CS456 Post-Midterm

Module 3

From slide 57

<u>TCP</u>
Point-to-point: one receiver, one sender
Reliable, in-order byte stream: no "message boundaries"
Pipelined: TCP congestion and flow control set window size
Full duplex data: bi-directional data flow with same connection
    -MSS: maximum segment size
Connection-oriented: handshaking (exchange of control msgs) inits sender, receiver state before data exchange
Flow controlled: sender will not overwhelm receiver

<u>TCP segment structure</u>
-source port #
-dest port #
-ack #
-receive window
-checksum
-payload

<u>TCP Sequence Numbers, and ACKS</u>
Sequence Numbers: byte stream "number" of first byte in segment's data
Acknowledgements: seq # of next byte expected from the other side
    -cumulative ACK

Out-of-order segments handled according to the implementer

<u>**HOST A   HOST B**</u>
Seq42 ack 79 data c ->

    <- Seq79 ack 43 data c

Seq43 ack80 ->

<u>TCP RTT, Timeout:</u>
Timeout: longer than RTT – varies
    -too short/too long – premature timeout or slow reaction to drop
Estimate RTT: SampleRTT measured time from segment transmission until ACK receipt
    -ignore retransmissions

-actual RTT will vary – want estimate

Formula

EstimatedRTT = $(1-\alpha)\times EstimatedRTT + \alpha\times SampleRTT$

-exponential weighted moving average

-influence of past sample decrease exponentially fast

-typical $\alpha = 0.125$

Timeout interval: EstimatedRTT plus "safety margin"

-large variation in EstimateRTT → Larger Safety Margin

Formula

DeviationRTT = $(1-\beta)\times DeviationRTT + \beta \times |\, SampleRTT - EstimatedRTT \,|$

$where\ \beta$ is usually 0.25

TimeoutInterval = EstimatedRTT + 4 x DeviationRTT

TCP reliable data transfer

-TCP creates a rdt on top of IP's unreliable service

-pipelined segments

-cumulative acks

-single retransmission timer

-retransmissions triggered by:

-timeouts

-duplicate acks

TCP Sender events

-data received from app

-create segment with seq #

-seq # is byte-stream number of first data byte in segment

-start timer if not already running

-timeout

-retransmit segment that caused timeout

-restart timer

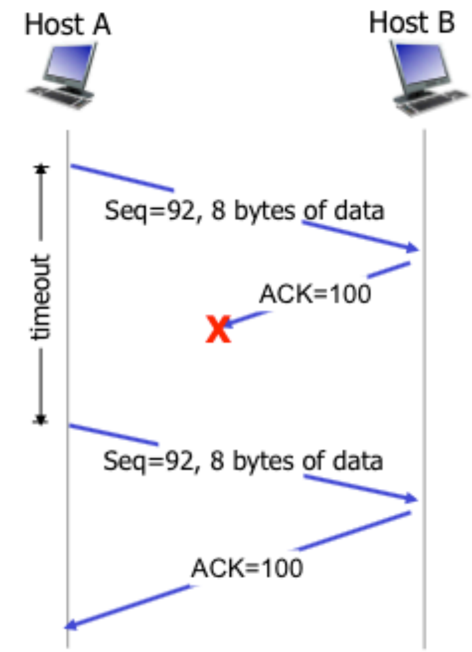
-ack received

-if ack acknowledges previously unacked segments

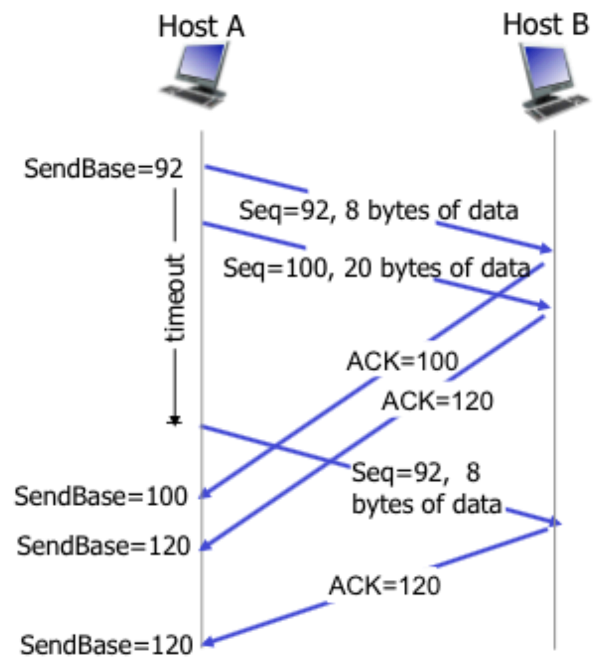
-update what is know to be acked

-start timer if there are still unacked segments

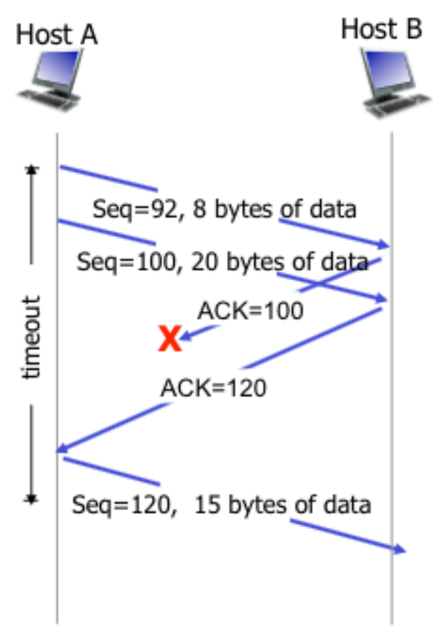
Retransmission Scenarios



lost ACK scenario



premature timeout



cumulative ACK

<u>TCP Receiver Events</u>

-received in-order segment

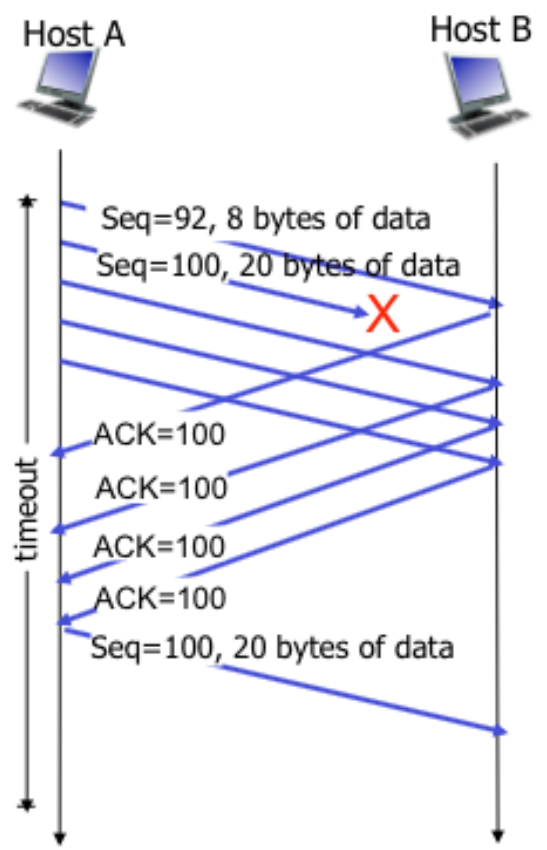
-delayed ack – wait upto 500ms if no other segment send ack as is

-received in-order segment – but with another segment with ack pending

-send single cumulative ack for both

-received out-of-order segment – high than expected seq #, gap detected
    -immediately send duplicate ack – indicating expected ack
-received segment that partially or completely fills gap
    -immediate ack – only if segment is at the lower end of gap

TCP Fast Retransmit
-time-out period –often relatively long
-detect lost segments via duplicate acks
    -sender often sends many segments back-to-back
-if sender receives 3 ACKs for same data "triple duplicate ACKs"
    -resend lowest unacked seq #



fast retransmit after sender
receipt of triple duplicate ACK

TCP Flow control
-flow control: receiver controls sender
    -so sender can't overflow receiver's buffer by transmitting too fast
-receiver "advertises" free buffer space by including **rwnd** in the TCP header
    -**RcvBuffer** size set via socket options (typical default is 4096)
    -many operating systems autoadjust RcvBuffer
-sender limits amount of unacked "in-flight" data to receiver's **rwnd** value
-guarantees not to overflow

Connection Management
-handshake
    -agree to establish connection
    -agree on connection parameters

2-way handshake
-variable delays
-retransmitted messages
-message reordering
-can't see other side

req_conn(x) and acc_conn(x)

TCP 3-way handshake

Module 4

Network layer
-Transport segment from sending to receiving host
-sender sends datagrams
-receiver delivers to transport layer
-network layer for every host and router
-router looks at header of all IP datagrams

Datagram networks
-no call setup at network layer
-router – no state about end-to-end connections
    -no network-level concept of "connections"
-packets forwarded using destination host address

Two key network-layer functions
Forwarding: move packets from router's input to appropriate router output
    -"getting through exchange"
Routing: determine route taken by packets from source to dest
    -routing algorithms
    -"whole trip"

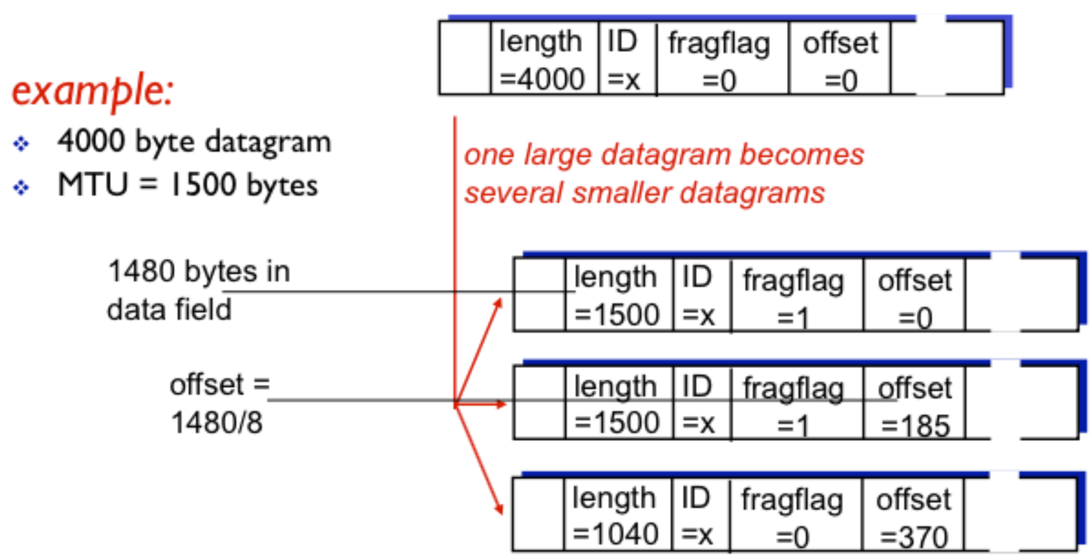
Longest Prefix Matching
-use longest address prefix when looking at forwarding table

What's in the Network layer
-routing, IP, ICMP protocols

IP datagram format
-protocol version
-length
-fragment offset
-time to live(num hops)
-checksum
-source, des tip
-payload
-20 bytes IP overhead and 20 bytes TCP overhead

IP Fragmentation, reassembly
-links have MTU(maximum transfer size)
-one datagram becomes several datagrams
-only reassembled at final destination
-IP header used to reassemble

Example:
Divide offset by 8 to reduce offset bit size

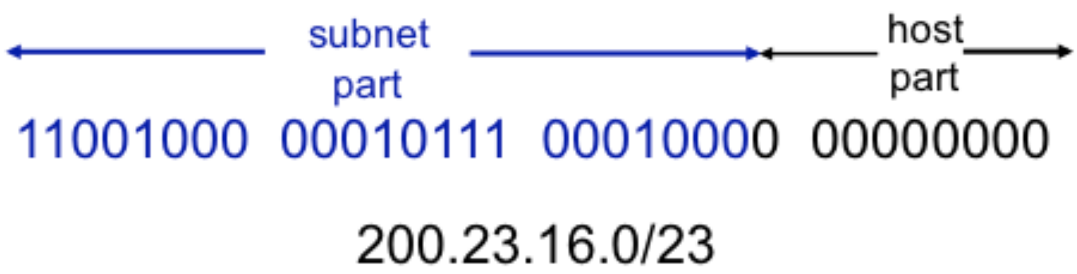


IP Addressing
IP Address: 32-bit identifier for host, router interface
Interface: connection between host/router and physical link
-routers have multiple interfaces
-eg: wireless, Ethernet

How interfaces are actually connected
-switches
-wifi base stations

Subnet
Recipe: determine the subnets, detach each interface from its host or router, creating islands of isolated networks
-each isolated network is a *subnet*

IP Addressing: CIDR
Classless InterDomain Routing: subnet portion of address of aritbrary length
-address format: a.b.c.d/x
where x is the # bits in the subnet portion (first part)



IP Address is hard-coded by system admin in a file

DHCP: Dynamic Host Configuration Protocol
-get address from server
-GOAL: allow host to dynamically obtain IP address when joining
-can revew its lease
-allows reuse of addresses
-support for mobile users

Overview:
Host broadcasts "DHCP discover"
DHCP server responds with "DHCP offer"
Host requests IP address "DHCP request"
DHCP server sends address "DHCP ack"

DHCP: more than IP addresses
-can return more than just IP on subnet
-address of first-hop router for client
-name and IP address of DNS server
-network mask (indicating network versus host portion of address)

How does network get subnet part of IP address
-allocated portion of ISP's address space

Hierarchical addressing: route aggregation
-efficient advertising of routing info
-"send me anything for this subnet"

IP addressing:
-ICANN provides blocks of address, DNS, domains

NAT: network address translation
-all datagrams leaving local network have same single source
  -NAT IP address with port numbers
MOTIVATION: LAN has IP to the world
  -different addresses for different hosts
  -change addresses of LAN without notifying outside
  -can change ISP without changing LAN
-replace incoming/outgoing IP and port and source and port
  -save to translation table

16-bit port number field
-60k simultaneous connections

ICMP: Internet control message protocol
-used by hosts and routers to communicate network-level info
  -erorrs
  -request/replies
-above IP

Traceroute and ICMP (how it works)
-source sends series of UDP segments to dest
  -TLL = 1 to n
  -when nth set of segments arrive –router discards datagrams
  -sends sources ICMP messages
    -includes name and IP and router

-stopping criteria:
  -UDP segment eventually arrives at dest
  -dest returns ICMP "port unreachable"
  -source tops

IPv6 :
-32-bit address space too small
-header format is faster
  -QoS
-fixed 40 byte header – no fragmentation

Datagram Format
-priority
-flow label – not well defined
-next header : identify upper layer protocol for data

Other changes from IPv4
-no checksum
-options in "next header" field
-ICMPv6
    -new messages
    -multicast group management functions

Transitions from IPv4 to IPv6
-tunneling: IPv6 datagram carried as IPv4 payload through IPv4 routers

Routing algorithm: determines end-to-end path through internet
Forwarding table: determines local forwarding at this router

$$cost(x_1, \ldots, x_n) = cost(x_1, x_2) + \cdots + cost(x_{n-1} + x_n)$$

Global or decentralized info
-global
    -all routers have complete topology
-decentralized
    -router knows cost to neighbours only
    -iterative
    -**distance vector** algorithms

-static: changes slowly over time
-dynamic: changes more quickly
    -periodic update
    -response to link cost changes

Dijkstra's Algorithm
-net topology – link costs known to all nodes
    -link state broadcast
    -all nodes same info
-computes least cost pathes from one node ("source") to all other nodes
    -gives forwarding table for that node
-iterative after k
-link state

c(x,y) – cost from x to y
D(v) – cost from current to v
p(v) – previous node along path from source to v
N' – set of nodes whose least cost path known

NEED TO LEARN HOW THE ALGO WORKS

Distance vector algorithm
Bellman-Ford equation (Dynamic programming)

$$d_x(y) = \ cost\ of\ least\ cost\ path\ from\ x\ to\ y$$
$$d_x(y) = \min\{c(x,v) + \ \ d_v(y)\}$$

-iterative approach
-distance vector

node x knows cost to each neighbor
    -needs to maintain neighbours' distance vectors
-from time-to-time, each node sends its own distance vector estimate to neighbours
    -once receive recalculate distance vector and resend

-iterative, asynchronous
    -each local iteration caused by:
        -local link cost change
        -DV update message from neighbor
-distributed:
    -each node notifies neighbours only when its DV changes and notify if necessary

link cost changes:
-node detects local link cost change
-updates routing info – recalculate distance vector
-if DV changes – notify
-good news travels fast, bad news travels slow

Poisoned reverse: if Z routes through Y to get to X
-Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)

Message complexity:
LS: with n nodes, E links O(nE)
DV: exchange between neighbours only
    -convergence time varies

Speed of convergence:
LS:: O(n^2) requires O(nE) messages
DV: convergence times varies

robustness: (malfunctioning router)
LS: each node computes only its table
DV: error propagates through network

<u>Hierarchical routing</u>
Scale: can't store all destinations in routing table
    -routing table exchange would swamp link
administrative autonomy: each network admin may want control routing in its own network

Aggregate routers into "autonomous systems" (AS)

- same AS run same routing protocol
- "intra-AS" routing
- routers in different AS can run different intra-AS routing protocol

gateway router: at "edge" of its own AS
-links to another AS

interconnected-AS: forwarding table
-controlled by both intra-AS and inter-AS routing algorithms
-intra-AS for internal
-inter-AS & intra-AS for external

<u>Inter-AS</u>
-propagate reachabilitiy to all internal routers

<u>Choosing among multiple ASes</u>
-AS2 and AS3 have path to dest
-**hot potato routing:** send packet towards closest of two routers (cost via intra-AS protocol)

<u>Intra-AS routing</u>
-AKA: Interior Gateway Protocols (IGP)
-ex: RIP, OSPF, IGRP

<u>RIP (Routing Information Protocol)</u>
-distance vector algorithm
-max 15 hops
-DV exchanged every 30 secs (advertisement)
-each advert list upto 25 dest subnets

Module 5

Data-link layer: transfer datagram from one node to physically adjacent node over a link

Ex: Ethernet, 802.11

Tourist = datagram
Transport segment = communication link
Transportation mode = link layer protocol
Travel agent = routing algorithm

Link layer services
Framing, Link access:
-encapsulate datagram into frame, adding header, trailer
-channel access if shared medium
-"MAC" addresses used in frame headers to identify source, dest

Reliable delivery between adjacent nodes
-learned how ch 3
-low error in fiber and twisted pair

Half-duplix and full-duplex

Flow control

Error detection

Error Correction

Where link layer
-every host
-NIC

Links communicating
-put datagram into frame
-rdt, flow, error checking/correcting done
-extract datagram and pass up the stack

Multiple access links, protocols
Point-to-point: PPP for dial-up, Ethernet switch
Broadcast (shared wire or medium): wireless, old fashion Ethernet

Collision: if node receives two or more signals at the same time

Media access control (MAC)
-distributed algorithm determines how nodes share channel
-communication about channel sharing must use channel itself

given: broadcast channel of rate R bps

wanted:
1. when a node wants to transmit, it can send at rate R
2. when M nodes want to transmit, each can send at average R/M
3. fully decentralized
   -no special node to coordinate transmissions
   -no synchronization of clocks, slots
4. simple

MAC protocols: taxonomy
-channel partitioning
  -divide channel into smaller "pieces" (time, frequency, code)
  -allocate price to node for exclusive use
-random access
  -channel not divided, allow collisions
  -"recover" from collisions
-"taking turns"
  -nodes take turns, but nodes with more to send can take longer turns

Channel Partitioning
-TDMA – time division multiple access
  -access to channel in "rounds"
  -each station gets fixed length slot (length = pkt trans time) in each round
  -unused slots go idle
-FDMA – frequency division multiple access
  -channel spectrum divided into frequency bands
  -each station assigned fixed frequency band
  -unused transmission times in frequency bands go idle

Random access protocols
-when node has packet to send
  -transmit at full channel data rate R
  -no a priori coordination among nodes
-two or more transmitting nodes → "collision
-random access Mac protocol specifies
  -how to detect collisions
  -how to recover from collisions (eg: via delayed retransmissions)
-examples of random access MAC protocols
  -ALOHA
  -slotted ALOGA
  -CSMA, CSMA/CD, CSMA/CA

Carrier Sense Multiple Access (CSMA)
-listen before transmit
-if channel sensed idle: transmit entire frame
  -if channel sensed busy, defer transmission
-human analogy: don't interrupt others

collisions can still occur
  -propagation delay means nodes may not hear each other
  -collision means entire packet transmission time wasted

CSMA/CD (collision detection)
  -collisions detected within short time
  -colliding transmissions aborted, reducing channel wastage

-collision detected:
    -easy in wired
    -difficult for wireless
-human analogy: polite conversationalist

<u>CSMA/CD algorithm</u>

1. NIC receives datagram from network layer and creates frame
2. NIC senses idle channel or wait until idle
3. If no collision detected during transmission we're done
4. If collision, abort – send 48-bit jam signal
5. After aborting, enter binary(exponential) backoff – backoff time grows exponentially

<u>CSMA/CD efficiency</u>

$$efficiency = \frac{1}{1 + \frac{5t_{prop}}{t_{trans}}}$$

<u>"Taking turns" MAC Protocols</u>
-channel partitioning MAC protocols
    -share channel fairly
    -inefficient low load – efficient high load
-random access MAC protocols
    -efficient at low load: single node fully utilize channel
    -high load:collision overhead
-best of both

<u>"Taking turns"</u>
-polling: master node "invites" slave nodes to transmit in turn
-typically used with "dumb" slave devices
-concerns:
    -polling overhead
    -latency
    -single point of failure (master)

-token passing: control token passed among nodes sequentially
-concerns:
    -token overhead
    -latency
    -single point of failure

<u>MAC Addresses and ARP</u>
32-bit IP address: network-layer address for interface
    -used for layer 3 forwarding
MAC (or LAN or physical or Ethernet) address:
    -used "locally" – gets from one interface to another physically-connected interface

-48-bit in hex

LAN Addresses
-IEEE
-manufacturer buys address space
-IP hierarchical address not portable
-depends on subnet

ARP (address resolution protocol) table:
-IP/MAC address mappings for some LAN modes
-TTL (time to live) for mappings 20 mins

if mapping not found **broadcast** peers will relay and host will be notified to return MAC address
-then save to ARP table

Routing to another LAN