CMPE 257: Wireless and Mobile Networking

Katia Obraczka
Computer Engineering
UCSC Baskin Engineering

Lecture 9

CMPE 257 Winter'11
Student Presentations

• Feb 28:
  – Mobility management: Tyler and Niosha.

• March 2:
  – DTNs: Philip and Rance.

• March 7:
  – Hybrid networks: Gregg and Darien.

• March 9:
  – Energy management: Mohamed.
  – Security: Jim.

• March 14:
  – Security: Chris and Seth.
Schedule

• Exam: Feb 23.
• Project presentations: March 17.
  – Time change: 5-8pm.
  – Approximately 15 minutes for each presentation.
Today

• End-to-end protocols.
E2E Protocols

• Reliable point-to-point.
• Reliable multipoint.
Reliable Point2Point Transport Layer: Outline

• TCP/IP basics.
• Impact of transmission errors on TCP performance.
• Approaches to improve TCP performance on wireless networks.
  – Classification.
• TCP on cellular.
• TCP on MANETs.
Internet Protocol (IP)

- Best-effort service:
  - Packets may be delivered out-of-order.
  - Packets may be lost.
  - Packets may be duplicated.
Transmission Control Protocol

- Reliable ordered delivery.
- Implements flow and congestion control.
- Reliability through retransmissions.
- End-to-end semantics:
  - ACKs sent to TCP sender confirm delivery of data by TCP receiver.
  - ACK for data sent only after it reached receiver.
TCP Basics

• Cumulative acknowledgements.
  – $ACK_i$ acknowledges receipt of packets through $(i-1)$.

• TCP uses byte sequence numbers.
  – For simplicity, usually refer to packet sequence numbers.
Cumulative ACKs

- **New** ACK generated only on receipt of *new in-sequence* packet.
Delayed ACKs

• ACK delayed until:
  – Another packet is received, or
  – Delayed ACK timer expires (200 ms typical)

• Reduces ACK traffic.
Delayed ACKs

New ACK not produced on receipt of packet 36, but on receipt of 37
Duplicate ACKs

• A **dupack** is generated whenever an **out-of-order** segment arrives at the receiver.
Duplicate ACKs

(Above example assumes *delayed acks*)

On receipt of 38

Dupack
Duplicate ACKs

• Duplicate ACKs are **not delayed**.
• Duplicate ACKs may be generated when:
  – a packet is **lost**, or
  – a packet is delivered **out-of-order (OOO)**.
Out-of-Order Packets

Dupack
On receipt of 37
Window-Based Control

• Sliding window protocol.
• Window size minimum of
  – receiver’s advertised window – function of available receiver buffer size.
  – congestion window - determined by sender; based on feedback from the network
Sliding Window

Sender’s window

Acks received

Not transmitted

Ack 5

CMPE 257 Winter'11
Self-Clocking

- New data sent when old data is ack’d.
- Helps maintain “equilibrium”.
- Congestion window size bounds the amount of data that can be sent per round-trip time.
- Throughput $\leq \frac{W}{RTT}$. 
Window Size

- Ideal size = delay * bandwidth
- What if window size < delay*bw?
  - Inefficiency (wasted bandwidth).
- What if > delay*bw?
  - Queuing at intermediate routers.
  - Potential for packet loss.
TCP Packet Loss Detection

• TCP assumes that packet loss indicates congestion.

• Packet loss detection:
  – Retransmission timeout (RTO).
  – Duplicate acknowledgements.
RTO

- For very packet transmitted, TCP sender starts timer.
- If acknowledgement for timed packet not received before timer=RTO, packet assumed lost.
- RTO dynamically calculated.
RTO Calculation

• RTO = mean + 4 mean deviation.
• Large variations in the RTT increase the deviation, leading to larger RTO.
Exponential Backoff

- Double RTO on each timeout

Packet transmitted

Time-out occurs before ack received, packet retransmitted

Timeout interval doubled

\[ T_2 = 2 \times T_1 \]
Duplicate ACKs

• Timeouts can take too long.
• How to initiate retransmission sooner?
  – Use Duplicate ACKs as loss indicator.
• Dupacks may be generated due to:
  – Packet loss, or
  – Out-of-order packet delivery.
• TCP sender assumes packet loss if it receives 3 consecutive dupacks.
Note on Duplicate ACKs

3 dupacks are also generated if a packet is delivered at least 3 places beyond its in-sequence location.
TCP Congestion Control

- Slow start.
- Congestion avoidance.
- Fast retransmit.
- Fast recovery.
- Selective acknowledgements (SACK).
Slow Start

- Initially, \( cwnd = 1 \) MSS (max. segment size).
- Increment \( cwnd \) by 1 MSS on each new ACK.
- Slow start ends when \( cwnd \) reaches the \textit{slow-start threshold}.
- \( cwnd \) grows \textit{exponentially} in slow start.
  - Factor of 1.5 per RTT if every other packet ack’d.
  - Factor of 2 per RTT if every packet ack’d.
  - Could be less if sender does not always have data to send.
Congestion Avoidance

• On each new ACK, increase $cwnd$ by $1/cwnd$ packets.
• $cwnd$ increases **linearly** with time during congestion avoidance.
  – $1/2$ MSS per RTT if every other packet ack’d.
  – $1$ MSS per RTT if every packet ack’d.
Assumes acks are not delayed.
Congestion?

• On detecting a packet loss, TCP sender assumes network congestion.
Timeout

- On a timeout, slow start is invoked.
  - cwnd is reduced to the initial value of 1 MSS.
- Slow start threshold is set to half the window size before packet loss, or:
  \[ \text{ssthresh} = \max(\min(\text{cwnd}, \text{receiver's advertised window})/2, 2 \text{ MSS}) \].
Timeout (cont’d…)

After timeout

\[ \text{ssthresh} = 8 \]

\[ \text{ssthresh} = 10 \]

\[ \text{cwnd} = 20 \]
Fast Retransmit

• When sender receives multiple (>= 3) duplicate ACKs, assumes packet lost without waiting for timeout.
  – Retransmits packet.

• TCP Tahoe: slow start, congestion avoidance, fast retransmit.
Fast Recovery

• Avoids slow start after single packet loss.
• Operates in conjunction with fast retransmit.
• After TCP sender receives 3 duplicate ACKs:
  – Retransmits one packet.
  – Reduces cwnd by half.
  – Every subsequent duplicate ACK clocks transmission.
  – New ACK causes sender to exit fast recovery.
Fast Recovery

• ssthresh = \( \min(cwnd, \text{receiver's advertised window})/2 \)  
  (at least 2 MSS)
• retransmit the missing segment (fast retransmit)
• \( cwnd = ssthresh + \text{number of dupacks} \)
• when a new ack comes: \( cwnd = ssthresh \)
  – Enter congestion avoidance.

Congestion window cut in half.
After fast retransmit and fast recovery window size is reduced in half.
TCP Reno

- Slow-start
- Congestion avoidance
- Fast retransmit
- Fast recovery
Other TCP Variants

Reno still suffers when multiple losses per RTT.

- **TCP New-Reno**
  - Stay in fast recovery until all packet losses in window are recovered.
  - **Can recover 1 packet loss per RTT without causing a timeout.**

- **Selective Acknowledgements (SACK)** provides information about out-of-order packets received by receiver.
  - **Can recover multiple packet losses per RTT.**
Impact of transmission errors on TCP performance
Random Errors

• If number of errors is small, they may be corrected by an error correcting code.

• Excessive bit errors result in 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 May Cause Fast Retransmit

Example assumes delayed ack - every other packet ack’d
Random Errors May Cause Fast Retransmit

Duplicate acks are not delayed
Random Errors May Cause Fast Retransmit
Random Errors May Cause Fast Retransmit

3 duplicate acks trigger fast retransmit at sender
Random Errors May Cause Fast Retransmit

• Fast retransmit results in:
  – Retransmission of lost packet.
  – Reduction in congestion window.

• Reducing congestion window in response to transmission errors is unnecessary.
Observations

• Sometimes congestion response may be appropriate in response to random errors.
• Example: errors may occur due to interference from other users or noise.
  – Interference due to other users is an indication of congestion, and thus it is appropriate to reduce congestion window.
  – If noise causes errors, it is not appropriate to reduce window.
• When a channel is in a bad state for a long duration, it might be better to let TCP backoff, so that it does not unnecessarily attempt retransmissions.
Burst Errors and Timeouts

• If wireless link remains unavailable for extended duration, multiple packets in a window’s 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.
Impact of Transmission Errors

- TCP cannot distinguish between packet losses due to congestion and transmission errors.
- Unnecessarily reduces congestion window.
- Throughput suffers.
Approaches to Improve Performance of TCP in Wireless Networks
Classification

• Based on who takes the action and
• What kind of action taken.

• E2E versus cross-layer approaches.
  – E2E approaches: connection end points try to distinguish between congestion and non-congestion losses.
  – Cross-layer approaches: combination of e2e and intermediate node participation.
Infrastructure-Based Wireless

• Earlier efforts to improve TCP’s performance focused on infrastructure-based wireless environments.
Cross-Layer Approaches

• Link layer error recovery.
• Link layer retransmission.
  – TCP-awareness.
  – TCP-unawareness.
• Split connection.
Link Layer Mechanisms: Error Correction

- Example: Forward Error Correction (FEC) can be used to correct limited 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

- Link level retransmission schemes retransmit a packet at the link layer, if errors are detected.
- Retransmission overhead incurred only if errors occur.
Link Layer Mechanisms

May combine both FEC and retransmissions:

- Use FEC to correct small number of errors.
- Use link level retransmission when FEC capability is exceeded.
Link Level Retransmissions

TCP connection

Link layer state

rxmt

CMPE 257 Winter'11
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, …)
Link Level Retransmissions

Issues

• How much time is required to trigger link layer retransmission?
  – Small fraction of end-to-end TCP RTT.
  – Multiple of end-to-end TCP RTT.

• Should 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: Summary

When is a reliable link layer beneficial to TCP performance?

- If it provides almost in-order delivery.
- TCP retransmission timeout large enough to tolerate additional delays due to link level retransmits.
Cross-Layer Approaches

- Link layer error recovery.
- Link layer retransmission.
  - TCP-awareness.
  - TCP-unawareness.
- Split connection.
TCP-Aware Link Layer
Snoop Protocol
[Balakrishnan95]

• Retains local recovery of Split Connection approach.
• Link level retransmissions.
• Differs from split connection schemes:
  – End-to-end semantics retained
  – Soft state at base station.
Snoop Protocol

TCP connection

Per TCP-connection state

application
transport
network
link
physical

application
transport
network
link
physical

application
transport
network
link
physical

FH
wireless
BS
MH

CMPE 257 Winter'11
Snoop Protocol

• **Buffers data** packets at base station.
  – Data sent by FH not yet ack’d by MH.
  – Allow link layer retransmission.

• When dupacks received by BS from MH (or local timeout), *retransmit* on wireless link, if packet in buffer.

• **Prevents fast retransmit** by TCP sender at FH by suppressing dupacks at BS.
Snoop: Example

TCP state maintained at link layer

Example assumes delayed ack - every other packet ack’d
Snoop : Example
Snoop: Example

Duplicate acks are not delayed
Snoop: Example

Duplicate acks
Snoop : Example

Dupack triggers retransmission of packet 37 from base station

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

Discard dupack

CMPE 257 Winter'11
Snoop: Example

CMPE 257 Winter’11
TCP sender does not fast retransmit
TCP sender does not fast retransmit
Snoop: Example

CMPE 257 Winter’11
Performance

2 Mbps Wireless link

CMPE 257 Winter'11
Snoop Protocol: Advantages

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

• If wireless link delay-bandwidth product less than 4 packets: simple (TCP-unaware) link level retransmission scheme can suffice.
  – Since delay-bandwidth product is small, retransmission scheme can deliver lost packet without causing MH to send 3 dupacks.
Snoop Protocol: Advantages

• Higher throughput can be achieved.
• 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.
Delayed Dupacks Approach

- TCP-unaware approximation of TCP-aware link layer.
- Attempts to imitate Snoop without making BS TCP-aware.
- Snoop implements two features at BS:
  - Link layer retransmission.
  - Dupack handling: reduced interference between TCP and link layer retransmissions (drop dupacks).
Delayed Dupacks

• Implements same two features:
  – at BS : link layer retransmission.
  – at MH : reducing interference between TCP and link layer retransmissions (by delaying dupacks).
Delayed Dupacks Protocols

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

• Delayed dupacks released after interval D, if missing packet not received.
• Link layer maintains state to allow retransmission.
Delayed Dupacks: Example

Example assumes delayed ack - every other packet ack’d
Link layer acks are not shown
Removed from BS link layer buffer on receipt of a link layer ack (LL acks not shown in figure)
Delayed Dupacks: Example

Duplicate acks are not delayed
Delayed Dupacks: Example

- Original ack
  - 36
  - 42
  - 43

- Duplicate acks
  - 36
  - 40
  - 41
  - 39
  - 37
  - 41
Delayed Dupacks: Example

Base station forwards dupacks
CMPE 257 Winter'11
Delayed Dupacks: Example

dupacks

36

36

36

36

CMPE 257 Winter’11
Delayed Dupacks : Example

TCP sender does not fast retransmit

Delayed dupacks are discarded if lost packet received before delay D expires
Delayed Dupacks [Vaidya99]

2 Mbps wireless duplex link with 20 ms delay
No congestion losses

20 ms 20 ms
10 Mbps 2 Mbps
Delayed Dupacks [Vaidya99]

5% packet loss due to congestion

CMPE 257 Winter'11
Delayed Dupacks:

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 D sufficient in such cases.
Delayed Dupacks: Disadvantages

- Right value of dupack delay $D$ dependent on wireless link properties.
- Mechanisms to determine $D$ needed.
- Delays dupacks for congestion losses too, delaying congestion loss recovery.
Cross-Layer Approaches

- Link layer error recovery.
- Link layer retransmission.
  - TCP-awareness.
  - TCP-unawareness.
- Split connection.
Split Connection Approach
Split Connection Approach

• End-to-end TCP connection is broken into one connection on the wired part of route and one over wireless part.
Split Connection Approach

• Connection between wireless host MH and fixed host FH goes through base station BS.

• $\text{FH-MH} = \text{FH-BS} + \text{BS-MH}$
Split Connection Approach

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

**Diagram:**
- FH: Fixed Host
- BS: Base Station
- MH: Mobile Host
Split Connection Approach
Split Connection Approach: Classification

- Hides transmission errors from sender
- Primary responsibility at base station
- If specialized transport protocol used on wireless, then wireless host also needs modification
Split Connection Approach: Example

• Indirect TCP [Bakre94]
  – FH - BS connection : Standard TCP.
  – BS - MH connection : Standard TCP.
Split Connection: Advantages

• BS-MH connection can be optimized independent of FH-BS connection.
  – Different congestion/error control.
• 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 per window.
  – Selective ACKs improve performance.
Split Connection:
Disadvantages

• End-to-end *semantics* violated.
  – ACK may be delivered to sender before data delivered to receiver.
Split Connection: Disadvantages

- BS retains hard state.
  - BS failure can result in loss of data.
    - If BS fails, packets 39 and 40 will be lost.
    - Both ack’d to sender; sender does not buffer.
Split Connection: 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.
Handoff

New base station

Hand-off

MH

BS

FH

40

39

38

37

36

39

40

40
Split Connection: Disadvantages

- **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.
Split Connection: Disadvantages

• May not be useful if data and acks traverse different paths.
  – Example: data on a satellite wireless hop, acks on a dial-up channel.

![Diagram showing FH, BS, and MH with data and ack labels]
E2E Approaches

• Strict E2E versus E2E with intermediate node involvement.
• E2E with intermediate node involvement:
  – Explicit notifications.
Explicit Notification Schemes

General Philosophy

• Approximate ideal TCP behavior.
  – Ideally, TCP sender should simply retransmit a packet lost due to transmission errors without taking any congestion control actions.

• A node determines whether packets are lost due to errors and informs sender using an “explicit notification”.

• Sender, on receiving the notification, does not reduce congestion window, but retransmits lost packet.
Explicit Notification Schemes

- Motivated by Explicit Congestion Notification (ECN) proposals [Floyd94].

Variations proposed in literature differ in:
- Who sends explicit notification.
- How they decide to send explicit notification.
- What sender does on receiving notification.
Explicit Loss Notification
[Balakrishnan98]

• MH is the TCP sender.
• Wireless link first on path from sender to receiver.
• Base station keeps track of holes in packet sequence.
• When a dupack is received from the receiver, BS compares the dupack sequence number with recorded holes.
  – If there is a match, an ELN bit is set in dupack.
ELN

- When sender receives dupack with ELN set, it retransmits packet, but does not reduce congestion window.
Explicit Loss Notification

[Biaz99thesis]

• Adapts ELN proposed in [Balakrishnan98] for the case when MH is receiver.
• Caches TCP sequence numbers at base station, similar to Snoop. But does not cache data packets, unlike Snoop.
• Duplicate acks are tagged with ELN bit before being forwarded to sender if sequence number for the lost packet is cached at BS.
• Sender takes appropriate action on receiving ELN.
ELN [Biaz99thesis]

Sequence numbers cached at base station

Dupack with ELN
Explicit Bad State Notification

[Bakshi97]

- MH is TCP receiver.
- BS attempts to deliver packets to MH using link layer retransmission scheme.
- If packet cannot be delivered using small number of retransmissions, BS sends a Explicit Bad State Notification (EBSN) message to TCP sender.
- When TCP sender receives EBSN, it resets RTO.
  - Timeout delayed when wireless channel in bad state.
Partial Ack Protocols

[Cobb95][Biaz97]

• Send two types of acknowledgements.
• Partial ACK informs sender that a packet was received by an intermediate host (typically, base station).
• Normal TCP cumulative ACK needed by sender for reliability purposes.
Partial Ack Protocols

- When packet for which partial ack is received detected to be lost, sender does not reduce its congestion window.
  - Loss assumed to be due to wireless errors.
Variations

• Base station may or may not locally buffer and retransmit lost packets.
Strict E2E Schemes
Receiver-Based Scheme

[Biaz98Asset]

- MH is TCP receiver.
- Receiver uses heuristics to guess cause of packet loss.
- When receiver believes that packet loss is due to errors, it sends notification to sender.
- TCP sender, on receiving notification, retransmits lost packet, without reducing congestion window.
Heuristics

• Receiver uses inter-arrival time between consecutively received packets to guess cause of 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

- Packet loss due to congestion
Receiver-Based Scheme

- Packet loss due to transmission error

```
   FH     BS     MH
  12  11  10
```

```
   FH     BS     MH
  12  11  10
      Error loss
```

CMPE 257 Winter’11
Sender-Based Discrimination Scheme
Sender-Based Discrimination

[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

• Define condition C as a function of congestion window size and observed RTTs.
• Condition C evaluated for new RTT.
• If (C == True) reduce congestion window.
Heuristics for Congestion Avoidance: Some proposals

- TCP Vegas [Brakmo94]

  expected throughput \( ET = \frac{W(i)}{RTT_{\text{min}}} \)

  actual throughput \( AT = \frac{W(i)}{RTT(i)} \)

  \[ \text{Condition C} = (ET - AT > \beta) \]
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 due to congestion.
  - Else assume packet loss 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!!

Reason

• 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.
Reliable Point2Point Transport Layer: Outline

✓ TCP/IP basics.
✓ Impact of transmission errors on TCP performance.
✓ Approaches to improve TCP performance on wireless networks.
  ✓ Classification.
✓ TCP on cellular.
  ▪ TCP on MANETs.