**+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Test 1 Review**

**+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**CHP 1 - Basics:**

* Protocols: Define format and order of messages sent and received among network entities, and actions taken on message transmission or message receipt
  + Human Protocols: Specific messages sent, and specific actions being taken when messages are received
  + Network Protocols: Involves machines rather than humans; all communication activity in the Internet is governed by protocols
* Network Edge: Computers and other devices that are connected to the Internet
* Access Networks: Physically connects an end system to the first router (“edge router”) on a path from the end system to any other distant end system
* Network Core: Mesh of packet switches and links that interconnects the Internet’s end systems
* How to connect end systems to edge router?
  + Residential access nets, institutional access networks (school, company), and mobile access networks
* Host Sending Function: Takes application message, breaks it into smaller chunks (known as packets) of length L bits, and transmits those packets into access network at transmission rate R
  + Packet Transmission Delay: L (bits) / R (bits/sec)
  + The transmission rate is otherwise known as the link capacity / link bandwidth
* Bit: Propagates between transmitter/receiver pairs
* Physical Link: What lies between the transmitter and the receiver
* Guided (Physical) Media: Signals propagate in solid media (copper, fiber, coax)
  + Twisted Pair (TP): Two insulated copper wires
  + Coaxial Cable: Two concentric copper conductors; bidirectional; broadband (multiple channels on the cable)
  + Fiber Optic Cable: Glass fiber carrying light pulses, each pulse being a bit; high-speed point-to-point transmission; low error rate due to repeaters being spaced apart; immune to electromagnetic noise (interference)
* Unguided Media: Signals propagate freely (e.g., radio)
  + Radio: Signal carried in electromagnetic spectrum; bidirectional; affected by reflection, obstruction by objects, and interference
    - Terrestrial Microwave: Up to 45 Mbps channels
    - LAN (e.g., WiFi): 11 Mbps, 54 Mbps
    - Wide-area (e.g., Cellular): 3G cellular; few Mbps
    - Satellite: Kbps to 45 Mbps channel (or multiple smaller channels); 270 msec end-to-end delay; geosynchronous versus low altitude
* Packet-Switching: Hosts break application-layer messages into packets; forward packets from one router to the next, across links on path from source to destination; each packet transmitted at *full* transmission rate of the link; allows more users to use a network, while giving the same performance as circuit switching with minimal simultaneous users
  + Takes L/R seconds to transmit (push out) L-bit packet into link at R bps
  + Store and Forward: Entire packet must arrive at router before it can be transmitted on next link
  + End-to-end Delay: NL/R, where N = number of links, R = the transmission rate for all the links, and L = length of the packet
    - For calculations involving more than one packet, multiply the formula above by the number of packets
  + It is great for bursty data and resource sharing; much simpler to implement, due to not having a call setup
  + Bandwidth guarantees are needed for audio/video apps, so it remains easier to use circuit switching for those applications
  + Excessive congestion is possible, due to packet delay / loss
    - Protocols are needed for reliable data transfer and congestion control
  + Loss/delay occurs when packets queue in router buffers; packet arrival rate to link (temporarily) exceeds output link capacity, so packets queue and wait for turn
    - Packet Loss: Queue (aka buffer) preceding link in buffer has finite capacity; packet arriving to the full queue is dropped (aka lost); the lost packet may be retransmitted by previous node, by source end system, or not at all
    - Nodal Processing Delay: Check bit errors, determine output link; typically < msec
    - Queuing Delay: Time waiting at output link for transmission; depends on congestion level of router
      * La/R (traffic intensity) can determine the queuing delay (where L = packet length in bits, R = link bandwidth in bps, and a = average packet arrival rate)
        + If La/R ~ 0: the average queuing delay is small
        + If La/R ~ 1: the average queuing delay is large
        + If La/R > 1: there is more “work” arriving than can be serviced, so the average delay is infinite!
    - Transmission Delay: L/R (where L = packet length in bits and R = link bandwidth in bps)
    - Propagation Delay: d/s (where d = length of physical link and s = propagation speed in medium); s is in the range of 2 \* 108 m/s to 3 \* 108 m/s
    - Total Nodal Delay: The sum of all four previous delays
    - End-to-End Delay: N(dproc + dtrans + dprop), where N = the number of routers + the destination host, dproc = the processing delay at each router and at the source host, dtrans = L/R (L = packet size, R = transmission rate), and dprop = the propagation on each link.
  + Throughput: Rate (bits/time unit) at which bits transferred between sender/receiver
    - Instantaneous: Rate at a given point in time (in bits/sec) at which Host B is receiving the file.
    - Average: Rate over a longer period of time; if the file consists of F bits and the transfer takes T seconds for Host B to receive all F bits, then it is F/T bits/sec.
    - Bottleneck Link: Link on end-to-end path that constrains end-to-end throughput
      * If RS < RC, then the bits pumped by the server will “flow” right through the router and arrive at the client at a rate of RS bps, giving a throughput of RS bps.
      * If RC < RS, then the router will not be able to forward bits as quickly as it receives them; bits leave the router at rate RC, giving an end-to-end throughput of RC bps.
      * For a simple two-link network, the throughput is min{RC, RS} (where RC = transmission rate of the link between the router and the client and RS = transmission rate of the link between the server and the router).
* Circuit Switching: End-to-end resources are allocated and reserved for “call” between source and destination; implemented with either FDM or TDM
  + Frequency-Division Multiplexing (FDM): The link dedicates a frequency band to each connection for the duration of the connection; the width of the band is called the bandwidth
  + Time-Division Multiplexing (TDM): Time is divided into frames of fixed duration, and each frame is divided into a fixed number of time slots; when the network establishes a connection across a link, the network dedicates one time slot in every frame to this connection
  + No sharing resources, giving a circuit-like (guaranteed) performance
  + The circuit segment remains idle if not being used by a call (no sharing); referred to as silent periods
* Routing Protocols: Determines source-destination route taken by packets via routing algorithms that automatically set the forwarding tables
* Forwarding Table: Maps destination addresses (or portions of the destination addresses) to that router’s outbound links; when a packet arrives at a router, the router examines the address and searches its forwarding table, using this destination address to find the appropriate outbound link
* Access Networks: End systems are connected to edge routers through residential access nets, institutional access networks (school, company), and mobile access networks.
  + PoP: A group of one or more routers (at the same location) in the provider’s network where customer ISPs can connect into the provider ISP
  + Peering: When a pair of nearby ISPs at the same level of the hierarchy directly connect their networks together so that all the traffic between them passes over the direct connection rather than through upstream intermediaries
  + IXP: A meeting point where multiple ISPs can peer together.
  + Regional ISP: An ISP in a region to which access ISPs in the region connect
  + Tier-1 Commercial ISPs: National and international coverage (e.g., Level 3, Sprint, AT&TT, and NTT); there are approximately 12 of these in the world
  + Content Provider Network: Private network that connects its data centers to Internet, often bypassing tier-1 and regional ISPs (e.g., Google)
  + Multi-Home: When an ISP (any ISP, other than a tier-1 ISP) connects to two or more provider ISPs
* Protocol Layering: Deals with complex systems; an explicit structure allows identification and a relationship of complex system pieces; can be implemented in software, hardware, or in a combination of the two.
  + Modularization: Eases maintenance and updating of system
    - A change of the implementation of a layer’s service is transparent to the rest of the system (e.g., a change in gate procedure doesn’t affect the rest of the system)
* Internet Protocol Stack:
  + Application: Supporting network applications (e.g., FTP, SMTP, HTTP)
  + Transport: Process-to-process data transfer (e.g., TCP, UDP)
  + Network: Routing of datagrams from source to destination (e.g., IP, routing protocols)
  + Link: Data transfer between neighboring network elements (e.g., Ethernet, 802.11 (WiFi), PPP); packets in this layer are referred to as frames
  + Physical: Bits “on the wire”; moves the *individual bits* within the frame from one node to the next
  + The transport layer transports application-layer messages (packets of information in the application layer) between application endpoints; the transport-layer protocol in a source host passes a transport-layer segment and destination address to the network layer, which then transports it to the transport-layer in the destination host; the link layer is responsible for receiving the datagram from the network layer and passing it to the next node along the route, where it passes it back up to the network layer; the physical layer moves the individual bits within each frame from one node to the next.
    - In this way, each layer relies on the services of the layer below it for data transportation to be possible.
  + ISO/OSI Reference Model: Has a presentation and session layer (in that order) between the application and transport layer
    - Presentation: Allows applications to interpret the meaning of data (e.g., encryption, compression, machine-specific conventions)
    - Session: Synchronization, checkpointing, and recovery of data exchange
    - The internet stack is “missing” these layers; if these services are needed, they must be implemented in the application
* Internet History:
  + 1961: Queuing theory shows effectiveness of packet switching
  + 1964: Packet-switching implemented in military nets
  + 1967: ARPAnet is conceived by Advanced Research Projects Agency
  + 1969: First ARPAnet node operational
  + 1970: ALOHAnet satellite network is released in Hawaii
  + 1972: ARPAnet public demo is released, NCP (Network Control Protocol) has first host-to-host protocol, the first e-mail program is released, and ARPAnet has 15 nodes
  + 1974: Cerf and Kahn - Architecture for interconnecting networks
  + 1976: Ethernet at Xerox PARC
  + 1979: ARPAnet has 200 nodes
  + Late 70’s: Proprietary architectures (DECnet, SNA, XNA) released; switching fixed length packets (ATM precursor)
  + 1982: SMTP e-mail protocol defined
  + 1983: Deployment of TCP/IP, and DNS defined for name-to-IP-address translation
  + 1985: FTP protocol defined
  + 1988: TCP congestion control
  + Early 1990’s: ARPAnet decommissioned, web hypertext released (HTML, HTTP)
  + 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned in 1995)
  + 1994: Mosaic released (later changed to Netscape)
  + Late 1990’s – 2000’s: Commercialization of the web, network security at a forefront, backbone links running at Gbps, 50 million hosts, 100 million+ users, and new apps released (P2P file sharing, instant messaging, etc)

**CHP 2 – Application Layer:**

* Network apps: E-mail, web, text messaging, remote login, P2P file sharing, multi-user network games, streaming stored video (YouTube, Hulu, Netflix), Voice over IP (VoIP, e.g., Skype), real-time video conferencing, social networking, searching, etc
* Creating a Network App:
  + Write programs that run on (different) end systems and communicate over a network (e.g., web server software communicates with browser software)
  + No need to write software for network-core devices – they do not run user applications
  + Applications on end systems allows for rapid app development and propagation
* Application Architectures: Can be structured as client-server or peer-to-peer (P2P)
  + Client-Server Architecture:
    - Server: Always-on host; permanent IP address; data centers for scaling
    - Clients: Communicate with the server; may be intermittently connected; may have dynamic IP addresses; do not communicate directly with each other
  + P2P Architecture: No always-on server; arbitrary end systems directly communicate with each other; peers request service from other peers, provide service in return to other peers
    - Self Scalability: New peers bring new service capacity, as well as new service demands
    - Peers are intermittently connected and change IP addresses – complex management
* Process: Program running within a host; within same host, two processes communicate using inter-process communication (defined by OS); processes in different hosts communicate by exchanging messages; applications with P2P architectures have client processes and server processes
  + Client Process: Process that initiates communication
  + Server Process: Process that waits to be connected
* Sockets: Process sends/receives messages to/from its socket; analogous to a door
  + Sending a process shoves a message out of the “door”; relies on transport infrastructure on other side of “door” to deliver message to socket at receiving process
* Addressing Processes:
  + To receive messages, process must have identifier; host-device has unique 32-bit IP address
  + Does the IP address of the host on which the process runs suffice for identifying the process?
    - No, *any* processes can be running on the same host
  + Identifier: Includes both IP address and port numbers associated with process on host (HTTP server port is 80, mail server port is 25, etc)
* The app-layer protocol defines the types of messages exchanged (e.g., request, response), the message syntax and semantics, and the rules for when and how processes send and respond to messages
  + Message Syntax: What fields in message and how fields are delineated
  + Message Semantics: Meaning of information in the fields
  + Open Protocols: Defined in RFCs; allows for interoperability (e.g., HTTP, SMTP)
  + Proprietary Protocols: Skype, etc
* What transport service does an app need?
  + Data Integrity: Some apps (e.g., file transfer, web transactions) require 100% reliable data transfer; other apps (e.g., audio) can tolerate some loss
  + Timing: Some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”
  + Throughput: Some apps (e.g., multimedia) require minimum amount of throughput to be “effective”; other apps (“elastic apps”) make use of whatever throughput they get
  + Security: Encryption, data integrity, etc
* TCP Service:
  + Reliable Transport between sending and receiving process
  + Flow Control: Sender won’t overwhelm receiver
  + Congestion Control: Throttle sender when network overloaded
  + Does Not Provide: Timing, minimum throughput guarantee, or security
  + Connection-Oriented: Setup required between client and server processes
  + Usually used for e-mail, file transfer, remote terminal access, and websites
* UDP Service:
  + Unreliable Data Transfer between sending and receiving process
  + Does Not Provide: Reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup
  + Usually used for multimedia streaming or internet telephony
* Securing TCP:
  + TCP and UDP: No encryption; cleartext passwords sent into socket, traverse internet in cleartext
  + SSL: Provides encrypted TCP connection; data integrity; end-point authentication; apps use SSL libraries, which “talk” to TCP
    - The socket API takes any cleartext passwords sent into the socket and traverses them across the internet after encryption
* A web page consists of objects; each object can be an HTML file, JPEG image, Java applet, audio file, etc.
  + Consists of base HTML-file which includes several referenced objects; each object is addressable by a URL
* HTTP (Hypertext Transfer Protocol): Web’s application layer protocol; is a client/server model
  + Client: Browser that requests, receives (using HTTP protocol), and “displays” web objects
  + Server: Web server sends (using HTTP protocol) objects in response to requests
  + Uses TCP: Client initiates TCP connection (creates socket) to server, port 80; server accepts TCP connection from client
    - HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server); TCP connection closed
  + HTTP is “Stateless”: Server maintains no information about past client requests
    - Protocols that maintain “state” are complex; past history (state) must be maintained; if server/client crashes, their views of “state” may be inconsistent, must be reconciled
  + Response Time (RTT): Time for a small packet to travel from client to server and back
  + Non-Persistent HTTP: At most one object sent over TCP connection, connection then closed; downloading multiple objects required multiple connections
    - HTTP Response Time: One RTT to initiate TCP connection, one RTT for HTTP request and first few bytes of HTTP response to return; file transmission time; response time = 2RTT + file transmission time
  + Persistent HTTP: Multiple objects can be sent over single TCP connection between client and server
* Suppose user enters URL [www.someschool.edu/somedepartment/home.index](http://www.someschool.edu/somedepartment/home.index) (contains text, references to 10 jpeg images)
  + HTTP client initiates TCP connection to HTTP server (process) at [www.someschool.edu](http://www.someschool.edu) on port 80
  + HTTP server at host [www.someschool.edu](http://www.someschool.edu) waiting for TCP connection at port 80. “accepts” connection, notifying client
  + HTTP client sends HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object somedepartment/home.index
  + HTTP server receives request message, forms response message containing requested object, and sends message into its socket
  + HTTP server closes TCP connection
  + HTTP client receives response message containing HTML file, displays HTML. Parsing HTML file, finds 10 referenced jpeg objects
  + Above steps repeated for each of 10 jpeg objects

**Other Important Information:**

* BYTE \* 8 = BIT
* 1 KBPS = 10^3 BPS
* 1 MBPS = 10^6 BPS
* Binomial Distribution: C(n, x) \* px \* (1 – p)n – x, where p = probability of a success on an individual trial, n = number of trials, x = total number of “successes” (pass or fail, heads or tails, etc.), and C(n, x) = n!/((n – x)!(x!)).