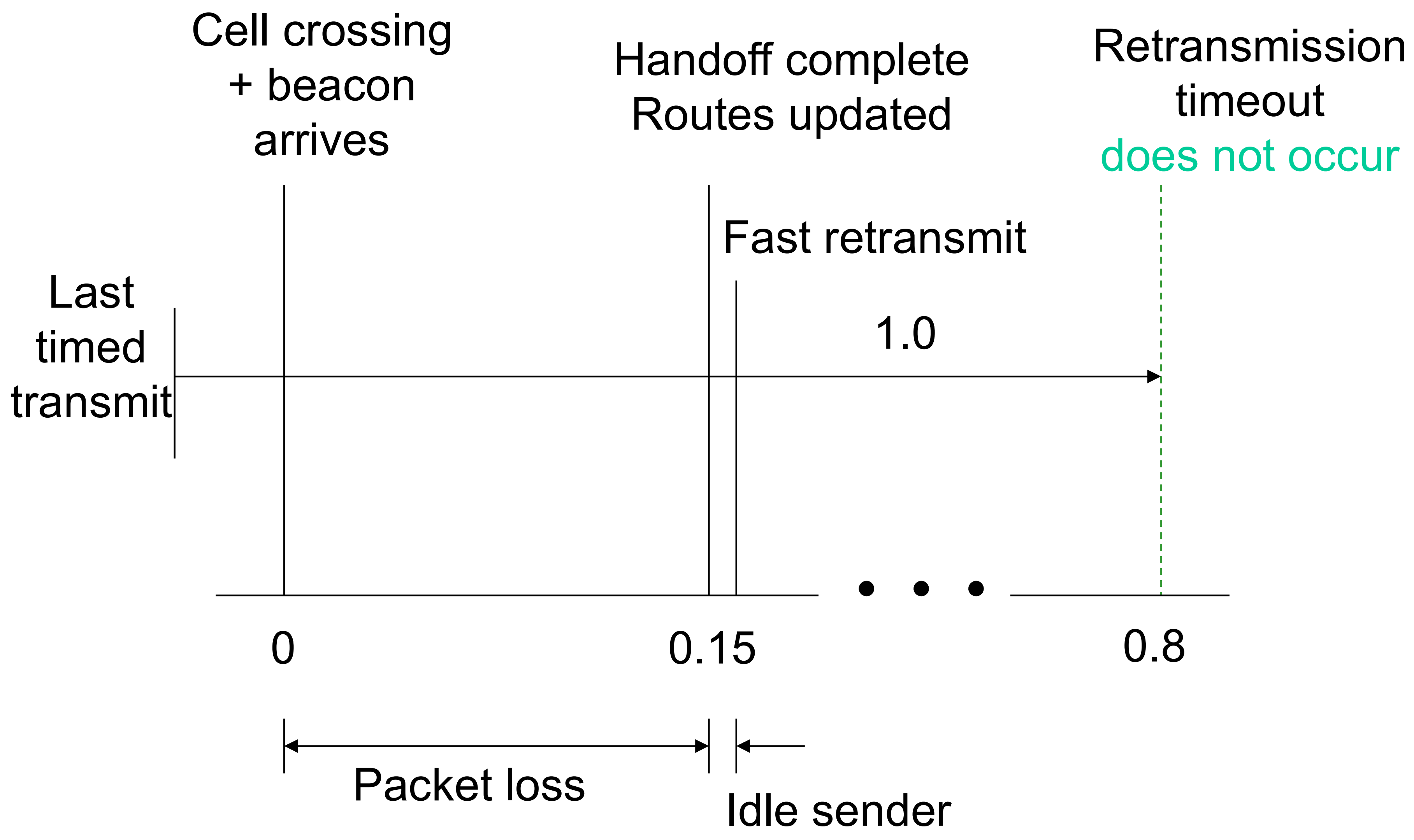
- Additional degradation in cases 3 and 4 due to packet loss and idle time at sender

## How to improve case 3 & 4?

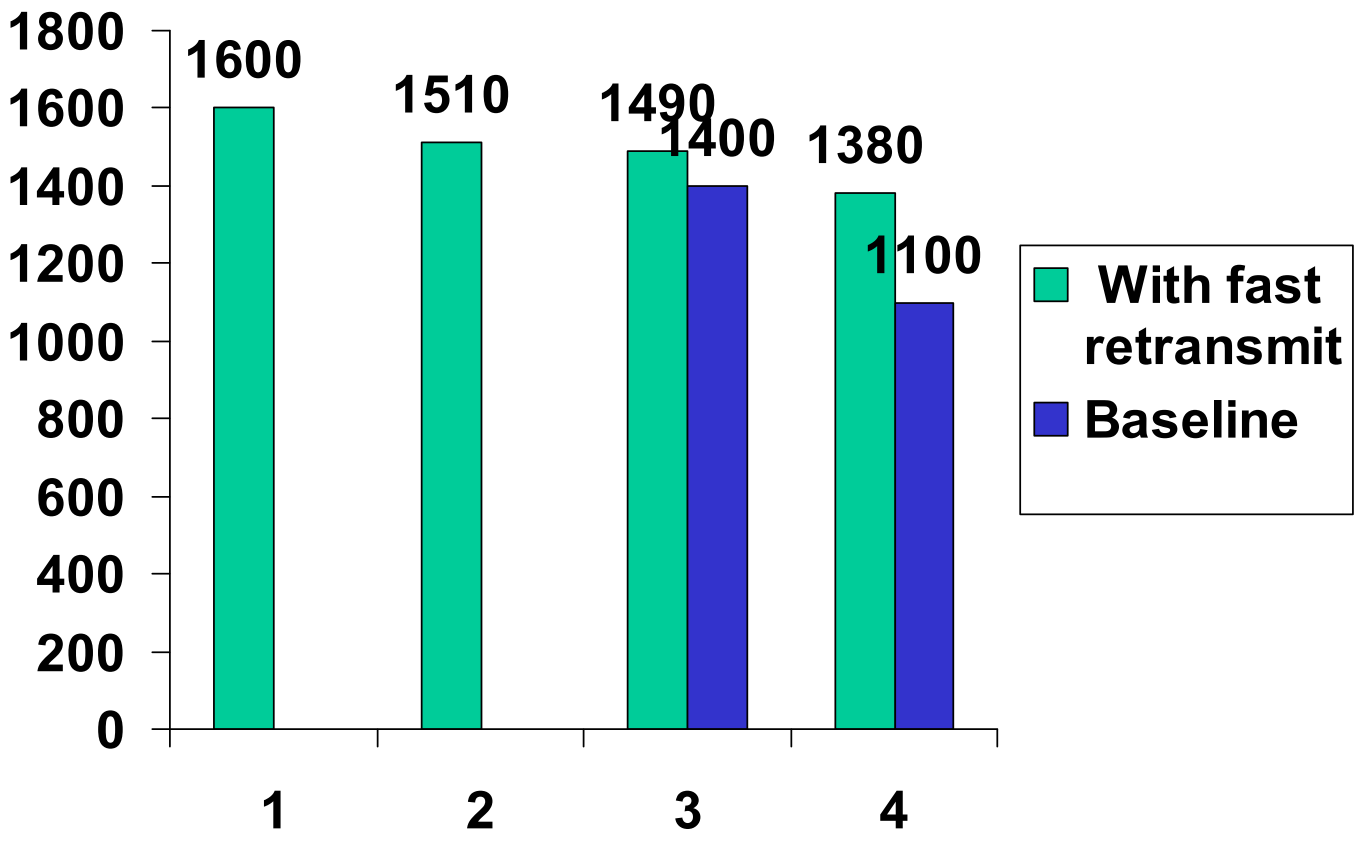
## Mitigation Using Fast Retransmit

- When MH is the TCP receiver: after handoff is complete, it sends 3 dupacks to the sender
  - this triggers fast retransmit at the sender
  - instead of dupacks, a special notification could also be sent
- When MH is the TCP sender: invoke fast retransmit after completion of handoff

# 0-second Rendezvous Delay Improvement using Fast Retransmit


Cell crossing + beacon arrives
Handoff complete Routes updated
Retransmission timeout does not occur
Fast retransmit
Last timed transmit
1.0
0
0.15
0.8
Packet loss
Idle sender

# TCP Performance Improvement



# TCP Performance Improvement

- No change in cases 1 and 2, as expected
- Improvement for non-overlapping cells
- Some degradation remains in case 3 and 4
  - fast retransmit reduces congestion window

## Other techniques to improve performance?

## Improving Performance by Smooth Handoffs [Caceres95]

- Provide sufficient overlap between cells to avoid packet loss

or

- Buffer packets at BS
  - Discard the packets after a short interval
  - If handoff occurs before the interval expires, forward the packets to the new base station
  - Prevents packet loss on handoff

## M-TCP [Brown97]

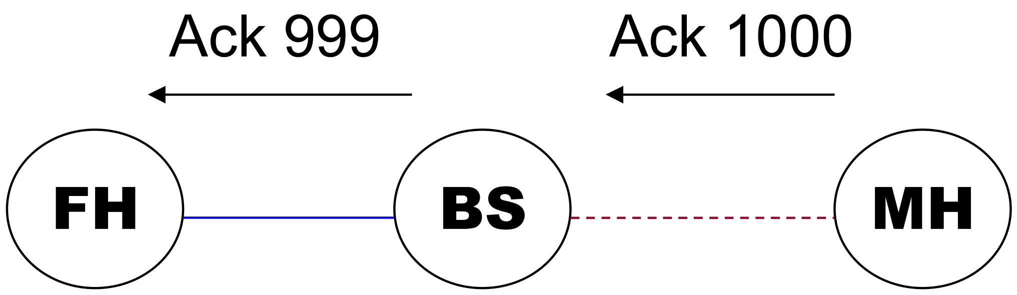
- In the fast retransmit scheme [Caceres95]
  - sender starts transmitting soon after handoff
  - BUT congestion window shrinks
- M-TCP attempts to avoid shrinkage in the congestion window

## M-TCP Uses TCP Persist Mode

- When a new ack is received with receiver's advertised window = 0, the sender enters persist mode
- Sender does not send any data in persist mode
  - except when persist timer goes off
- When a positive window advertisement is received, sender exits persist mode
- On exiting persist mode, RTO and cwnd are same as before the persist mode

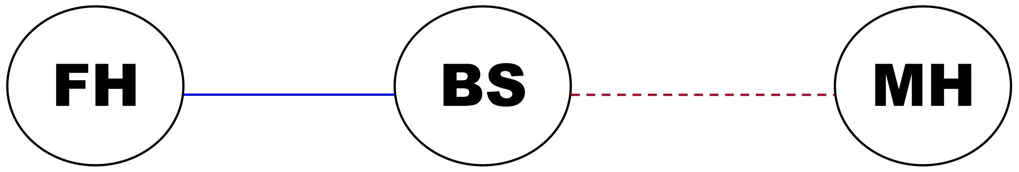
# M-TCP

- Similar to the split connection approach, M-TCP splits one TCP connection into two logical parts
  - the two parts have independent flow control as in I-TCP
- The BS does not send an ack to FH, unless BS has received an ack from MH
  - maintains end-to-end semantics
- BS withholds ack for the last byte ack'd by MH



## M-TCP

- Withheld ack sent with window advertisement = 0, if MH moves away (handoff in progress)
- Sender FH put into persist mode during handoff
- Sender exits persist mode after handoff, and starts sending packets using same cwnd as before handoff



Withholding the last ack may cause timeout near the end of transfer. How to avoid it?

## M-TCP

- The last ack is not withheld, if BS does not expect any other ack from the MH
  - this happens when the BS has no other unack'd data buffered locally
  - this is required to prevent a sender timeout at the end of a transfer (or end of a burst of data)

# M-TCP

- Avoids reduction of congestion window due to handoff, unlike the fast retransmit scheme
  - simulation-based performance results look good
- Important Question unanswered : Is not reducing the window a good idea?

## M-TCP

- Avoids reduction of congestion window due to handoff, unlike the fast retransmit scheme
    - simulation-based performance results look good
- Important Question unanswered : Is not reducing the window a good idea?

  When host moves, route changes, and new route may be more congested than before.

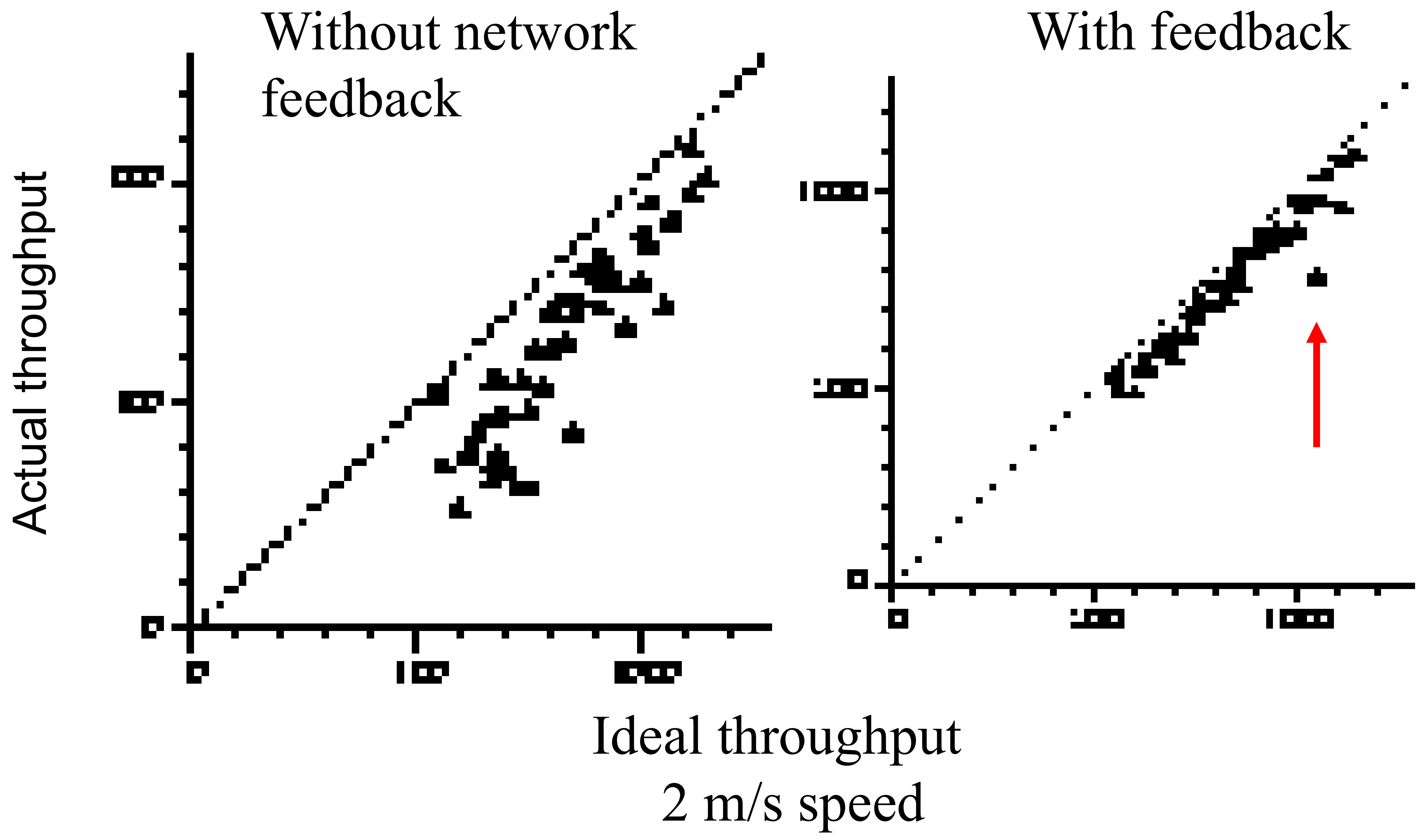
  Starting full speed after handoff may be too aggressive.

## TCP in Mobile Ad Hoc Networks

## How to Improve Throughput

- Network feedback
- Inform TCP of route failure by explicit message
- Let TCP know when route is repaired
  - Probing
  - Explicit notification
- Reduces repeated TCP timeouts and backoff

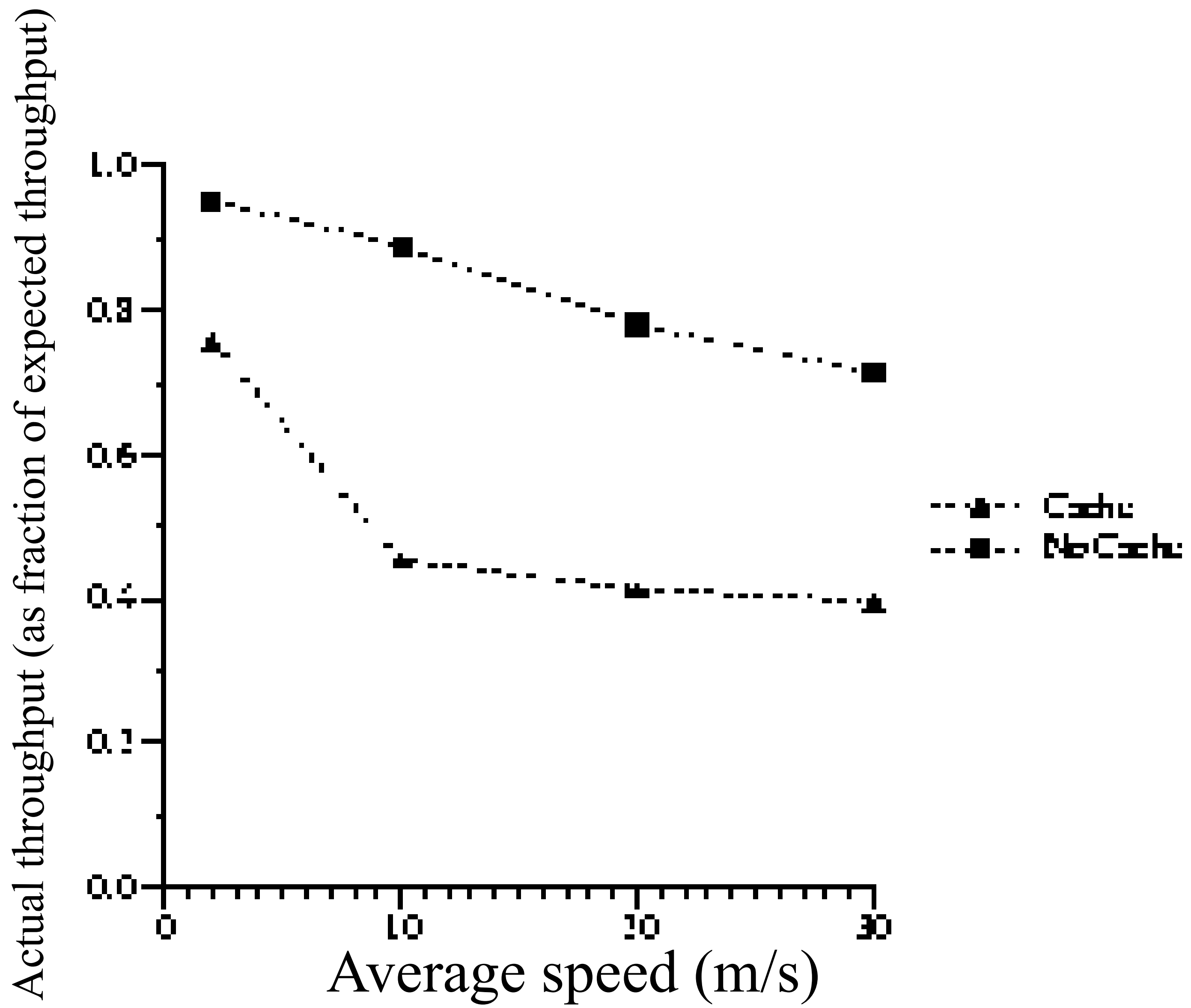
# Performance Improvement



# Impact of Route Caching

- Route caching has been suggested as a mechanism to reduce route discovery overhead [Broch98]
- Each node may cache one or more routes to a given destination
- When a route from S to D is detected as broken, node S may:
  - Use another cached route from local cache, or
  - Obtain a new route using cached route at another node

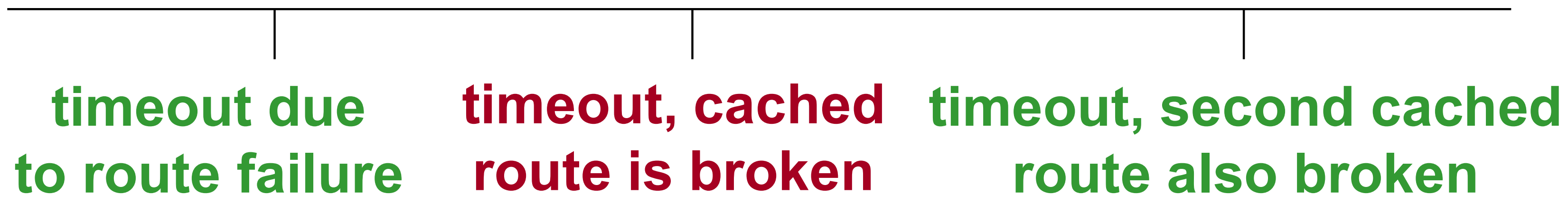
# To Cache or Not to Cache



# Why Performance Degrades With Caching

- When a route is broken, route discovery returns a cached route from local cache or from a nearby node

- After a time-out, TCP sender transmits a packet on the new route.
  However, the cached route has also broken after it was cached


timeout due to route failure
timeout, cached route is broken
timeout, second cached route also broken

- Another route discovery, and TCP time-out interval
- Process repeats until a good route is found

# Issues: To Cache or Not to Cache

- Caching can result in faster route "repair"
- Faster does not necessarily mean correct
- If incorrect repairs occur often enough, caching performs poorly
- Need mechanisms for determining when cached routes are stale

## Caching and TCP performance

- Caching can reduce overhead of route discovery even if cache accuracy is not very high
- But if cache accuracy is not high enough, gains in routing overhead may be offset by loss of TCP performance due to multiple time-outs

# A Comparison of Mechanisms for Improving TCP Performance over Wireless Links

Hari Balakrishnan
Venkata N. Padmanabhan
Srinivasan Seshan
Randy H.Katz

UC Berkeley

What questions would you like to answer when comparing different schemes to improve TCP?

## Goals

- What combination of mechanisms result in best performance?
  - Link layer approach
    - TCP-aware vs. TCP-agnostic
  - TCP variants
- How important is it for LL-scheme to be TCP-aware?
- How useful is SACK and SMART for dealing with (bursty) lossy links?
- Is it important to split TCP to get good performance?

## Lessons

- TCP-aware link layer is better than link layer approach alone
    - TCP-aware link-layer protocol with SACK performs the best
- End-to-end schemes (e.g., ELN, SMART, SACK) are useful
- Split connection is not necessary for good performance

## Review

- How does TCP work?
- Why does TCP perform poorly in wireless networks?
- What are two classes of solutions?
- How do they work?
- What issues do mobility introduce to TCP performance?
- How can we address them?

## Review

- Why does TCP not perform well in wireless networks?
- What are two classes of solutions?
- How do they work?
- What issues do mobility introduce to TCP performance?
- How can we address them?