CS 78 Computer Networks

Congestion Control

Andrew T. Campbell
campbell@cs.dartmouth.edu

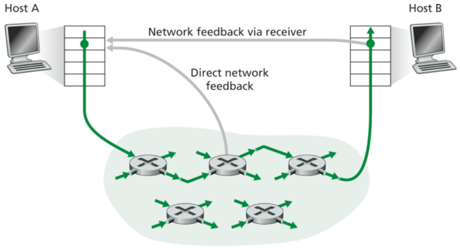
What is congestion and why is it an important problem for Internet?

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)
- Can be a serious problem

How does the source determine congestion?



Two approaches towards congestion control - what’s the tradeoffs?

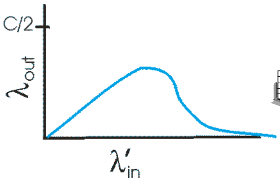
End-end congestion control

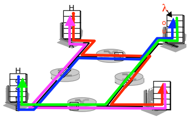
- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

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

Congestion Scenarios

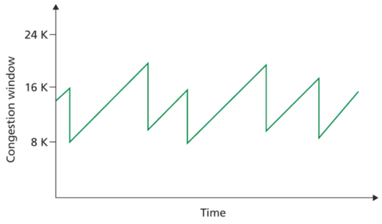




Another “cost” of congestion:

- when packet dropped, any “upstream transmission capacity used for that packet was wasted!

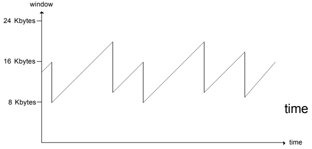
TCP's end-to-end approach AIMD (Additive Increase, Multiplicative Decrease) Algorithm



TCP congestion control: additive increase, multiplicative decrease

- Approach: increase transmission rate (window size), probing for usable bandwidth, until loss occurs
  - additive increase: increase **CongWin** by 1 MSS every RTT until loss detected
  - multiplicative decrease: cut **CongWin** in half after loss

Saw tooth behavior: probing for bandwidth



TCP Congestion Control

- Sender limits transmission:
  `LastByteSent-LastByteAcked ≤ CongWin`
- Roughly,

$$\text{rate} = \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

- `CongWin` is dynamic, function of perceived network congestion

How does sender perceive congestion?

- loss event = timeout *or* 3 duplicate acks
- TCP sender reduces rate (`CongWin`) after loss event

Three mechanisms:

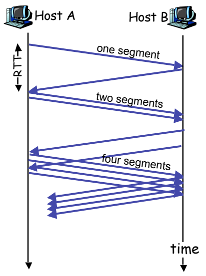
- AIMD
- slow start
- conservative after timeout events

TCP Slow Start

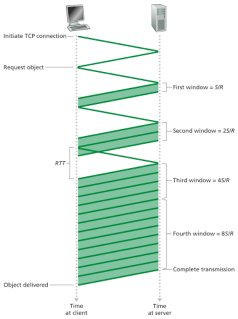
- When connection begins, `CongWin` = 1 MSS
  - Example: MSS = 500 bytes & RTT = 200 msec
  - initial rate = 20 kbps
- available bandwidth may be >> MSS/RTT
  - desirable to quickly ramp up to respectable rate
- When connection begins, increase rate exponentially fast until first loss event

TCP Slow Start (more)

- When connection begins, increase rate exponentially until first loss event:
  - double `CongWin` every RTT
  - done by incrementing `CongWin` for every ACK received
- Summary: initial rate is slow but ramps up exponentially fast



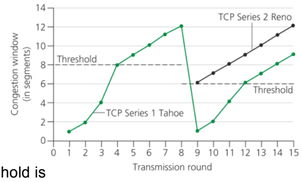
TCP timing during slow start



Refinement

Q: When should the exponential increase switch to linear?

A: When **`CongWin`** gets to 1/2 of its value before timeout.



Implementation:

- Variable Threshold
- At loss event, Threshold is set to 1/2 of CongWin just before loss event

Inferring loss

- After 3 dup ACKs:
  - `CongWin` is cut in half
  - window then grows linearly
- But after timeout event:
  - `CongWin` instead set to 1 Max Seg Size (MSS);
  - window then grows exponentially
  - to a threshold, then grows linearly

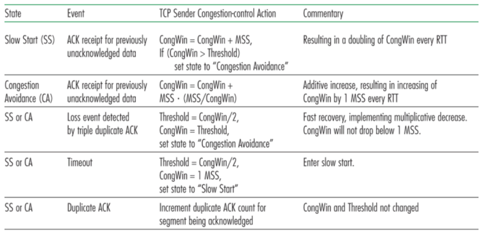
• 3 dup ACKs indicates network capable of delivering some segments

•timeout indicates a “more alarming” congestion scenario

TCP Congestion Control

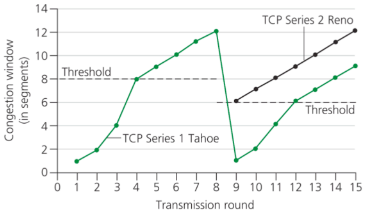
- When **`CongWin`** is below **`Threshold`**, sender in slow-start phase, window grows exponentially.
- When **`CongWin`** is above **`Threshold`**, sender is in congestion-avoidance phase, window grows linearly.
- When a triple duplicate ACK occurs, **`Threshold`** set to **`CongWin/2`** and **`CongWin`** set to **`Threshold`**.
- When timeout occurs, **`Threshold`** set to **`CongWin/2`** and **`CongWin`** is set to 1 MSS.

TCP sender congestion control



State	Event	TCP Sender Congestion-control Action	Commentary
Slow Start (SS)	ACK receipt for previously unacknowledged data	CongWin = CongWin + MSS, If (CongWin > Threshold) set state to “Congestion Avoidance”	Resulting in a doubling of CongWin every RTT
Congestion Avoidance (CA)	ACK receipt for previously unacknowledged data	CongWin = CongWin + MSS · (MSS/CongWin)	Additive increase, resulting in increasing of CongWin by 1 MSS every RTT
SS or CA	Loss event detected by triple duplicate ACK	Threshold = CongWin/2, CongWin = Threshold, set state to “Congestion Avoidance”	Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.
SS or CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, set state to “Slow Start”	Enter slow start.
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acknowledged	CongWin and Threshold not changed

Congestion control’s evolution

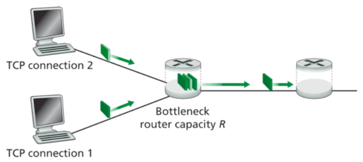


TCP throughput

- What’s the average throughout of TCP as a function of window size and RTT?
  - Ignore slow start
- Let W be the window size when loss occurs.
- When window is W, throughput is W/RTT
- Just after loss, window drops to W/2, throughput to W/2RTT.
- Average throughout: .75 W/RTT

TCP: The fairness issue

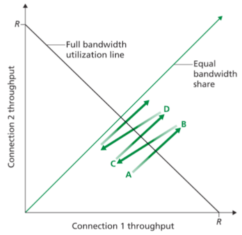
if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



Why is TCP fair?

Two competing sessions:

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



Fairness

Fairness and UDP

- Multimedia apps often do not use TCP
  - do not want rate throttled by congestion control
- Instead use UDP:
  - pump audio/video at constant rate, tolerate packet loss
- Research area: TCP friendly

Fairness and parallel TCP connections

- nothing prevents app from opening parallel connections between 2 hosts.
- Web browsers do this
- Example: link of rate R supporting 9 cnctions;
  - new app asks for 1 TCP, gets rate R/10
  - new app asks for 11 TCPs, gets R/2 !