TCP/IP five-layer network model

Yuyuan: WIFI (Data link layer)

Michael: IP address (networking layer)

Jingzhi: TCP/UDP (Transport layer)

**Intro:**

The original OSI model describes seven layers that computer systems use to communicate over a network. It was the first standard model for network communications, adopted by all major computer and telecommunication companies in the early 1980s. However, the simpler model, TCP/IP is a five-layer model. The modern Internet is based on TCP/IP model. It helps visualize and communicate how networks operate and helps isolate and troubleshoot networking problems.

**WIFI (Data link layer)**

There are three main ways that a wireless network can be configured: ad-hoc networks where nodes all speak directly to each other, wireless LANS or WLANS where one or more access points act as a bridge between a wireless and a wired network, and mesh networks which are kind of a hybrid of the two. Ad-hoc network is the simplest one among the three, but it is not the most commonly used one. Practical example includes some smartphones can establish ad-hoc networks with other smartphones in the area so that people can exchange photos, video or contact information. The most common type of wireless network you’ll run into in the business world is a wireless LAN or WLAN. A wireless LAN consists of one or more access points, which act as bridges between the wireless and wired networks. The wired network operates as a normal LAN. Like the types we’ve already discussed, the wired LAN contains the outbound internet link. In order to access resources outside of the WLAN, wireless devices would communicate with access points. They then forward traffic along to the gateway router, where everything proceeds like normal. Mesh networks are kind of like ad-hoc networks, since lots of the devices communicate with each other wirelessly, forming a mesh. If you were to draw lines for all the links between all the nodes, most mesh networks you’ll run into are made up of only wireless access points and will still be connected to a wired network. This kind of network let’s you deploy more access points to the mesh without having to run a cable to each of them. With this kind of setup, you can really increase the performance and range of a wireless network.

The concept of channels is one of the most important things to understand about wireless networking. Channels are individual, smaller sections of the overall frequency band used by a wireless network. Channels are super important because they help address a very old networking concern, collision domains. A collision domain is anyone network segment where one computer can interrupt another. Communications that overlap each other can’t be properly understood by the receiving end. So when two or more transmissions occur at the same time, also called a collision, all devices in question have to stop their transmissions. They wait a random amount of time and try again when things quiet down. This really slows things down. The problem caused by collision domains has been mostly reduced on wired networks through devices called switches. Switches remember which computers live on which physical interfaces. So traffic is only sent to the node It’s intended for. Wireless networking doesn’t have cables, so there aren’t physical interfaces for a wireless device to connect to. That means we can have something that works like a wireless switch. Wireless devices are doomed to talk over each other. Channels help fix this problem to a certain extent. The point is to understand how collision domains are a necessary problem with all wireless networks and how you can use your knowledge in this space to optimize wireless network deployments. You want to make sure that both your own access points and those of neighboring businesses overlap channels as little as possible.

**TCP/UDP:**

A socket is the instantiation of an endpoint in a potential TCP connection. TCP sockets require actual programs to instantiate them. You can contrast this with a port which is more of a virtual descriptive thing. In other words, you can send traffic to any port you want, but you're only going to get a response if a program has opened a socket on that port. TCP sockets can exist in lots of states. LISTEN. Listen means that a TCP socket is ready and listening for incoming connections. You'd see this on the server side only. SYN\_SENT. This means that a synchronization request has been sent, but the connection hasn't been established yet. You'd see this on the client side only. SYN\_RECEIVED. This means that a socket previously in a listener state, has received a synchronization request and sent a SYN\_ACK back. But it hasn't received the final ACK from the client yet. You'd see this on the server side only. ESTABLISHED. This means that the TCP connection is in working order, and both sides are free to send each other data. You'd see this state on both the client and server sides of the connection. This will be true of all the following socket states, too. FIN\_WAIT. This means that a FIN has been sent, but the corresponding ACK from the other end hasn't been received yet. CLOSE\_WAIT. This means that the connection has been closed at the TCP layer, but that the application that opened the socket hasn't released its hold on the socket yet. CLOSED. This means that the connection has been fully terminated, and that no further communication is possible.

A TCP header itself is split into lots of fields containing lots of information. First, we have the source port, and the destination port fields. The destination port is the port of the service the traffic is intended for. A source port is a high numbered port chosen from a special section of ports known as ephemeral ports. For now, it's enough to know that a source port is required to keep lots of outgoing connections separate. A destination port, say port 80, is needed to make sure traffic reaches a web server running on a certain IP. Similarly, a source port is needed so that when the web server replies, the computer making the original request can send this data to the program that was requesting it. It is in this way that when it web server responds to your requests to view a webpage that this response gets received by your web browser and not your word processor. Next up is a field known as the sequence number. This is a 32-bit number that's used to keep track of where in a sequence of TCP segments this one is expected to be. You might remember that lower on our protocol stack, there are limits to the total size of what we send across the wire. In Ethernet frame, it's usually limited in size to 1,518 bytes, but we usually need to send way more data than that. At the transport layer, TCP splits all the data into many segments. The sequence number in a header is used to keep track of which segment it is. The next field, the acknowledgment number, is a lot like the sequence number. The acknowledgment number is the number of the next expected segment. In very simple language, a sequence number of one and an acknowledgement number of two could be read as this is segment one, expect segment two next. The data offset field comes next. This field is a four-bit number that communicates how long the TCP header for this segment is. This is so that the receiving network device understands where the actual data payload begins. The next field is a 16-bit number known as the TCP window. A TCP window specifies the range of sequence numbers that might be sent before an acknowledgement is required. TCP is a protocol that's super reliant on acknowledgements. This is done in order to make sure that all expected data is being received and that the sending device doesn't waste time sending data that isn't being received. The next field is a 16-bit checksum. It operates just like the checksum fields at the IP and Ethernet level. Once all this segment has been ingested by a recipient, the checksum is calculated across the entire segment and is compared with the checksum in the header to make sure that there was no data lost or corrupted along the way. The Urgent pointer field is used in conjunction with one of the TCP control flags to point out segments that might be more important than others. This is a feature of TCP that hasn't ever seen adoption and you'll probably never find it in modern networking. Next up, we have the options field. Like the urgent pointer field, this is rarely used in the real world, but it's sometimes used for more complicated flow control protocols. Finally, we have some padding which is just a sequence of zeros to ensure that the data payload section begins at the expected location.

**IP address / Network Layer:**

The network layer has two main functions. One is breaking up segments into network packets and reassembling the packets on the receiving end. The other is routing packets by discovering the best path across a physical network. The network layer uses network addresses (typically Internet Protocol addresses) to route packets to a destination node.

If the two devices communicating are on the same network, then the network layer is unnecessary. For Example, Tulane students are able to visit the Tulane Website faster on campus because it is in the same system of the network under a series of servers. The network layer breaks up segments from the transport layer into smaller units, called packets, on the sender’s device and reassembling these packets on the receiving device. The network layer also finds the best physical path for the data to reach its destination; this is known as routing.

IP Routing describes the process of determining the path for data to follow in order to navigate from one computer or server to another. A packet of data traverses from its source router through a web of routers across many networks until it finally reaches its destination router using a routing algorithm. The routing algorithm takes into account factors such as the size of a packet and its header to determine the most efficient route to the destination. When a packet has reached a router, the source and destination address of the packet is used in conjunction with a routing table (list that contains the routes to a certain network) to determine the next hop address. This process is repeated for the next router using its own routing table until the packet has reached its destination. Because the data is divided into packets, each packet travels independently from each other and is treated as such. As a result, each packet can be sent through a different route to the destination if necessary.

Reference

<https://www.sangoma.com/how-ip-routing-works/#:~:text=IP%20Routing%20describes%20the%20process,router%20using%20a%20routing%20algorithm>.

<https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/>

<https://microchipdeveloper.com/tcpip:tcp-ip-five-layer-model>

<https://www.imperva.com/learn/ddos/tcp-transmission-control-protocol/#:~:text=TCP%20wraps%20each%20data%20packet,port%20%E2%80%93%20The%20receiving%20device's%20port>.

<https://www.geeksforgeeks.org/differences-between-tcp-and-udp/>

<https://witestlab.poly.edu/blog/channel-planning-in-802-11-understanding-the-effect-of-nearby-networks/>