CS118 Final – Nathan Tung (004059195)

0. Security between Bob/Alice (people, web browsers/servers, online bank clients/servers, DNS servers, routers exchanging routing tables, etc.)
   a. Confidentiality: only sender/intended receiver should understand (sender encrypts, receiver decrypts); Authentication: sender/receiver confirm each other's identity; Message integrity: sender/receiver ensure message not altered; Access and available: services must be accessible and available
   b. Symmetric/private key cryptography: sender/receiver share secret key; asymmetric/public has encryption key known to all, private decryption key known only by receiver
      i. RSA (Rivest-Shamir-Adleman algorithm): public key cryptosystem involving the asymmetry found in the factoring of the product of two large prime numbers (known as the factoring problem)
      ii. KB-(KB+(m)) = m; KB- is Bob's private decryption key, KB+ is Bob's public encryption key
      iii. No matter what we do, Trudy could always perform a man-in-the-middle attack
   c. Digital signature: Bob signs message m by encrypting with his private key KB- to create a "signed" message; Alice can use public key KB+ and prove that Bob signed it
      i. Digital signature message digest: goal is to have fixed-length, easily-computed digital "fingerprint" (since it's computationally expensive to public-key-encrypt long messages)
      ii. Apply hash function H (often MD5 or SHA-1, the US standard) to m to obtain fixed-size message digest H(m)
         1. Bob sends digitally signed message: large message m → H → H(m) → digital signature KB- (encrypt) → encrypted message digest KB-(H(m))
         2. Alice verifies signature and integrity of message: large message m → H → H(m) =?= encrypted message digest KB-(H(m)) → digital signature KB+ (decrypt) → H(m)
      iii. Message digest: AKA cryptography checksum/hashcode, a function-produced number that is extremely difficult to reverse; also a hash function that converts variable-length input into small fixed-length value
   d. CA (certification authorities): bind public key to specific entity E; that is, E(person, router) registers public key with CA; then Alice applies CA's public key to Bob's certificate, obtaining Bob's public key
1. Multimedia Networking
   a. Application types
      i. Streaming/stored (audio, video): streaming begins playout before downloading entire file; interactivity; stored at server can transmit faster than audio/video will be rendered, thus storing/buffer at client [YouTube, Netflix, Hulu]
         1. UDP streaming, HTTP streaming, adaptive HTTP streaming (latter two used more); use client buffering to absorb variations in server-to-client delay and allow continuous playback until buffer is drained
         2. CDN (content distribution networks) manages servers in distributed locations, stores video copies and content in servers, and directs users to a CDN location with best user experience
            a. Private (Google/YouTube) or third-party (Akamai for Netflix/Hulu]
            b. Large mega-server is a bad idea and doesn't scale; "Enter Deep" (deploy server clusters in access ISPs all over the world, close to end users to improve delay and throughput) (Akamai) vs. "Bring Home" (bring ISPs home by building larger clusters at smaller number of key locations and connect them near tier-1 ISPs for lower maintenance/management overhead but probably more delay and lower throughput) (Limelight)
         3. Netflix: 30% downstream US traffic in 2011, little infrastructure due to using 3rd party services (Amazon cloud services), multiple CDN providers (Akamai, Lightlight, Level-3),
      ii. Conversational (voice/video over IP): interactive nature of human-to-human conversation limits delay tolerance [Skype]
         1. Network loss (IP datagram lost due to network congestion – router buffer overflow), delay loss (IP datagram arrives too late for playout due to processing, queuing, etc.; max tolerable delay 400 ms), loss tolerance (1-10% ok)
         2. Receiver tries to playout each chunk exactly q ms after it's generated; with time stamp t, play at t+q (large q = less packet loss, small q better interactive experience)
         3. Recovery from packet loss can use FEC (2D parity); for every group of n chunks, create redundant chunk by XORing n original chunks; send n+1 chunks, increasing bandwidth by 1/n; can reconstruct we lose up to 1 chunk
         4. Skype: proprietary application-layer protocol using encrypted messages; uses P2P components like clients, SN (super nodes, or Skype peers with special functions), overlay network (among SNs to locate clients), login server
            a. Join by contacting SN (ip address cached) using TCP; login to centralized login server; obtain IP for callee from SN, SN overlay, or client buddy list; initiate call
            b. If Alice and Bob are behind NATs, no outside peer can connect to inside peer; relay solution: Alice signals SN to connect to Bob, and her SN connects to Bob's SN; Bob's SN connects to initiated connection for his SN
      iii. Streaming live (audio, video): live sporting event [futbol]
   b. RTP (real-time protocol): specifies packet structure for packets carrying audio or video data
      i. RTP packet provides payload type identification, packet sequence number, time stamping
      ii. RTP runs in end systems; RTP packets encapsulated in UDP segments; interoperability: two VoIP apps running RTP may be able to work together
      iii. RTP libraries provide transport-layer interface extending to UDP (port nums, IP, payload type identification, packet sequence number, time-stamping)
      iv. Example: sending 64 kbps PCM-encoded voice over RPT; app collects encoded data in chunks (160 bytes/chunk every 20 ms); audio chunk and RPT header form the RPT packet, encapsulated in UDP
         1. RPT header indicates type of audio encoding in each packet (which sender can change during conference); RTP header also contains sequence numbers, timestamps
         2. Payload type (encoding type used, 7 bits), seq # (one per RTP packet sent, detects packet loss/disorder, 16 bits), timestamp (sample of data's first byte, 32 bits), SSRC (IDs source of RTP stream, 32 bits),
      v. No mechanisms ensuring timely data delivery or other quality of service guarantees; RTP encapsulation only at end systems (not at best-effort intermediate routers; no effort to ensure they arrive on time)
   c. RTCP (real-time control protocol): works with RTP, participants in RPT periodically send RTCP control packets to other participants; contain sender/receiver reports (# packets sent/lost) for feedback and performance
      i. For multiple multicast senders, each RTP session typically has a single multicast address and all RTP/RTCP packets use that address; RTP, RTCP packets distinguished via port numbers; RTCP traffic reduced when more participants join
      ii. SIP (session initiation protocol) goal: all conference or phone calls use internet, people identified by name/email, not phone numbers, and callee can be reached no matter what IP callee is currently using or roaming with
   d. Multiple classes of service: provide protection/isolation for one class from others; allocate fixed, non-shared bandwidth to flow (inefficient use if flow doesn't use); while providing isolation, use resources as efficiently as possible
      i. Traffic Policing: limit traffic to not exceed declared parameters, forcing source to adhere to bandwidth allocations (marking or policing at network edge)
         1. (Long term) average rate: how many packets can be sent per unit time (in the long run) (e.g. 100 packets/s or 6000 packets/min)
         2. Peak rate: (e.g. 6000 packets/min – ppm on average; 1500 ppm peak rate)
         3. (Max) burst size: max number of packets sent consecutively (no intervening idle)
         4. Implementation: token bucket (limits input to specified burst size and average rate); bucket holds b tokens, but tokens generate at r token/s unless bucket is full
         5. Policing via token bucket and WFQ together provide guaranteed upper bound on delay – a quality of service guarantee!
      ii. Scheduling: choosing next packet to send on link
         1. FIFO scheduling: send in order of queue arrival (discard policy: new packet comes into full queue. Who gets dropped? Tail drop new arriving packet? Drop other packets by priority? Randomly?)
         2. Priority scheduling: send highest priority queued packet; multiple classes with different priorities
         3. Round Robin (RR) scheduling: multiple classes; cyclically scan class queues, send one complete packet from each class; WFQ (weighted fair queuing): generalizes RR, weighted amount of service per cycle
      iii. Leaky (token) bucket: algorithm used in packet-switched networks to check that transmitted data packets conform to limits on bandwidth and burstiness (bucket holds b tokens, generated at rate r tokens/s unless full)
         1. Models the buffering requirements for smooth playback. In this model, the decoder maintains a buffer. Encoded data goes from the network into the buffer, and from the buffer into the decoder. If the buffer underflows, it means the decoder is removing data from the buffer faster than the network is delivering it. If the buffer overflows, it means the network is delivering data faster than the decoder consumes it.
         2. Host sends packet by placing into bucket; bucket (network interface) leaks (transmits) at constant rate; bursty traffic becomes constant traffic via bucket; bucket is finite queue with finite rate of output
         3. If bucket or queue is full, we just discard those packets from host
      iv. Scalability: simple functions in network core, relatively complex functions at edge routers/hosts; don't define service classes, but rather provide functional components to build service classes
      v. COS (class of service): parameter used in data/voice protocols to differentiate between types of payloads contained in the transmitted packet; used as a means to manage traffic in network by grouping together in classes; best-effort
2. Wireless and Mobile Networking
   a. WiFi 802.11 (b up to 11 Mbps at 2.4-5 GHz, a up to 54 Mbps at 5-6 GHz, g up to 54 Mbps at 2.4-5 GHz, n with multiple antennae up to 200 Mbps at 2.4-5 GHz) (g is most popular)
      i. All of these use CSMA/CA for multiple access; use same frame structure for link-layer frames; can reduce transmission rate to increase distance; communicate with base-stations (or AP, access points) and ad-hoc network versions
      ii. BSS (basic service set, or "cell", a group of communicating stations) in infrastructure mode contains wireless hosts, base stations/APs, and ad hoc mode (hosts only)
   b. CSMA/CA (carrier sense multiple access – collision avoidance) (more overhead than CD, reducing performance)
      i. 802.11 has no collision detection (only CSMA – sense before transmitting); difficult to sense collisions due to weak signals (like in hidden terminal), that's why we use CSMA/CA; goal is to avoid collisions
      ii. Sender: if sense channel idle for DIFS, transmit entire frame; else channel is busy, count down random backoff time and transmit when timer expires; if no ACK, increase random backoff interval and repeat
      iii. Receiver: if frame received ok, return ACK after SIFS (ACK needed for hidden terminal problem)
      iv. SIFS (short inter-frame space): time required for station to send end of frame and start transmitting; DIFS (DCF inter-frame space): time to wait before starting backoff interval ("contending") (DIFS=SIFS+2 slots)
      v. Difference with Ethernet: 802.11 uses CA, not CD; 802.11 uses link-layer acknowledgement/retransmit (ARQ) due to high BER (bit error rates) in wireless
      vi. 802.11 has no CD, so stations that begin transmitting a frame will transmit it to completion (no turning back!); long frames being transmitted with collisions can degrade performance of WiFi very much
         1. This is the need for SIFS and DIFS – if two stations transmit immediately, collision occurs and WiFi wouldn't abort transmissions
         2. 802.11 backoff: if two stations sense busy channel, both enter random backoff (hopefully with different backoff values); the "winning" channel begins transmitting and the "losing" one waits for complete transmission
      vii. Link-layer acknowledgment scheme: when dest station receives frame that passed CRC, it waits for SIFS then sends back ACK frame; if no ACK received, assume error and retransmit frame using CSMA/CA protocol to access channel
   c. Hidden terminal: A←→B←→C; B, A hear each other and B, C hear each other; A, C cannot hear each other due to interference at B (signal attenuation/fading); this can lead to collision, unfortunately
   d. Dealing with hidden terminal: RTS (request-to-send) and CTS (clear-to-send)
      i. Allows sender to reserve channel rather than random access of data frames (avoid collisions in long data frames)
      ii. Sender first transmits small RTS packets to BS using CSMA (RTSs might collide, but they're short); if clear, BS broadcasts CTS specifically for that sender in response
      iii. CTS heard by all nodes; sender transmits data frame, other stations defer transmissions; now we can avoid data frame collisions completely using small reservation packets
      iv. Process: DIFS; source sends RTS to dest; SIFS; dest sends CTS to source and all nodes; SIFS; source sends data to dest; SIFS; dest sends ACK to source and all nodes (at this point, data transmit is done and channel opens for other nodes)
3. Link Layer (datagram transferred by different link protocols over different links; hosts and routers are nodes, communication channels that connect nodes are links, layer-2 packets are frames that encapsulate datagram)
   a. EDC (error detection/correction): receiver IDs and corrects bit errors without retransmitting; not 100% reliable, but rarely; larger EDC fields is more accurate in [Data][EDC]
      i. Single bit or 2D parity checking; internet checksum, CRC (cyclic redundancy check) using D*2^r XOR R = nG or D*2^r = nG XOR R
   b. FEC (forward error correction), ARQ (automatic repeat request)
      i. ARQ requires retransmitting of certain portions with error detected; FEC uses codes that allow receiver to detect/correct some number of errors without retransmit
      ii. ARQ: receiver discards erroneous frames, ACKs errorless frames, NAKs erroneous frames; sender retransmits NAKed frames or retransmits unACKed frame on timeout
         1. Stop and Wait (slow sequential process), GBN (go-back-n), SR (selective-repeat)
      iii. FEC: use extra bits to correct errors (or character bits are sent multiple times and receiver accepts the conforming ones) (often used in audio storage and playback devices, sometimes in conjunction with ARQ)
         1. Given two messages A (10101010) and B (01010101) and receiver knows it's one or the other, calculate # of unmatched bits to A and B, the one with shorter distance is most likely correct
         2. Use redundancy in transmitting info using some predetermined algorithm; redundant bit(s) may be some function of original bits; use FEC when retransmits are costly/impossible, like broadcasting to multiple receivers
         3. Checksum (bytes of data converted to 16-bit int and summed; 1s comp of sum becomes checksum in segment header; little packet overhead of only 16-bits but not as good as CRC) (used at transport layer)
         4. CRC (cyclic redundancy check): D * 2^r XOR R (used at link layer)
   c. Multiple access links and protocol: single shared broadcast channel (links include point-to-point or broadcast/shared wire/medium)
      i. Interference: multiple simultaneous transmissions by nodes; collision: multiple signals received by a node at same time
      ii. Multiple access protocol: distributed protocol that determines how nodes share channel (when to transmit); communication about channel sharing uses channel itself
         1. Ideally, we have rate R bps; 1 node will use rate R, M nodes will each send at R/M; fully decentralized and simple, no clock synchronization or special coordinating node
         2. Channel partitioning protocols like TDM or FDM (time/freq division multiplexing) are simple, eliminate collision, and very fair; every node gets R/M speed; however, inefficient, since a single node sending still only gets R/M
         3. CDMA (code division multiple access) uses something similar with unique codes assigned to nodes; used commonly in cellular telephony
         4. Random access MAC protocol (the ALOHAs, the CSMAs): how to detect collisions and recover from them; on collision, wait a random delay and try retransmitting the frame after
      iii. Slotted ALOHA: when node gets fresh frame, transmits in next slot; if no collision, node sends new frame in next slot; else, retransmits in each slot after until successfully transmitted
         1. Single active node continuously transmits at full rate of channel, highly decentralized and simple, only sync'd slots in nodes; however, collisions waste slots, slots can be idle, nodes might be able to detect collision in less time than transmit time, clock synchronization
      iv. Pure ALOHA: transmit frame immediate when it first arrives without synchronizing; fully decentralized; collision probability increases (frame at t0 collides with frames sent within [t0-1, t0+1])
      v. Computing efficiency, or long-run fraction of successful slots (given many nodes N sending many frames at probability p)
         1. Slotted ALOHA: P(given node has success in slot) = $p(1-p)^{N-1}$, P(any node has success)=$Np(1-p)^{N-1}$
            a. Max efficiency for $Np(1-p)^{N-1}$ with N going to infinity gives 1/e = 0.37; at best, channel is successfully used to transmit 37% of the time
            b. Used in low-data-rate tactical satellite communications, subscriber-based satellite comm networks, mobile telephone call setups, set-top box communications, and contactless RFID
         2. Pure ALOHA: P(given node has success) = P(node transmits)*P(no node transmits at t0-1)*P(no node transmits t+1)=$p(1-p)2^{N-1}$
            a. Max efficiency with optimum p and N going to infinity is 1/(2e) = 0.18; at beast, channel is successfully used to transmit 18% of the time; worse than slotted ALOHA
            b. Apparently used in Ethernet to achieve better efficiency than slotted ALOHA(although Ethernet is considered a CSMA/CD system for detecting another nodes that are transmitting)
   d. CSMA (carrier sense multiple access): listen before transmit; if channel seems idle, transmit entire frame; else if busy, defer transmission (don't interrupt others!)

i. Collisions can still occur; propagation delay means nodes might not hear others' transmissions, wasting entire transmission time

e. CSMA/CD (collision detection): detect collisions within short time, aborting colliding transmissions, reducing channel wastage
   i. Easy in wired LANs to measure signal strengths and compare transmitted/received signals; difficult in wireless, signal strength overwhelmed by local transmission strength (polite conversationalist)
   ii. Tprop = max prop delay between 2 LAN nodes; Ttrans = time to transmit max-size frame; efficiency = 1/(1+5Tprop/Ttrans) = 1 as Tprop→0, Ttrans→infinity; better than ALOHA (cheap, decentralized)
   iii. Process: adapter obtains datagram from network layer, readies link-layer frame and puts the frame adapter buffer; if channel is idle (no signal energy entering adapter from channel), start frame transmit; else, wait until no signal energy; while transmitting, monitor for signal energy from other adapters on that broadcast channel; if we finish transmit without detecting signal, finished; else, abort transmit, wait a random time, and retry at detecting idle channel
      1. Binary exponential backoff algorithm: the more collisions experienced by a frame, the longer the interval we choose our random wait time from

f. MAC address (6-bytes/48-bits); LAN/physical/Ethernet address burned in NIC (network interface card) ROM used locally to get frame from one interface to another in same physical network (eg. 1A-2F-BB-76-09-AD)
   i. Administered by IEEE; unique, MAC flat address is portable since LAN card can be moved to another LAN (unlike IP, which depends on IP subnet attached to node)
   ii. ARP (address resolution protocol): used to determine MAC address given its IP address (encapsulated within link-layer frame but contains both network and link-layer addresses; considered to straddle both layers)
      1. Each IP node (host, router) on LAN has table that maps IP/MAC addresses for some LAN nodes <IP; MAC; TTL>, with TTL (time to live, usually 20 min) limiting how long the mapping is remembered
      2. A wants to send datagram to B on same LAN but doesn't have B's MAC in ARP table; A broadcasts ARP query packet with B's IP to all nodes in LAN; B is one of these, receives ARP packet and replies with its own MAC address as it's sent back to A; A caches IP-MAC address pairs in ARP until TTL (soft state: info that times out unless refreshed); ARP is plug-and-play (nodes create ARP w/o admin)
      3. A wants to send datagram to B via R on another LAN and A knows B's IP address
         a. A creates IP datagram with IP source A, dest B, and then it creates link-layer frame with R's MAC address as dest with A-to-B IP datagram; frame is sent to R
         b. R removes datagram from frame and passes up to IP; R forwards datagram with IP source A, dest B, then creates link-layer frame with B's MAC address as dest with A-to-B IP datagram

g. Ethernet: first and dominant LAN tech; simple and cheap at $20 for NIC
   i. Bus topology (all nodes in same collision domain, can collide with each other) used to be popular; now we use star topology (active switch in center; spokes run separate Ethernet protocol, no collision)
   ii. Ethernet frame encapsulates IP datagram with 7-byte preamble to synchronize receiver/sender clock, 6-byte dest MAC address, higher layer protocol type, and CRC
   iii. Ethernet is connectionless (no handshaking between NICs), unreliable (no ACKs or NAKs, data in dropped frames recovered only if initial sender using rdt like TCP), uses unslotted CSMA/CD with binary backoff

4. Network Layer (logical communication between hosts) (host: computer or device connected to network; host is a node that has network layer host address, last 2 bits of IP) (node: any connection point to network)
   a. VC (virtual circuits) are network-layer connections forming paths (links/routers) between source and desitnation hosts with VC numbers and entries in forwarding table; includes setup, data transfer, and VC teardown
   b. Datagram networks, unlike VCs, do not have to setup or maintain state information; end system simply stamps packet with address of destination system prior to sending
   c. IPv4, subnets
      i. IP: 32-bit identifier for host/router interface (connection from host/router to physical link)
      ii. Subnet: isolated subdivision of IP network, where devices interface and physically reach each other without router; example IP address 223.1.1.x/24, where the first 24-bits define network address
      iii. Example of three routers interconnecting six subnets (223.1.1.x, 223.1.2.x, 223.1.3.x, 223.1.7.x, 223.1.8.x, 223.1.9.x)
      iv. CIDR (classless interdomain routing) generalizes subnet addressing (32-bit IP divided into two parts, with form a.b.c.d/x, x indicating first number of bits in network prefix)
      v. Classful addressing: network prefix of IPs with 8,16, or 24 bits in length were (before CIDR) known as class A, B, and C networks, each supporting up to 2^x-2 hosts (2^8-2=254 usable hosts for C) (# subnets = 2^(x-#x for class))
      vi. IP broadcast address 255.255.255.255: when sent to this address, message delivered to all hosts on same subnet
   d. Intra-domain (AS: autonomous systems, or regionally-aggregated routers): Link State (OSPF, Dijkstra), Distance Vector (RIP, Bellman-Ford)
      i. LS vs DV: LS has O(nE) messages sent (n nodes, E links) and oscillates with $O(n^2)$; LS nodes might advertise wrong link cost but nodes compute own table; DV only exchanges between neighbors with varying convergence time and might route loops or count-to-infinity problem; DV nodes might advertise wrong path cost, but node tables are used by others, may propagate error through network
         1. LS doesn't scale too well, since it stores least-cost path changes for lots of hosts (wastes memory, bandwidth overhead); companies often prefer administrative autonomy (use their preferred algorithm)
      ii. OSPF (open shortest path first): link state algorithm uses Dijkstra
         1. Advertises directly over IP (not TCP or UDP), sent to entire AS via flooding; ad carries one entry per neighbor router
         2. Secured by authentication; multi-path (more than one same-cost path, unlike RIP); hierarchical in large domains
      iii. RIP (routing information protocol): distance vector uses Bellman-Ford
         1. $d_v(z)=\min\{c(u,v)+d_v(z), c(u,x)+d_x(z), c(u,w)+d_w(z), \ldots\}$; nodes send updated distance vector estimate to neighbors; neighbors recalculate and update
         2. Good news travels fast; bad news travels slow (count-to-infinity problem); set a max number of hops (max = 15) to counter the count-to-infinity problem
         3. "Poisoned reverse": if Z goes through Y to reach X, Z tells Y that distance from Z→X is infinite so Y doesn't route through Z to get to X (reduces chances of forming loops)
         4. Advertises through UDP packets, sometimes repeated; RIP routing tables managed by application-level process called route-d (daemon)
   e. Inter-domain: BGP (border gateway protocol)
      i. Most common standard, allows each AS to obtain/propagate subnet reachability from neighbor ASs and determine good routes; subnets advertise own existence; used by ISPs to establish routing with other ISPs
      ii. BGP peers, or router pairs, exchange routing info over semi-permanent TCP connections; AS2 advertising prefix to AS1 means AS2 promises it will forward address datagrams toward that prefix
         1. The peers' TCP connection is called a session; it can be either eBGP (external and spanning two ASs) or iBGP (internal, between routers in same AS)
         2. "route" = prefix + attributes (destinations are not hosts but are prefixes, each prefix being a subnet or collection of subnet)
   f. Intra-AS vs. Inter-AS routing
      i. Policy: Inter-AS admin wants control over how traffic is routed and who routes through its net; Intra-AS has single admin, doesn't need policy decisions
      ii. Performance: Inter-AS may have policy dominate over performance; Intra-AS can focus on performance
      iii. Scale: hierarchical routing saves table size, reduced update traffic
   g. Broadcast routing algorithm: transmitting a packet to be received by every device on the network
      i. Deliver packets from source to all other nodes; idea that source duplication and in-network duplication is inefficient
      ii. (Uncontrolled) Flooding: node receives broadcast packet, sends copy to all neighbors (cycles/broadcast storm) except neighbor from where it came
         1. Simple and elegant; however, fatal flaw in infinite cycles (packet gets passed over and over – this is the broadcast storm that renders the network useless with so many duplicated packets)
      iii. Controlled Flooding: only broadcast packet if it hasn't broadcasted same packet before
         1. Keep track of already broadcasted packet IDs or use RPF (reverse path forwarding) to forward packet only if it arrived on shortest path between node and source
         2. Spanning tree has no redundant packets received by any nodes; that is, nodes forward copies only along spanning tree (in both directions)
      iv. N-way-unicast: given N destination nodes, source node makes N copies of packet and addresses each copy to different destination, then transmits them to the N destinations using unicast routing
         1. Simple, no new network-layer routing protocol, packet duplication, or forwarding functionality needed (unicast: sending of messages to single network destination w/ unique address)
         2. Inefficient (say source node is connected to rest of network via single link; it should just send to that one and have it duplicate and forward it to the rest); some overhead for recipient information
      v. Sequence-number-controlled flooding: source node puts address and broadcast sequence number into broadcast packet, then sends packet to all neighbors
         1. Nodes maintain list of source address and sequence number of each broadcast packet it has already received, duplicated, and forwarded
         2. When node receives broadcast packet, check to see if packet is in this list; if so, drop it; else, duplicate and forward to neighbors except the node from where it came
      vi. RPF controlled flooding (reverse path forwarding/broadcast): router transmits packet on all outgoing links (except from where it came) iff the packet arrived on the link that is on its own shortest unicast path back to source; otherwise, packet is discarded (we know the router will receive packet sooner or later, guaranteed)
         1. RPF does not use unicast; router also does not need to complete shortest path from itself to source; it needs to know next neighbor on its unicast shortest path to sender (determining whether to flood)
      vii. Spanning-tree broadcast: the above two don't prevent redundant broadcast packets, so we build a minimum spanning tree and each node sends packets only to those adjacent to it in the tree
         1. Nodes only need to know the neighbors; but difficulty lies in creation and maintenance of spanning tree
         2. Center-based approach: define a center node/core; nodes then unicast tree-join messages addressed to center node, forwarded via unicast until arrives at center or at spanning tree (new path is branch!)
   h. Multicast routing: allow single source node to send copy of a packet to a subnet of other network nodes (for bulk data transfer like software updates, or streaming continuous media, shared data apps, data feeds, etc.)
      i. Construct trees of routers with local multicast group members
         1. Tree (not all paths between routers used), source-based tree (different tree from each sender to receiver); shared tree (same tree used by all group members)
      ii. Using group-shared tree: center-based tree that includes all edge routers with attached hosts belonging to multicast group; join message forwarded using unicast routing toward center until it hits center or multicast tree
         1. Biggest question is how we choose the center; this option has per group overhead with higher delays and more traffic concentration
      iii. Using source-based tree: multicast routing tree for each source (rather than routing tree for all senders, like above); RPF with source node x is used to create multicast forwarding tree for datagrams originating from x
         1. Unfortunately, routers with no attached hosts that are joined to multicast group still receive unwanted multicast packets; "pruning" messages can be sent/forwarded upstream to router to prevent wasted packet sending
         2. Per source overhead with better delay properties

5. Transport Layer (logical communication between processes)
   a. TCP (transmission control protocol): reliable, in-order, congestion control, flow control, requires connection setup
      i. Slow start, congestion avoidance
      ii. Flow control: TCP receiver has buffer and flow control keeps sender from overflowing receiver's buffer by transmitting too much/fast; goal is to match speed adding/removing from buffer
         1. Advertise spare room in segments using RcvWindow = RcvBuffer – [LastByteRcvd – LastByteRead]; unACKed data only goes into RcvWindow, guaranteeing no overflow
         2. Point: control (throttle) amount of sent data going into buffer so receiver can handle it
      iii. Congestion control: too many sources sending too much data too fast for network to handle (not flow control); lost packets or long delays
         1. Option 1: end-end congestion control (no explicit feedback from network, congestion inferred from end-system observed loss and delay; approach used by TCP)
         2. Option 2: network-assisted congestion control (routers give feedback to end systems, gives single bit SNA, DECbit, TCP/IP ECN, or ATM indicating congestion and explicit rate sender should send at)
         3. Point: control entire network load (traffic entering network) so everyone has a fair share of the network resources
      iv. Fast retransmit: timeout period is too long of a delay before resending lost packets, detect lost segments via 3 DUPACKs with same number and resend that packet before timeout occurs
         1. TCP Tahoe treats these as timeouts, performs "fast retransmit" and sets slow start threshold to ½ CWnd, set Cwnd to 1 MSS, and resets to slow start state
         2. TCP Reno halves Cwnd, sets the slow start threshold to new Cwnd, performs fast retransmit, then enter fast recovery (if ACK times out, reset to slow start state like Tahoe)
   b. TCP congestion control (additive increase, multiplicative decrease): increase transmission rate (window size) while probing for usable bandwidth until loss occurs
      i. Additive increase: increase CongWin by 1 MSS every RTT until loss detected; Mutiplicative decrease: cut CongWin in half after loss (saw tooth behavior - probing for bandwidth)
      ii. Sender limits transmission (LastByteSent – LastByteAcked <= CongWin); roughly, rate = CongWin/RTT Bytes/s; for all of these, CongWin is a dynamic function of perceived network congestion
      iii. Sender perceives congestion by loss event (timeout or 3 DUPACKs) or TCP sender reduces rate and CongWin after loss event; CongWin changed by AIMD, slow start, or conservative after timeout events
   c. Congestion control algorithm
      i. Slow start: connection begins with CongWin=1 MSS (max segment size) so rate is approx. MSS/RTT; increase rate exponentially until first loss event (double CongWin every RTT for every ACK received)
      ii. Congestion avoidance: increase goes from exponential to linear (+ 1 MSS every RTT) when CongWin gets to ½ of value before timeout (threshold set to ½ of CongWin just before loss event)
         1. Minor congestion: after 3 DUPACKs, CongWin is cut in half and then grows linearly (go to fast recovery)
         2. Major congestion: after timeout event (more alarming), CongWin set to 1 MSS, window grows exponentially to threshold, then grows linearly (go back to slow start)
      iii. Fast retransmit (see above)
      iv. Fast recovery: add 1 MSS to cwnd for each DUPACK received for missing segment that caused fast recovery
      v. If 3 DUPACKs, Tahoe sets Cwnd to 1 MSS, Reno sets it to ½ Cwnd; if Timeout, both Tahoe and Reho sets Cwnd to 1 MSS

6. Application Layer
   a. DNS (domain name system): distributed database hierarchizing many name servers; application-layer protocol in which host, routers, and name servers communicate to resolve names (maps name/IP translation)
      i. Core internet function, complexity at network's "edge"
      ii. Not centralized: single point of failure, traffic volume from all DNS queries (all HTTP requests/email messages), distant centralized DB from some clients, maintaining all Internet records; basically it can't scale
      iii. Client wants amazon.com's IP: queries root server to find .com DNS server, then queries .com DNS server to get amazon.com DNS server, then queries amazon's authoritative DNS server to get amazon's IP
      iv. When host sends initial DNS query to root DNS server, it's sent to local DNS server to act as proxy, forwarding query into hierarchy
      v. After name server larns of a mapping, it caches it until timeout in format (hostname/domain/alias, value/IP, type A/NS/CNAME/MX, ttl)
   b. HTTP (hypertext transfer protocol): web's application layer protocol for fetching web pages consisting of objects (HTML, JPEGs, base-HTML, etc.)
      i. HTTP uses TCP, stateless; RTT (response time

ii. HTTP 1.0 nonpersistent (one object set over TCP connection), HTTP 1.1 persistent (multiple objects sent over same TCP connection)
iii. Nonpersistent HTTP, not in parallel: 2*RTT to download base HTML + n*2RTT to download n referenced objects
iv. Nonpersistent HTTP, x in parallel: 2*RTT to download base HTML + ceil(n/x)*RTT to download n referenced objects with x objects in parallel
v. Persistent HTTP, pipelined: 2*RTT to download base HTML + RTT to download n referenced objects

c. DHCP (dynamic host configuration protocol): host dynamically gets IP address from AS server upon joining; plug-and-play protocol means it automates network aspect of connecting host to network
i. Client-server protocol, where client is newly-arrived host trying to obtain network config info, including IP for itself; often each subnet has its own DHCP server
ii. Host discovers, DHCP server offers some IP with a lease time (time that IP will remain valid), host requests a specific IP (sometimes from more than one offer), server ACKs and sends address

d. NAT (network address translation): NAT switch/router looks like one device with single IP; traffic leaving/entering home router must have single IP; using NAT translation table, we can further separate these to individual hosts within that network by using ports (although many argue ports should be for processes only, not hosts, and that we should just use IPv6 rather than trying to patch up IPv4 using NAT)

e. ICMP (internet control message protocol): the third main component of network layer (the other two being IP protocol and Internet routing protocols like RIP, OSPF, BGP)
i. Used by hosts and routers to communicate network-layer info to each other, usually for error reporting; often ICMP is considered part of IP but is architecturally above IP (carried via IP datagrams, just like TCP/UDP)
ii. Messages contain type and code field and contain header and first 8 bytes of the IP datagram that caused the ICMP message to be generated (thus sender can determine datagram that caused error)

7. Web Page Request (procedure through application, transport, network, link layers)
a. Student attaches laptop to campus network, requests and receives www.google.com; laptop needs its own IP address, IP of first-hop router, and IP of DNS server; use DHCP
b. DHCP request encapsulated in UDP → in IP → in 802.3 Ethernet; Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server; Ethernet demuxed to IP → to UDP → to DHCP
c. DHCP server formulates DHCP ACK containing client's IP address, IP of first-hop router for client, name and IP of DNS server
d. Encapsulation at DHCP server, frame forwarded (switch learning) through LAN, demultiplexing at client
e. DHCP client receives DHCP ACK reply; now clent has IP address, IP of first-hop router, and knows name/IP of DNS server
f. Before sending HTTP request, need IP of google.com; use DNS
g. DNS query created and encapsulated in UDP → in IP → in Ethernet; need MAC of router interface to send frame to router; use ARP
h. ARP query broadcast received by router, sending ARP reply with MAC of router interface; now client has MAC of first-hop router and can send frame containing DNS query
i. IP datagram containing DNS query forwarded via LAN switch from client to first-hop router
j. IP datagram forwarded from campus network into Comcast network, routed to DNS server (tables made by RIP, OSPF, IS-IS and/or BGP routing protocols); demuxed to DNS server
k. DNS server replies client with IP of google.com; to send HTTP request, client opens TCP socket to web server; TCP SYN segment (starts 3-way handshake) inter-domain routed to web server, gets back TCP SYNACK
l. TCP connection established! HTTP request sent to TCP socket; IP datagram containing HTTP request routed to google.com; web server responds with HTTP reply containing web page; IP datagram containing HTTP reply routed back to client

CS118 Midterm

- Packet switching: messages sent as packets (small data chunks) with delay=Length/TransmissionRate (L/R)
  - For P packets over series of N links, delay=(N+P-1)L/R
- Circuit Switching vs Packet Switching
  - Circuit switching: reserved communication; establishes connection (circuit), guaranteed constant transmission rate (Ex: Telephone network, video conference; good for real-time services)
    - Frequency-division multiplexing (FDM) or time-division multiplexing (TDM); divided by frequency or time
  - Packet switching: no links reserved, so packet may delay in buffer before transmission; no guarantees, unpredictable, but better sharing of transmission capacity, and easier/cheaper than circuit switching (allocates link use on demand)
- Processing delay + queuing delay + transmission delay + propagation delay = total nodal delay
  - Processing delay: time to process packet header to decide where packet goes; Queuing delay: time waiting to be transmitted into link; Transmission delay: L/R, time taken to push packet bits into link (for a 100 Mbps Ethernet link, R=100 Mbps); Propagation delay: time taken to propagate to next router (usually at rate of 2 or 3 * 10^8 m/s, or speed of light)
    - Propagation delay is d/s, where d is distance between routers, s is propagation speed
  - Traffic intensity=La/R, where a is rate of packets/s that arrive at queue; La/R should be <= 1
- Internet protocol stack: Application (HTTP, SMTP, FTP, etc.) > Transport (UDP, TCP) > Network > Link > Physical
- Communication architectures: client-server vs P2P (self-scalable)
- Socket: software interface between application and transport layer that sends/receives messages (window analogy)
- IP address (address of the host) and port number (identifier for the receiving process/socket) are needed to communicate
- Reliable data transfer (RDT): guaranteed data delivery service
  - Transport-layer protocol could guarantee throughput rate (rate of bits delivered to recipient); Guarantee timing (say, 100 ms) in which every bit must be received in; Security can be boosted by encrypting all data before leaving socket and decrypting it on arrival
- TCP vs UDP (transport protocols) [no throughput or timing guarantees for either, although Internet usually works anyway]
  - TCP is connection-oriented, RDT (all data in proper order, no errors), congestion-control
    - Congestion control throttles sending process when network between client/server is congested; it also limits TCP connection to fair share of bandwidth; promotes general wellness of the Internet, not the client
  - UDP is lightweight, minimal, connection-less (no handshaking), unreliable (not all data, may not be in order)
  - Most applications use TCP (email-SMTP, remote-Telnet, web-HTTP, file-FTP) but internet telephone uses UDP first
- HTTP is stateless
  - Non-persistent (separate TCP connections for multiple objects) vs persistent (multiple responses over same TCP)
    - RTT (round-trip time) is time taken for packet to reach server and back to client; Includes packet propagation delays, packet queuing delays
    - 1 RTT to handshake, 1 RTT to request file and receive it, plus transmission time at the server of HTML file
  - Persistent (TCP connections established and left open, maintained for each requested object; used with pipelining)
  - HTTP request message: request line + header lines + empty line + entity body
    - GET /blah.txt HTTP/1.1 (request line: method + URL + HTTP version)
      Host: www.google.com (start of header lines) Connection: close User-agent: Mozilla/5.0 Accept-language: fr
    - HTTP response message: status line + 6 header line + empty line + entity body: HTTP/1.1 200 OK
  - Cookies, Web Caches (proxy servers), Conditional GET
- FTP: user provides authorization from FTP client to server; if passed, client's local files can mingle with server's remote files
  - HTTP vs FTP; both are file transfer on top of TCP; FTP keeps state, uses two parallel TCP connections: control, data
    - Control information sent "out-of"band like SMTP; unlike HTTP, which sends it in one "in-band" message
    - Control connection remains open with user's ID and information; data connection is non-persistent
- SMTP: user agents send/retrieve via mail servers through SMTP; uses RDT of TCP with persistence
  - Does not use intermediate mail servers; email stays in sending server's queue until recipient's server is free; HELO (HELLO), MAIL FROM, RCPT TO, DATA, QUIT
  - Unlike HTTP's pull protocol (TCP initiated by viewer), SMTP is a push protocol (that is, TCP initiated by sender)
  - Also, SMTP message only works in 7-bit ASCII; SMTP objects are placed in one message, not encapsulated by HTTP
  - Agent A --SMTP--> Mail Server A --SMTP--> Mail Server B --POP, IMAP, HTTP--> Agent B
  - Post Office Protocol v3 (POP3), Internet Mail Access Protocol (IMAP), or HTTP used to pull messages after pushed
    - POP: authorization, transaction, and update stages; download-and-keep/delete options
      - Simple, no means to save messages on remote folders (unlike IMAP); Also, IMAP allows a user to obtain components of a message (say, in low bandwidth conditions)
- DNS Servers & Caching
- Transport-layer protocol provides logical communication (as though directly connected) between application processes
  - Network-layer protocol provides logical communication between hosts (USPS deliver to house, rather than people)
- A packet in transport-layer is known as a "segment"; in network-layer, a "datagram"
  - Although TCP uses segment, UDP might use datagram
- IP vs UDP vs TCP: IP uses best-effort, no guarantees in communication (unreliable) ;UDP provides process-to-process data delivery and error checking (also unreliable); TCP provides RDT (flow control, sequence numbers, acks, timers) and congestion control for the general internet
  - Demultiplexing: delivering data in transport-layer segment to correct socket (receiving and sorting);
  - Multiplexing: gather data chunks at source host from sockets, encapsulate with headers, pass to network layer
  - For TCP, four-tuples (src IP, src port, dest, IP, dest port) are used to demultiplex segment to correct socket
  - UDP almost only provides demuxes/muxes service to pass data between network layer and application processes
    - DNS uses UDP; UDP allows more application level control (there's no built-in congestion control, auto resend, etc.), no connection establishment/handshake (fast!), and no connection state, small overhead
  - UDP uses checksum (before sending A, B, C, add them and then 1s complement; add at end to see if it's 11…11)
- RDT (using unidirectional data transfer as an example – notice the layer below RDT protocol is unreliable, like TCP on IP)
  - rdt1.0 has send/rcv FSMs with 1 state each for a perfectly reliable channel
    - On rdt_send(data), sending side makes packet and udt_sends; On rdt_rcv(packet), receiving side extracts data from packet and delivers_data
  - rdt2.0 over channel with bit errors (stop-and-wait via ACK/NAK); error detection, receiver feedback, retransmission
    - On rdt_send, we make packet and send, then resend on NAK; on ACK, return; On rdt_rcv, if corrupt, keep resending NAK; else, extract data, deliver, and send ACK
  - rdt2.1 fixes duplicate resent packets and garbled ACK/NAK via sequence numbers 0 and 1
    - On rdt_send, make packet with 0; if corrupt or NAK, resend; if ACK, proceed to repeat with packet 1, etc.
    - On rdt_rcv with noncorrupt seq0 packet, extract/deliver, then send ACK; if corrupt, send NAK; remain if next packet is still seq0; if noncorrupt seq1 packet received, extract/deliver, send ACK; etc.
  - rdt2.2 tacks on seq number to ACK or NACK, as in ACK0 or ACK1, etc.
  - rdt3.0 over lossy channel with bit errors and lost packets
    - Again, using alternating-bit protocol (0, 1, 0, 1, …) and is a stop-and-wait protocol
    - On rdt_send, make seq0 packet and start timer; wait for ACK0 (ignore corrupt ACK or ACK1) until timeout, then resend packet and start timer, etc.
    - Once ACK0 is received and noncorrupt, stop timer and wait for next rdt_send, which sends seq1 packet
    - Now we wait for ACK1 (ignore corrupt ACK or ACK0) until timeout, then resent packet and start timer, etc.
    - For the sender, anything that screws up means we retransmit (retransmit is a panacea)
    - As for the receiver, if it is sent the same packet seq# twice in a row, send ACK# both times but drop all repeated packets
  - Not good performance from rdt3.0; d_trans = L/R = (8000 bits/pkt)/(10^9 bits/s) = 8 microseconds
    - Assuming RTT is 30 ms, then t = 0.5RTT + L/R = 30.008 ms
    - Utilization U_sender = (L/R)/(RTT + L/R) = 0.008/30.008 = 0.00027 [very bad!]
  - Solution is to pipeline by transmitting multiple packets before waiting for ACK; requires more seq# and buffering
  - Pipelined error recovery: Go-Back-N (GBN) or Selective Repeat (SR)
    - GBN, or sliding-window protocol: given some N window size (max number of unACKed packets to send)
      - Invoked from above: if <N unACKed packets, create/send packet; otherwise, window is full
      - ACK receipt: ACK #n means cumulative acknowledgement up to the nth packet
      - Timeout event: on timeout, sender resends all send but unACKed packets
      - On the receiver side, send ACK n if we've received up to n-1; otherwise, send last correct number

- Out-of-order packets are discarded for simplicity and retransmitted later
- GBN is similar to the RDT aspect of TCP

- SR: need this, since GBN sucks with large N and bandwidth-delay product; it retransmits many packets unnecessarily
  - SR individually ACKs correct packets, still using N, and accepts out of order packets
  - Receiver's reACKs already received packets with seq # below current window base
  - Sender and receiver each maintain their own window
  - Important: Window Size N = Sequence Number Space is WRONG for SR
  - Solution: Window Size N <= ½ Sequence Number Space [this is for SR]

- TCP: neither TDM nor FDM, no TCP connection state kept either; TCP provides point-to-point, full-duplex service
  - No multicasting allowed (that is, one sender to many receivers, or vice versa)
  - When client establishes connection with server, we have a three-way-handshake
    - Max segment size (MSS) is max data amt that can be grabbed into a segment to send as buffered data
  - Timer, using latest EstimatedRTT and DevRTT, is doubled an interval on retransmit for some congestion control
  - Fast Retransmit using duplicate ACKs in special cases

    Arrival of in-order segment with expected sequence number. All data up to expected sequence number already acknowledged.
    Delayed ACK. Wait up to 500 msec for arrival of another in-order segment. If next in-order segment does not arrive in this interval, send an ACK.
    Arrival of in-order segment with expected sequence number. One other in-order segment waiting for ACK transmission.
    Immediately send single cumulative ACK, ACKing both in-order segments.
    Arrival of out-of-order segment with higher-than-expected sequence number. Gap detected.
    Immediately send duplicate ACK, indicating sequence number of next expected byte (which is the lower end of the gap).
    Arrival of segment that partially or completely fills in gap in received data.
    Immediately send ACK, provided that segment starts at the lower end of gap.
  - TCP looks like GBN since it doesn't ACK individually; it maintains smallest number of transmitted, unACKed byte
  - However, TCP often buffers correct but out-of-order segments; also, TCP retransmits only one segment at most
    - Selective acknowledgement, a modification to TCP, allows TCP to ACK out-of-order segments selectively; TCP is a hybrid of GBN and SR protocols
- Flow Control: service that prevents sender overflowing the receiver's buffer
  - Speed matches between sender's speed of sending and receiver's speed of reading
  - Sender maintains a receive window that tells it how much buffer space is free on the receiver's side
    - rwnd = RcvBuffer - (LastByteRcvd – LastByteRead)
    - For flow control, maintain that LastByteSent – LastByteAcked <= rwnd
    - Say Host B has full buffer such that rwnd=0; if space clears up but B never sends to A, A doesn't know
    - Host A is required by TCP to continue sending 1 data byte segment even when B's rwnd=0 to get ACKs
- TCP Connection Management
  - SYN: no app data, but SYN bit flag is 1 and involves a client_isn sequence number to start connection with server
  - SYNACK: no app data; server extracts TCP SYN, allocates buffers/variables to connection; SYN bit is again 1, acknowledges with client_isn+1, and gives back server_isn (its own initial sequence number)
  - ACK: client allocates buffers/variables to connection; sends segment (app data!) to acknowledge with server_isn+1 and SYN bit 0
  - This ends three-way handshake; now client-server can send data segments to each other with SYN=0
  - To close, client sends TCP segment to server with FIN bit=1; server ACKs, then sends its own FIN=1, and client ACKs
- Congestion Control (network congestion: too many sources sending data at too high a rate)
  - Need to throttle senders when network is congested
  - Scenario 1: 2 hosts share sending rates (R/2 for each)
    - Large queuing delays result as packet sending rate nears link capacity
  - Scenario 2: 2 hosts have reliable connection with retransmission allowed and router's buffer is now finite; then congestion depends on retransmission
    - If congested, retransmissions must compensate for buffer overflow's dropped packets
    - If congested, unnecessarily retransmissions in large delays may waste router link bandwidth in order to forward copies of packets that are unneeded
  - Scenario 3: 4 hosts overlapping two-hop paths with finite buffers
    - Hosts A-C and B-D must share router R2's buffer space; as B-D's load in R2 increases, A-C's load falls; as offered load approaches infinity, B-D packets fill R2 and A-C throughput goes to zero
    - Heavy traffic/congestion means A-C has no throughput (since B-D took over)
    - Dropping a packet due to congestion wastes transmission capacity of upstream links that forwarded it
  - End-to-end congestion control
    - Network layer (IP) gives no feedback; TCP must check its own segment less via timeout or 3 DUPACK
    - TCP must decrease window size accordingly; or use increased RTT delay values to indicate congestion
  - Network-assisted congestion control
    - Network-layer (routers) give feedback to sender about network congestion; used in ATM available bit-rate (ABR) congestion control; XCP protocol tells how each source should change transmission rate
    - Network can also send a choke packet that notifies sender of congestion indication (takes a full RTT)
- TCP Congestion Control: have sender limit rate of sending traffic into connection (based on perceived network congestion)
  - Variable congestion window (cwnd) limits rate of sending; LastByteSent – LastByte Acked <= min{cwnd, rwnd}; sender's rate is approx. cwnd/RTT bytes/s
  - Congestion detected by a loss event at TCP sender: timeout or 3 DUPACKs from receiver
  - No loss events (but rather successful ACKs) means cwnd can be increased for faster transmission rate; faster ACK rate means higher cwnd
  - TCP is self-clocking since it changes congestion window size and thus transmission rate based on its own received ACKs
    - Lost segments implies congestion; TCP sender's rate decreased when segment lost
    - ACKed segment means network segments are received, so rate increases when ACK arrives from previously unACKed segment
    - Bandwidth probing: increase rate in response to arriving ACKS until loss occurs, then try again and keep probing
- TCP Congestion-Control Algorithm
  - Slow Start
    - TCP usually initializes cwnd to small value of 1 MSS, so initial sending rate is around MSS/RTT
    - Value of cwnd increases by 1 MSS every time transmitted segment is fist ACKed, exponentially doubling cwnd by MSS every RTT
      - 1 segment sent out; ACKed, so cwnd=2 MSS; 2 segments sent out; two ACKs, so cwnd=4 MSS; 4 segments sent out, etc…
      - On loss event, cwnd=1 and restart process; stop once cwnd_new = cwnd_old/2 (or variable ssthresh, slow start threshold)
      - Once cwnd hits ssthresh, enter congestion-avoidance mode, where cwnd is increased more cautiously
      - Slow start also ends if 3 DUPACKs are detected, and a fast retransmit is done before entering fast recovery
  - Congestion Avoidance
    - At this point, cwnd is about half of the congestion-causing value; now we add 1 MSS to cwnd every RTT
      - Say MSS is 1460 bytes, cwnd=14600 bytes, then RTT sends 10 segments; each ACK adds 1/10 to cwnd for 1 MSS after a RTT
      - Again, ssthresh is ½ cwnd at loss event (or 3 DUPACKs, in which case we add back 3 MSS) and we end to go to fast recovery
  - Fast Retransmit (see above before)
  - Fast Recovery
    - Add 1 MSS to cwnd for each duplicate ACK received for missing segment that caused TCP to enter fast-recovery state
    - When ACK arrives for missing segment, TCP enters congestion-avoidance after deflating cwnd; a timeout causes a transition to slow-start instead after setting cwnd to 1 MSS and ssthresh to half of cwnd when loss event occurred
    - This is optional; TCP Tahoe cuts cwnd to 1 MSS and enters slow start after timeout OR 3 DUPACK; TCP Reno uses fast recovery
  - TCP congestion control is often called additive-increase, multiplicative decrease (AIMD) due to 1 MSS increments ½cwnd decrements; sawtooth behavior on graph comparing cwnd x time
  - Reno algorithm is commonly implemented in TCP; TCP Vegas detects router congestion before packet loss occurs and linearly lowers rate of sending packets; lengthier RTT → congestion
- TCP Average Throughput: 0.75*W/RTT or 1.22*MSS/(RTT*sqrt(L))
- Forwarding (moving packet at router's input link to right output link on path) and Routing (packet route that flows from sender to receiver)
  - Forwarding is router-local action; routing is network-wide process that determines end-to-end paths from source to destination
  - Routers (AKA packet switches in the link-layer) have forwarding table that indicates destination or connection based on header value
  - Connection setup: some network-layer architectures require routers from source to destination to handshake to setup data packet flow
- Network service model of end-to-end packet transport
  - Sending host could guarantee delivery or delivery with bounded delay
  - Other possibilities: in-order packet delivery, guaranteed min bandwidth, guaranteed max jitter (amount of time between transmitting two packets equal to time between their receipt at destination), or security services (secret session key to encrypt)
  - Internet network layer provides best-effort service only (basically, no guarantees)
  - Constant bit rate (CBR) Asynchronous Transfer Mode (ATM) network service: packet flow (cells) with delays and lost/late cells less than guaranteed values
  - ABR ATM network service: cells might be lost, but won't be reordered and a minimum cell transmission rate (MCR) is guaranteed
- Host-to-host connection (virtual-circuit, VC) or connectionless service (datagram) networks
  - Connection networks are implemented in routers in network core and in end systems
- Internet is a datagram network, but ATM and frame relay are VC networks consisting of a path (link/router series), VC numbers, and entries in forwarding table
  - An entry is added/removed in the forwarding table whenever a VC is established or terminates
    - Replaced numbers link-t-link reduces length of packet header VC field and simplifies VC setup
  - VC setup: network layer determines path between sender and receiver, finds VC number for each path, and adds entry in forwarding table in each router
  - Data transfer: packets flow once VC is established
  - VC teardown: network layer updates forwarding tables in each router to indicate VC is now nonexistent
  - All routers along path of two end systems are aware and involved in VC setup and VC network; signaling messages/protocols initiate/terminate the VC
- Datagram network simply involves end system stamping packet with address of destination, and the packet is placed into the network; no setup, no state info
  - Routers use packet destination address to forward via forwarding table; use 32 bit destination address to match the longest prefix for a link interface
- Router forwards/switches packets from incoming to outgoing links
  - Input ports (where forwarding table is consulted; performs link-layer functions); switching fabric connects input to output; output ports; routing processor
  - Routing processor computes/updates forwarding table; switching fabric moves the actual input to output based on table
- Switching techniques: by memory (lookup/forwarding by memory; 1 packet at a time), by bus (input transfers directly to output over shared bus with a temporary switch-internal label in the header to indicate output port) ; interconnection network