ICS 451: Today's plan

- UDP
- Transmission Control Block (TCB)
- Nagle Algorithm
- TCP Retransmissions and Binary Exponential Backoff
- TCP Congestion Control

UDP

- ports, length, checksum
  - no options
  - length is superfluous
    - IP also records length
- checksum can be sent as zero (no checksum)
  - but bad practice
- maximum payload size is 8 less than maximum IP payload
  - 65,507 bytes
- exercise: what is the IP header size?

Transmission Control Block

- All the information for a socket has to be stored
- The collection of information for a socket is called a Transmission Control Block, or TCB
- TCBs contain variables, including
  - local and remote port numbers and IPs
  - the state of the TCP finite state machine
  - Maximum Segment Size (MSS)
  - snd.nxt, snd.una, snd.wnd
  - rcv.nxt, rcv.wnd
  - incoming/outgoing buffers for data

Nagle Algorithm

- goals:
  - on LANs, send data as soon as possible
  - on WANs, send maximum-sized segments whenever possible
- strategy:
  - when no acks are pending (snd.nxt == snd.una) send what we are given as soon as possible
  - when acks are pending, only send if we have at least 1MSS worth of data

TCP Retransmission

- On timeout, TCP retransmits the segment beginning at snd.una (first unacked sequence)
  - and doubles the retransmission timeout
    - this is called *Binary Exponential Backoff*
- but timeout is slow: how to retransmit faster?
- *fast retransmit*:
  - receiver sends duplicate ack for out-of-order segments
  - sender resends snd.una on 3 duplicate acks

TCP Selective Acknowledgement

- SACK is a TCP header option that reports which out-of-order segments have been received
- Sack-Permitted option sent with SYN
- SACK option has list of blocks of sequence numbers that have been received out of order
  - each block is represented by two sequence numbers, the first and the last

Flow Control and Congestion Control

- The window is a mechanism to slow down a sender that is sending too fast for the receiver
  - this is called *Flow Control*
- What if the sender and receiver are both fast, but the network is slow?
  - the sender will fill router buffers, eventually leading to packet loss
  - sending data that will be discarded is not useful
- so TCP provides *Congestion Control*

  to slow down senders too fast for the network

Congestion Collapse

- Suppose routers have very large buffers
- if more traffic is offered than can be carried,
  - queue will grow and delays increase
- after sufficient time, TCP will retransmit
  - each packet is then carried multiple times
  - exactly when the network is overloaded!!!
- This *congestion collapse* was observed in the late 1980s

Bandwidth Allocation, Fairness

- Ideally, everyone who shares a link should get the same bandwidth
- Problems with this ideal include:
  - we only manage connections, not users
  - users change over time – how to be fair?
  - different users (e.g. backup vs. interactive) need different things
  - fairness can only be provided at any given router
  - users who pay more would expect more
  - should fully use congested links

TCP Congestion Control Principles

- TCP uses *voluntary* congestion control:
  - any connection experiencing congestion should slow down
- TCP defines a *window* on the sender that may slow the sender down
  - beyond what the flow-control window requires
- This sender-only window is called the *congestion control window* or *cwnd*
- Packet loss signals congestion

TCP Congestion Control

- Increase the congestion window regularly
  - the increase is linear: 1MSS/RTT
- Decrease the congestion window once congestion is detected
  - the decrease is multiplicative: reduce to half
- So TCP congestion control is AIMD: Additive Increase and Multiplicative Decrease
  - AIMD is important for network stability

cwnd Transient Behavior

- cwnd is 1 MSS at start of connection
- cwnd grows by 1MSS for every MSS acked
  - this exponential growth is called *slow start*
- When congestion detected, cwnd set to 1 MSS
- then grows exponentially up to half the previous window
  - half the previous window is called the slow start threshold or *ssthresh*

cwnd Linear Growth

- after completion of slow start, cwnd grows by 1MSS every RTT
- this is done by adding to the window, on every new ack,

$$(MSS \times new\text{-}bytes\text{-}acked) / cwnd$$

Congestion Response

- On a fast retransmit, cwnd = cwnd / 2, and cwnd resumes linear growth
- On timer expiration,
  - ssthresh = cwnd / 2,
  - cwnd = 1MSS
  - slow start up to ssthresh, then linear growth

Performance

- Assuming constant available bandwidth, TCP:
- finds it quickly through slow start, then repeatedly:
- drops to half and increases linearly again to the full bandwidth
- so on average, TCP only uses ¾ of the available bandwidth

- Connections with shorter RTT increase faster

Summary

- packet loss means congestion
  - safe assumption, not always accurate
- use cwnd on sender to control speed
- on congestion, slow down multiplicatively
  - cut the window to half
- most of the time, grow linearly
- all this is known as TCP Reno

- fairness is impossible, but worth trying for

Alternatives

- Since queue lengths increase as congestion begins, RTTs also increase
- measuring the RTT (or related measures) allows early detection of congestion
  - unfortunately RTT measurements are noisy
- area for research!
- UDP has no congestion control
- Other Internet transport protocols are supposed to implement “TCP-like” congestion control