Week 7

* TCP: Overview
  + Point to point
    - One sender, one receiver
  + Reliable, in-order byte steam
  + Full duplex data
    - Bi-directional data flow in same connection
    - MSS: maximum segment size
  + Cumulative ACK’s
  + Pipelining
    - TCP congestion and flow control set window size
  + Connection-oriented
    - Handshaking
* TCP segment Structure
  + Will include port numbers of both the source and destination
  + The sequence numbers
  + Acknowledgment number
  + Check sum
  + Receive window
  + Heavily suggested that you check the week 7 slide 4 for the diagram
* TCP sequence numbers, ACKs
  + Sequence numbers
    - Byte stream “number” of first byte in segment’s data
  + Acknowledgment
    - Seq # of next byte expected from the other side
    - Also serves as cumulative ACK
* TCP round trip time, timeout
  + How to set TCP timeout value?
    - Longer than RTT, but the RTT value changes?
    - Too short: premature timeout, unnecessary retransmissions
    - Too long: slow reaction to segment loss
    - How to estimate the RTT
      * Sample the RTT and take an average to smooth over the result
  + Exponential Weighted Moving Average
    - Influence of past sample decreases exponentially fast
    - Typical value = .125
* TCP reliable data transfer SUMMARY
  + TCP creates reliable service on top of IP’s unreliable service
    - Pipelined segments, cumulative ACKs, single retransmission timer
  + Retransmissions triggered by
    - Timeout attacks
* TCP Flow Control
  + Sender must transmit data no faster than it can be consumed by the receiver
    - Receiver might be a slow machine
    - App might consume data slowly
  + Flow control:
    - Receiver controls sender, so sender won’t overflow receiver’s window
* TCP connection management
  + Before exchanging data, sender/receiver “handshake”
    - Agree to establish connection
    - Agree on connection parameters
* TCP 3-way Handshake
* Congestion Control
  + Congestion
    - Informally, too many sources sending too much data for the network to handle
  + Manifestations
    - Long delays and packet loss
* TCP congestion Control
  + A lost segment implies congestion
    - The TCP sender’s rate should be decreased when a segment is lost

Thursday:

* Network Layer
  + Forwarding and Routing
* Network-Layer Services and Protocols
  + Transport segment from sending to receiving host
    - Sender: encapsulates segments into datagrams, passes to link layer
    - Receiver: delivers segments to transport layer protocol
  + Network layer protocols in every internet device
    - Host. Routers, clients, etc.
    - Unlike the application and transport later
  + Routers:
    - Examines header fields in all IP datagrams passing through it
* Two Key network layer functions
  + Network layer functions
    - Forwarding: move packets from a router’s input link to appropriate router output link
    - Routing: determine route taken by packets from source to destination
      * Routing algorithms
    - Analogy used in class:
      * Forwarding: process of getting through single interchange
      * Routing: planning a travel by using a map prior to leaving
* Switching Fabrics
  + Transfer packet from input link to appropriate output link
  + Switching rate: rate at which packets can be moved from input links to output links
  + Three major types of switching fabrics
    - Bus, memory, interconnection networks
      * First gen routers
      * Computers with switching under direct control of CPU
      * Packet copied to systems memory
      * Speed limited by memory bandwidth
* Output Port Queuing
  + Buffering
    - Required when datagrams arrive from fabric faster than link transmission rate
  + Drop Policy: which datagrams to drop if no free buffers
    - Tail drop: drop arriving packet
    - Priority: drop/remove on priority basis
  + Scheduling discipline
    - Chooses among queued datagrams for transmission
      * First come first served priority
* Routing algorithm
  + The goal is to determine the best path from sending hots to receiving host, through network of routers
    - Path: sequence of routers that the packets traverse to reach destination
  + Good: least cost, fastest, least congested
* Graph Abstraction
  + Graph G = (nodes, edges)
    - N = set of routers = {u,v,w,x,y,z}
    - E = set of links = {(u,v), (u,x),(v,x),(v,w),…}
* Dijkstra’s Link-state routing algorithm
  + Centralized (global): network topology, link costs known to all nodes
    - Accomplished via “link state broadcast”
    - All nodes have same info
  + Computes least cost paths from one node (“source”) to all other nodes
    - Gives forwarding table for that node
  + Iterative: after k iterations, know least cost path to k destinations
  + Cx,y = direct link cost from node x to y
  + D(v) : current estimate of cost of least-cost-path from source to destination v
  + P(v) predecessor node along path from source to v
  + N’: set of nodes whose least-const-path definitively known at this point in the calculation

Initialization

N’ = {u}

for all nodes a

if a adjacent to u

then D(a) = Cu,a

else D(a) = infinite

Loop

find a not in N’ such that D(a) is a minimum

add a to N’

update D(b) for all b adjacent to a and not in N’

D

Network Layer Summary