# CS/ECE 439: Wireless Networking

## Transport Layer – TCP over Wireless

 Fall 2024

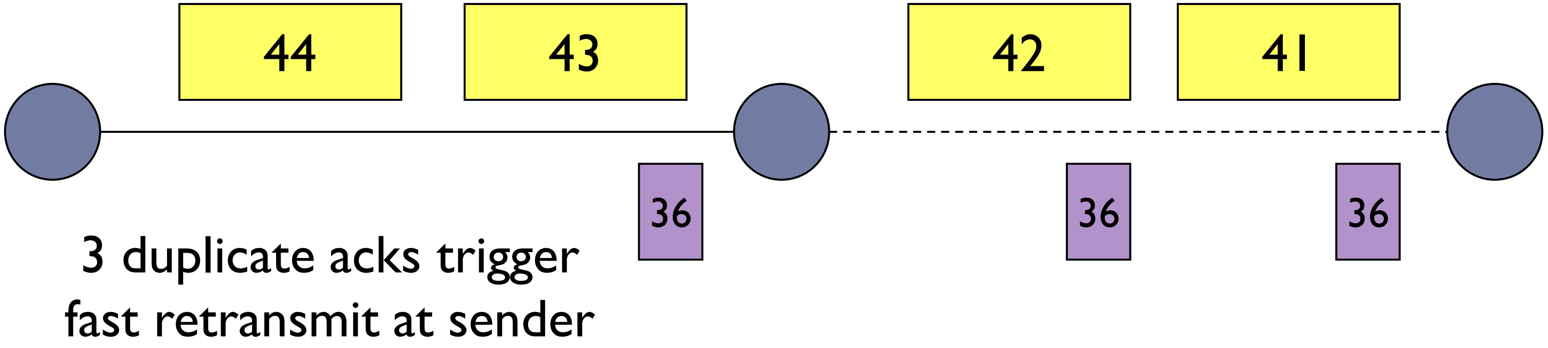
# Wireless Characteristics

- Low bandwidth
- Long or variable latency

- Random Errors
  - If number of errors is small
    - May be corrected by an error correcting code
  - Excessive bit errors
    - Result in a packet being discarded, possibly before it reaches the transport layer

# Random Errors

- May cause fast retransmit
  - Example assumes delayed ack - every other packet ack'd



# Random Errors

- Fast retransmit results in
  - Retransmission of lost packet
  - Reduction in congestion window
- Reducing congestion window
  - Unnecessary response to errors
  - Reduces the throughput

# Random Errors

- Sometimes congestion response is appropriate
  - Interference due to other users
    - Reduce congestion window
  - Bad channel for a long duration
    - Let TCP backoff
    - Do not unnecessarily attempt retransmissions while the channel remains in the bad state
- But what about errors for which reducing congestion window is an inappropriate response?
  - Noise
    - Do not reduce window

# Timeouts

- Burst errors may cause timeouts
  - If wireless link remains unavailable for extended duration, a window worth of data may be lost
    - Driving through a tunnel
    - Passing a truck
  - Timeout results in slow start
    - Slow start reduces congestion window to 1 MSS, reducing throughput
    - Reduction in window in response to burst errors?
- Random errors may also cause timeouts
  - Multiple packet losses in a window can result in timeout when using TCP-Reno
    - And to a lesser extent when using SACK

# Impact of Transmission Errors

- TCP cannot distinguish between packet losses due to congestion and transmission errors
  - Unnecessarily reduces congestion window
  - Throughput suffers

# Ideal Behavior

- Ideal TCP behavior
  - Simply retransmit a packet lost due to transmission errors
  - Take no congestion control actions
    - Ideal TCP typically not realizable
- Ideal network behavior
  - Transmission errors should be hidden from the sender
  - Errors should be recovered transparently and efficiently
- Proposed schemes attempt to approximate one of the above two ideals

# Techniques

- Nature of actions taken to improve performance
  - Hide error losses from the sender
    - Sender is unaware of error-based losses
      - Will not reduce congestion window
  - Let sender know, or determine, cause of packet loss
    - Sender knows about cause of packet loss
      - Will not reduce congestion window

# Techniques

- Where modifications are needed
  - At the sender node only
  - At the receiver node only
  - At intermediate node(s) only
  - Combinations of the above

# 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

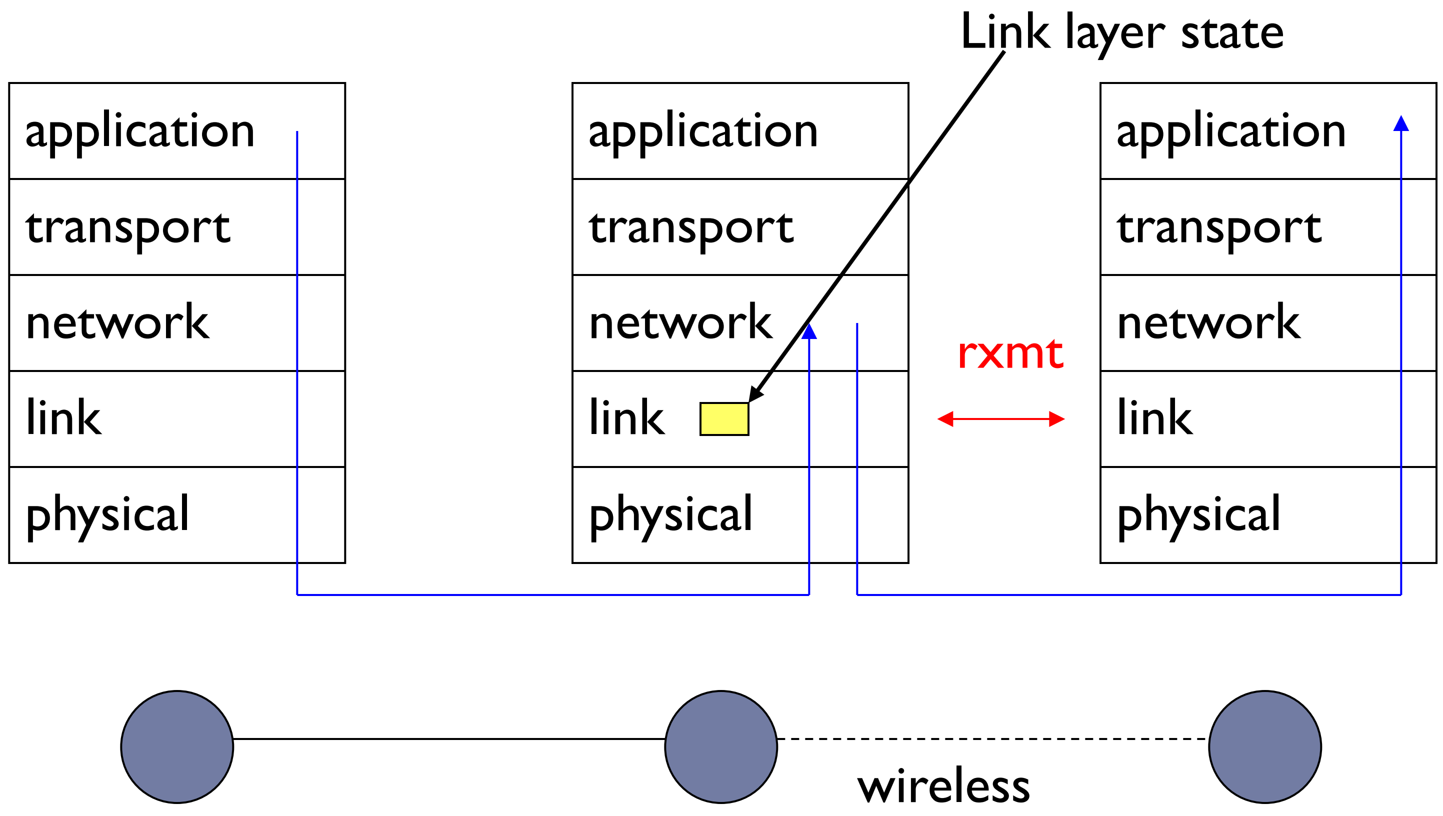
# Link Layer Mechanisms: Forward Error Correction

- Forward Error Correction (FEC) can be used to correct small number of errors
  - Correctable errors hidden from the TCP sender
  - FEC incurs overhead even when errors do not occur
    - Adaptive FEC schemes can reduce the overhead by choosing appropriate FEC dynamically

# Link Layer Mechanisms: Link Level Retransmissions

- Retransmit a packet at the link layer, if errors are detected
- Retransmission overhead incurred only if errors occur
  - Unlike FEC overhead
- In general
  - Use FEC to correct a small number of errors
  - Use link level retransmission when FEC capability is exceeded

# Link Level Retransmissions



# Link Level Retransmissions

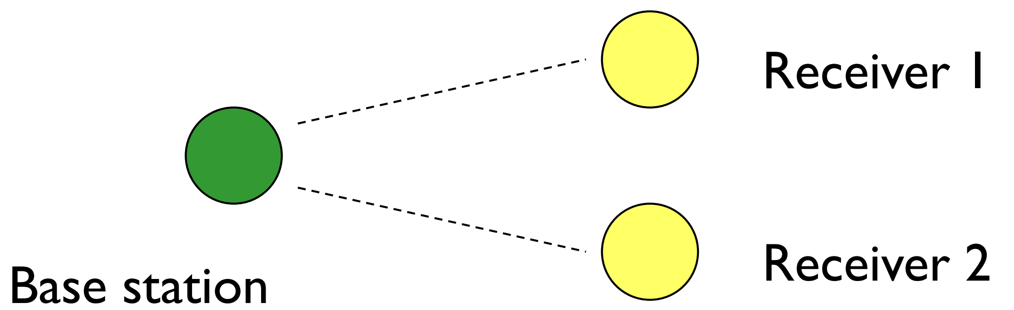
- How many retransmissions at the link level before giving up?
  - Finite bound -- semi-reliable link layer
  - No bound -- reliable link layer
- What triggers link level retransmissions?
  - Link layer timeout mechanism
  - Link level acks (negative acks, dupacks, …)
  - Other mechanisms (e.g., Snoop, as discussed later)

# Link Level Retransmissions

- 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

- Retransmissions can cause
  - Head-of-the-line blocking
  - Congestion losses



# Link Level Retransmissions

- The sender's Retransmission Timeout (RTO)
  - Function of measured RTT (round-trip times)
  - Link level retransmits increase RTT, therefore, RTO
- Infrequent errors
  - RTO will not account for RTT variations due to link level retransmissions
- Frequent errors
  - Increase RTO significantly on slow wireless links

# Link Level Retransmissions

- Not all connections benefit from retransmissions
  - Audio
- Need to be able to specify requirements on a per-packet basis
  - Should the packet be retransmitted?
  - How many times?
- Need a standard mechanism to specify the requirements

# Link Layer Schemes: Summary

- When is a reliable link layer beneficial to TCP performance?
  - If TCP retransmission timeout is large enough to tolerate additional delays due to link level retransmits

# Link Layer Mechanisms: Hiding Losses

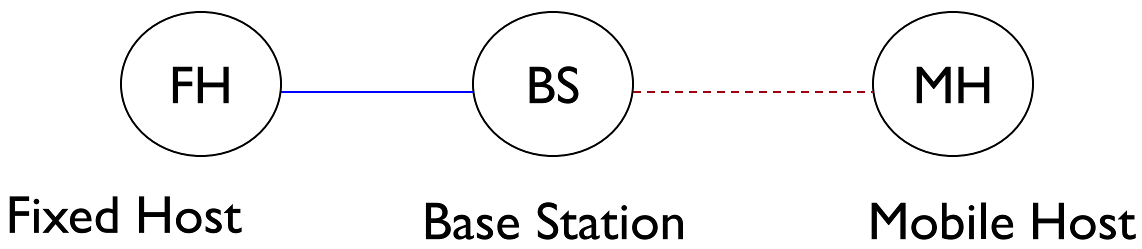
- Hide wireless losses from TCP sender
- Link layer modifications needed at both ends of wireless link
  - TCP need not be modified

# Split Connection Approach

- End-to-end TCP connection is broken into
  - One connection on the wired part of route
  - One over wireless part of the route
- A single TCP connection split into two TCP connections
  - If wireless link is not last on route
    - More than two TCP connections may be needed

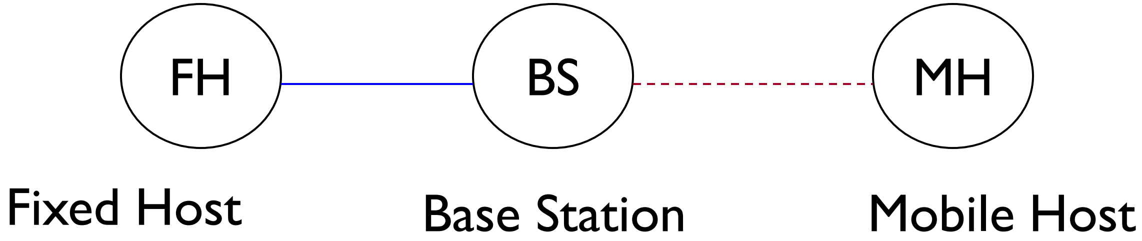
# Split Connection Approach

- Connection between wireless host MH and fixed host (FH) goes through base station (BS)
  - FH -> MH = FH -> BS + BS -> MH



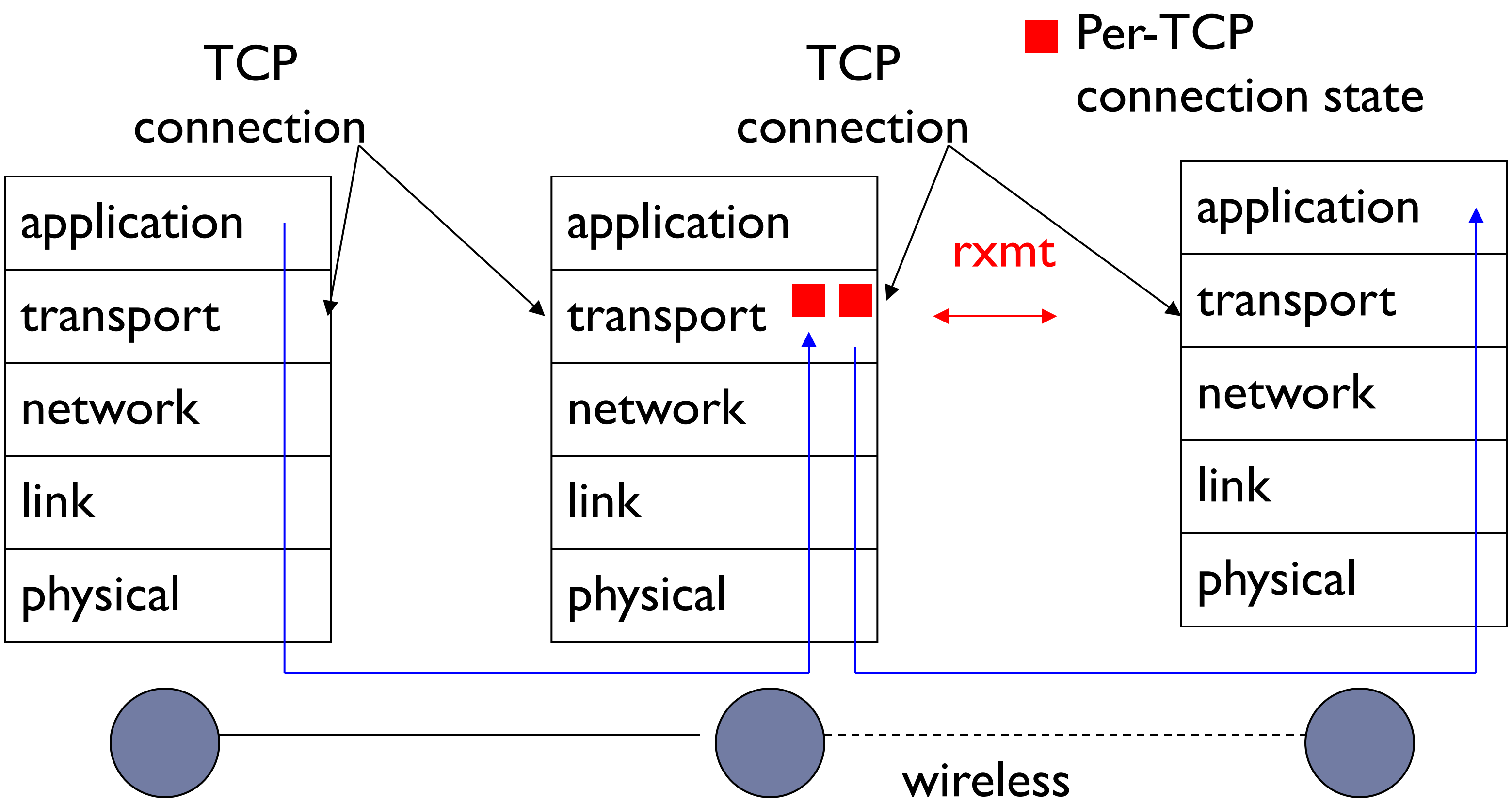
# Split Connection Approach

- Split connection results in independent flow control for the two parts
- Flow/error control protocols, packet size, time-outs, may be different for each part



# Split Connection Approach

1 TCP = ½ TCP + ½ (TCP or XXX)



# Split Connection Approach

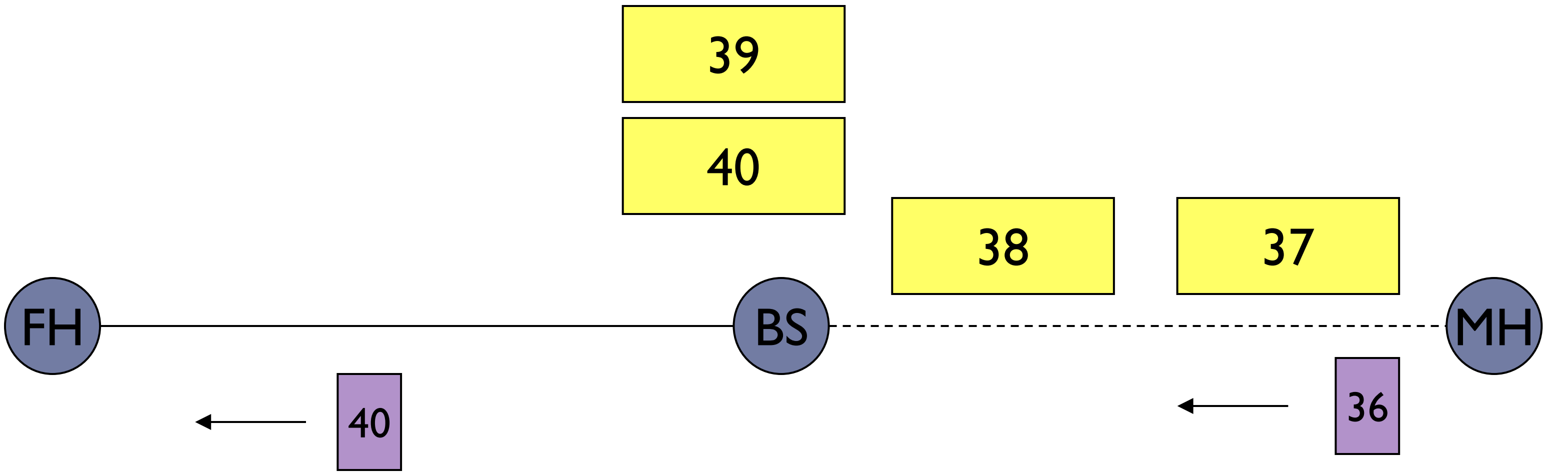
- Indirect TCP
  - FH -> BS connection : Standard TCP
  - BS -> MH connection : Standard TCP
- Selective Repeat Protocol (SRP)
  - FH -> BS connection : standard TCP
  - BS -> FH connection : selective repeat protocol on top of UDP
  - Performance better than Indirect-TCP (I-TCP)
    - Wireless portion of connection can be tuned to wireless behavior

# Split Connection Approach: Advantages

- BS-MH connection can be optimized independent of FH-BS connection
- Local recovery of errors
- Good performance achievable using appropriate BS-MH protocol
  - Standard TCP on BS-MH performs poorly
  - Selective acks improve performance

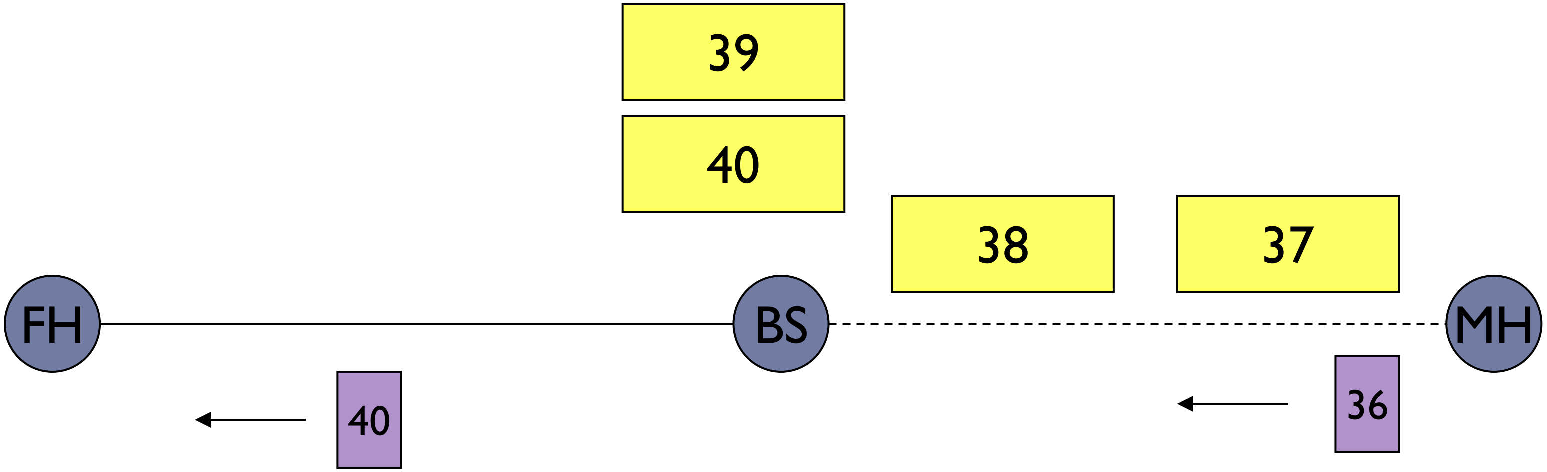
# Split Connection Approach : Disadvantages

- 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



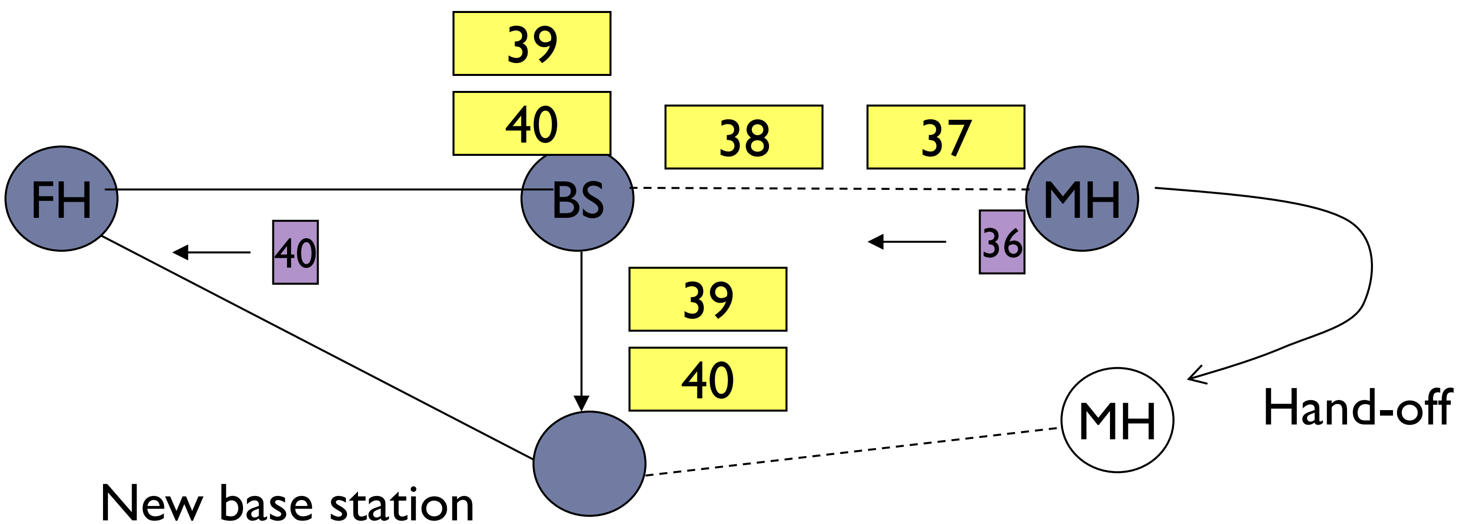
# Split Connection Approach : Disadvantages

- BS retains hard state
  - BS failure can result in loss of data (unreliability)
    - If BS fails, packet 40 will be lost
    - Because it is ack'd to sender, the sender does not buffer 40



# Split Connection Approach : Disadvantages

- BS retains hard state
  - Hand-off latency increases due to state transfer
    - Data that has been ack'd to sender, must be moved to new base station

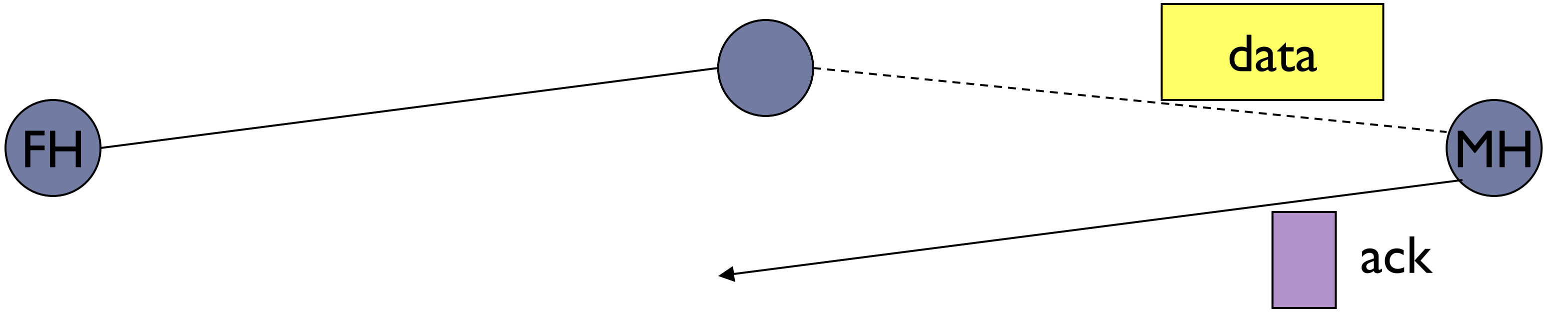


# Split Connection Approach : Disadvantages

- Buffer space needed at BS for each TCP connection
  - BS buffers tend to get full with a slow wireless link slower
    - One window of data on wired connection could be stored at base station for each split connection
- Window on BS-MH connection reduced in response to errors
  - May not be an issue for wireless links with small delay-bw product

# Split Connection Approach : Disadvantages

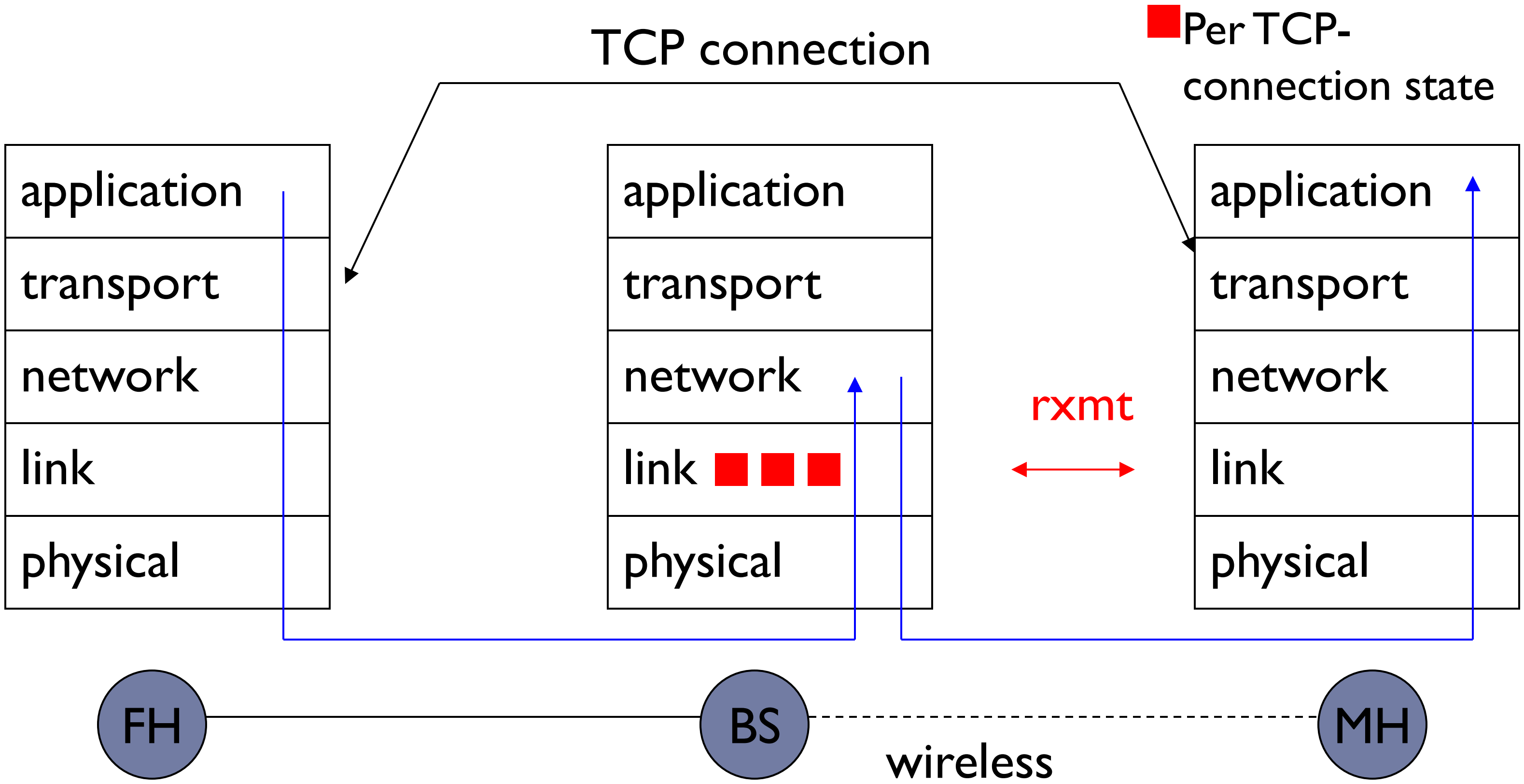
- Extra copying of data at BS
  - Copying from FH-BS socket buffer to BS-MH socket buffer
  - Increases end-to-end latency
- May not be useful if data and acks traverse different paths (both do not go through the base station)
  - Example: data on a satellite wireless hop, acks on a dial-up channel



# TCP-Aware Link Layer

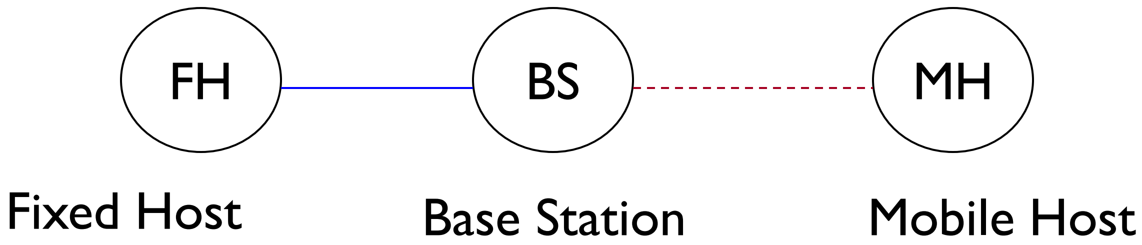
- Snoop Protocol
  - Retains local recovery of Split Connection approach and link level retransmission schemes
  - Improves on split connection
    - End-to-end semantics retained
    - Soft state at base station, instead of hard state

# Snoop Protocol

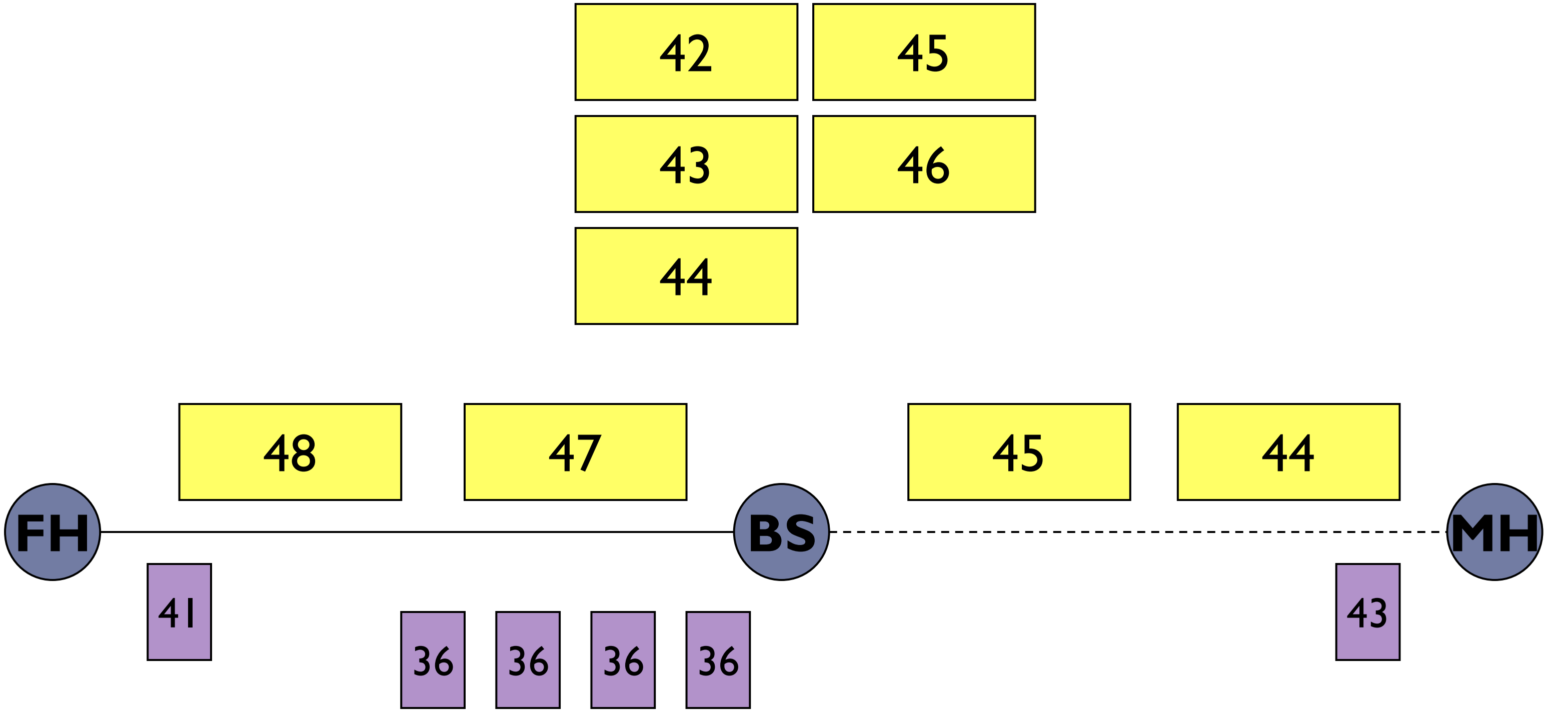


# Snoop Protocol

- 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 Protocol



# Snoop Protocol: When Beneficial?

- Snoop
  - Prevents fast retransmit despite transmission errors on the wireless link
- If wireless link level delay-bandwidth product is less than 4 packets
  - Simple (TCP-unaware) link level retransmission scheme can suffice
  - Since delay-bandwidth product is small
    - Retransmission scheme can deliver the lost packet without resulting in 3 dupacks from the TCP receiver

# Snoop Protocol: Advantages

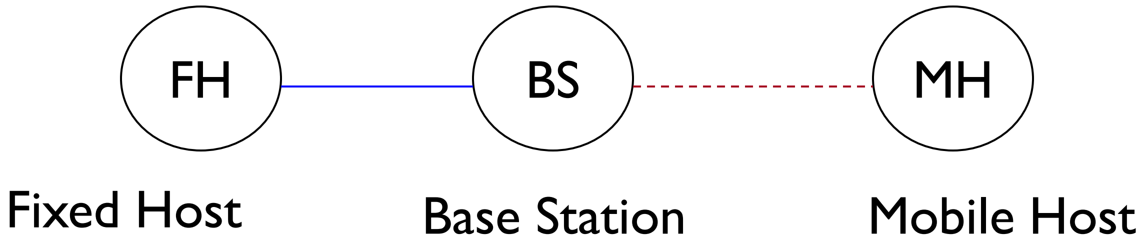
- High throughput
  - Performance further improved using selective acks
- Local recovery from wireless losses
- Fast retransmit not triggered at sender
- End-to-end semantics retained
- Soft state at base station
  - Loss of the soft state affects performance, but not correctness

# 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)

# WTCP Protocol

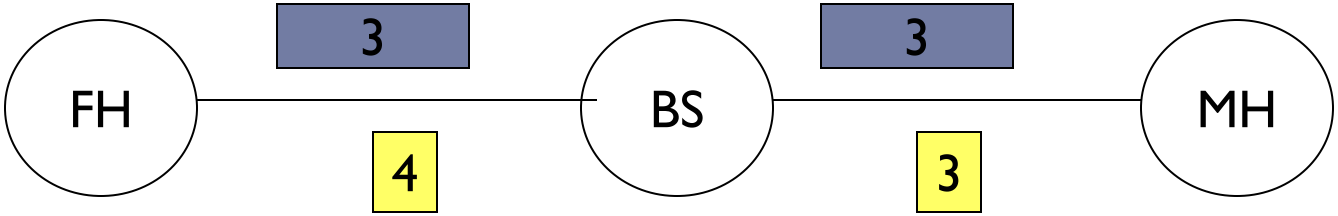
- Snoop hides wireless losses from the sender
  - But sender's RTT estimates may be larger in presence of errors
  - Larger RTO results in slower response for congestion losses



# WTCP Protocol

- Local recovery
- Timestamp option to estimate RTT
- The base station
  - Adds base station residence time to the timestamp when processing an ack received from the wireless host
- Sender's RTT estimate
  - Not affected by retransmissions on wireless link

# WTCP Protocol




Numbers in this figure are timestamps

Base station residence time is 1 unit

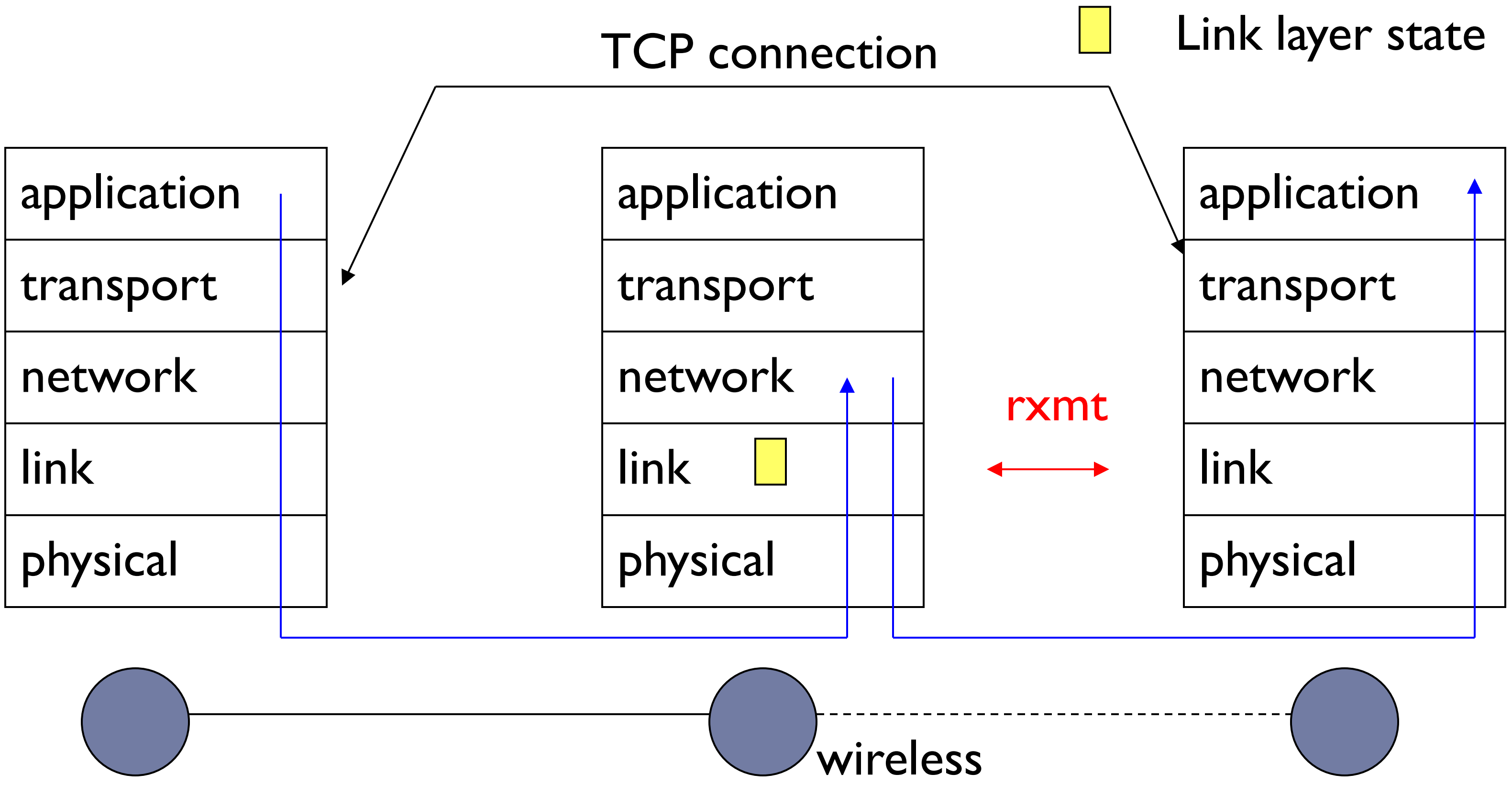
# WTCP : Disadvantages

- Requires use of the timestamp option
- May be useful only if retransmission times are large
  - Link stays in bad state for a long time
  - Link frequently enters a bad state
  - Link delay large
- WTCP does not account for congestion on wireless hop
  - Assumes that all delay at base station is due to queuing and retransmissions
  - Will not work for shared wireless LAN, where delays also incurred due to contention with other transmitters

# TCP-Unaware Approximation of TCP-Aware Link Layer

- Delayed Dupacks Protocol
  - 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 dupacks)

# Delayed Dupacks Protocol



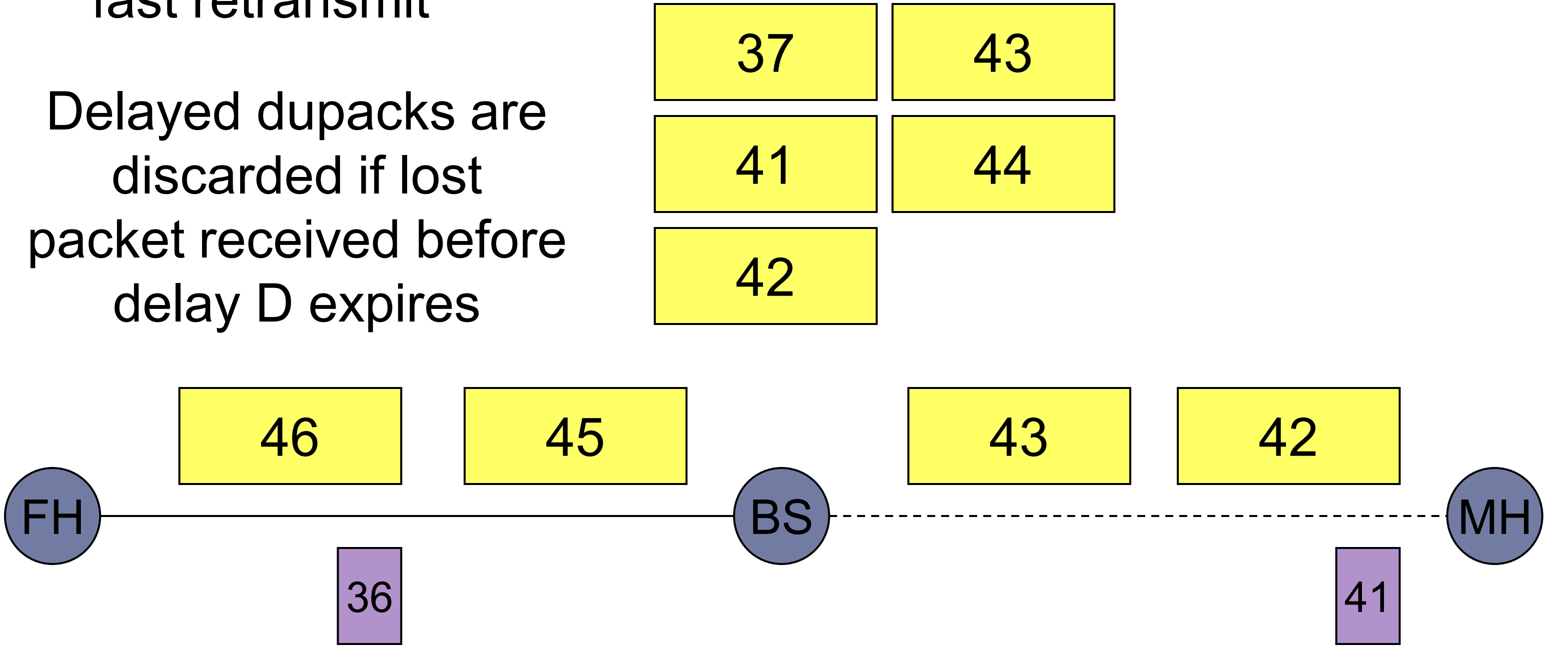
# Delayed Dupacks Protocol

- Delayed dupacks released after interval D, if missing packet not received by then
- Link layer maintains state to allow retransmission
- Link layer state is not TCP-specific

# Delayed Dupacks Protocol

TCP sender does not fast retransmit

Delayed dupacks are discarded if lost packet received before delay D expires



# Delayed Dupacks Scheme: Advantages

- Link layer need not be TCP-aware
- Can be used even if TCP headers are encrypted
- Works well for relatively small wireless RTT (compared to end-to-end RTT)
  - Relatively small delay D sufficient in such cases

# Delayed Dupacks Scheme: Disadvantages

- Right value of dupack delay D dependent on the wireless link properties
- Mechanisms to automatically choose D needed
- Delays dupacks for congestion losses too, delaying congestion loss recovery

# Explicit Notification Schemes

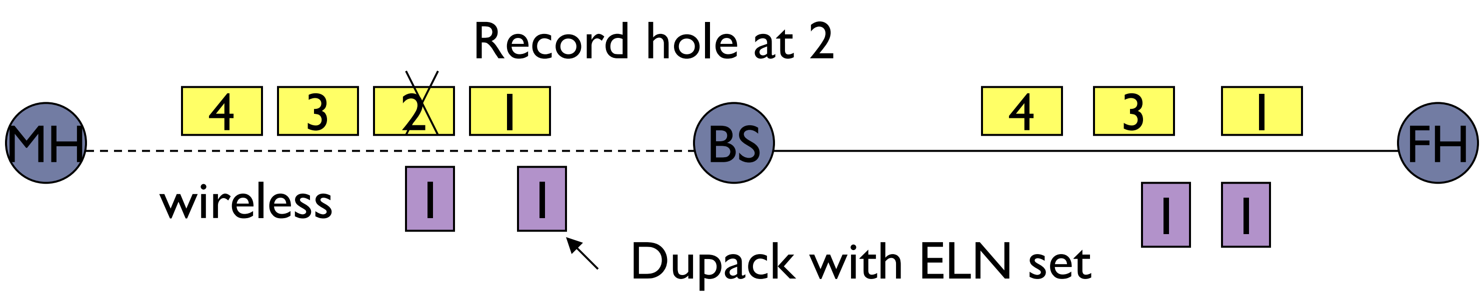
- General Philosophy
  - Approximate Ideal TCP behavior
    - TCP sender should simply retransmit a packet lost due to transmission errors
    - No congestion control actions
  - Wireless node
    - Determines that packets are lost due to errors
    - Informs sender using an explicit notification
  - Sender - on notification
    - Does not reduce congestion window
    - Retransmits lost packet

# Explicit Notification Schemes

- Motivated by the Explicit Congestion Notification (ECN) proposals
- Variations proposed in literature differ in
  - Who sends explicit notification
  - How they know to send the explicit notification
  - What the sender does on receiving the notification

# Explicit Loss Notification – MH as TCP Sender

- Wireless link first on the path from sender to receiver
- Base station
  - Keeps track of holes in the packet sequence
  - Dupack from receiver
    - Base station compares the dupack sequence number with  recorded holes
    - If there is a match, an ELN bit is set in the dupack
- Sender - Dupack with ELN set
  - Retransmit packet
  - Do not reduce congestion window



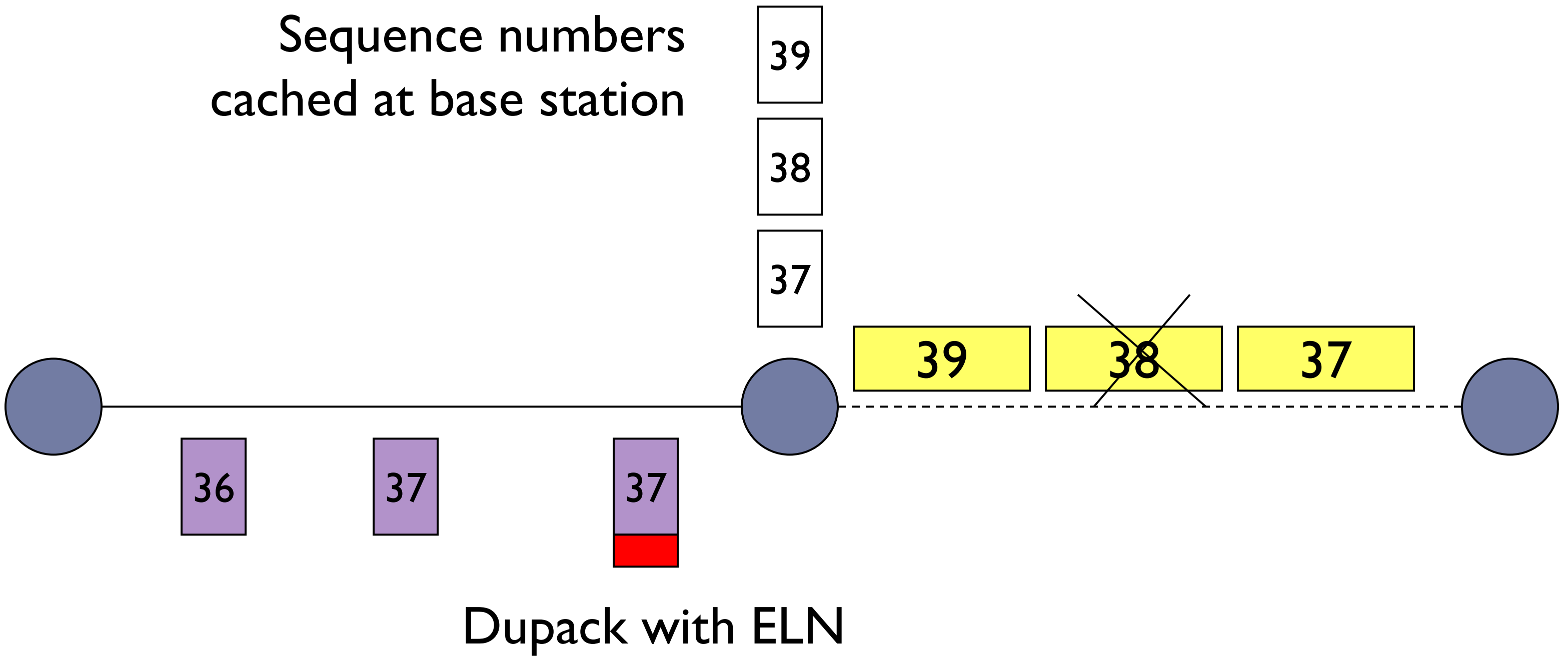
# Explicit Loss Notification – MH as TCP Sender

- Base station
  - Attempts to deliver packets to the MH using a link layer retransmission scheme
  - If packet cannot be delivered using a small number of retransmissions
    - BS sends a Explicit Bad State Notification (EBSN) message to TCP sender
- When TCP sender receives EBSN, it resets its timer
  - Timeout delayed, when wireless channel in bad state

# Explicit Loss Notification - MH as TCP receiver

- Approximate hypothetical ELN
- Base station
  - Caches TCP sequence numbers
  - Does not cache data packets
- If sequence number for lost packet is cached at the base station
  - Duplicate acks are tagged with ELN bit before being forwarded to sender
- Sender takes appropriate action on receiving ELN

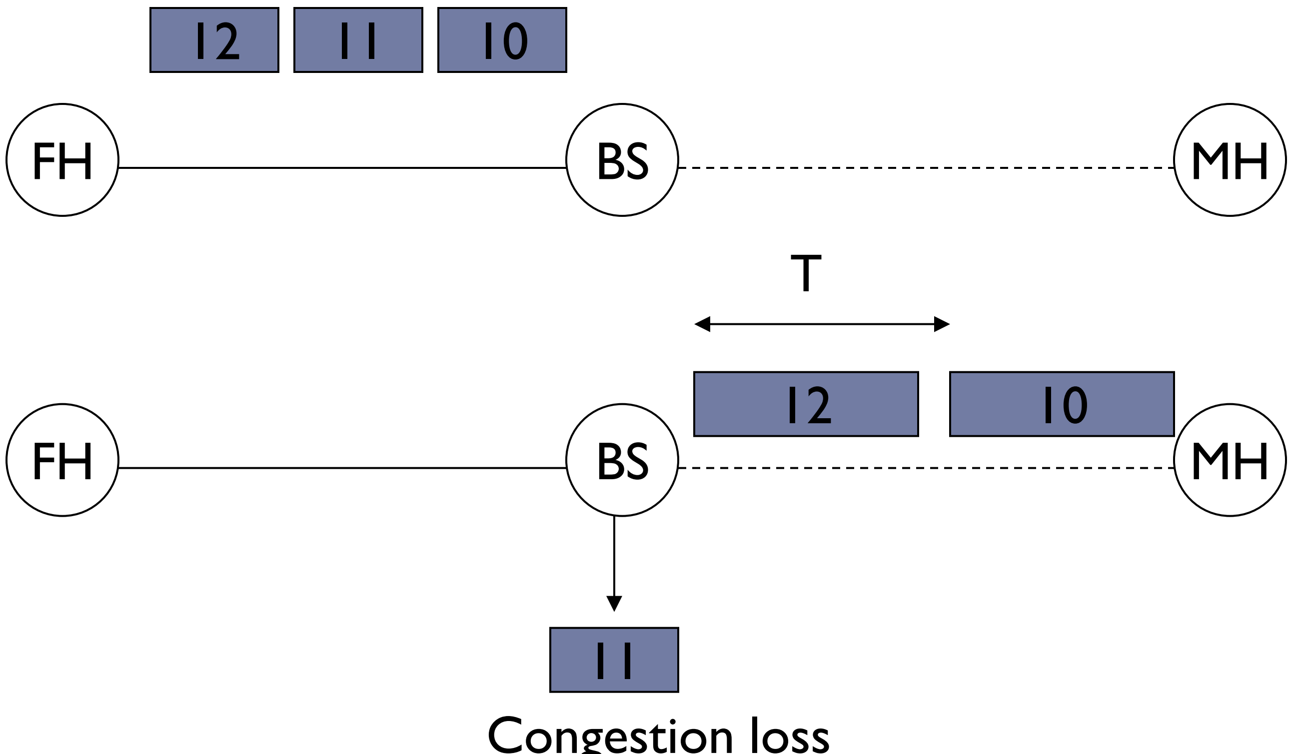
# Explicit Loss Notification - MH as TCP receiver



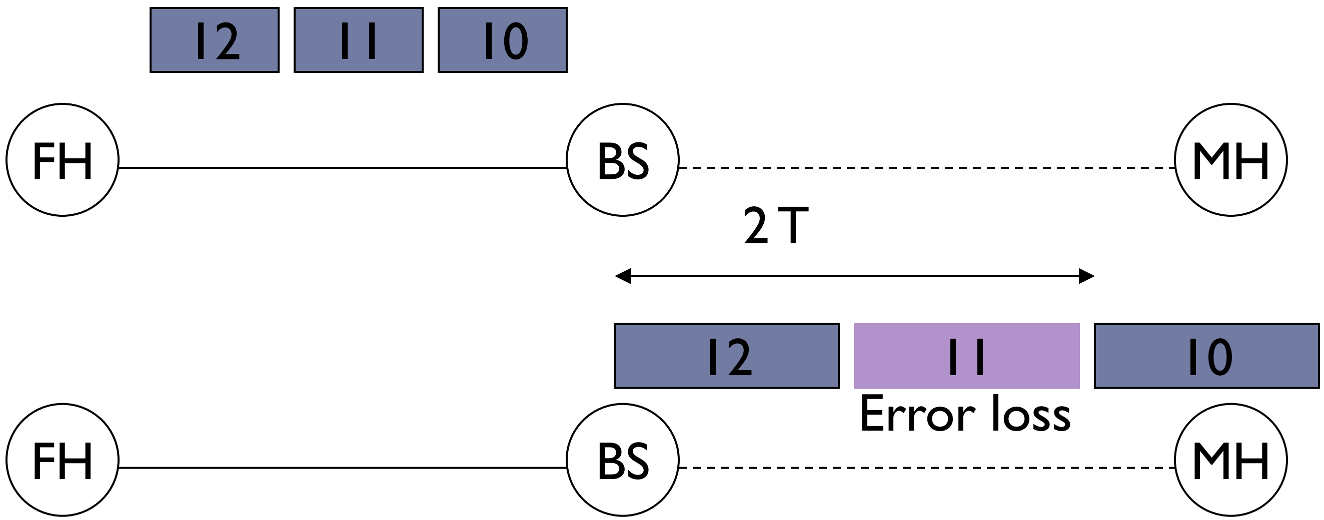
# Receiver-Based Discrimination Scheme

- MH is TCP receiver
  - Use heuristics to guess cause of packet loss
  - If packet loss is "due" to errors
    - Send a notification to the TCP sender
- TCP sender - on notification
  - Retransmit lost packet
  - Do not reduce congestion window

# Receiver-Based Scheme



# Receiver-Based Scheme



# Receiver-Based Scheme

- Receiver uses the inter-arrival time between consecutively received packets to guess the cause of a packet loss

- On determining a packet loss as being due to errors, the receiver may
  - Tag corresponding dupacks with an ELN bit, or
  - Send an explicit notification to sender

# Receiver-Based Scheme: Disadvantages

- Limited applicability
- The slowest link on the path must be the last wireless hop
  - To ensure some queuing will occur at the base station
- The queueing delays for all packets (at the base station) should be somewhat uniform
  - Multiple connections on the link will make inter-packet delays variable

# Receiver-Based Scheme: Advantages

- Can be implemented without modifying the base station (an "end-to-end" scheme)
- May be used despite encryption, or if data & acks traverse different paths

# Sender-Based Discrimination Scheme

- Sender can attempt to determine cause of a packet loss
- If packet loss determined to be due to errors, do not reduce congestion window
- Sender can only use statistics based on round-trip times, window sizes, and loss pattern
  - Unless network provides more information (example: explicit loss notification)

# Sender-Based Heuristics: Disadvantage

- Does not work quite well enough as yet !!
- Reason
  - Statistics collected by the sender garbled by other traffic on the network
  - Not much correlation between observed short-term statistics, and onset of congestion

# Sender-Based Heuristics: Advantages

- Only sender needs to be modified
- Needs further investigation to develop better heuristics
  - Investigate longer-term heuristics

# TCP in Presence of Transmission Errors: Summary

- Many techniques have been proposed, and several approaches perform well in many environments
- Recommendation: Prefer end-to-end techniques
  - End-to-end techniques are those which do not require TCP-Specific help from lower layers
  - Lower layers may help improve TCP performance without taking TCP-specific actions.
    - Examples:
      - Semi-reliable link level retransmission schemes
      - Explicit notification