ICS 351: Today's plan

- port numbers
- congestion control
- Ethernet switching
- Spanning Tree Protocol

TCP Behavior

- TCP control bits (SYN, FIN, ACK, RST) help maintain TCP connections
- three-way handshake is SYN, with SYN-ACK in answer, and a final ACK to confirm receipt of the second packet
- 32-bit sequence number, ack number count bytes rather than packets
- an ack is sent, almost for free (piggyback) in every packet except the first
- window tells the recipient how many more bytes (past the ack) the sender of this packet is willing to receive -- flow control, slowing down the sender to avoid overwhelming a slow receiver
- this is the flow control window
- setting the window to zero forces the sender to stop
- in general, TCP can send one window every RTT (round-trip time)

port numbers

- an IP address identifies an interface, and by extension a machine
- a port number identifies an application within a machine
- servers *listen* on specific, *well-known* ports
- each local port can be used for multiple *sockets*, as long as (at least) one of these is different: local/remote IP, local/remote port, protocol
- note:
- o a socket has a **local** and a **remote** port (and likewise for IP addresses)
- o a packet has a **source** and a **destination** port (and likewise for IP addresses)
- local and remote make sense on a host, whereas source and destination make sense for a packet

Congestion Collapse

- reminder: the network hardware might be working fine, but if the software fails, the network goes down
- e.g. if the routing tables include loops, packets will not get delivered
- imagine a retransmission mechanism where, when a packet is lost, I resend the lost packet and also a new one
- if a packet is lost due to congestion, the first little congestion experienced will likely lead to more congestion
- this happened a few times in the 1970's -- the network hardware was working fine, but almost no data would get through

TCP Congestion Control

- to control congestion, TCP slows down substantially (half the speed) whenever packets are lost
- TCP then slowly speeds up its transmission rate when no packets are lost.
- this is controlled by a window that (unlike the flow control window above) is maintained on each sender, and never communicated: the congestion window
- when packets are lost, the congestion window shrinks to about half its previous size (the details are complicated!)
- every RTT when no packets are lost, the congestion window grows by one packet
- the effective window is the smaller of the flow control window and the congestion window
- since each TCP can send one window every RTT, shrinking the window slows down sending
- TCP also has other mechanisms to lessen congestion, including binary exponential backoff on retransmissions, and adaptive timers to more reliably detect packet loss

Ethernet Equipment

- much experience so far in lab with Ethernet
- different equipment used to connect Ethernet segments:
- o hubs: broadcast everything
- o switches: broadcast packets to 0xff:ff:ff:ff:ff:ff, and packets for destinations that are not known. Selectively transmit where possible
- o routers: forward packets among different IP networks
- hubs and switches work within a single IP network as a single broadcast medium (but switches don't always broadcast)
- traditional bridges had two interfaces, and forwarded everything from one interface to the other -- hubs and switches both implement this bridging function

Learning Switches

- if a switch gets a packet from *A* on interface *I*, it forwards the packet,
- and remembers that *A* can be reached on interface *I*
- the next time a packet for *A* is received on interface *I'*, it is only forwarded on interface *I* (unless *I* == *I'*, and then it is not forwarded)
- if there is no record of communication from *A* (within the last 60 seconds), the packet is broadcast on all interfaces except *I'*

Broadcast Storms

- given a network with redundant links
- if the network is connected by hubs, every packet will cause collisions with itself
- if the network is connected by switches, any broadcast packet will live forever
- packets may even be multiplied if there is more than one loop
- this is useless traffic that gets in the way of useful traffic -- a "**broadcast storm**"

Preventing Broadcast Storms

1. have no redundant links in the network, or
2. restrict "broadcast" forwarding by switches:

    * select a root switch, based on priority, using MAC addresses to break ties in case of equal priority

    * find a least-cost path to the root, reached via the root port for this switch

    * for each segment, determine a least-cost switch port to use to reach the root, the designated port for this segment

    * only forward broadcasts along root ports and designated ports

    * root ports and designated ports form a *Spanning Tree*

Rapid Spanning Tree Protocol RSTP

- the regular spanning tree protocol can take tens of seconds to converge after a topology change
- instead, a switch can pre-select alternate ports that also lead to the root bridge
- broadcast data is only sent on alternate ports when it is determined that the root port is disconnected
- similarly for backup paths to individual segments
- switches also actively exchange their information, so one switch can quickly hand off forwarding to another switch