The Transport Layer
Congestion control in TCP

Jasleen Kaur

April 2, 2019

Transport Layer Protocols & Services

Outline

◆ Fundamental transport layer services
  » Multiplexing/Demultiplexing
  » Error detection
  » Reliable data delivery
  » Pipelining
  » Flow control
  » Congestion control

◆ Internet transport protocols
  » UDP
  » TCP
Congestion Control

Congestion control v. Flow control

- In flow control, the sender adjusts its transmission rate so as not to overwhelm the receiver.
  - One source is sending data too fast for a receiver to handle.

- In congestion control, the sender(s) adjust their transmission rate so as not to overwhelm routers in the network.
  - Many sources independently work to avoid sending too much data too fast for the network to handle.

- Symptoms of congestion:
  - Lost packets (buffer overflow at routers)
  - Long delays (queuing in router buffers)

The Causes and Effects of Congestion

Scenario 1: Two equal-rate senders share a single link

Two sources send as fast as possible to two receivers across a shared link with capacity $R$.
- Data is delivered to the application at the receiver at rate $\lambda_{out}$.
- Packets queue at the router.
  - Assume the router has infinite storage capacity.
  - (Thus no packets are lost and there are no retransmissions.)
The Causes and Effects of Congestion
Scenario 1: Two equal-rate senders share a single link

- The maximum achievable per connection throughput is constrained by \( \frac{1}{2} \) the capacity of the shared link
- Exponentially large delays are experienced when the router becomes congested
  » The queue grows without bound

The Causes and Effects of Congestion
Scenario 2: Finite capacity router queue

- Assume packets can now be lost
  » Sender retransmits upon detection of loss
- Define offered load as the original transmissions plus retransmissions
  » \( \lambda'_{\text{in}} = \lambda_{\text{in}} + \lambda_{\text{retransmit}} \)
The Causes and Effects of Congestion

Scenario 2: Throughput analysis

- By definition $\lambda_{out} = \lambda_{in}$
- Retransmission scenarios:
  - "Perfect" — Retransmissions occur only when there is loss
  - Premature — Delayed packets are retransmitted
  - $\lambda_{out} = \text{"goodput"}$
TCP Congestion Control
Congestion window and transmission rate

- If \( w \times \frac{MSS}{R} < RTT \), then the maximum rate at which a TCP connection can transmit data is

\[
\frac{w \times MSS}{RTT} \text{ bytes/sec}
\]

- \( w \) is the minimum of the number of segments in the receiver's window or the congestion window

TCP Congestion Control
Congestion window control

- TCP connections probe for available bandwidth
  » Increase the congestion window until loss occurs
  » When loss is detected decrease window, then begin probing (increasing) again
- The congestion window grows in two phases:
  » Slow start — Ramp up transmission rate until loss occurs
  » Congestion avoidance — Keep connection close to sustainable bandwidth
- A window size “threshold” distinguishes between slow start and congestion avoidance phases
TCP Congestion Control

**Slowstart**

- Exponential increase in window size each RTT until:
  - Loss occurs
  - $cong\text{Win} = \text{threshold}$
  (Not so slow!)

- Note: TCP implementations detect loss using:
  - Timeout or three duplicate ACKs

**Congestion avoidance**

- Increase congestion window by 1 segment each RTT, decrease by a factor of 2 when packet loss is detected
  - "Additive Increase, Multiplicative Decrease" (AIMD)
TCP Congestion Control

Slow-start v. Congestion avoidance

- The threshold is an estimate of a “safe” level of throughput that is sustainable in the network
  » The threshold specifies a throughput that was sustainable in the recent past

- Slow-start quickly increases throughput to this threshold

- Congestion avoidance slows probes for additional available bandwidth beyond the threshold

Assume \( RTT > \frac{w \times MSS}{R} \)

Initial Threshold is 1 MB = 700 segments

TCP Congestion Control

Slow-start v. Congestion avoidance

- Loss (at any time) reduces the “safe” throughput estimate to 1/2 of the current throughput
  » This is the throughput that resulted in loss

- Slow-start begins anew whenever there is loss

- Throughput at initial threshold = 1 MB/RTT
  » At 1st threshold: 16MSS/RTT
  » At 2nd threshold: 10MSS/RTT

Assume \( RTT > \frac{w \times MSS}{R} \)
TCP Congestion Control

**Major TCP Variants**

- **TCP Tahoe:**
  - Loss signaled by timeout
  - \(\text{threshold} = \frac{\text{congWin}}{2}\)
  - \(\text{congWin} = 1\ \text{MSS}\)

- **TCP Reno:**
  - "Fast retransmit" — Receipt of 3 duplicate ACKs also signals a packet loss
  - "Fast recovery" — Skips slowstart and continues in congestion avoidance new slowstart threshold

- **Others:** TCP NewReno, SACK, ...

![Diagram of TCP Congestion Control](image)

**TCP Congestion Control**

**Summary (TCP Reno)**

- \(\text{cwnd} \geq \text{ssthresh}\)
- \(\text{cwnd} = \text{cwnd} + \text{MSS} \times \left(\frac{\text{MSS}}{\text{cwnd}}\right)\)
- \(\text{dupACKcount} = 0\)
- \(\text{new ACK}\)
- \(\text{cwnd} = \text{cwnd} + \text{MSS}\)
- \(\text{dupACKcount} = 0\)
- \(\text{xmit new segs, as allowed}\)
- \(\text{cwnd} = \frac{\text{cwnd}}{2}\)
- \(\text{cwnd} = 1\ \text{MSS}\)
- \(\text{dupACKcount} = 0\)
- \(\text{re-xmit missing seg}\)
- \(\text{dupACKcount} = 3\)
- \(\text{ssthresh} = \text{cwnd}/2\)
- \(\text{cwnd} = \text{ssthresh} + 3\ \text{MSS}\)
- \(\text{re-xmit missing seg}\)
- \(\text{new ACK}\)
- \(\text{cwnd} = \text{cwnd} + \text{MSS}\)
- \(\text{dupACKcount} = 0\)
- \(\text{xmit new segs, as allowed}\)
- \(\text{cwnd} = \frac{\text{cwnd}}{2}\)
- \(\text{cwnd} = 1\ \text{MSS}\)
- \(\text{dupACKcount} = 0\)
- \(\text{re-xmit missing seg}\)
- \(\text{dupACKcount} = 3\)
- \(\text{ssthresh} = \text{cwnd}/2\)
- \(\text{cwnd} = \text{ssthresh} + 3\ \text{MSS}\)
- \(\text{re-xmit missing seg}\)
- \(\text{new ACK}\)
- \(\text{cwnd} = \text{cwnd} + \text{MSS}\)
- \(\text{dupACKcount} = 0\)
- \(\text{xmit new segs, as allowed}\)
Advanced Topics: TCP over “long, fat, pipes”

“High speed TCP”

- Can an end system transmit at 10 Gbps over TCP?
  - Assume 1,500 byte segments and a 100 ms RTT
- 10 Gbps would requires $W = 83,333$ segments (with no loss)
- Throughput in terms of segment loss probability, $L$ is

\[
\text{TCP throughput} = \frac{1.22 \cdot \text{MSS}}{\text{RTT} \cdot \sqrt{L}}
\]

- Thus, to achieve 10 Gbps throughput, we need a loss rate of $L = 2 \cdot 10^{-10}$ (a crazy small loss rate!)
- For these reasons, new versions of TCP for “high-speed” networks exist
  - Beyond a certain window size, the window grows faster each RTT and decreases less on a loss

Approaches to Congestion Control

End-to-end v. Hop-by-hop

- End-to-end congestion control
  - End-systems receive no feedback from network
  - Congestion inferred by observing loss and/or delay
- Hop-by-hop congestion control
  - Routers provide feedback to end systems
    - Network determines an explicit rate that a sender should transmit at
    - Network signals congestion by setting a bit in a packet’s header (SNA, DECbit, TCP/IP ECN, ATM)
A router can detect it is becoming congested before packet loss occurs.
ECN allows a router to “mark” a packet to provide an indication of congestion to an end-system.
This enables end-systems to slow down before loss occurs.

Two bits in IP header (ToS field) can be set by a router to indicate “early” congestion.
This indication is carried to receiving host.
The receiver detects the congestion indication in an arriving datagram and sets the ECN-Echo (ECE) bit on the next ACK to notify the sender of congestion.
When the source receives the ECE indication it reduces its congestion window as for a packet drop.

- The source acknowledges the congestion indication by sending a segment with the congestion window reduced (CWR) bit set.

- The receiver keeps transmitting ACKs with the ECE bit set until it receives a segment with the CWR bit set.