

# Explicit Notification

# Explicit Notification Schemes

## General Philosophy

- Approximate **Ideal TCP behavior**: Ideally, the TCP sender should simply retransmit a packet lost due to transmission errors, **without** taking any congestion control actions
- A wireless node somehow determines that packets are lost due to errors and informs the sender using an explicit notification
- Sender, on receiving the notification, does **not reduce congestion window**, but **retransmits** lost packet

# Explicit Notification Schemes

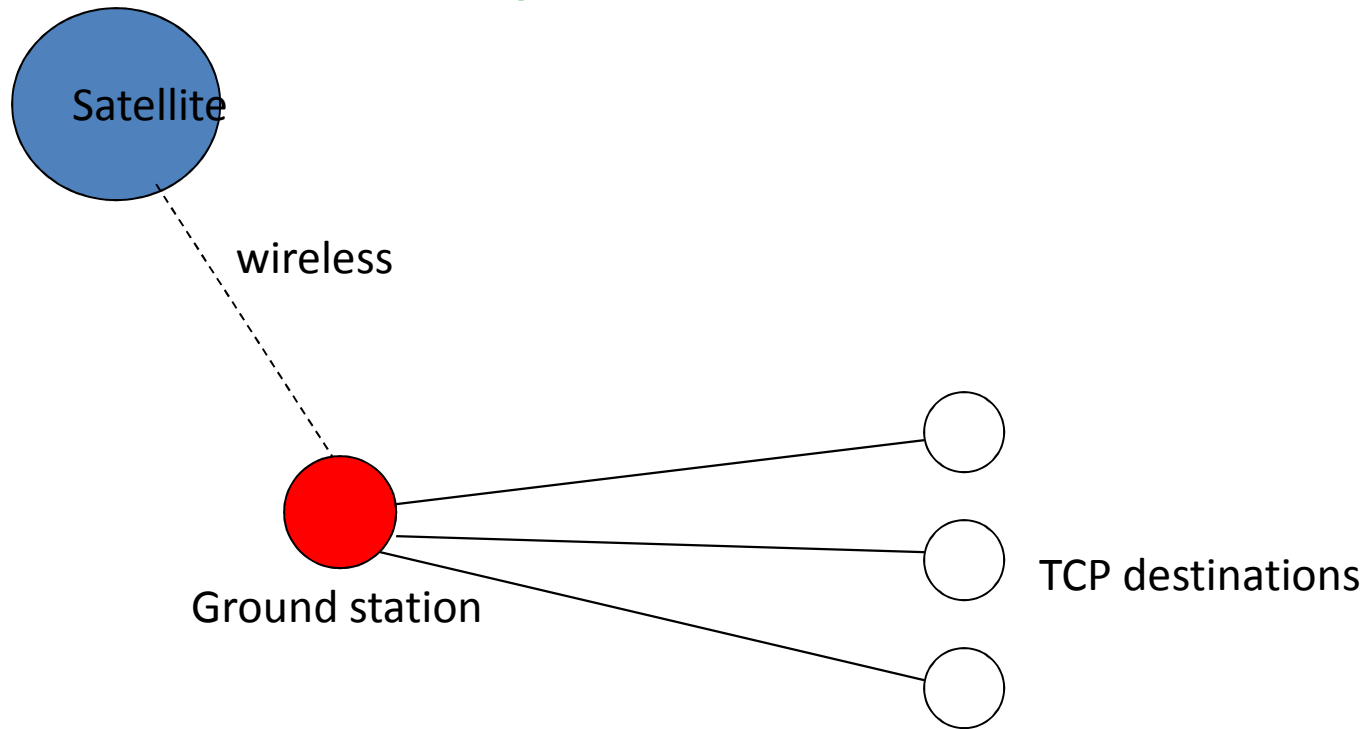
- Motivated by the **Explicit Congestion Notification (ECN)** proposals [[Floyd94](#)]

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 Notification

## Space Communication Protocol Standards-Transport Protocol (SCPS-TP)



# Space Communication Protocol Standards-Transport Protocol (SCPS-TP)

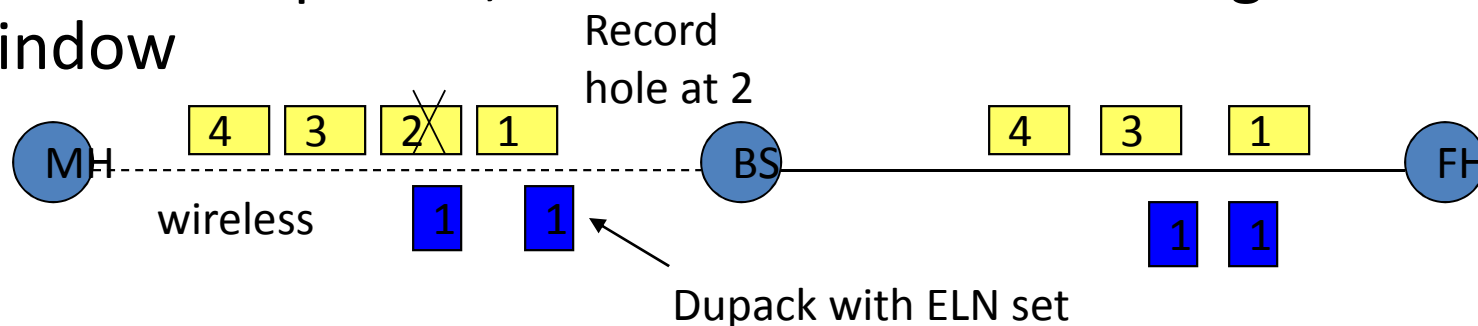
- The receiving ground station keeps track of how many packets with errors are received (their checksums failed)
- When the error rate exceeds a threshold, the ground station sends **corruption experienced** messages to destinations of recent error-free TCP packets
  - destinations are cached
- The TCP destinations tag acks with corruption-experienced bit
- TCP sender, after receiving an ack with corruption-experienced bit, does not back off until it receives an ack without that bit (even if timeout or fast retransmit occurs)

# Explicit Loss Notification

[Balakrishnan98]

when MH is the TCP sender

- Wireless link first on the path from sender to receiver
- The base station keeps track of **holes** in the packet sequence received from the sender
- When a dupack is received from the receiver, the base station compares the dupack sequence number with the recorded holes
  - if there is a match, an ELN bit is set in the dupack
- When sender receives dupack with ELN set, it retransmits packet, but does not reduce congestion window



# Explicit Bad State Notification

[Bakshi97]

when MH is TCP receiver

- 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

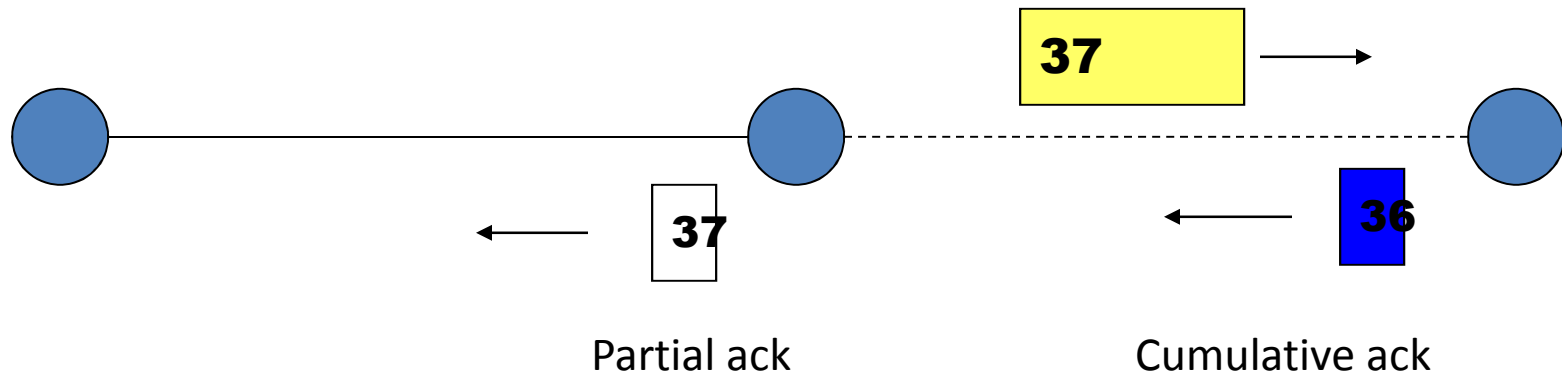
# Partial Ack Protocols

[Cobb95][Biaz97]

- Send two types of acknowledgements
- A partial acknowledgement informs the sender that a packet was received by an intermediate host (typically, base station)
- Normal TCP cumulative ack needed by the sender for reliability purposes

# Partial Ack Protocols

- When a packet for which a **partial ack is received** is detected to be lost, the sender does not reduce its congestion window
  - loss assumed to be due to wireless errors



# Various Schemes

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

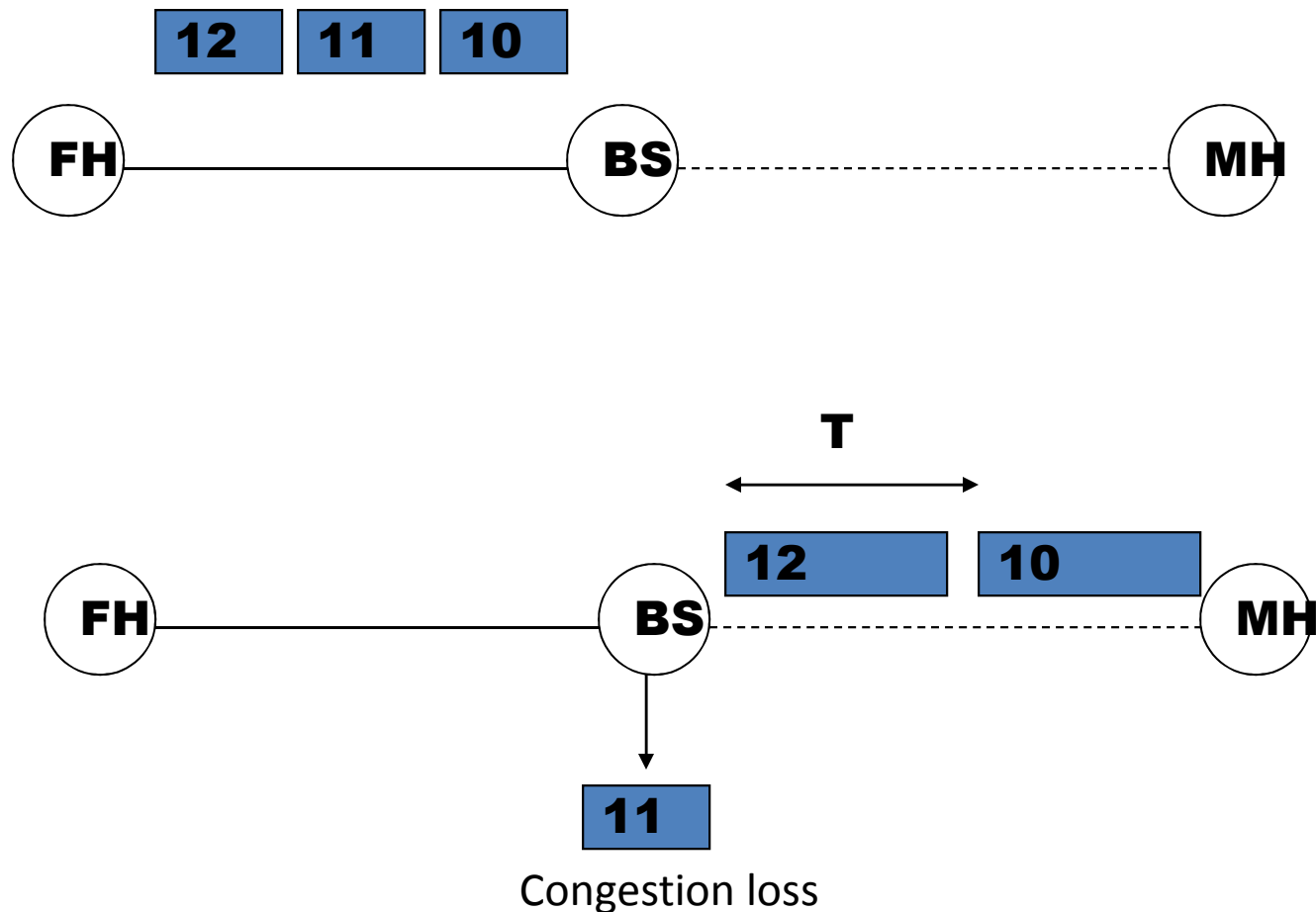
# Receiver-Based Discrimination Scheme

# Receiver-Based Scheme [Biaz98Asset]

- MH is TCP receiver
- Receiver uses a heuristic 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, but does not reduce congestion window

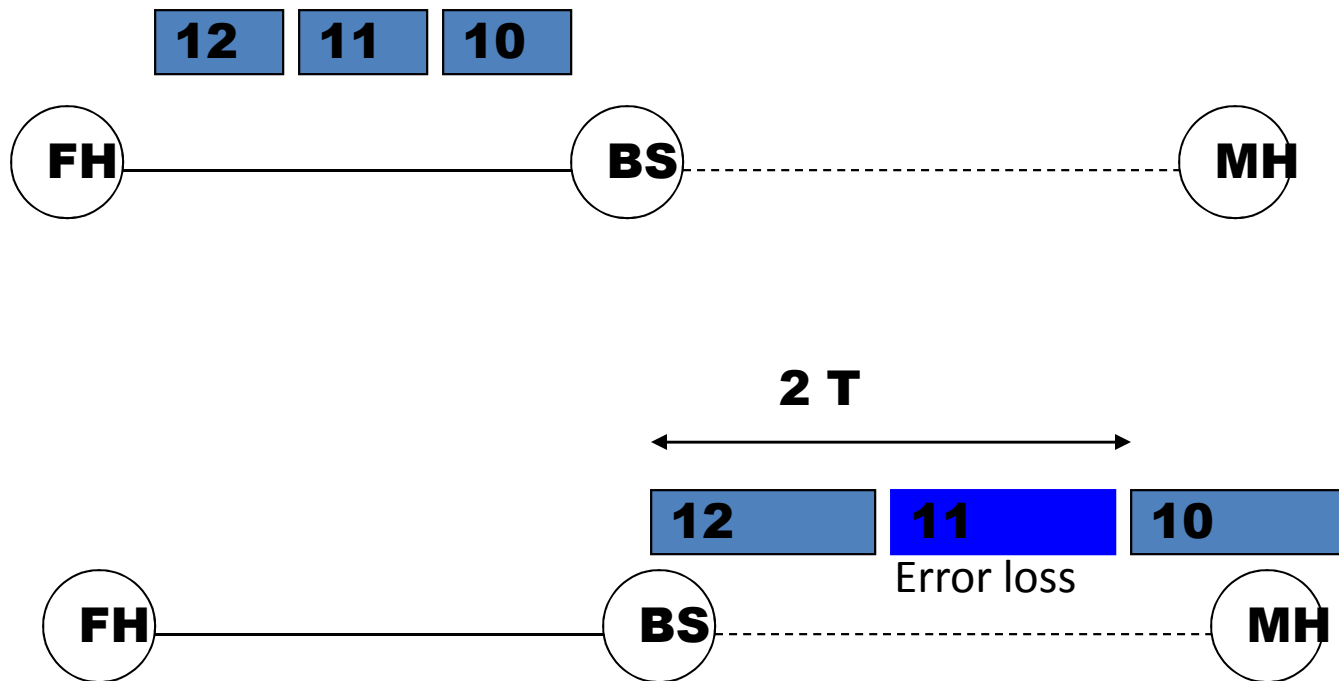
# Receiver-Based Scheme

- Packet loss due to congestion



# Receiver-Based Scheme

- Packet loss due to transmission error



# 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

# Various Schemes

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

# Sender-Based Discrimination Scheme

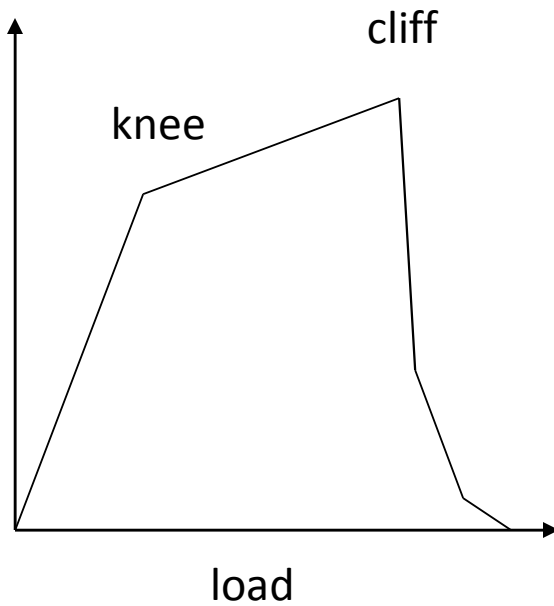
# Sender-Based Discrimination Scheme

## [Biaz98ic3n,Biaz99techrep]

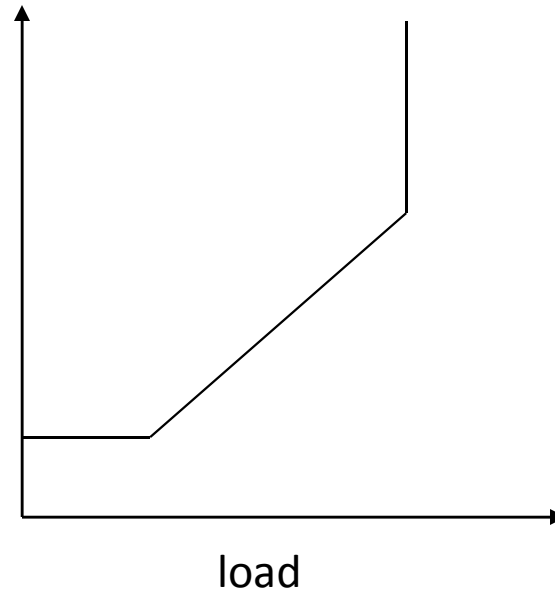
- Sender can attempt to determine cause of a packet loss
- If packet loss determined to be due to errors, do not reduce congestion window
- Sender can only use statistics based on round-trip times, window sizes, and loss pattern
  - unless network provides more information (example: explicit loss notification)

# Heuristics for Congestion Avoidance

throughput



RTT



# Heuristics for Congestion Avoidance

- Define condition C as a function of congestion window size and observed RTTs
- Condition C evaluated when a new RTT is calculated
  - condition C typically evaluates to 2 or 3 possible values
  - for now assume 2 values: TRUE or FALSE
- If (C == True) reduce congestion window
- Several proposals for condition C

# Heuristics for Congestion Avoidance

## Some proposals

- Normalized Delay Gradient [jain89]

$$r = [\text{RTT}(i) - \text{RTT}(i-1)] / [\text{RTT}(i) + \text{RTT}(i-1)]$$

$$w = [W(i) - W(i-1)] / [W(i) + W(i-1)]$$

$$\text{Condition } C = (r/w > 0)$$

# Heuristics for Congestion Avoidance

## Some proposals

- Normalized Throughput Gradient [Wang91]

Throughput gradient

$$TG(i) = [T(i) - T(i-1)] / [W(i) - W(i-1)]$$

Normalized Throughout Gradient

$$NTG = TG(i) / TG(1)$$

Condition C = (NTG < 0.5)

# Heuristics for Congestion Avoidance

## Some proposals

- TCP Vegas [Brakmo94]

expected throughput  $ET = W(i) / RTT_{min}$

actual throughput  $AT = W(i) / RTT(i)$

Condition C = (  $ET - AT > \beta$  )

# Sender-Based Heuristics

- Record latest value evaluated for condition C
- When a packet loss is detected
  - if last evaluation of C is TRUE, assume packet loss is due to **congestion**
  - else assume that packet loss is due to **transmission errors**
- If packet loss determined to be due to errors, do not reduce congestion window

# 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

# Statistical Techniques

## Future Work

- Other statistical measures ?
- Mechanisms that achieve good TCP throughput despite not-too-good diagnostic accuracy

# 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

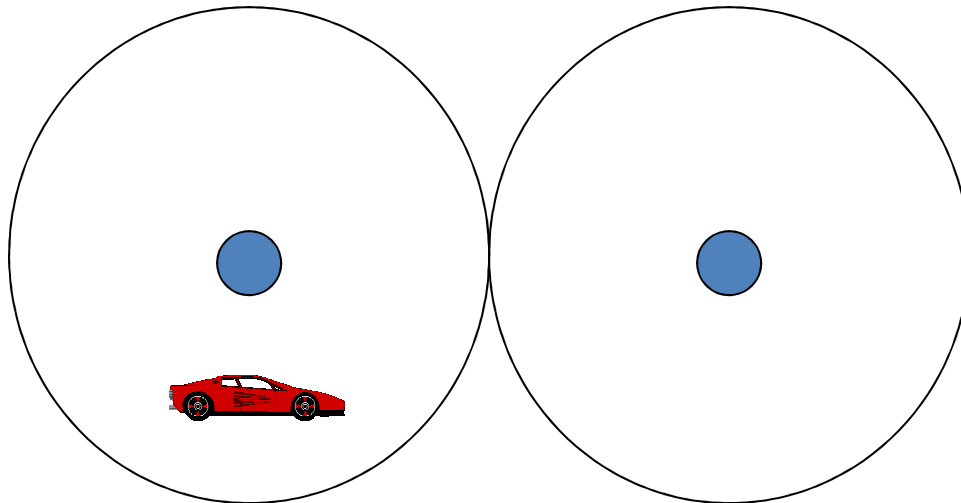
# Outline

- Schemes to improve TCP performance in presence of transmission errors
- **Impact of mobility on TCP performance**
- Approaches to improve TCP performance in presence of mobility
- Issues in multi-hop wireless networks
- Issues needing further work
- References

# Impact of Mobility on TCP Performance

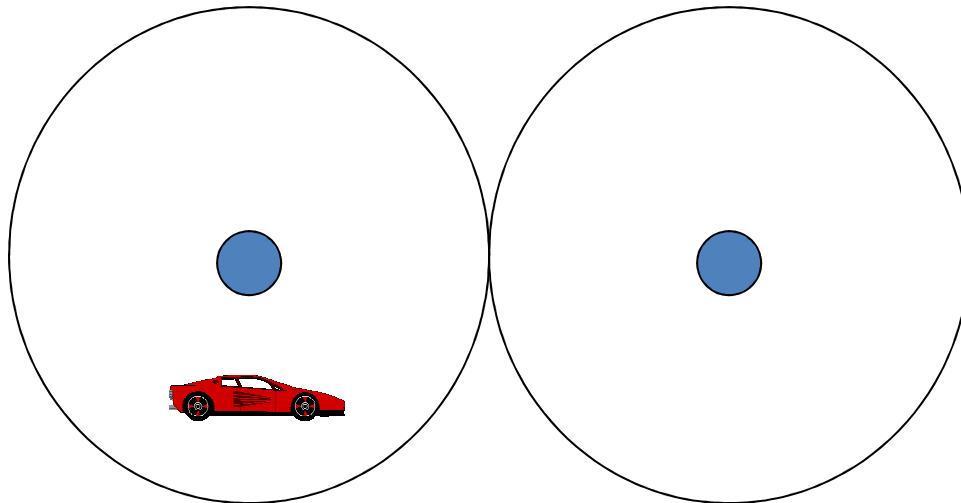
# Impact of Mobility

- Hand-offs occur when a mobile host starts communicating with a new base station (in cellular wireless systems)



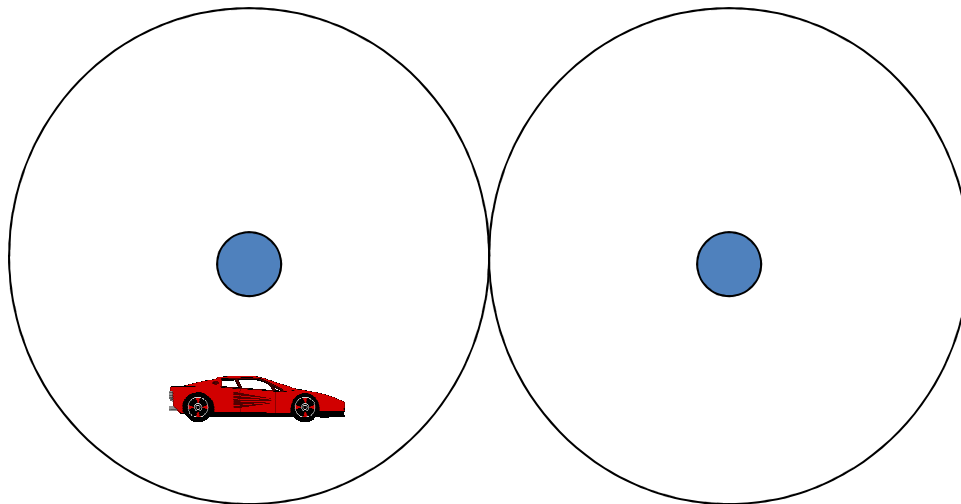
# Impact of Mobility

- If link layer performs hand-offs and guarantees reliability despite handoff, then TCP will not be aware of the handoff
  - except for potential delays during handoff

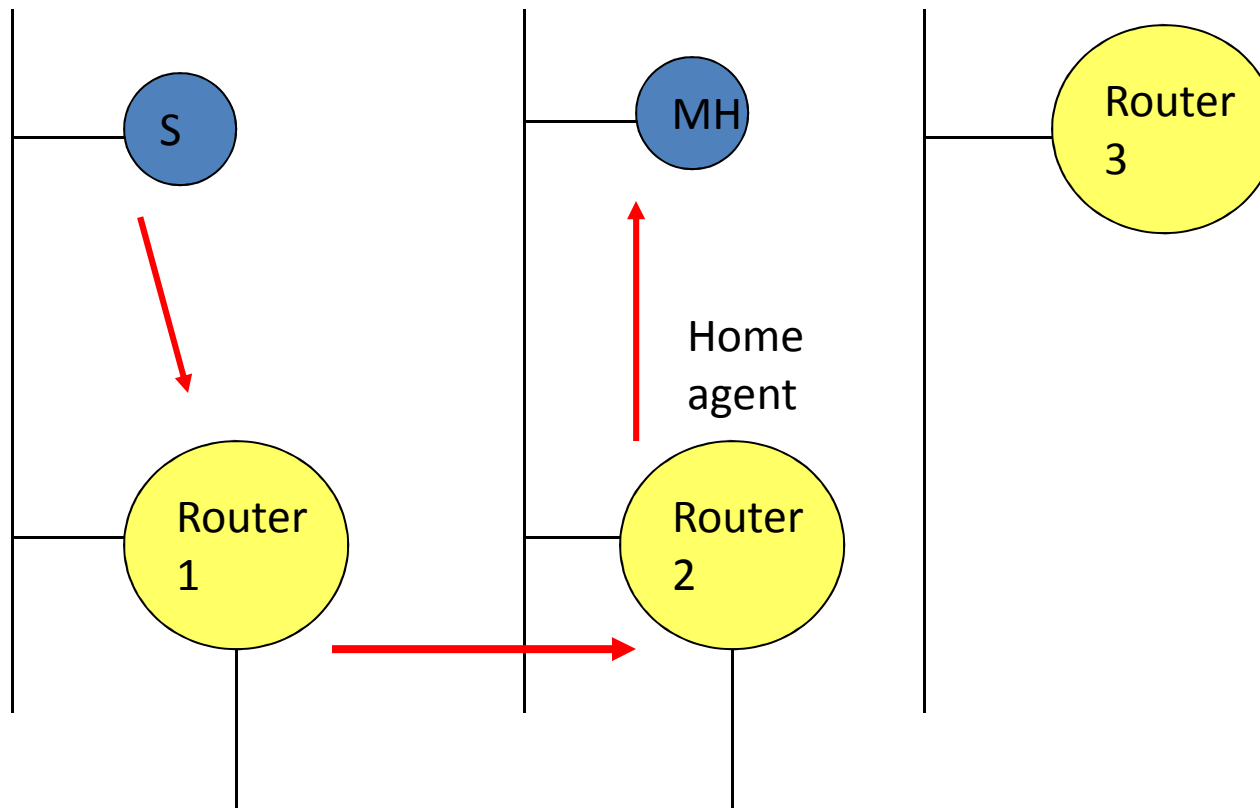


# Impact of Mobility

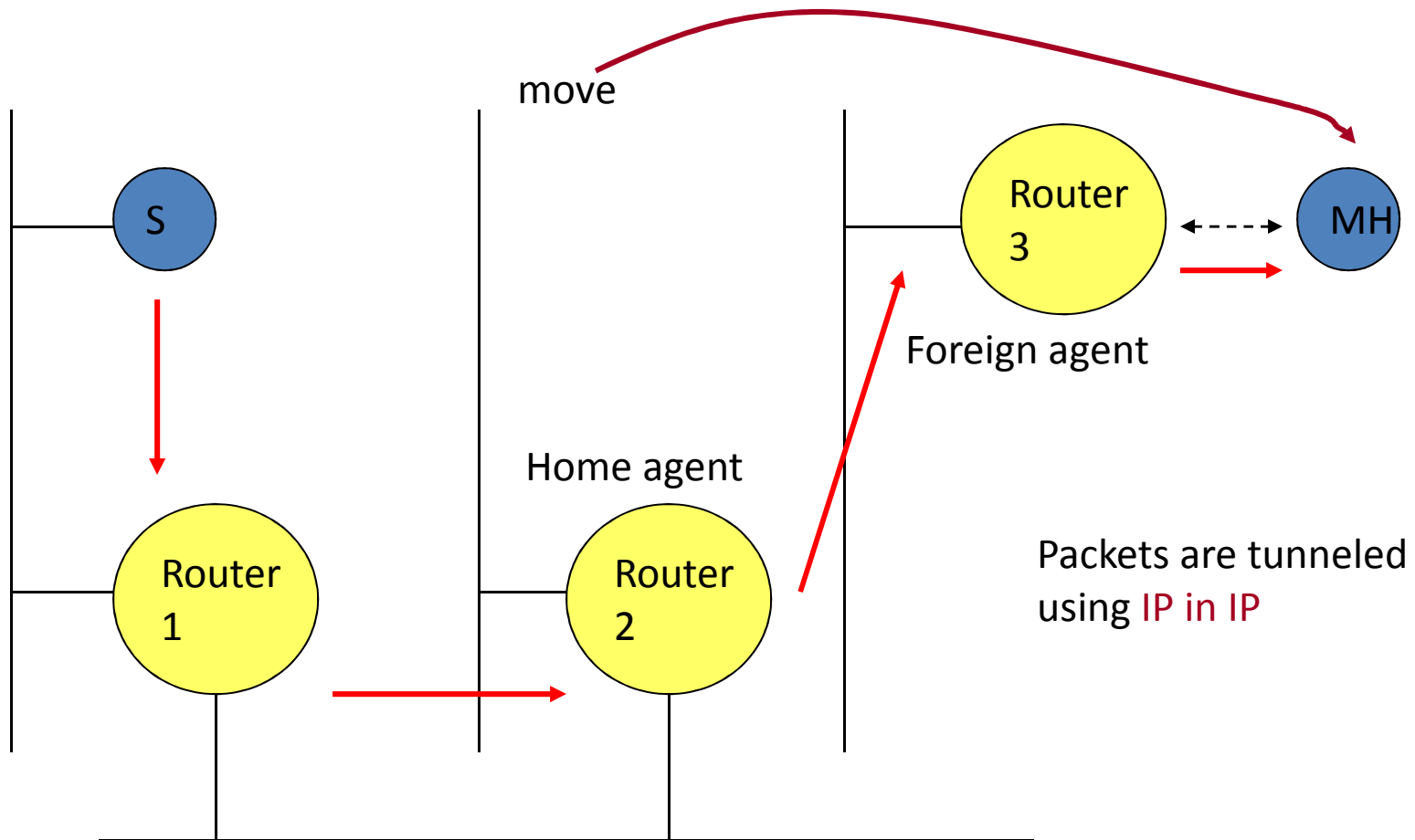
- If hand-off visible to IP
  - Need Mobile IP [[Johnson96](#)]
  - packets may be lost while a new route is being established reliability despite handoff
- We consider this case



# Mobile IP [Johnson96]

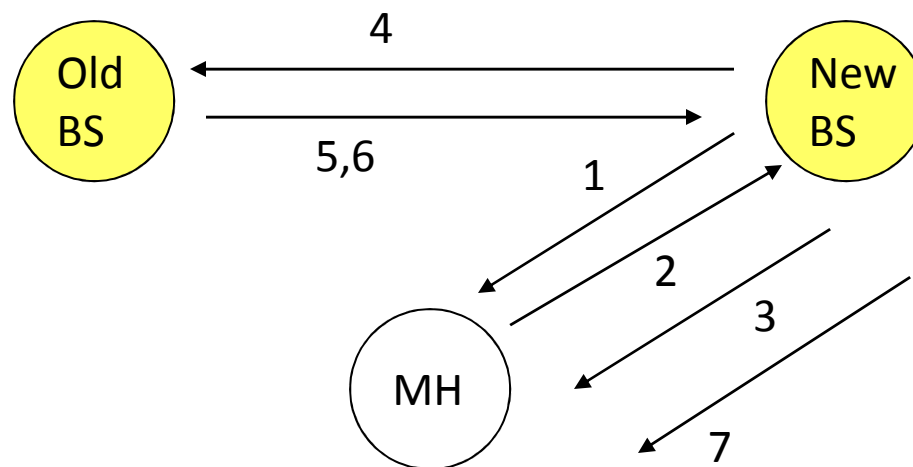


# Mobile IP [Johnson96]



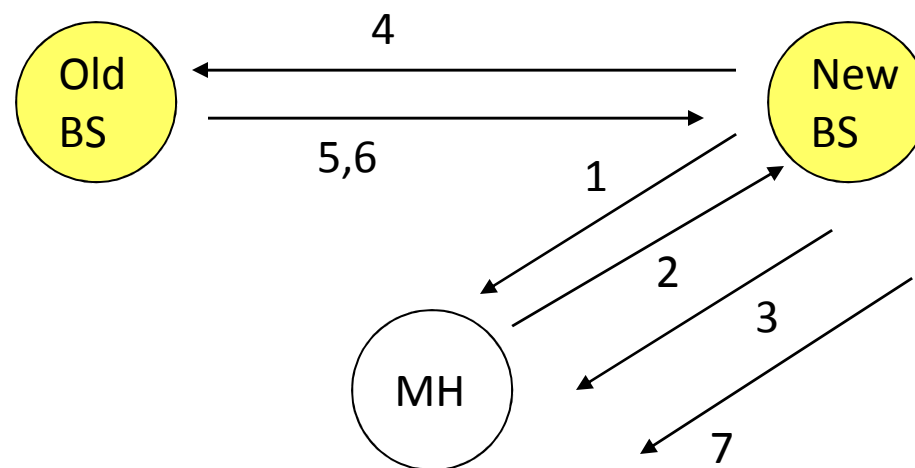
# Example Hand-Off Procedure

1. Each base station periodically transmits beacon
2. Mobile host, on hearing stronger beacon from a new BS, sends it a greeting
  - changes routing tables to make new BS its default gateway
  - sends new BS identity of the old BS



# Hand-Off Procedure

3. New BS acknowledges the greeting, and begins to route the MH's packets
4. New BS informs old BS
5. Old BS changes routing table, to forward any packets for the MH to the new BS
6. Old BS sends an ack to new BS
7. New BS sends handoff-completion message to MH



# Mobile IP

- Mobile IP would need to modify the previous hand-off procedure to inform the home agent the identity of the new foreign agent
- Triangular optimization can reduce the routing delay
  - Route directly to foreign agent, instead of via home agent

# Hand-off

- Hand-offs may result in temporary **loss of route** to MH
  - with non-overlapping cells, it may be a while before the mobile host receives a beacon from the new BS
- While routes are being reestablished during handoff, MH and old BS may attempt to send packets to each other, resulting in **loss of packets**

# Impact of Handoffs on Schemes to Improves Performance in Presence of Errors

- Split connection approach
  - hard state at base station must be moved to new base station
- Snoop protocol
  - soft state need not be moved
  - while the new base station builds new state, packet losses may not be recovered locally
- Frequent handoffs a problem for schemes that rely on significant amount of hard/soft state at base stations
  - hard state should not be lost
  - soft state needs to be recreated to benefit performance

Techniques to  
Improve TCP Performance  
in Presence of **Mobility**

# Classification

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

# Using Fast Retransmits to Recover from Timeouts during Handoff

## [Caceres95]

- During the long delay for a handoff to complete, a whole window worth of data may be lost
- After handoff is complete, acks are not received by the TCP sender
- Sender eventually times out, and retransmits
- If handoff still not complete, another timeout will occur
- Performance penalty
  - Time wasted until timeout occurs
  - Window shrunk after timeout

# 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

# 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]

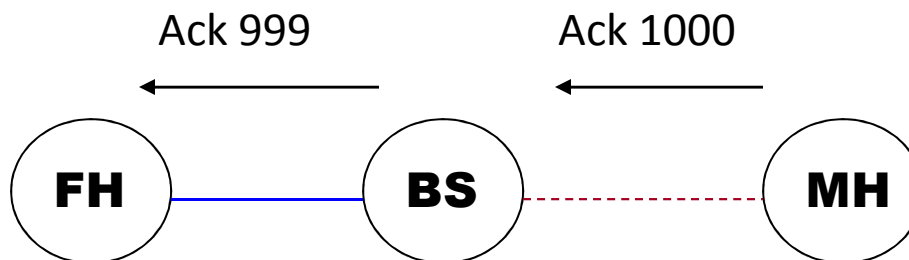
- In the fast retransmit scheme [Caceres95]
  - sender starts transmitting soon after handoff
  - BUT congestion window shrinks
- M-TCP attempts to avoid shrinkage in the congestion window

# 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

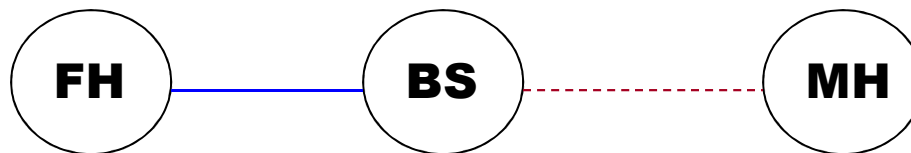
# M-TCP

- 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



# 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

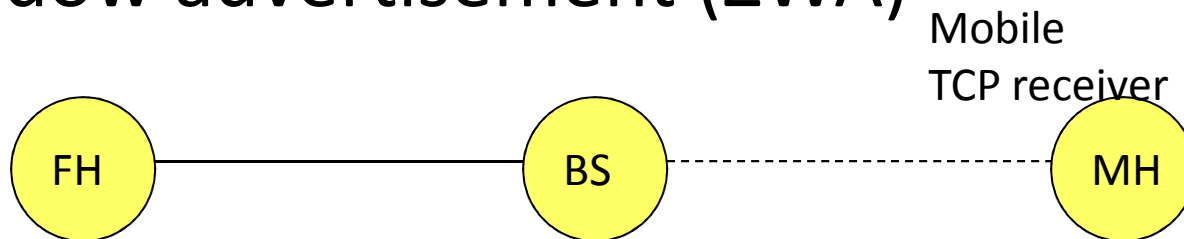
- Avoids reduction of congestion window due to handoff, unlike the fast retransmit scheme
  - simulation-based performance results look good
- **Important Question** unanswered : Is not reducing the window a good idea?

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

It is not obvious that starting full speed after handoff is right.

# FreezeTCP [Goff99]

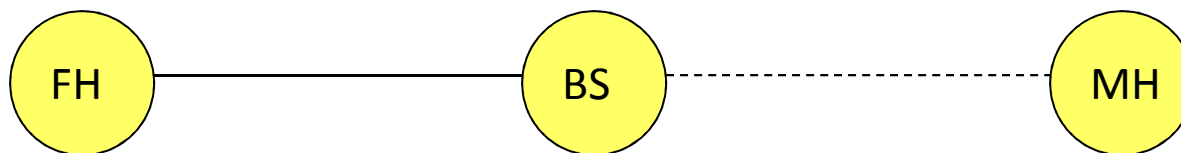
- M-TCP needs help from base station
  - Base station withholds ack for one byte
  - The base station uses this ack to send a zero window advertisement when a mobile host moves to another cell
- **FreezeTCP** requires the receiver to send zero window advertisement (ZWA)



# FreezeTCP [Goff99]

- TCP receiver determines if a handoff is about to happen
  - determination may be based on signal strength
- Ideally, receiver should attempt to send ZWA 1 RTT before handoff
- Receiver sends 3 dupacks when route is reestablished
- No help needed from the base station
  - an end-to-end enhancement

Mobile  
TCP receiver



# Using Multicast to Improve Handoffs

## [Ghai94,Seshan96]

- Define a group of base stations including
  - current cell of a mobile host
  - cells that the mobile host is likely to visit next
- Address packets destined to the mobile host to the group
- Only one base station transmits the packets to the mobile host
  - if rest of them buffer the packets, then packet loss minimized on handoff

# Using Multicast to Improve Handoffs

- Static group definition [[Ghai94](#)]
  - groups can be defined taking physical topology into account
  - static definition may not take individual user mobility pattern into account
- Dynamic group definition [[Seshan96](#)]
  - implemented using IP multicast groups
  - each user's group can be different
  - overhead of updating the multicast groups is a concern with a large scale deployment

# Using Multicast to Improve Handoffs

- Buffering at multiple base stations incurs memory overhead
- Trade-off between buffering overhead and packet loss
- Buffer requirement can be reduced by starting buffering only when a mobile host is likely to leave current cell soon

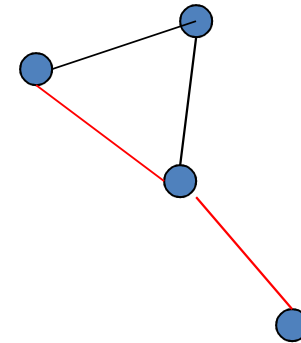
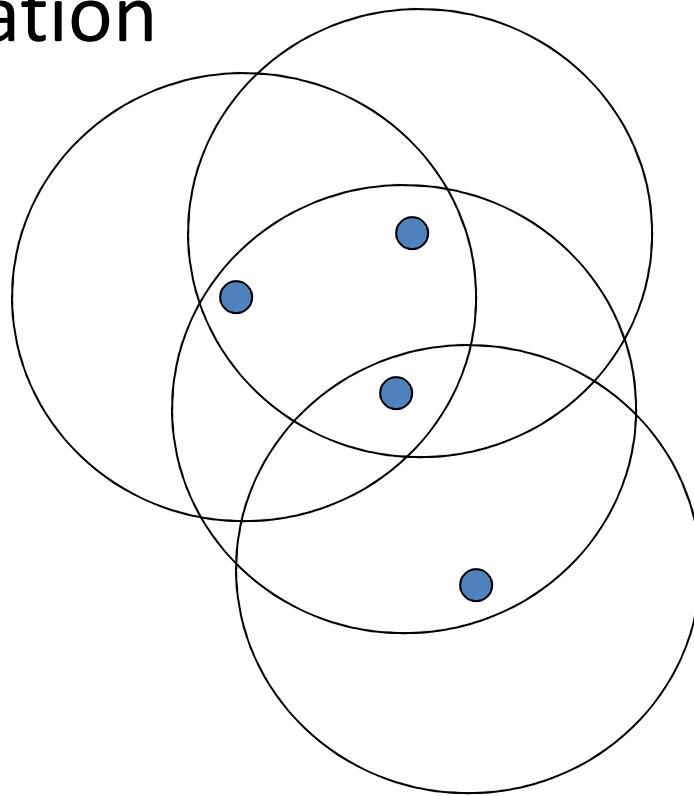
# Outline

- Impact of mobility on TCP performance
- Approaches to improve TCP performance in presence of mobility
- Issues in multi-hop wireless networks
- Issues needing further work
- References

# TCP in Mobile Ad Hoc Networks

# Mobile Ad Hoc Networks (MANET)

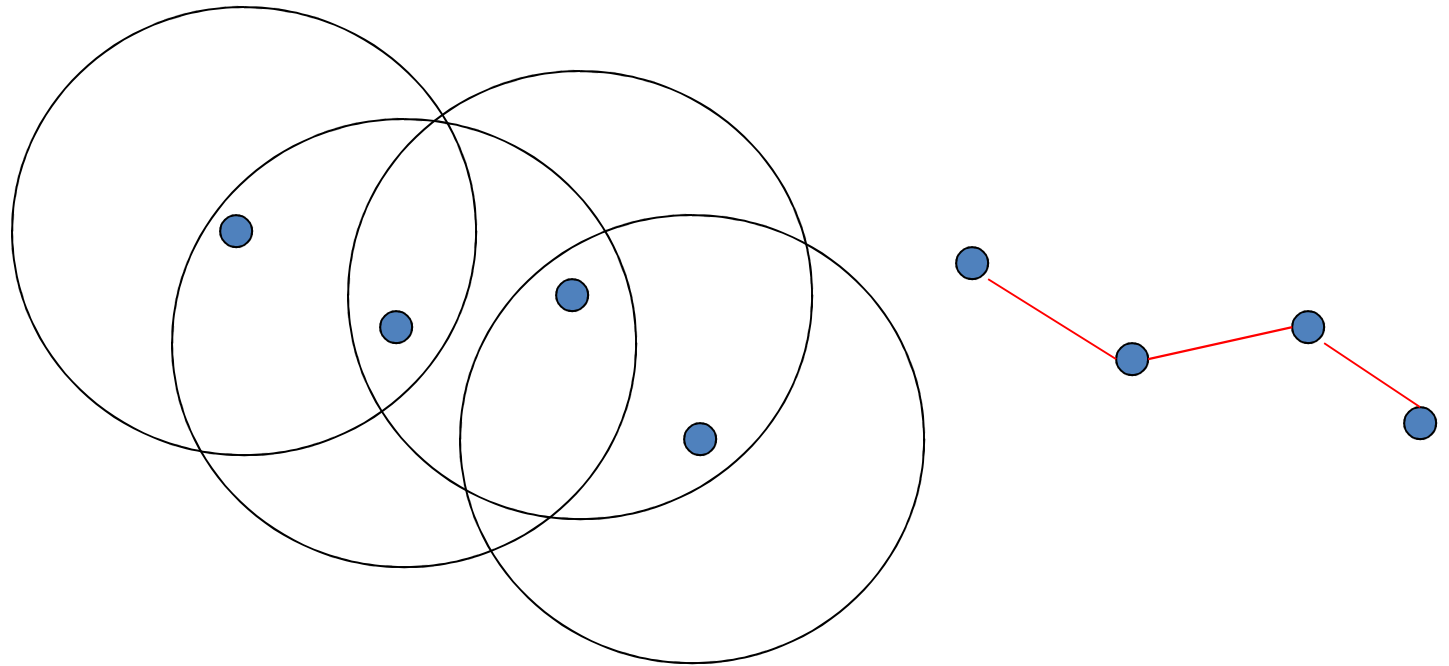
- May need to traverse multiple links to reach a destination



# Mobile Ad Hoc Networks

## [IETF-MANET]

- Mobility causes route changes



# TCP in Mobile Ad Hoc Networks

## Issues

- Route changes due to mobility
- Wireless transmission errors
  - problem compounded with multiple hops
- Out-of-order packet delivery
  - frequent route changes may cause out-of-order delivery
  - TCP does not perform well if packets are **significantly** OOO
- Multiple access protocol
  - choice of MAC protocol can impact TCP performance significantly
- Half-duplex radios
  - cannot send and receive packets simultaneously
  - changing mode (send or receive) incurs overhead

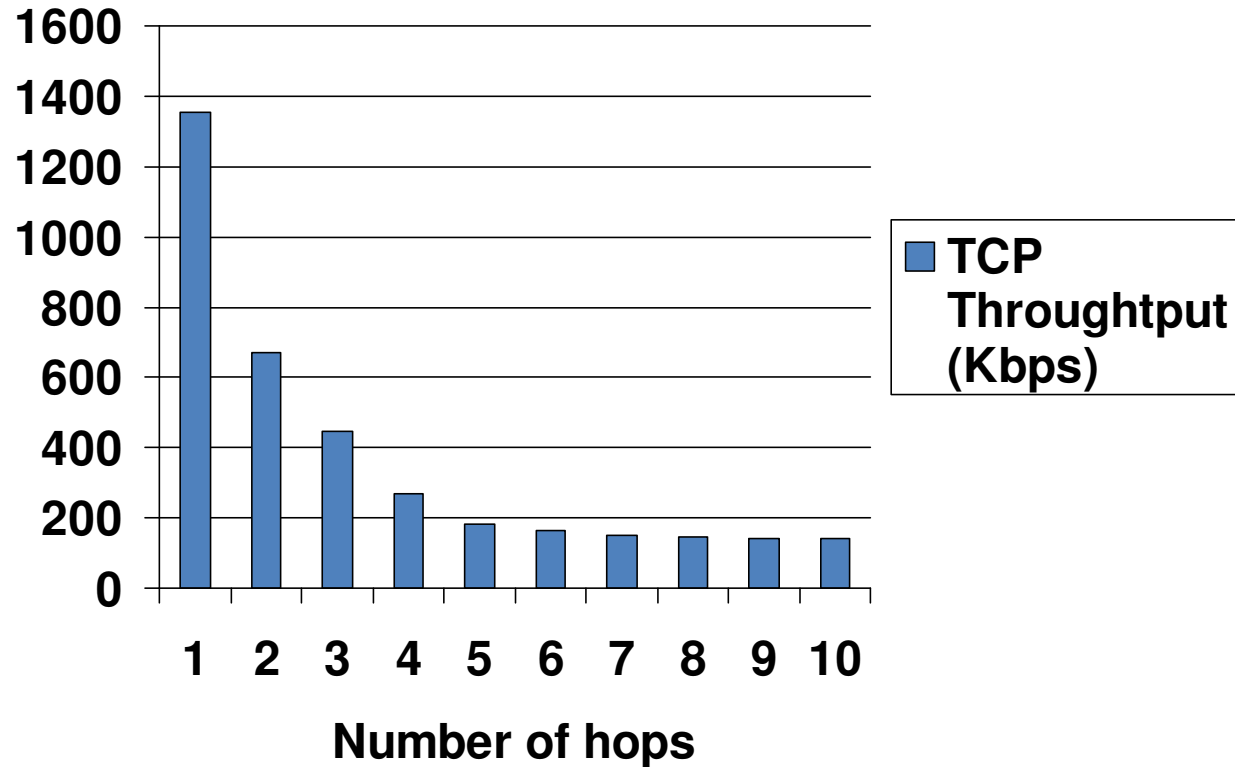
# Throughput over Multi-Hop Wireless Paths

## [Gerla99]

- When contention-based MAC protocol is used, connections over multiple hops are at a disadvantage compared to shorter connections, because they have to contend for wireless access at each hop
  - extent of packet delay or drop increases with number of hops

# Impact of Multi-Hop Wireless Paths

## [Holland99]



TCP Throughput using 2 Mbps 802.11 MAC

# Impact of mobility to TCP

- Speed?
  - Higher speed --- route breaks often
  - Higher speed --- route is also repaired sooner.
- Path length?

# Why Does Throughput Improve?

## General Principle

- TCP timeout interval somewhat (not entirely) independent of speed
- Network state at higher speed, when timeout occurs, may be more favorable than at lower speed
- Network state
  - Link/route status
  - Route caches
  - Congestion

# How to Improve Throughput (Bring Closer to Ideal)

- 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

# Issues

## Network Feedback

- Network knows best (why packets are lost)
- + Network feedback beneficial
- Need to modify transport & network layer to receive/send feedback
- Need mechanisms for information exchange between layers

# Impact of 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

# Why Performance may 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



- 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

# Issues

## Window Size After Route Repair

- Same as before route break: may be too **optimistic**
- Same as startup: may be too **conservative**
- **Better be conservative** than overly optimistic
  - Reset window to small value after route repair

# Issues

## RTO After Route Repair

- Same as before route break
  - If new route long, this RTO may be too small, leading to timeouts
    - Except when RTT small compared to clock granularity
- Same as TCP start-up (6 second)
  - May be too large
  - Will result in slow response to future losses
- **Proposal:** new RTO = function of old RTO, old route length, and new route length
  - Example:  $\text{new RTO} = \text{old RTO} * \text{new route length} / \text{old route length}$
  - Not evaluated yet

# Impact of MAC - Delay Variability

- As wireless medium is shared between multiple sources, the round-trip delay is variable
- Also, on slow wireless networks, delay is large
  - made larger by send-receive turnaround time
- Large and variable delays result in larger RTO
- On packet loss, timeout takes much longer to occur
- Idle source (waiting for timeout to occur) lowers TCP throughput

# Impact of MAC - Delay Variability

## [Balakrishnan97]

Several techniques may be used to mitigate problem, based on minimizing ack transmissions

- to reduce frequency of send-receive turnaround and contention between acks and data
- **Piggybacking** link layer acks with data
- **Sending fewer TCP acks** - ack every  $d$ -th packet ( $d$  may be chosen dynamically)
  - but need to use rate control at sender to reduce burstiness (for large  $d$ )
- **Ack filtering** - Gateway may drop an older ack in the queue, if a new ack arrives
  - reduces number of acks that need to be delivered to the sender

# Out-of-Order Packet Delivery

- Route changes may result in out-of-order (OOO) delivery
- Significantly OOO delivery confuses TCP, triggering fast retransmit
- Potential solutions:
  - Avoid OOO delivery by ordering packets before delivering to IP layer
    - can result in **variable delay**
  - turn off fast retransmit
    - can result in **poor performance** in presence of congestion