Congestion Control

Srinidhi Varadarajan

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 (queuing in router buffers)
- a top-10 problem!

Causes/costs of congestion: scenario 1
- two senders, two receivers
- one router, infinite buffers
- no retransmission

Causes/costs of congestion: scenario 2
- one router, finite buffers
- sender retransmission of lost packet

Causes/costs of congestion: scenario 2
- always: \( \lambda_{in} = \lambda_{out} \) (goodput)
- “perfect” retransmission only when loss \( \lambda_{in} > \lambda_{out} \)
- retransmission of delayed (not lost) packet makes \( \lambda_{in} \) larger (than perfect case) for same \( \lambda_{out} \)

Causes/costs of congestion: scenario 3
- four senders
- multihop paths
- timeout/retransmit

Q: what happens as \( \lambda_{in} \) and \( \lambda_{out} \) increase?
Causes/costs of congestion: scenario 3

Another “cost” of congestion:
- when packet dropped, any “upstream transmission capacity used for that packet was wasted!

Approaches towards congestion control

Two broad approaches towards congestion control:

End-end congestion control:
- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

Network-assisted congestion control:
- routers provide feedback to end systems
  - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
  - explicit rate sender should send at

Case study: ATM ABR congestion control

ABR: available bit rate:
- “elastic service”
- if sender’s path “underloaded”:
  - sender should use available bandwidth
- if sender’s path congested:
  - sender throttled to minimum guaranteed rate

RM (resource management) cells:
- sent by sender, interspersed with data cells
- bits in RM cell set by switches (“network-assisted”)
  - NI bit: no increase in rate (mild congestion)
  - CI bit: congestion indication
- RM cells returned to sender by receiver, with bits intact

Case study: ATM ABR congestion control

- two-byte ER (explicit rate) field in RM cell
  - congested switch may lower ER value in cell
  - sender’ send rate thus minimum supportable rate on path
- EFCI bit in data cells: set to 1 in congested switch
  - if data cell preceding RM cell has EFCI set, receiver sets CI bit in returned RM cell

TCP Congestion Control

- end-end control (no network assistance)
- transmission rate limited by congestion window size, Congwin, over segments:

<table>
<thead>
<tr>
<th>send_base</th>
<th>Congwin</th>
<th>send</th>
<th>not send</th>
<th>MSS</th>
<th>already sent</th>
<th>Congwin</th>
<th>sent, not yet sent</th>
<th>not usable</th>
<th>usable, not yet sent</th>
</tr>
</thead>
</table>

- w segments, each with MSS bytes sent in one RTT:

  throughput = \( \frac{w \times MSS}{RTT} \) Bytes/sec

TCP congestion control:

- “probing” for usable bandwidth:
  - ideally: transmit as fast as possible (Congwin as large as possible) without loss
  - increase Congwin until loss (congestion)
  - loss: decrease Congwin, then begin probing (increasing) again

- two “phases”
  - slow start
  - congestion avoidance

- important variables:
  - Congwin
  - threshold: defines threshold between two slow start phase, congestion control phase
TCP Slowstart

**Slowstart algorithm**
- initialize: Congwin = 1
- for (each segment ACKed)
- Congwin++
- until (loss event OR CongWin > threshold)

- exponential increase (per RTT) in window size (not so slow!)
- loss event: timeout (Tahoe TCP) and/or three duplicate ACKs (Reno TCP)

TCP Congestion Avoidance

**Congestion avoidance**

- */ slowstart is over */
- */ Congwin > threshold */
- Until (loss event) {
  - every w segments ACKed: Congwin++
- }

- threshold = Congwin/2
- Congwin = 1
- perform slowstart¹

¹: TCP Reno skips slowstart (fast recovery) after three duplicate ACKs

AIMD

TCP congestion avoidance:

- **AIMD**: additive increase, multiplicative decrease
  - increase window by 1 per RTT
  - decrease window by factor of 2 on loss event

TCP Fairness

**Fairness goal**: if N TCP sessions share same bottleneck link, each should get 1/N of link capacity

Why is TCP fair?

Two competing sessions:

- Additive increase gives slope of 1, as throughout increases
- Multiplicative decrease decreases throughput proportionally

TCP latency Modeling

**Q**: How long does it take to receive an object from a Web server after sending a request?

- TCP connection establishment
- Data transfer delay

**Notation, assumptions**:

- Assume one link between client and server of rate R
- Assume: fixed congestion window, W segments
- S: MSS (bits)
- O: object size (bits)
- no retransmissions (no loss, no corruption)

**Two cases to consider**:

- $WS/R > RTT + S/R$: ACK for first segment in window returns before window’s worth of data sent
- $WS/R < RTT + S/R$: wait for ACK after sending window’s worth of data sent

**Case 1**: latency = $2RTT + O/R$

**Case 2**: latency = $2RTT + O/R + (K-1)[S/R + RTT - WS/R]$

K := O/WS

TCP latency Modeling
TCP Latency Modeling: Slow Start

- Now suppose window grows according to slow start.
- Will show that the latency of one object of size $O$ is:

$$
\text{Latency} = 2RTT + \frac{O}{R} + P \left[ RTT + \frac{S}{R} \right] \left(2^P - 1\right) \frac{S}{R}
$$

where $P$ is the number of times TCP stalls at server:

$$
P = \min(Q, K - 1)
$$

- where $Q$ is the number of times the server would stall if the object were of infinite size.
- and $K$ is the number of windows that cover the object.

Example:

- $O/S = 15$ segments
- $K = 4$ windows
- $Q = 2$
- $P = \min(K-1, Q) = 2$

Server stalls $P=2$ times.

TCP Latency Modeling: Slow Start (cont.)

$$
\text{latency} = \frac{O}{R} + 2RTT + \frac{S}{R} + P(2RTT - 2 + \frac{S}{R} (2^P - 1)) \frac{S}{R}
$$

$\text{RTT}$ = time from when server starts to send segment until server receives acknowledgement

$\text{S/R}$ = time to transmit the kth window

$\text{2S/R}$ = stall time after the kth window

$\text{3S/R}$ = stall time after the kth window

$\text{4S/R}$ = complete transmission

$\text{5S/R}$ = time at client

$\text{6S/R}$ = time at server