Wireless and Mobile Networks

CMPE 257

Spring 2006
End-to-End Protocols
Announcements

• Exam:
  – June 7th in class.

• Project presentations and demos:
  – June 12th. 4-7pm.
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 single-hop infrastructure-based wireless networks.
- 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 \( i \).

• TCP uses byte sequence numbers.
  – For simplicity, we 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
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

On receipt of 37, Dupack is triggered.
Number of Dupacks

40
39
37
38

41
40
39
37

42
41
40
39

New Ack
New Ack
New Ack
New Ack

34
34
36
36

Dupack
Dupack
Dupack
Dupack
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

1 2 3 4 5 6 7 8 9 10 11 12 13

Acks received

Ack 5

Not transmitted

1 2 3 4 5 6 7 8 9 10 11 12 13

Sender's window
Self-Clocking

- New data sent when old data is ack’ed.
- 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 * available 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

- \[ \text{RTO} = \text{mean} + 4 \times \text{mean deviation}. \]
- 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

\[ T_2 = 2 \times T_1 \]
Reacting to 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

12 8 11 10 9 7

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.
- TCP variants.
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 **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.
Timeout (cont’d…)

After timeout

Congestion window (segments)

Time (round trips)

ssthresh = 8

ssthresh = 10

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

- \( ssthresh = \frac{\min(cwnd, \text{receiver's advertised window})}{2} \) (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

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 Infrastructure-Based Wireless Networks

• Earlier efforts to improve TCP performance focused on single-hop infrastructure-based wireless environments.
  – I.e., networks where hosts are directly connected to AP through wireless link.
Cross-Layer Approaches

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

• Try to shield upper layers (e.g., transport) from errors that can be recovered at lower layers.

• Used over the wireless link.
  – Between AP and MH.

• Overhead incurred at adjacent nodes.
Link Layer Error Recovery

TCP connection

application
transport
network
link
physical

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 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.
- Can complement error correction.
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

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.

and

- 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 Retransmission
Snoop Protocol
[Balakrishnan95]

- More sophisticated link-level retransmission scheme.
- 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

rxmt

CMPE 257 - Wireless and Mobile Networking
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

Example assumes delayed ack - every other packet ack’d
Snoop : Example
Duplicate acks are not delayed
Snoop : Example

Duplicate acks
Discard dupack

Dupack triggers retransmission of packet 37 from base station

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

BS needs to be TCP-aware to interpret TCP headers
Snoop : Example
Snoop: Example

TCP sender does not fast retransmit
Snoop : Example

TCP sender does not fast retransmit
Snoop : Example
Performance

2 Mbps Wireless link

1/error rate (in bytes)

bits/sec

0 400000 800000 1200000 1600000 2000000

16K 32K 64K 128K 256K no error

base TCP Snoop
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
Delayed Dupacks: Example

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

Duplicate acks
Delayed Dupacks: Example

Base station forwards dupacks

Dupack

Dupacks

Delayed dupack
Delayed Dupacks: Example

dupacks

Delayed dupacks
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

10 Mbps 2 Mbps

---

**Graph Details:**
- **x-axis:** 1/error rate
- **y-axis:** base TCP, dupack delay 80ms + LL Retransmit, Only LL retransmit
- **Data Points:**
  - 16384
  - 32768
  - 65536
  - 1E+05

---

**Diagram Notes:**
- 20 ms delay
- 2 Mbps wireless link
- No congestion losses
Delayed Dupacks [Vaidya99]

5% packet loss due to congestion
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.