

---

## LT12 - TCP

Pipelined protocols

Pipelining - Sender allows multiple “in-flight”, yet-to-be-acknowledged packets

- Range of sequence numbers must be increased
- Buffering at sender and/or receiver

Two generic forms

### Go-Back-N (TCP)

- Sender can have up to N unacked packets
- Receiver only sends cumulative ack
- Doesn't ack packet if there's a gap
- Sender has timer for oldest unacked packet
- When timer expires retransmit all unacked packets

#### Sender

- k-bit sequence number in packet header
- “window” of up to N, consecutive unacked packets allowed

#### TODO

- ACK(n): ACKs all packets up to and including sequence number N - *cumulative ack*
- May receive duplicate ACKs
- Timer for oldest in flight packet TODO

#### Receiver

- Always send ACK for correctly-received packet with highest in-order sequence number
- May generate duplicate ACKs
- Out of order packets discarded (no receiver buffering)
- re-ACK packets with highest in-order sequence number

### Selective Repeat

- Sender can have up to N unacked packets in pipeline

- 
- Receiver sends individual ack for each packet
  - Sender maintains timer for each unacked packet
  - When timer expires retransmit only that unacked packet

### **Sender**

- Only resends packets for which ACK was not received TODO

### **Receiver**

- Individually acknowledges all correctly received packets
- Buffer packets as needed for eventual in-order delivery to upper layer TODO

### **TCP Overview**

- Point to Point
- One sender, one receiver
- Reliable, in-order byte stream

TODO

Expected to know various elements of TCP segment structure, not necessarily drawing the whole segment.

LOTS OF TODO

### **Summary**

TODO

### **More TCP Issues**

- Options
- Sack enables selective acknowledgements
- Window scaling deals with  $\text{bandwidth} \times \text{delay} > 64k$

TODO

---

## Selective ACK

- Instead of sending a cumulative ACK, instead send a set of (*upto*, *startingfrom*) pairs
- Allows the acknowledgement to indicate that parts of the sequence space have been received
- Other algorithms, like fast retransmission, have made it less useful, but it is still widely supported

## Window Scaling

- TCP “Receive Window” is advertised in every packet and indicates how much data the receiver is willing to accept right now
- It is a 16 bit quantity, so limited to 64k
- At 100MBps (GigE), this means any network with RTT
- TODO
- Solution is to negotiate a scaling factor for the receive window
- Increases effective window size to  $2^{30}$  (shift window left by 14 bits)
- Allows bandwidth\*delay up to 1GB before ACK
- Hence GigE up to 10s RTT, 40GigE up to 250ms RTT

## Paws

## Slow Start