Running TCP on Wireless Links

- **TCP interprets any packet loss as a sign of congestion.**
  - TCP sender reduces congestion window.
- **On wireless links, packet loss can *__also__* occur due to random channel errors, or cellular or WLAN handoffs.**
  - Temporary loss not due to congestion.
  - Reducing congestion window may be too conservative.
  - Leads to poor throughput.

Running TCP on Wireless Links

- **Fundamental question: How to distinguish loss due to congestion from loss due to other wireless/mobility reasons?**
- **Hard to do: TCP is fundamentally end-to-end.**
  - We just know that packet is lost, not why it is lost.
- **Existing solutions break the end-to-end principle to some extent.**
  - Also must be compatible with existing TCP.

Broad Approaches

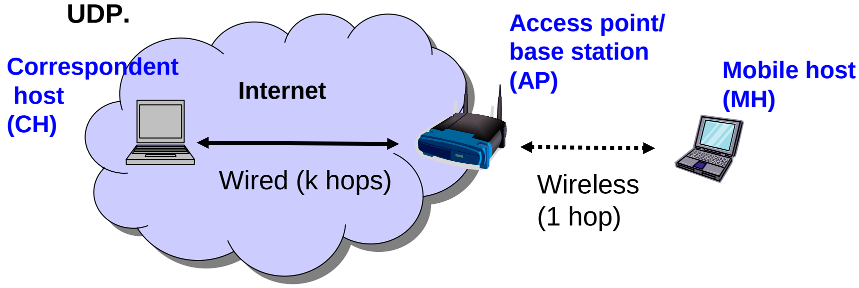
- **Two broad approaches to run TCP over wireless links.**
  1. Mask wireless loss from the TCP sender.
     - Then TCP sender will not reduce congestion window.
  2. Explicitly notify the TCP sender about cause of packet loss.
     - TCP sender will not reduce congestion window for wireless losses.
- **Some additional approaches designed to explicitly handle mobility.**
- **Solutions may be at the TCP sender, at the TCP receiver, or at an intermediate node (typically, wireless basestation or WLAN access point).**

Techniques to Mask Wireless Losses from TCP Sender

- **Split connection approach**
  - I-TCP [Bakre-Badrinath-ICDCS-95]
- **Snoop TCP** [Balakrishnan-et-al-ACM-Winet-95].
- **These solutions assume that the wireless part is just one hop (traditional cellular or WLAN network).**
- **All losses on wireless side assumed not connected with congestion.**
  - Note that this may not true always; e.g., losses due to collision is because of congestion. But such subtleties are ignored. Assume that link layer is able to overcome congestion losses.

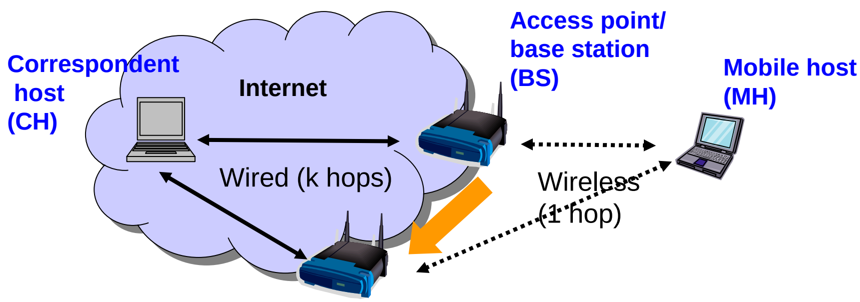
Indirect TCP (I-TCP)

- Segment the TCP connection into two.
- No changes to the TCP protocol for hosts connected to the wired Internet (correspondent host or CH).
- Split the TCP connection at AP into 2 TCP connections, one between CH and AP, the other between AP and MH. No real end-to-end connection.
- The connection between AP and MH does not need to be a real TCP. Can be a custom transport protocol that is tuned for the wireless hop. For example, selective repeat over UDP.



I-TCP Socket and State Migration

- On handoff, connection state must be migrated.



I-TCP Critique

- **Advantages**
  - No changes in the fixed network necessary.
  - Transmission errors on the wireless link do not propagate into the fixed network. Local recovery from errors.
  - Possibility of using custom (optimized) transport protocol for the AP-MH hop.

- **Disadvantages**
  - Loss of end-to-end semantics, an ACK to sender does now not any longer mean that a receiver really got a packet. Problem if there is a crash at AP.
  - Large buffer space may be needed at AP.
  - AP must maintain per-TCP connection state.
  - State must be forwarded to new AP on handoff. May cause higher handoff latency.

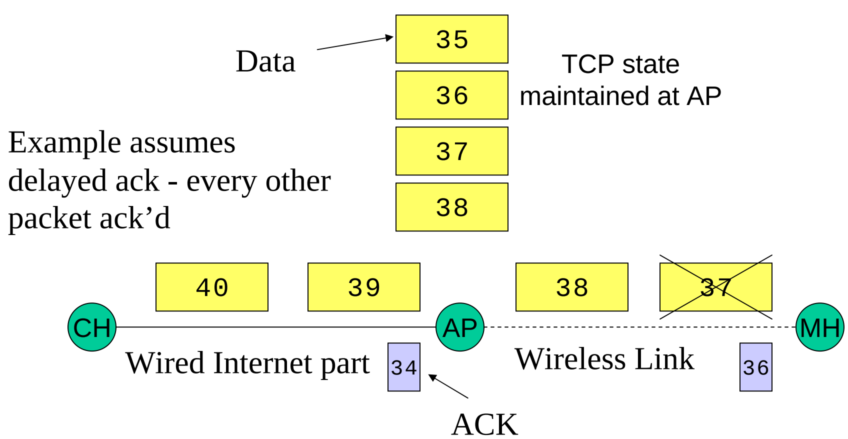
Snoop TCP

- **Removes the limitation of I-TCP**
  - No more split connection.
  - Single end-to-end connection like regular TCP.
- **Only access-point (AP) modified for a base implementation.**
  - Modification on MH improves over the base implementation. But not mandatory.
- **AP "snoops" on all TCP packets. It buffers packets for the MH.**

Snoop TCP (contd.)

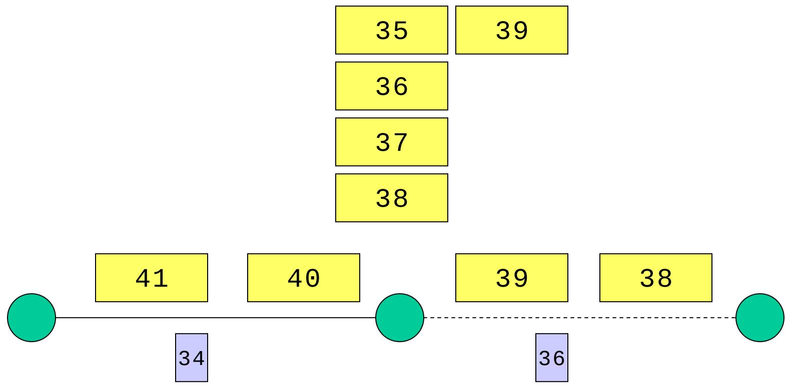
- **Data transfer to MH**
  - AP buffers data until it receives ACK from MH, AP detects packet loss via dupacks or time-out, and retransmits packet.
  - CH unaware of loss or retransmission. No reduction in congestion window.
- **Data transfer from MH**
  - AP detects packet loss on the wireless link via missing sequence numbers, AP answers directly with a NACK to the MH.
  - MH can now retransmit data with only a very short delay.
  - This requires modification on the MH.

Snoop : Example



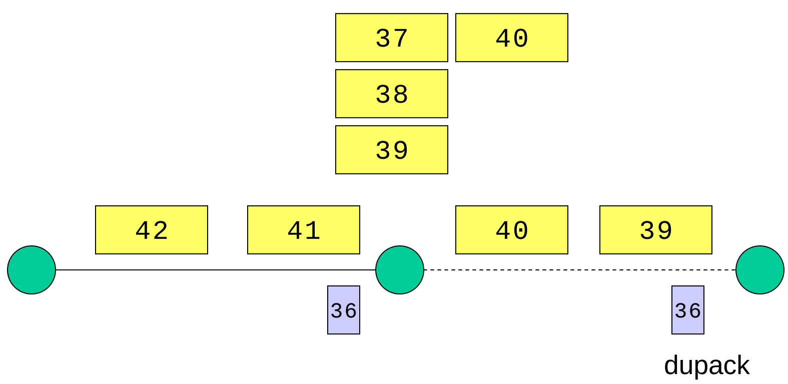
[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

Snoop : Example



[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

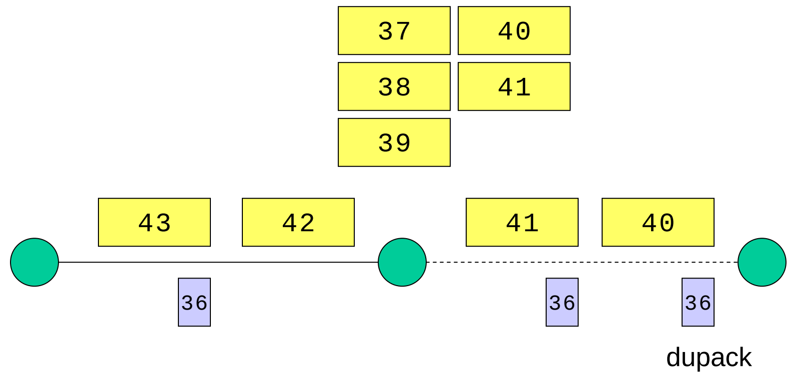
Snoop : Example



Duplicate acks are not delayed

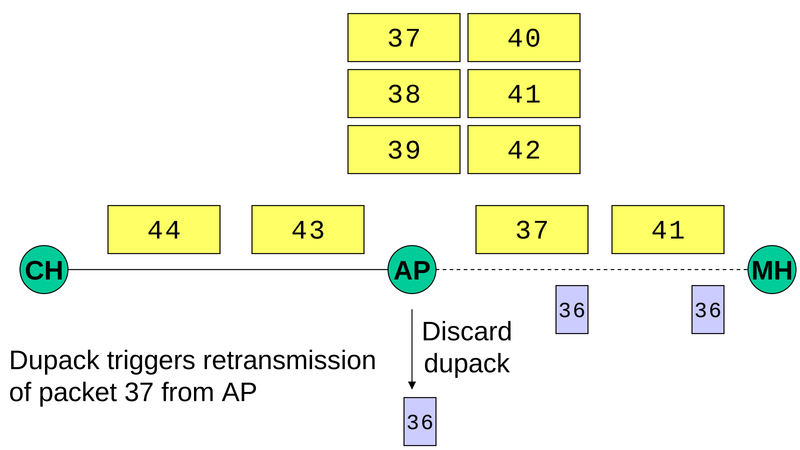
[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

Snoop : Example



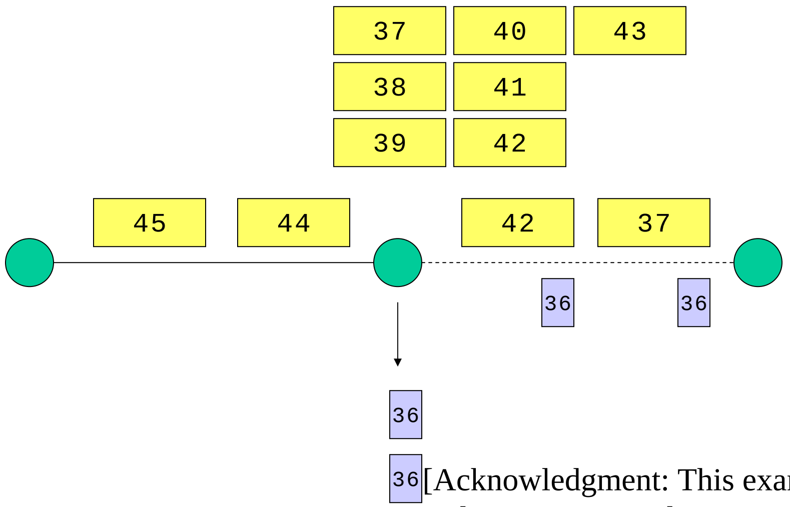
[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

Snoop : Example



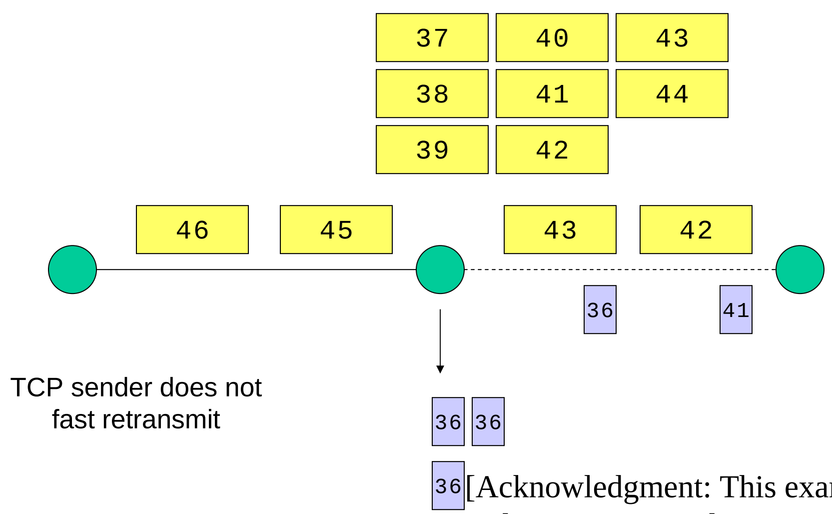
[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

Snoop : Example



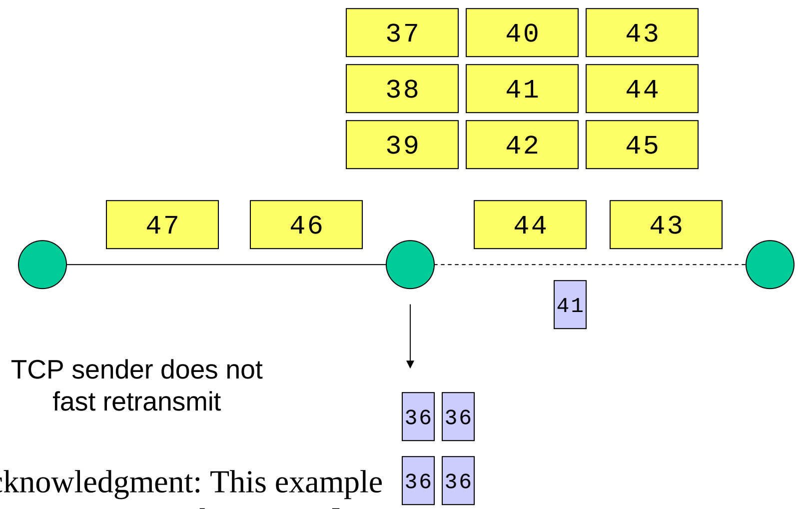
[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

Snoop : Example



[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

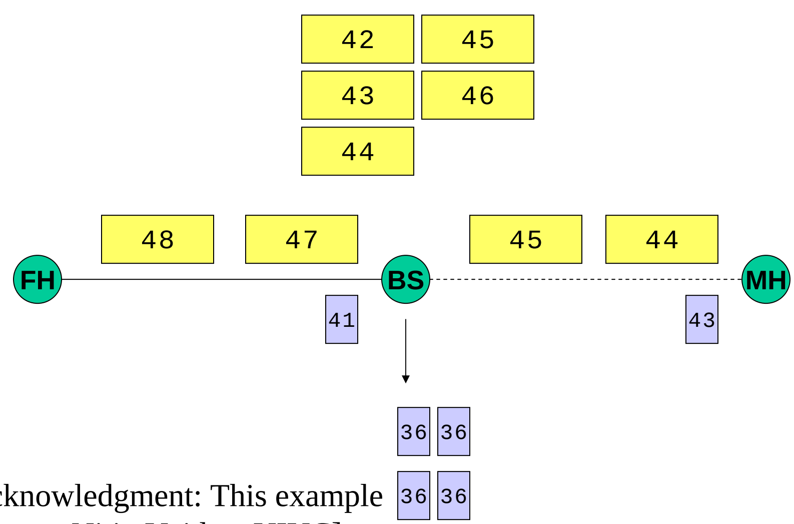
Snoop : Example



TCP sender does not fast retransmit

[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

Snoop : Example



[Acknowledgment: This example is due to Nitin Vaidya, UIUC]

Critique of Snoop TCP

- **Advantages:**
  - Can work without modification on MH.
  - Preserves end-to-end semantics. Crash does not affect correctness, only performance.
  - After handoff, new AP does not need to understand snoop TCP for communication to continue. Can automatically fall back on to regular TCP.
  - No state needs to be migrated. But if done, this can improve performance.
    - Note such "state" is called soft state. Good if available. But can work if not available.
- **Disadvantages:**
  - For the NACK scheme to work MH still needs to be modified.
  - Does not work with encrypted TCP headers.
  - Does not work for asymmetric routes.

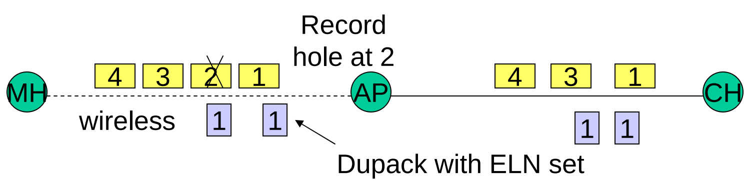
Explicit Notification-Based Approach

- **Send notification to the TCP sender about wireless packet loss.**
- **Upon notification, TCP sender retransmits packet, but does not reduce congestion window.**
- **Motivated by the Explicit Congestion Notification (ECN) Approach [Floyd-94].**
- **Many design options: Who sends notification? How? How notification is interpreted at sender?**
- **We will discuss one example approach.**

Explicit Loss Notification (ELN) Approach

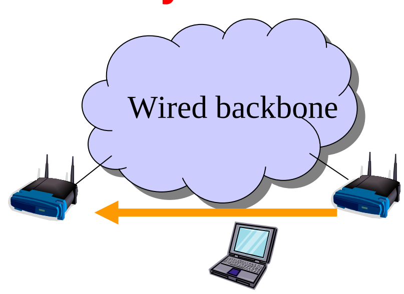
[Balakrishnan-Katz-Globecomm-98]

- **Assume MH is the TCP sender.**
- **AP keeps track of holes in the packet sequence received from the sender**
- **When a dupack is received from the receiver (CH), AP compares the dupack sequence number with the recorded holes**
  - if there is a match, an ELN bit is set in the dupack
- **When sender (MH) receives dupack with ELN set, it retransmits packet, but does not reduce congestion window.**



[Example due to Nitin Vaidya, UIUC]

Impact of Mobility on TCP Performance



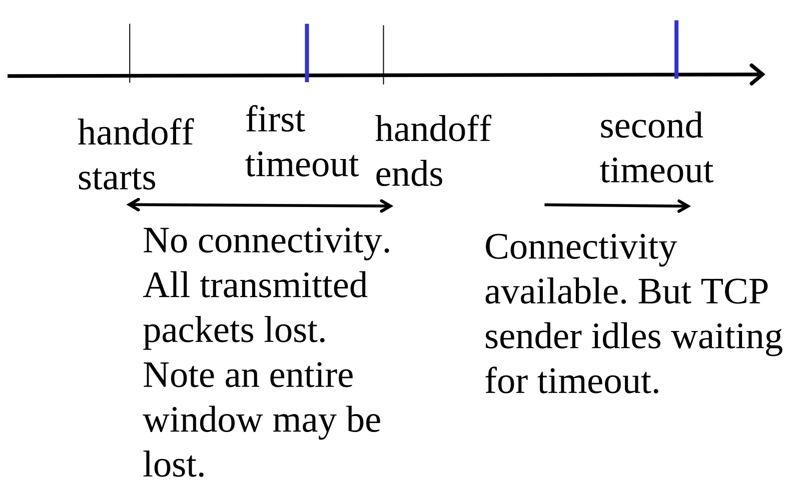
- **Handoff can be either at the link layer (IP does not know) or at the network layer (IP is aware).**
- **Link layer handoff may not impact TCP much.**
  - Other than a transient increase in RTT.
- **Network layer handoff (e.g., Mobile IP) is slow. This is because routing must be updated.**
  - Packets can be lost.
  - TCP is impacted.
  - We are interested in such handoffs.

Fast Retransmit-Based Solution

- **During the long delay for a mobile-IP handoff to complete, a whole window worth of packets may be lost.**
  - Assuming no buffering/forwarding.
- **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.

[Caceres-Iftode-95]

Illustrative Timeline



Fast Retransmit-Based Solution

- **Assumption: MH is aware of handoff process.**
- **When MH is the TCP receiver: after handoff is complete, it sends 3 dupacks to the sender**
  - this triggers fast retransmit at the sender.
- **When MH is the TCP sender: invoke fast retransmit after completion of handoff.**
- **Advantages**
  - no slow start after handoff.
  - Retransmissions immediately after handoff instead of waiting for timeout.
  - Very minor change on TCP on MH only.
- **Disadvantages**
  - Only handles losses due to handoff.
  - Retransmitted packets will still traverse the entire network.
  - Congestion window still reduces upon handoff.

Mobile TCP (M-TCP)

- **The fast-retransmit based solution can start retransmission immediately after handoff is complete. But it cannot prevent reduction in congestion window.**
- **M-TCP also prevents reduction in congestion window.**
- **How? Using <u>persist mode</u> of TCP.**

[Brown-Singh-97]

M-TCP Uses TCP's Persist Mode

- **TCP fact: When a new ACK is received with receiver's advertised window = 0 (in TCP header), the sender enters persist mode.**
  - Means receiver does not have space to accept more packets.
- **Sender does not send any data in persist mode.**
- **When a positive window advertisement is received again, sender exits persist mode.**
- **On exiting persist mode, RTO and cwnd are same as before the persist mode.**

M-TCP Details

- **Similar to split connection approach (I-TCP).**
  - But maintains end-to-end semantics. AP forwards ACK only after it receives ACK.
- **When the AP detects handoff or disconnection**
  - AP advertises zero receiver window to sender.
  - This forces sender into persist mode.
  - After handoff is complete (connectivity is regained) new AP advertises correct receive window size.
- **How is the zero window advertisement is sent?**
  - AP withholds the ACK for the last byte.
  - This ACK carries the zero window advertisement on handoff.

Critiquing M-TCP

- **Some argue that not reducing the congestion window may not always be a good idea.**
  - Level of congestion on new route is unknown!
- **M-TCP needs help from AP for zero window advertisement.**
  - It is possible for the receiver to do this, when it is the MH.

Concluding Remarks

- **Need extra knowledge on wireless side to detect loss due to wireless/mobility effects that is unconnected to congestion.**
- **MH and/or AP may know such information.**
- **Approaches modify TCP on MH or introduce a support protocol on AP (or do both).**
  - Doing anything on AP contradicts end-to-end principle.
- **Some approaches only provide specific help. For example, improvements only when MH is TCP receiver or sender, but not both.**