Transport Layer


Reference: Computer Networks: A Systems Approach. Larry Peterson, Bruce Davie, Morgan Kaufmann
Transport Layer

- Multiplexing and Demultiplexing
- Connectionless Transport: UDP
- Reliable Data Transfer
- Connection-oriented Transport: TCP
  - Segment Structure
  - Reliable Data Transfer
  - Flow Control
  - Connection Management

✓ Congestion Control
- TCP Congestion Control
Principles of Congestion Control

Congestion:
• Informally: Too many sources sending too much data too fast for network to handle.

• Different from flow control!

• Manifestations:
  • Lost packets (buffer overflow at routers)
  • Long delays (queueing in router buffers)

• A top-10 problem!
Causes & Costs of Congestion: First Scenario

- Two senders, two receivers
- One router, infinite buffers
- Output link capacity: $R$
- No retransmission
Causes & Costs of Congestion: First Scenario

• Maximum per-connection throughput: $\frac{R}{2}$

• Large delays as arrival rate, $\lambda_{in}$, approaches capacity
Causes & Costs of Congestion: Second Scenario

• One router, **finite** buffers
• Sender retransmission of timed-out packet
• Application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$
• Transport-layer input includes **retransmissions**: $\lambda'_{in} \geq \lambda_{in}$
Causes & Costs of Congestion: Second Scenario

Idealization: perfect knowledge

• Sender sends only when router buffers available
Causes & Costs of Congestion: Second Scenario

- **Idealization: Known loss**
  - Packets can be lost, dropped at router due to full buffers
- Sender only resends if packet **known** to be lost
Causes & Costs of Congestion: Second Scenario

When sending at R/2, some packets are retransmissions but asymptotic goodput is still R/2 (why?)

\[ \lambda_{\text{out}} \]

\[ \lambda_{\text{in}} \]

\[ \lambda_{\text{in}}^{\prime} \] : original data, plus retransmitted data

\[ \lambda_{\text{out}} \]

free buffer space!
Causes & Costs of Congestion: Second Scenario

Realistic: duplicates

- Packets can be lost, dropped at router due to full buffers
- Sender times out prematurely, sending two copies, both of which are delivered

\[
\lambda_{\text{out}} \quad \lambda_{\text{in}} \quad \frac{R}{2}
\]

- when sending at \( \frac{R}{2} \), some packets are retransmissions including duplicated that are delivered!
Causes & Costs of Congestion: Second Scenario

Costs of congestion:

• More work (retransmit) for given goodput

• Unneeded retransmissions: link carries multiple copies of packet
  • Decreasing goodput
Causes & Costs of Congestion: Third Scenario

- Four senders
- Multi-hop paths
- Timeout/retransmit

Q: what happens as $\lambda_{in}$ and $\lambda_{in}'$ increase?
A: as red $\lambda_{in}'$ increases, all arriving blue pkts at upper queue are dropped, blue throughput $\rightarrow 0$
Another cost of congestion

• When packet dropped, any upstream transmission capacity used for that packet was wasted!
Transport Layer

• Multiplexing and Demultiplexing
• Connectionless Transport: UDP
• Reliable Data Transfer
• Connection-oriented Transport: TCP
  • Segment Structure
  • Reliable Data Transfer
  • Flow Control
  • Connection Management

• Congestion Control
✓ TCP Congestion Control
TCP Congestion Control: AIMD

- **Approach:** sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs

  - **Additive Increase**
    - Increase $cwnd$ by 1 MSS every RTT until loss detected

  - **Multiplicative Decrease**
    - Cut $cwnd$ in half after loss
TCP Congestion Control: AIMD

AIMD saw tooth behavior: probing for bandwidth

**Additively increase window size ... until loss occurs (then cut window in half)**
TCP Congestion Control: Details

• Sender limits transmission

\[ \text{LastByteSent - LastByteAcked} \leq \text{cwnd} \]

\[ \text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec} \]

cwnd is dynamic, function of perceived network congestion

TCP sending rate:

• **Roughly:** send cwnd bytes, wait RTT for ACKS, then send more bytes
TCP Slow Start

• When connection begins
  increase rate exponentially until first loss event
  • Initially $cwnd = 1$ MSS
  • Double $cwnd$ every RTT
  • Done by incrementing $cwnd$ for every ACK received

• Summary: initial rate is slow but ramps up exponentially fast
TCP: Detecting & Reacting to Loss

• Loss indicated by timeout:
  • \texttt{cwnd} set to 1 MSS;
  • Window then grows exponentially (as in slow start) to threshold, then grows linearly

• Loss indicated by 3 duplicate ACKs: TCP RENO
  • Duplicate ACKs indicate network capable of delivering some segments
  • \texttt{cwnd} is cut in half window then grows linearly

• TCP Tahoe always sets \texttt{cwnd} to 1 (timeout or 3 duplicate acks)
TCP: Switching from Slow Start to CA

• **Q:** When should the exponential increase switch to linear?

• **A:** when \( cwnd \) gets to \( \frac{1}{2} \) of its value before timeout.

• **Implementation**
  
  • Variable \( ss\text{thresh} \)
  
  • On loss event, \( ss\text{thresh} \) is set to 1/2 of \( cwnd \) just before loss event
TCP Congestion Control

- **Start:**
  - $\text{cwnd} = 1 \text{ MSS}$
  - $\text{dupACKCount} = 0$
  - $\Lambda$

- **Fast Recovery:**
  - $\text{dupACKCount} = 3$
  - $\text{sssthresh} = \text{cwnd}/2$
  - $\text{cwnd} = \text{sssthresh} + 3$
  - Retransmit missing segment

- **Congestion Avoidance:**
  - $\text{cwnd} = \text{sssthresh} / 2$
  - $\text{dupACKCount} = 0$
  - $\Lambda$

- **New ACK:**
  - $\text{cwnd} = \text{cwnd} + \text{MSS} \times \left( \frac{\text{MSS}}{\text{cwnd}} \right)$
  - $\text{dupACKCount} = 0$
  - Transmit new segment(s), as allowed

- **Duplicate ACK:**
  - $\text{dupACKCount}++$

- **Timeout:**
  - $\text{sssthresh} = \text{cwnd}/2$
  - $\text{cwnd} = 1 \text{ MSS}$
  - $\text{dupACKCount} = 0$
  - Retransmit missing segment
TCP Throughput

• Average TCP throughput as function of window size, RTT?
  • Ignore slow start, assume always data to send

\[
\text{avg TCP throughput} = \frac{3}{4} \frac{W}{\text{RTT}} \text{ bytes/sec}
\]
TCP Fairness

- **Fairness goal**: if $K$ TCP sessions share the same bottleneck link of bandwidth $R$, each should have an average rate of $R/K$. 
WHY TCP is Fair?

Two competing sessions:
- Additive increase gives slope of 1, as throughout increases
- Multiplicative decrease decreases throughput proportionally
Fairness

Fairness and UDP

- Multimedia applications often do not use TCP: Do not want rate throttled by congestion control
- Instead use UDP: Send audio/video at constant rate, tolerate packet loss

Fairness, parallel TCP connections

- Application can open multiple parallel connections between two hosts
- Web browsers do this
- Example, link of rate $R$ with 9 existing connections:
  - New applications asks for 1 TCP, gets rate: $R/10$
  - New applications asks for 11 TCPs, gets: $R/2$
Explicit Congestion Notification

**Network-assisted congestion control:**

- Two bits in IP header (ToS field) marked by network router to indicate congestion
- Congestion indication carried to receiving host
- Receiver (seeing congestion indication in IP datagram) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion
Acknowledgements

The following materials have been used in preparation of this slide set:

   7th Edition
   James Kurose, Keith Ross
   Pearson
   2016

   5th Edition
   Larry Peterson, Bruce Davie
   Morgan Kaufmann
   2011