Lecture 5: 
Flow Control

CSE 123: Computer Networks
Alex C. Snoeren
Pipelined Transmission

- Keep multiple packets “in flight”
  - Allows sender to make efficient use of the link
  - Sequence numbers ensure receiver can distinguish frames

- Sender buffers outstanding un-acked packets
  - Receiver ACKs the highest *consecutive* frame received
    - ACKs are *cumulative* (covers current frame and all previous)

Ignored!
Go-Back-\(N\)

- Retransmit all packets from point of loss
  - Packets sent after loss event are ignored (i.e., sent again)

- Simple to implement (receiver very simple)
- Sender controls how much data is “in flight”
Send Window

- Bound on number of outstanding packets
  - Window “opens” upon receipt of new ACK
  - Window resets entirely upon a timeout

- Limits amount of waste
  - Still lots of duplicates
  - We can do better with selective retransmission

Go-Back-N Example with window size 3
Sliding Window

- Single mechanism that supports:
  - Multiple outstanding packets
  - Reliable delivery
  - In-order delivery
  - Flow control

- Sender and receiver each maintain “window” abstractions to track outstanding packets
  - At the core of all modern ARQ protocols

- Go-Back-N is a special case
  - Receive window size of one
Sliding Window – Sender

- Window bounds outstanding unACKed data
  - Implies need for buffering at sender
- “Last” ACK applies to in-order data
- What to do on a timeout?
  - Go-Back-N: resend all unacknowledged data on timeout
  - Selective Repeat: timer per packet, resend as needed
Sliding Window – Receiver

Receiver: ...

Receive Window

“Last” Received Largest Accepted

- Receiver buffers too:
  - data may arrive out-of-order
  - or faster than can be consumed
    - **Flow control**: tell sender how much buffer left at receiver

- Receiver ACK choices:
  - Cumulative, Selective (exempt missing frames), Negative
Deciding When to Retransmit

- How do you know when a packet has been lost?
  - Ultimately sender uses timers to decide when to retransmit

- But how long should the timer be?
  - Too long: inefficient (large delays, poor use of bandwidth)
  - Too short: may retransmit unnecessarily (causing extra traffic)

- Right timer is based on the round-trip time (RTT)
  - Which can vary greatly for reasons well see later
Can we shortcut the timeout?

- Timeout is long in practice
  - Lots of variation in RTT and timeout must be conservative

- If packets are usually *in order* then *out-of-order* ACKs imply that a packet was lost
  - Negative ACK
    » Receiver requests missing packet
  - Fast retransmit
    » When sender receives multiple duplicate acknowledgements resends missing packet
Fast retransmit

- Don’t bother waiting
  - Receipt of duplicate acknowledgement (dupACK) indicates loss
  - Retransmit immediately

- Used in TCP
  - Need to be careful if frames can be reordered
  - Today’s TCP identifies a loss if there are three duplicate ACKs in a row
Connection Establishment

- Both sender and receiver must be ready before we start to transfer the data
  - Sender and receiver need to agree on a set of parameters
  - Most important: sequence number space in each direction
  - Lots of other parameters: e.g., the Maximum Segment Size

- Handshake protocols: setup state between two oblivious endpoints
  - Need to deal with delayed and reordered packets
Two-way handshake?

Active participant (client)  Passive participant (server)

SYN, SequenceNum = x
SYN, SequenceNum = y

+data

What’s wrong here?
Two-way handshake?

Active participant (client)

Old SYN, SequenceNum = x
New SYN, SequenceNum = q
SYN, SequenceNum = y
+data

Passive participant (server)

Delayed old SYN
Rejected
Three-Way Handshake

- Opens both directions for transfer

Active participant (client)  Passive participant (server)

- SYN, SequenceNum = x
- SYN + ACK, SequenceNum = y,
  Acknowledgment = x + 1
- ACK, Acknowledgment = y + 1
- +data

CSE 123 – Lecture 5: Flow Control
For Next Time

- Read 5-5.1 in P&D
- (Keep) going on the project…