Layer Zero UNLV 2018-19

Introduction to Networking Layers & Protocols - Jinger Siu

Quick Reference

Layer	Data Type	Addressing Scheme	Relevant Protocols	Associated Physical Device
Layer 1	bits	N/A	N/A	hubs, wires, microwaves
Layer 2	frames	MAC addresses	ARP, Ethernet	switches
Layer 3	packets	IP addresses (v4/v6)	IP, routing protocols	routers
Layer 4	segments	port numbers	TCP, UDP	N/A
Layer 7	data	N/A	HTTP(S), FTP, DHCP	web browsers

Clarification

Terms like “server” and “client” are thrown around a lot in this text. If you’re comfortable with this vernacular, you can skip this section. Whenever I use “server,” I just mean “a computer that has some specific, ad hoc function”. Whenever I use “client,” I mean “a computer that is attempting to access a server’s resources/functions.” Any one computer can be both a host and a client. For example, I could be hosting my own website, as well as accessing other websites on the internet.

For clarity, examples will include a *dedicated* server/client. Throughout any given example, a client device will act only as a client, and a server device will act only as a server.

Overview

- much of networking can be described as having a “client-server” architecture
    - e.g. web client/server, dhcp client/server, etc.
- networking is also very dependent on the idea of “source” and “destination”
    - e.g. source/destination IP address, source/destination MAC address, etc.
- people logically segmented the networking process into “layers”
    - this helps humans understand how networking works
    - we focus on the OSI model
- there’s a lot of different ways to identify clients on the network
    - e.g. MAC address (layer 2) → IP address (layer 3) → port numbers (layer 4)
- some addressing schemes/identification mechanisms for network communication
    - MAC addresses: 48-bit hexadecimal strings
        - e.g. 00:A0:C9:14:C8:29 or 00A0.C914.C829
    - IP addresses: IPv4 (32-bit decimal strings) and IPv6 (128-bit decimal strings)
        - e.g. IPv4 131.23.44.250
        - e.g. IPv6 2001:0db8:85a3:0000:0000:8a2e:0370:7334
    - port numbers: reserved ports and client-generated “ephemeral ports”

The OSI Model

The OSI model is a way for people to understand the inner workings of the networking process. When we think about a certain networking protocol, we first think of where it lays in the OSI model. There are 7 layers, but for now we are only really concerned with 5 of those layers.

There is also the TCP/IP model, but it’s pretty much the same thing. Think in terms of whichever model clicks with you better logically.

- ***Layer 1*** Physical
    - physical medium for transmission:
        - copper cables (ex. Ethernet cables), fiber optic cables, microwaves (Wi-Fi)
    - electronic devices that don't do any form of switching or routing:
        - hubs and repeaters
        - considered to be in layer 1 because they just take electronic signals from one interface connection, and send it out all other interfaces
    - electronic signals/waves translated into machine-readable 1s and 0s
- ***Layer 2*** Data Link
    - the 1s and 0s in Layer 1 are encapsulated into a *frame* (usually found to be using hexadecimal notation)
    - this encapsulation happens by adding a *frame header* and *frame trailer* at the end of the series of 1s and 0s from the physical layer
        - the header contains information about the source MAC address and the destination MAC address
    - Address Resolution Protocol (ARP): translates an IP address into a MAC address
- ***Layer 3*** Network
    - frames are encapsulated into *packets*
    - encapsulation happens by adding a *packet header*
        - the header contains information about the source IP address and the destination IP address
    - Internet Control Message Protocol (ICMP)
        - includes *ping*, which is the simplest way to communicate over the network layer
            - ex. in a Linux terminal, type `ping 127.0.0.1` to see the results of what a *ping* look like (press Ctrl-C to stop)
        - usually used for diagnostics or error-checking
        - does **not** use any layer 4 protocol, so it does **not** have a port number
- ***Layer 4*** Transport
    - packets are encapsulated into *segments*

        - just like packets are just a bunch of frames abstracted into a different data structure, *segments* are just a bunch of packets
        - there are two main protocols for the transport layer
            - Transport Control Protocol (TCP)
            - User Datagram Protocol (UDP)
- ***Layer 7*** Application
    - typically user-facing programs and applications
    - some examples
        - File Transfer Protocol (FTP): transfer files (TCP ports 20 & 21)
        - Telnet: interactive communication over the command line (TCP port 23)
        - Dynamic Host Configuration Protocol (DHCP): dynamically assign IP addresses to clients (UDP ports 67 & 68)
        - Domain Name Service (DNS): translates human-readable URLs to network-readable IP addresses, and vice versa (TCP port 53 and UDP port 53)

The important thing to remember is that any given layer is *dependent* on all the layers below it. There is no way for me to get to google.com if my computer isn't plugged into the network via an ethernet cable, or if I'm not connected to a wireless network via microwaves.

At any given instance in layer 3 network connectivity, there needs to be a connection at layer 2, which means there also needs to be a connection at layer 1. This is why we need protocols like ARP. ARP is used when we know the IP address of who we want to talk to, but we don't have their MAC address. If we don't have the layer 2 MAC address communication, we can't have layer 3 IP address communication.

Alternatively, layers are *transparent* to those above it. This means we don't need the higher layers in order for the lower layers to work. Take *ping* for example. Ping doesn't use any ports, so it doesn't use the transport layer at all. But that doesn't matter, as long as layers 3, 2, and 1 are working properly.

NOTE ABOUT PORT NUMBERS/STANDARDIZATION

Recall the *Layer 7* section above. Notice how the ports for a given protocol are relatively variable. Some include only TCP ports, some only UDP ports, some both, etc. Ports, like other aspects of networking, exist based off standardization.

The reason we use TCP port 80 for HTTP, for instance, is because a bunch of people agreed upon it as the standard. If I was running an HTTP server and I decided to use a different port, I could. I could be running one web service out of port 8080, one out of port 34429, and one out of port 1.

This doesn't inherently present any issues. Problems can happen, though, when other people try to utilize your server's resources. Port numbers are linked to protocol through standardization. There needs to be a standard way to communicate throughout the networking process (i.e. throughout all 7 layers).

Imagine if, instead of determining languages based off rules and commonality, we all just decided to speak to each other in whichever series of sounds we personally liked. It would be challenging, if not impossible. Standardization makes sure we're all speaking the same language.

Because of this, port numbers 0 to 1024 are reserved for "privileged services." This includes FTP, SSH, TELNET, HTTP, etc. Here is a quick reference for these kinds of ports: https://www.webopedia.com/quick_ref/portnumbers.asp

TCP vs UDP

TCP and UDP are both transport layer protocols, and they both abstract packets into segments. However, they are used for different situations. TCP is used to create "reliable" connections between devices, and UDP is to create fast connections.

TCP has the notion of segment sequencing. That is, segments (which are just series of packets) are numbered canonically before they are sent. If any segments are lost during

transmission to the destination device, TCP allows the source device to resend those segments. The segments are also checked for integrity.

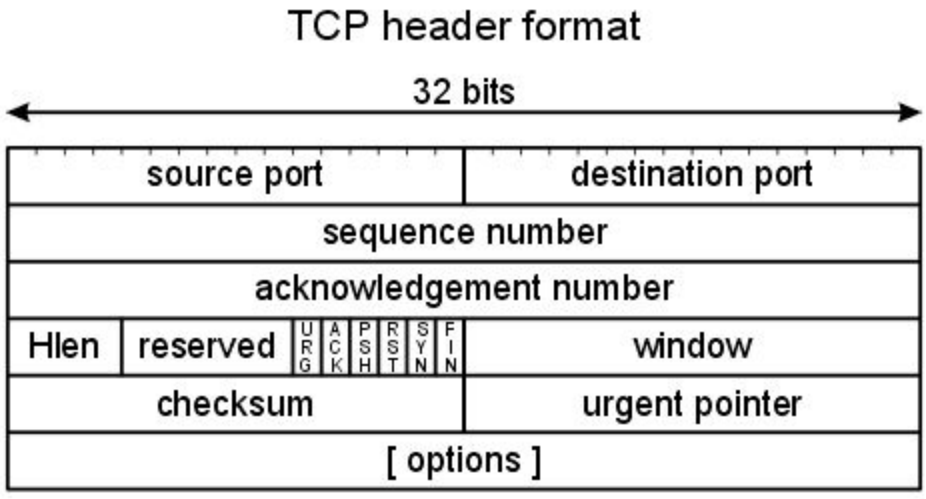
UDP does not take these extra steps. It sends segments without sequencing, and is considered "unreliable." UDP gives up on creating reliable connections in order to have low overhead and fast transmission. This is why UDP is used for things like live streaming, DNS queries, DHCP, etc.

THE TCP 3-WAY / 4-WAY HANDSHAKE

The TCP 3-way handshake describes the process of how two devices create a TCP connection.

Let's assume some client is attempting to SSH (TCP port 22) to some server. The client first sends a *SYN segment*, where SYN stands for "synchronize." Recall that, through encapsulation, *segments* are really just a series of 1s and 0s. So a *SYN segment* is a segment that has the *SYN bit* in the TCP header set to be 1.

The SYN bit itself isn't anything magic. It's just a specific bit in the TCP header that has been designated for synchronization. It's the standard.



TCP header. The pertinent information for us is the section with *URG, ACK, PSH, RST, SYN*, and *FIN*

The server receives the SYN segment, and then sends back a *SYN/ACK segment* to the client. ACK stands for “acknowledge.” This is a segment whose header has both the SYN and the ACK bits set to 1.

The third and final step is that the client sends the server an *ACK segment*. This segment has only the ACK bit set to 1.

The 4-way handshake is a similar concept, but it terminates a TCP connection as opposed to creating one. The first device sends a *FIN segment*, the other device replies with an *ACK segment* and then with a *FIN/ACK segment*. Finally, the first devices responds by sending an *ACK segment.*