Lecture 23: TCP, Sockets

- TCP
  - Piggybacking, Windowing, timeouts

- Sockets
  - API for interacting with network
TCP (cont.)

A→B: *...
B→A: #...

A: DATA 1
   ACK 1
   DATA 2
   ACK 2
B: DATA 1
   ACK 2
   DATA 3
   ACK 3

Combine (piggybacking)

A: DATA 40
B: DATA 40

ack means "I've received all packets for 0...n"

Z: A has received 0...3

Ack this again

Recall immediately if receive duplicate ACK
Windowing:

- Acknowledgment:
  - Pro: fast
  - Con: must keep track of resend packets

Resending:

- Receiver always re-acks their prefix that they've seen.
- Sender always resends what they think is lost (can be dup), should be ignored.

Data 0

Data 1

Data 2
Timeouts:
want to time out just after you would have received an ACK (RTT) $A \to B$
- timeout too long:
  spend time waiting
- too short: premature
  resends

estimate RTT: every time you receive ACK, measure time since you sent DATA, give an estimate of RTT.

update your estimate of RTT,
new $\text{est} = d \cdot \text{old estimate} + (1-d) \cdot \text{observed RTT}$

$0 \leq d \leq 1$, $d = 0$: no history
$d = 1$: ignore observations.
Establishing/ending connections.

log. stream of data.

[SYN #0]

A | B

B starts stream
B acting as SYN:
A acts B

SYN 
ACK #0

SYN #0

SYN

ACK #0

ACK #0

"three-way handshake: SYN, SYN/ACK, ACK"
Sockets & Multiplexing

- TCP communication endpoints identified by
  
  IP address + port +
  (identifies service)
  e.g. port 80: web traffic (HTTP)

- Multiple clients connect to port 80 of
  server.

- Client connects to port 80 from
  port (say 0,000)

- Server chooses a high-numbered port
  arbitrarily, sends SYN/ACK from
  high port (say 37973) back to client
  port 10,000

- Subsequently, all traffic is on
  port 37973 of server & 10,000 of
  client.

Server socket listens for incoming
connections, creates client sockets.