CMPE 257: Wireless and Mobile Networking

Spring 2005
E2E Protocols (point-to-point)
Announcements

Final day switched!
June 9\textsuperscript{th} from 4-7pm.
Project presentations.
Final exam on June 2\textsuperscript{nd}. 
Today

E2E 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.
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 $i$.

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

40  39  38  37

33  35

41  40  39  38

35  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

40 39 37 38

41 40 39 37

36 36

Dupack

On receipt of 37
Number of Dupacks

40  39  37  38

41  40  39  37

42  41  40  39

New Ack 34  New Ack 36  New Ack 36  New Ack 36
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

Ack 5

Not transmitted

Sender’s window

Ack 5

Acks received

Not transmitted
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 $\leq W / RTT$. 
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.
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 = \text{mean} + 4 \text{ 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

\[ T_2 = 2 \times T_1 \]

Timeout interval doubled
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 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.
On a timeout, slow start is invoked.

cwnd is reduced to the initial value of $1\text{ MSS}$.

Slow start threshold is set to half the window size before packet loss, or:

$$\text{ssthresh} = \text{maximum}(\text{min}(\text{cwnd, receiver’s advertised window})/2, 2\text{ MSS}).$$
Timeout (cont’d…)

After timeout

Congestion window (segments)

Time (round trips)

mol = 20

ssthresh = 8

ssthresh = 10
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

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

Duplicate acks
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.
Single-Hop Wireless

Earlier efforts to improve TCP’s performance focused on single-hop wireless environments.
Cross-Layer Approaches

Link layer error recovery.
Link layer retransmission.
  TCP-awareness.
  TCP-unawareness.
Split connection.
Link Layer Error Recovery

Used over the wireless link.
Between AP and MH.
Try to recover transmission error at lower layers.
Overhead incurred at adjacent nodes.
Link Layer Error Recovery

TCP connection

- Application
- Transport
- Network
- Link
- Physical

- Application
- Transport
- Network
- Link
- Physical

Link layer state

rxmt

wireless
Forward Error Correction

Example of link layer error correction mechanism.
Transmitter adds redundant information that can be used to correct errors at the receiver.
E.g., fully replicate information.
Trade-offs?
Link Layer Retransmission

Only performed if errors are detected.

Issues:

How many times?

How long to wait before re-transmitting?

Can complement error correction.