**Computer Networking**

*(Learning from Udemy’s COMPTIA Network (N10-007))*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Network Models** | | | | | | |
| - Explain devices, applications, protocols and services at their appropriate OS1 layers  - Models are used to represent how networks function  - There are two very popular network models  1. OSI seven-layer model  2. TCP/IP model  - Each layer  - Has a distinct function  - Has a name and a number | | | | | | |
| **OSI model** | | | | **TCP/IP model** | | |
| - Specifies 7 distinct functions that a network must do  - Older, more detailed | | | | - Looks at applications as applications  - E.g. email, FTP, telnet  - More modern, not as detailed but not as complicated | | |
| Layers | | | |  | | |
| 7 - Application | - The APIs etc built into an application that makes the application “network aware”  - E.g. MS Words knows how to search and show what files to open | | | 4 - Application | | - Everything to do with application works in this layer |
| 6 - Presentation | - Converts data into a format that your applications can read | | |
| 5 - Session | - Actual connection between two systems  - Defines what’s taking place in terms of how that connectivity really works  - E.g. TCP connection between a web server and a web client, sending an email, sharing a folder | | | 3 - Transport | | - Assembly, disassembly  - What it takes to connect to other system to make sure data gets there  - E.g. TCP, UDP |
| 4 - Transport | - Assembly, disassembly area for data as it gets broken down into individual packets  - Makes sure that the packets gets into the other system in good order | | |
| 3 - Network | - Has to do with logical addresses  - E.g. IP addresses, routers | | | 2 - Internet | | - IP addresses, routers |
| 2 - Data Link | - Anything that works with a MAC address works at the data link layer  - E.g. Network cards, switches | | | 1 - Network Interface/Link | | - Hardware  - Covers all the physical cabling, MAC addresses, network cards  - Exceptions: routers |
| 1 - Physical | - What can of cables to use and stuff like that | | |
| Flow  (from Layer 1 up) | | **Layer 1/1 (OSI/TCP/IP)**  1. Network layer POV: network card waits for data  2. Network card receives an ethernet frame  3. Checks incoming MAC address and verify that it’s for it (the correct system)  **Layer 2/1**  4. Frame check sequence, check the entire Ethernet frame to make sure its in good shape  5. Strip off MAC address and store it elsewhere in memory, left with an IP packet  6. Network card passes IP packet (frame + source,dest IP address) to next layer  **Layer 3/2**  7. Network layer POV: Look at IP addresses, make sure that its for it  8. Discard its own IP address but keeps IP address of where information came from  9. Depending on what type of packet, what’s left can be e.g. TCP segment (designed and ready for computer, just need to get it to the right applications and in the right format)  10. Passes it to the next higher layer  **Layer 4/3**  11. Transport layer POV: act as the assembler/disassembler  12 (1). If its big-sized data *going out,* disassemble  - Break it into bite-sized chunks  - Disassemble and add sequencing numbers to individual TCP/IP segments  12 (2). If its bunch of data *going in*, reassemble  - Reassemble all the data using sequencing number  13. Pass complete chunks of data *and* port numbers to next layer  **Layer 5-7/4**  14. Session layer POV (layer 5): designed to connect a server to a client on a remote system (try to connect to something)  - E.g. MS Word isn’t network aware (only local), so needed a distinct session layer to connect to a remote system  15. Presentation layer POV (layer 6): gets data to a particular application but not in the form that the application itself could use  16. Application layer POV: looks at port numbers  - Keeps return port numberin memory somewhere  17. Sends data to right application using destination port number | | |  | |
| Frames | | - Devices on a network send and receive data (in discreet chunks called frames or packets)  - Frames are created and destroyed inside the network interface card (NIC)  - If frame not meant for that network, network card will just consume it (data will not go past network frame) | | |  | |
| - Packetized data  - A single frame can be up to 1 500 bytes long/in size (10 000 ones and zeroes)  - Generated inside the network card, data comes from application  - Eaten up by network cards and sent up to whatever software that needs it, and wiped out on the card  - Have a discrete beginning and end | | |  | |
| MAC address | | - Media Access Control (MAC)  - How frames know to get to the right computer  - Frame payload does not identify the destination  - 48-bit address  - 6 pairs, 12 hexadecimal values (each representing 4 binary characters)  - In total, 48 binary characters  - First 3 pairs, represents a unique OEM (Original Equipment Manufacturer), issued to the maker of the network cards  - Last 3 pairs, represents a unique ID  - Every network card in existence has a unique MAC address | | | Terminal command:  *ipconfig /all*  - See all information about all network cards  - MAC address is under “physical address” | |
| Cyclic redundancy check (CRC) | | A way to verify that the data is good  - If data is bad, network knows to resend it | | |  | |
| Hardware | | Hub | - Allows multiple computers to connect to each other to share resources  - E.g. webpages, documents  - Where LAN came from  - Receives one signal from source computer, and repeats signal to send it out to all other connected computers in hub | | - IP packets never change but never travel by themselves but within a frame | |
| Router | - Have two connections or more  - Connects multiple local area networks  - Usually includes a switch  - MAC addresses from source computers (especially over the internet) usually points to the router’s MAC address  - Flow:  1. Router receives frame  2. Identifies MAC address as its own (correct delivery)  3. Strips away its MAC address from frame  4. Uses routing table to find the computer IP packets is meant for  5. Adds back to frame MAC address of the destination computer (found by routing table)  - Has a routing table - tells based on network information, where to send data | |
| Switch | - Local Area Connection (LAC) | |
|  |  | |
|  |  | |
|  |  | |
| Data | | Source MAC + Dest MAC  + Source IP + Dest IP  + Source port no. + Dest port  + Frame  + CRC | | |  | |
| **Broadcast** | | | | **Unicast** | | |
| - A broadcast transmission is sent to every device in a broadcast domain | | | | - Unicast transmission is addressed to a single device on a network | | |
| - Computer does not know who it is sending out to  **How it works**  - One computer sends out a frame that propagated to everyone else  - But for the destination MAC address, it puts **“FF-FF-FF-FF-FF-FF”** (broadcast address)  - Every computer that receives this information will accept it can pass it up through the layers  - Very important for a local area network (LAN)  - Broadcast domain  - When a group of computers are in a broadcast domain, they can hear each other’s broadcast  - When plugged into a hub, all connected computers are in the same single broadcast domain  - E.g. Broadcast for “Tom’s” computer to send back his MAC address | | | |  | | |
| - Problem of broadcast domain and MAC addresses, sending out too many broadcast messages that nothing gets done | | | |  | | |
| Logical addressing | | - Needed especially in a very big network (using only MAC addresses isn’t sufficient)  - E.g. IP addressing  - Not fixed with card  - Can be used to identify a particular network  - First three octets identifies computers on the same network, last octet uniquely identifies the computer on the network  - Now MAC address refers to the router | | |  | |
| Default gateway | | Connection router itself | | |  | |
| Packets | | - Packets have sequence numbers so the network software can reassemble the file correctly  - TCP (Transmission Control Protocol)  - Connection-oriented conversation between 2 computers  - Sequencing numbers  - Acknowldegement  - UDP (Use Datagram Protocol)  - Connectionless, sends and hopes you’re ready for it  - If not, send again | | |  | |
| Ports | | - Port numbers help direct packet traffic between the source and destination  - How computer knows which web page to send IP packet to  - Port range is 0 - 65535  - First 1024 port numbers are reserved  - Port 80: HTTP  - Port 20, 21: FTP | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
| **TCP/IP Basics** | |  | | |  | |
| IP addresses | | - Each computer on a TCP/IP network must have a unique IP address  - IPv4 addresses are written as four octets  - E.g. 192.168.4.12  - Each octet represents a binary string  - E.g. 192 is 11000000 | | |  | |
|  | | - Usually given in dotted decimal notation  - Dots in an address has no meaning  - Actually made up of 32-long ones&zeros  - Number of combinations: 28 = 256  - If separate bits into octets (groups of 8, 4 groups), each octet is valued between 0 and 255  - How to convert binary (1 octet) to decimal  1. Start with binary  - E.g. 11000101  2. Use code = **“128 64 32 16 8 4 2 1”**  - Which is 128 divided by 2 repeatedly  3. To get decimal,  - decimal = sum(code[bit] for bit in binary)  - i.e. sum up numbers in code with index corresponding to 1s’ index in code  4. Therefore, decimal = 128 + 64 + 4 + 1 = 197  5. Conclusion, octet 11000101 = 197  - How to convert decimal to binary  1. Using code, start from “128” and continue towards “1”  2. If decimal is larger than current number in code, set bit 1 for that index and substract from decimal  3. Else if decimal is smaller than number, set bit 0 for that index  4. Continue to next number until 8 bits are filled | | | Examples:  00000000 = 0  11111111 = 255  10000000 = 128  00000001 = 1 | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |
|  | |  | | |  | |