Google @ UVic | Fall 2022

Building Your Technical Career:

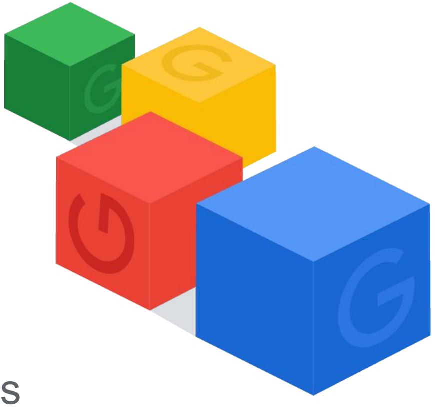
- Learn how to make the most of your degree
- Prepare for a career in software engineering

Interview Prep Workshop:

- Watch a mock interview
- Solve sample problems as a group

STEP Resume Workshop (Tech):

- Learn about Google´s STEP internship for 1st/2nd year students
- Get resume tips on how to best present your skills



Hosted by Google Software Engineers **Ian Sutton** and **Faesal Murad**

Google @ UVic | Fall 2022

Tuesday, October 18th

- **Building Your Technical Career**: *ECS 660* | 5pm - 6:30pm

Wednesday, October 19th

- **Office hours for 1st/2nd year students**: *ECS 223 & ECS 227* | 1pm - 3pm

Thursday, October 20th

- **Interview Prep Workshop**: *ECS 660* | 12pm - 1pm
- **STEP Resume Workshop (Tech)**: *ECS 660* | 5:30pm - 6:30pm

Friday October 21st:

- **Virtual office hours**: *Online* | 12pm - 1pm

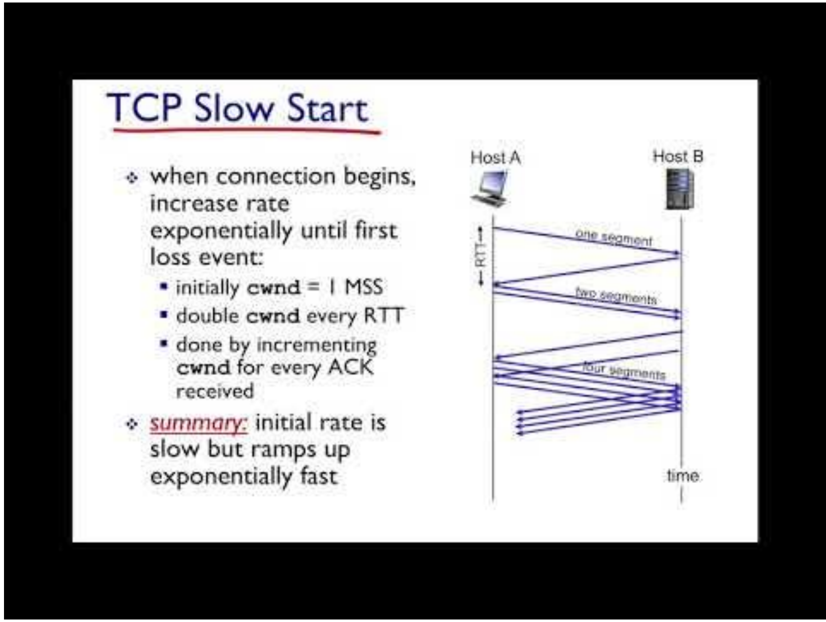
Please RSVP at https://goo.gle/UVictoriaFall2022Events

Check out more career-related content online at https://careersonair.withgoogle.com/

Computer Networks

TCP Congestion Control

Jianping Pan
Fall 2022



TCP Slow Start

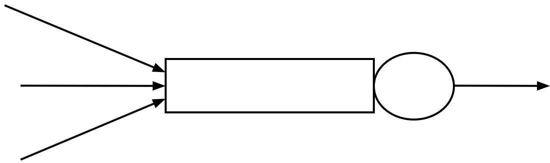
- when connection begins, increase rate exponentially until first loss event:
  - initially **cwnd** = 1 MSS
  - double **cwnd** every RTT
  - done by incrementing **cwnd** for every ACK received
- *summary:* initial rate is slow but ramps up exponentially fast

Review: TCP mechanisms

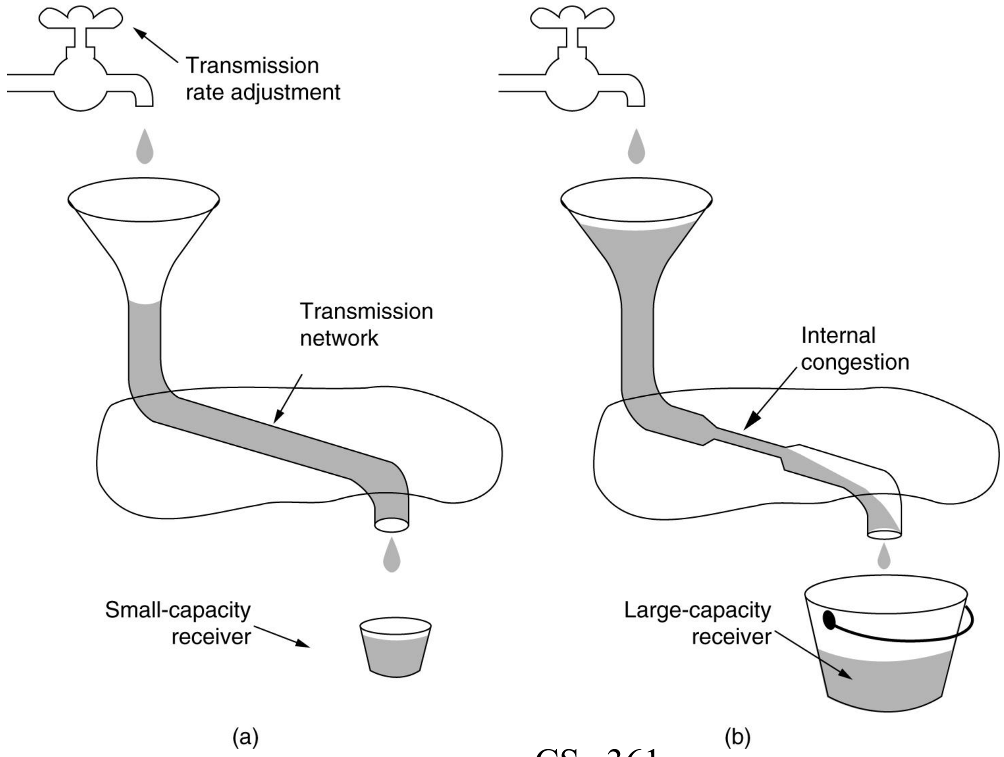
- Connection management
  - packet handshake
- Flow control
  - sliding variable window
- Error control
  - error detection
  - error recovery

Why congestion control?

- Flow control
  - coordinate sender and receiver (buffer)
- Network congestion
  - coordination between the sender and network
    - avoid a sender to overflow a router
  - coordination among many senders
    - traffic aggregation from many senders
  - congestion syndrome
    - increasing queuing delay, packet drop



Flow vs congestion control



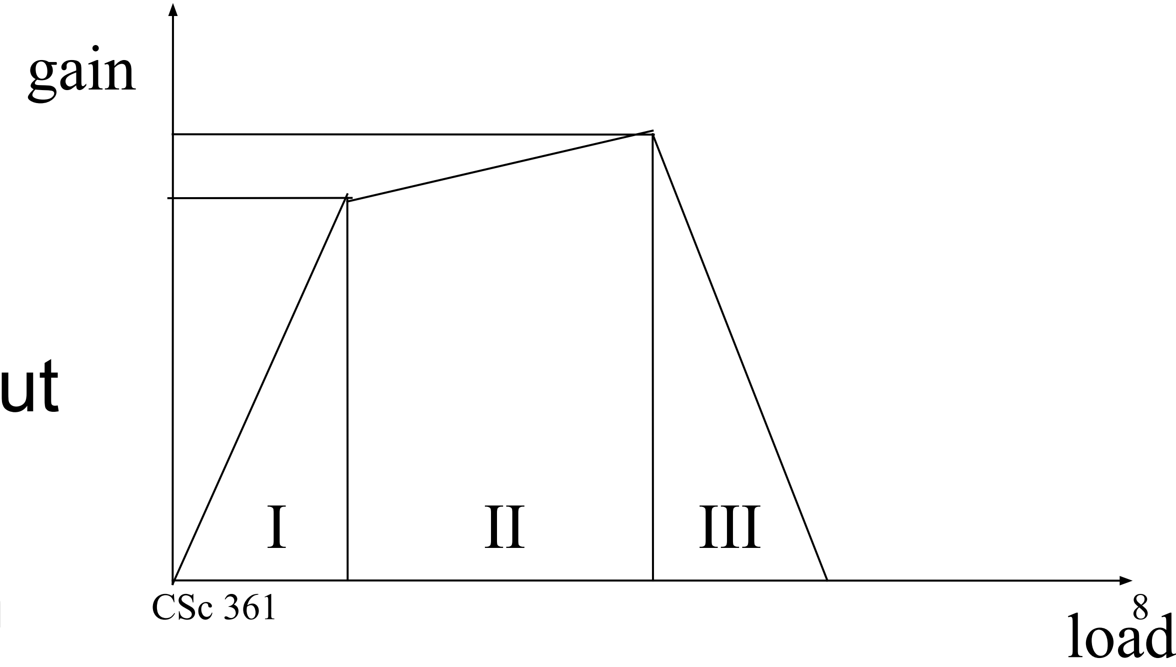
Congestion control **approaches**

- End-to-end approach
  - congestion indicators for the endpoint
    - packet loss
      - also can be caused by transmission error
    - increasing round-trip time
      - also can be caused by alternative routes
- Network-assisted approach
  - ICMP source quenching
  - explicit congestion notification (ECN)

Q: why packet loss and increasing rtt?

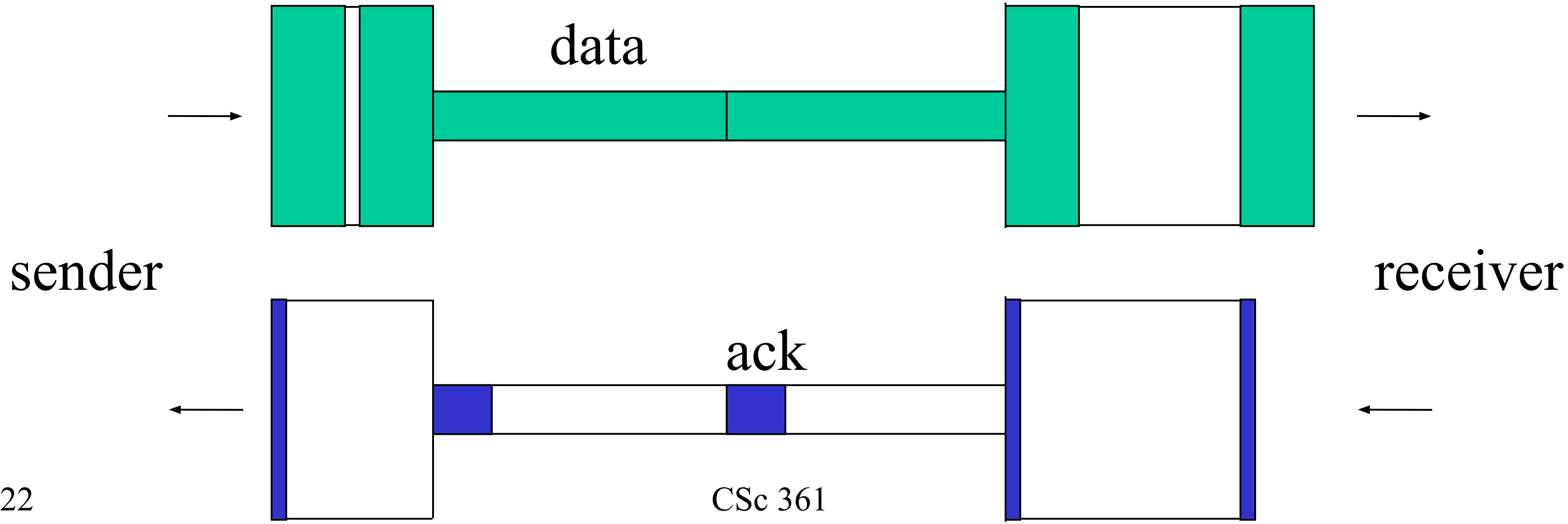
Load-gain curve

- Gain: ~ throughput/delay
- Load-gain curve
  - low load
  - medium load
  - high load
- Congestion
  - low throughput
  - high delay
  - very low gain



Congestion control principles

- Principle: packet conservation in steady state
  - acknowledgment self-clocking



Q: what does ack spacing mean?

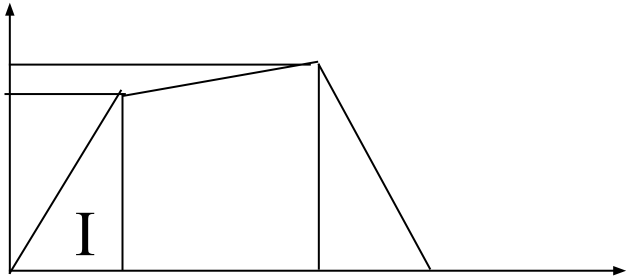
TCP congestion control

- Changes at sender only
  - an add-on to TCP flow control
- Sender variables
  - congestion window (cwnd)
    - sender window = min {…, …, cwnd}
    - initially, cwnd = 1 MSS (maximal segment size)
  - slow-start threshold (ssthresh)
    - initial ssthresh

Q: why sender-only is better?

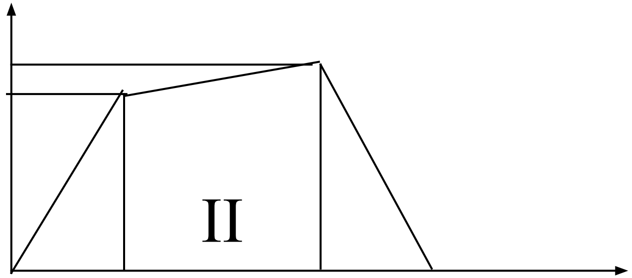
Slow start

- Slow start
  - when cwnd < ssthresh
  - on each new ack
    - cwnd += 1 MSS
    - effectively, doubling cwnd every RTT
  - "start small, but grow really fast"
    - Q: why?



Congestion avoidance

- Congestion avoidance
  - when cwnd > ssthresh
  - on each new ack
    - cwnd += MSS^2/cwnd
    - effectively, cwnd += 1 MSS every RTT
    - linear increment



Q: why increase cwnd to “avoid” congestion?

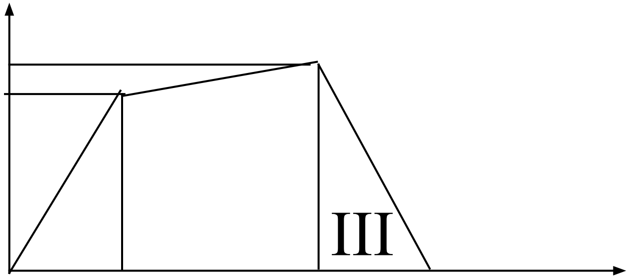
Network congestion

- cwnd is always increased in slow-start and congestion avoidance
  - network congestion is inevitable
- Network congestion indicator
  - TCP treats packet loss as network congestion
- Packet loss indicators
  - acknowledgment timeout
  - 3 duplicate acknowledgments

Q: packet loss == network congestion?

Timeout retransmit

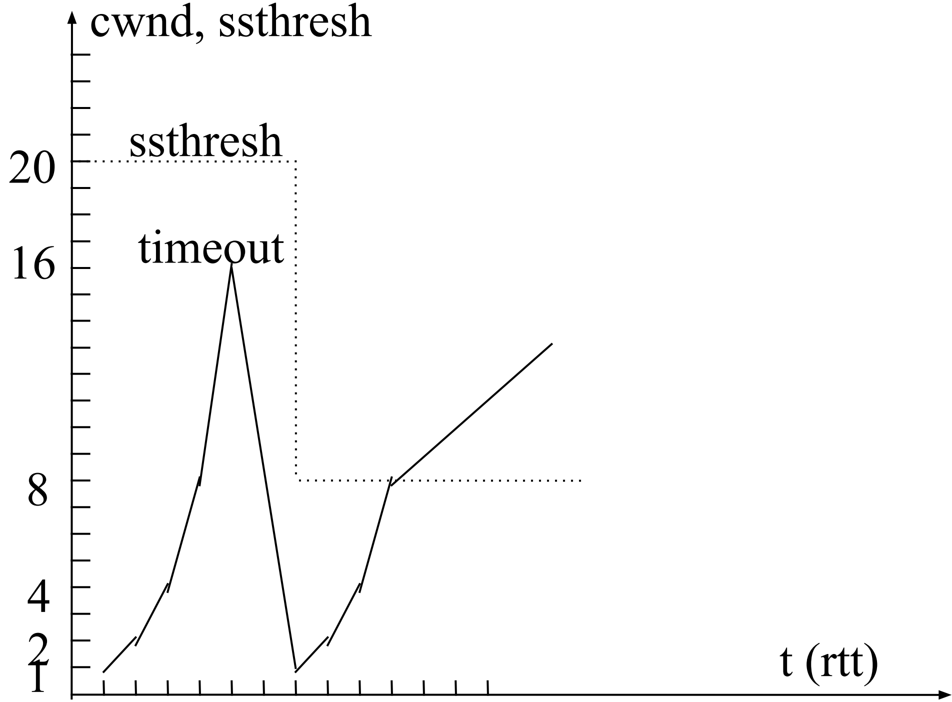
- Timeout
  - RTO = d (SRTT + c RTTV)
- Congestion control
  - ssthresh = cwnd / 2
  - cwnd = 1 MSS
  - followed by slow-start
- Error control
  - retransmit packet
  - backoff timer



Q: how to calculate SRTT and RTTV?

Congestion window

- Slow-start
- Congestion avoidance
- Timeout retransmission
  - TCP timeout is quite conservative
  - pay attention to how ssthresh is adjusted!

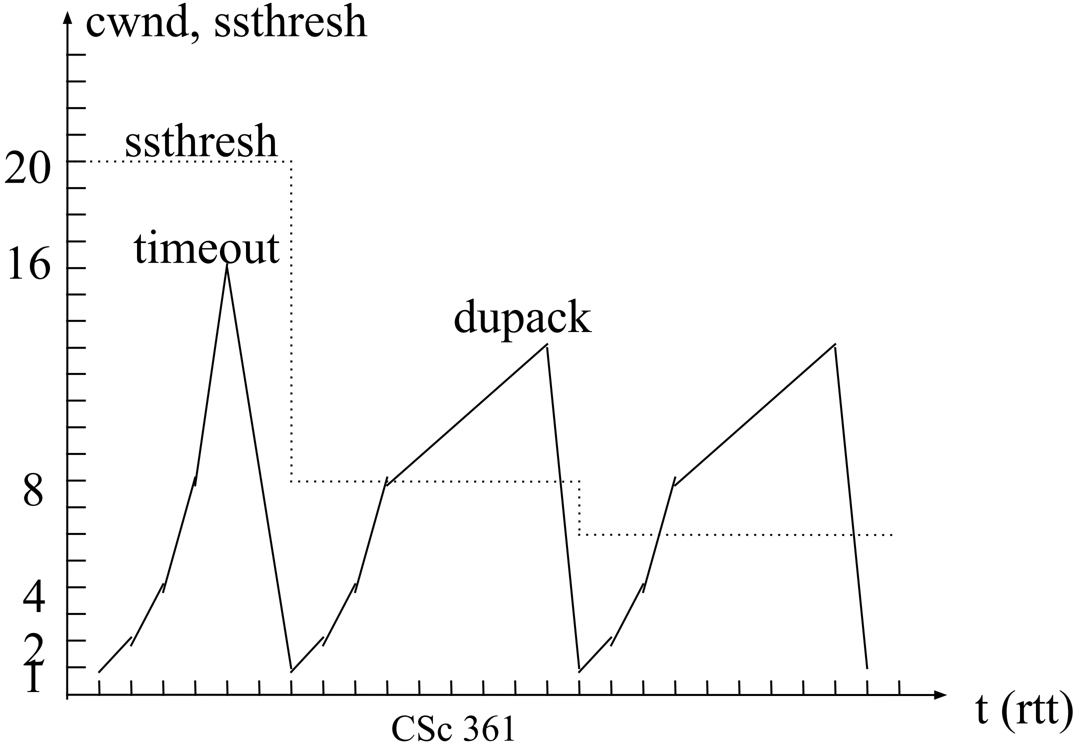


Fast retransmit

- Duplicate acknowledgment
  - example
    - rcv: [0, 499], [500, 999], [1500, 1999], [2000, 2499], [2500, 2999]
    - ack: 500, 1000, 1000, 1000, 1000 (3rd dupack)
- Congestion control (fast retransmit)
  - on 3rd dupack: ssthresh=cwnd/2; cwnd=1 MSS
  - followed by slow start
- Error control
  - retransmit: [1000,1499]

Q: dupack == loss?

Fast retransmit: cwnd



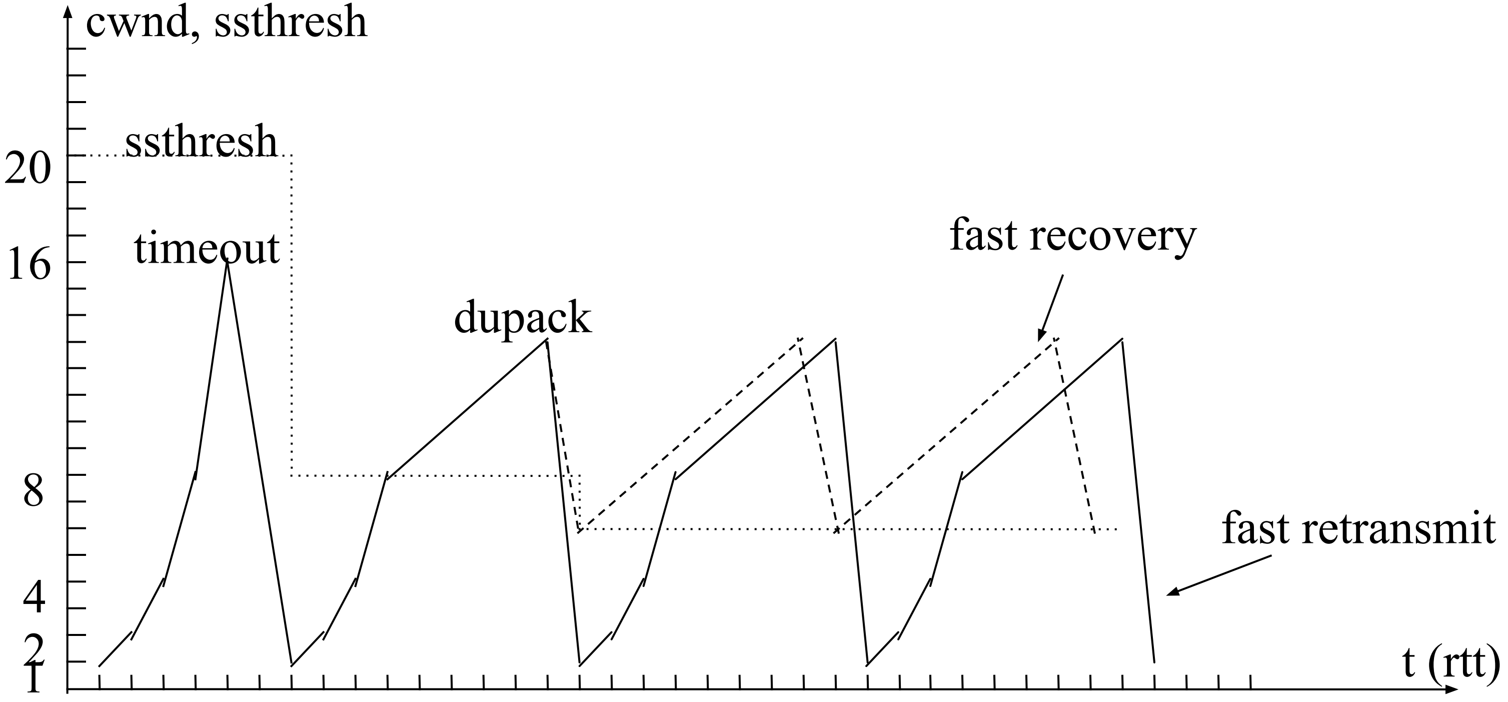
Q: why “fast” retransmit?

Fast recovery

- Duplicate acknowledgment
  - – example
    - rcv: [0, 499], [500, 999], [1500, 1999], [2000, 2499], [2500, 2999]
    - ack: 500, 1000, 1000, 1000, 1000 (3rd dupack)
- Congestion control (fast recovery)
  - – on 3rd dupack: cwnd=ssthresh=cwnd/2
  - – followed by congestion avoidance
- Error control
  - – retransmit: [1000,1499]

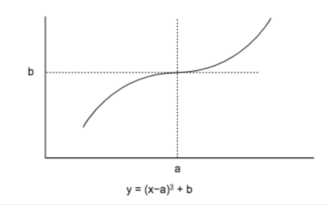
Q: fast transmit vs recovery?

Fast recovery: cwnd



Q: why fast recovery?

TCP congestion control



- Tahoe (packet loss as congestion signal)
  - slow start, cong avoid, timeout, fast retransmit
- Reno (packet loss)
  - slow start, cong avoid, timeout, fast recovery
- Vegas (packet delay): react earlier than loss
- *New*Reno (widely used): cwnd inflation
- SACK (selective ack): more info on holes
- CUBIC in the current Linux, MacOS, Win

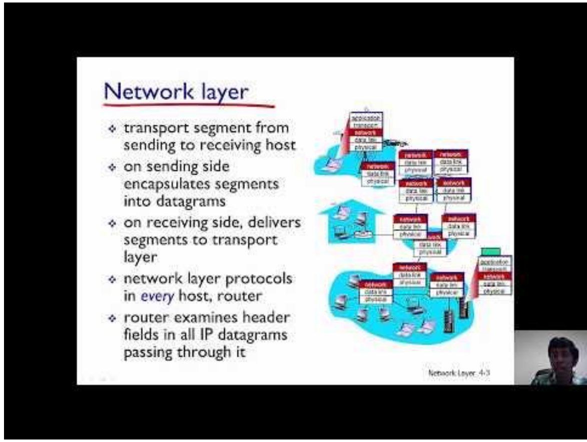
* TCP over wireless (http://www.cs.uvic.ca/~pan/tcpair), TCP in data centers, etc

This lecture

- Congestion control
  - purpose and approach
  - TCP congestion control
    - slow-start, congestion avoidance
    - timeout retransmit
    - fast retransmit, fast recovery
- Explore further
  - TCP congestion control [RFC2581]
  - http://www.cs.uvic.ca/~pan/csc485

Next chapter

- Network layer: IP addressing and routing
  - K&R4: Computer Networking
    - Chapter 4 (except 4.3 and 4.7)
    - if you have a newer edition, it has been split into Chapter 4 (Data Plane) and 5 (Control Plane)
- Next Wednesday (Oct 26)
  - Guest lecture
    - Dr Kaiyang Liu: TCP CUBIC/BBR
      - * UVic visiting PhD, 2016–2018
      - * IEEE TCCLD Best PhD Thesis
      - * UVic Postdoc, since 2019



Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it

Network Layer 4-3