Wide-Area Congestion Control

Lecture 19, Computer Networks (198:552)
Fall 2019
Review: TCP congestion control

• Keep some in-flight (un-ACK’ed) packets: congestion window

• Adjust window based on several algorithms:
  • Startup: slow start
  • Steady state: AIMD
  • Loss: fast retransmission, fast recovery

• Window versus rate-based protocols
Queue Dynamics with TCP

Steady-state behavior
Network model

- Single flow
- Packet-switched core network
- Bottleneck queue (max size B)
- Queuing delay
- Propagation delay $RT_{prop}$
- Link rate C
Sender behavior at steady state

- Congestion avoidance: Additive increase, multiplicative decrease (AIMD)
- Steady state isn’t static: lose pkts, grow cwnd, lose pkts, …

![Graph showing congestion window over time with loss events and window halving.](image-url)
Sender behavior at steady state

- How does the queue size at the bottleneck look, over time?
  - Case 1: $B = C \times RT_{prop}$
  - Case 2: $B > C \times RT_{prop}$
  - Case 3: $B < C \times RT_{prop}$
Network model

A few flows (say 3—4)

Q: how does queue size look now?
Network model

Many flows (say hundreds)

Queuing delay

Propagation delay $RT_{prop}$

Bottleneck queue (max size $B$)

Packet-switched core network

Q: how does queue size look now?
How big should router buffers be?

• Classic buffer-sizing rule: $B = C \times RT_{prop}$
  • BDP buffer
  • Single TCP flow halving its window still gets a throughput of 100% link rate

• Q: should buffers be BDP-sized?

• Significant implications:
  • Massive pkt buffers (e.g., 40 Gbit/s with 200ms $RT_{prop}$): high cost
  • Massive pkt delays: bufferbloat
TCP BBR
1. Estimate the bottleneck link rate $C$
2. Estimate the propagation delay $RT_{prop}$
3. Send at rate $C$ with at most $k \times C \times RT_{prop}$ packets in flight
(1) Estimating the bottleneck link rate

- Data can’t be delivered to a receiver faster than the bottleneck link rate

- Measure the **data delivery rate**
  - And use the maximum value over the recent past
  - Important: measurements time out after a certain period
  - Occasionally send higher (PROBE_BW cycling) to see if changed

- Q: how would you measure delivery rate at the receiver?

- Q: how would you measure delivery rate at the sender?
Measuring delivery rate at the sender

Data that is unACKed at the time of transmitting packet

Normal case: All that data (and only that data) is ACKed by this point

unACKed data at pkt transmit time

Round trip time between pkt-ACK

Packets

ACKs
Quirk: Often, ACKs are “aggregated”

More data appears to be in flight than there actually is.

Idea: use minimum of sent rate and received rate.

Q: how would you measure the rate at which data was sent? (Note: packets of received data and sent data must be the same)
(2) Estimating RT$_{prop}$

- Use the minimum of the RTT values experienced so far
- If you’re sending at high rate, it is difficult to see the true RT$_{prop}$ of the path
  - Q: why?
- Occasionally send just a few packets in an RTT to measure RT$_{prop}$ (PROBE_RTT cycling)
- Also allows achieving fairness among BBR flows
Issues specific to wide-area
The Internet: Many things to consider…

- Bufferbloat
- Token-bucket policers
- Cellular base station scheduling
- Sometimes compete with few streams, sometimes many
- Delayed and aggregated ACKs (WiFi)
- Coexisting with legacy protocols (e.g., Cubic)