Understanding the OSI Model

First, let's take a look at the OSI model, which stands for Open Systems Interconnection. It is a conceptual model for how data travels from one computer to another across a network and is organized into seven layers. Each with its own function. Here is a summary of the seven layers:

1. Physical layer - the physical side of networking hardware, cables, Wi-Fi, and how the signals are sent (you can think of it as the road the cars (data) are traveling on).

2. Data Link layer - ensures data is sent and received properly between two devices on the same local network (think of it as the traffic lights and road signs).

3. Network layer - the layer where routing, or figuring out the best possible path for the data to take, occurs. (think GPS for your data).

4. Transport layer - this layer makes sure your entire message is properly delivered (think making sure your puzzle has all the pieces).

5. Session layer - establishes connections, manages the connection, and determines when they are finished. (think starting and ending a conversation at a party).

6. Presentation layer - this layer translates data for applications, like varying languages.

7. Application layer - this is the user interface to the network, where users access network services, email, web browsing, etc.

I will admit it was confusing at first, but comparing the layers to building a house helped—foundation to roof. For example, when I send an email, the Application Layer handles the email program, but the Physical Layer is what sends the signal through my Wi-Fi.

Understanding the TCP/IP Model

It is different from OSI because it is the model used on the internet and is designed to be simpler, with four layers instead of seven:

Network Access Layer- The combination of Physical and Data Link layer functionality, which actually provides access to the network.

Internet Layer- This contains functions that are similar to the Network layer in the OSI model, addressing and routing packets to their destination.

Transport Layer- This applies TCP and UDP protocols, which are used to control the flow of data between applications.

Application Layer- This combines the top three layers of the OSI model, where the applications are actually making the connection through the network.

It is a less detailed model than OSI because the TCP/IP model is just suited for the purposes of the Internet. For example, when I use my browser to see web pages, TCP/IP takes care of making sure my browser connects with the web server without me needing to know what the details are in between.

TCP and UDP Comparison

Finally, it's worth saying a word about the two protocols that exist in the transport layer: TCP and UDP:

TCP (Transmission Control Protocol) - This protocol is reliable, meaning connection must be established before transmission (Like making a phone call). Reliable service is typically slower service but it always guarantees delivery of data to the destination. This is why this service is used to read emails, transfer files, or make web page requests when correct delivery of the data is important. For example, if I download a movie, I will always want to be that every byte of information I expect to receive is the correct amount.

UDP (User Datagram Protocol) has no reliability, very fast, and no connection like sending a text message. This protocol is great for real-time mediums in which a few lost packets will not matter -- like games, video streams, or VoIP. When I stream a live sports game, UDP gets used. If there's a frame missed, no biggie, just keep the streaming going without all the buffering.

Personal Reflections and Real-Life Examples

As stated, this was a little tricky to learn at first. I came up with a mnemonic, Please Do Not Throw Sausage Pizza Away for the OSI layers to help me remember, and it worked. The real-life examples were great - such as Spotify streaming music using UDP (if a note skipped, oh well) instead of using TCP, like when downloading software/firmware updates (it needs to all be intact when it finishes). This step is like layers of abstraction - the lower layers do all the engine/technical stuff, while the upper layers do the driving/hard parts, which make networks easier to manage.

Detailed Examination of the OSI Model

The OSI (Open Systems Interconnection) model is a theoretical framework that divides network communication into seven layers, each with specific functions.

LAYER	NAME	FUNCTIONS	REAL-LIFE EXAMPLE
1	Physical Layer	Handles physical transmission like cables and signals, bit synchronization, bit rate control	Wi-Fi signals sending data through your router
2	Data Link Layer	Ensures node-to-node delivery, framing, physical addressing (MAC), error control, flow control	Ethernet cables connecting devices on the same network
3	Network Layer	Manages routing and addressing across networks logical addressing (IP)	GPS-like routing for data packets across the internet
4	Transport Layer	Ensures end-to-end delivery, segmentation and reassembly, flow/error control	Ensuring a downloaded file arrives complete
5	Session Layer	Establishes, maintains, terminates, connections; synchronization dialog controller	Starting and ending a video call
6	Presentation Layer	Data translation, encryption/decryption, compression	Converting data for an email app to read
7	Application Layer	Provides network access for applications, NVT, FTAM, mail services directory, services	Using a web browser to access a website

This model is mostly theoretical, but it's essential to learn about network architecture, given it isn't widely practical until maybe now. Imperva and Cloudflare both touched on this, and pointed out its value when troubleshooting when needed. I enjoyed thinking about it as if you were drafting a house, each layer as a part of the construction, from foundation to roof, successful construction depends on every part completing its task.

Deep Dive into the TCP/IP Model

The TCP/IP model, in contrast, is more practical, with four layers, developed before the OSI model by the Department of Defense (DoD) in the 1970s, based on standard protocols.

Layer Name	Function	Key Responsibilities	Real-Life Example
Application Layer	Provides services and interfaces for end-user applications to access network resources	Supports application protocols enables communication between software applications handles data formatting	Using HTTP for web browsing like accessing Google
Transport Layer	Ensures reliable or unreliable delivery of data between hosts	TCP: Reliable connection-oriented; UDP: Faster connectionless; Manages flow control and segmentation	TCP for downloading files UDP for streaming videos
Internet Layer	Determines the best path for data to travel across networks	IP: Provides addressing and routing; Handles packet forwarding fragmentation logical addressing	Routing data packets across different networks
Network Access Layer	Manages the physical transmission of data over the network hardware	Handles how data is physically sent over cables Wi-Fi etc.; Manages MAC addressing framing error detection	Wi-Fi connecting your laptop to the internet

Its basic approach is easier to apply, but is naturally without OSI's depth of segmentation for troubleshooting such as Fortinet. I found it to be like a simplified map than the detailed OSI map; here it is tailored to focus on what we need for the internet when browsing the web - something that we all do daily - like ensuring that my browser can communicate with a web server on the website I'm browsing.

Comparing TCP and UDP Protocols

At the Transport layer, both models use TCP and UDP, which I found fascinating for their contrasting approaches.

Basis	TCP	UDP
Type of Service	Connection-oriented requires establishing connection before data transfer	Datagram-oriented no connection overhead efficient for broadcast/multicast
Reliability	Reliable guarantees delivery	Unreliable delivery not guaranteed
Error checking mechanism	Extensive includes flow control and acknowledgment	Basic uses checksums
Acknowledgment	Present	No acknowledgment segment
Sequence	Sequencing of data packets arrive in order	No sequencing order managed by application layer
Speed	Slower than UDP	Faster simpler and more efficient
Retransmission	Possible for lost packets	No retransmission of lost packets
Header Length	20-60 bytes variable length	8 bytes fixed-length header
Weight	Heavy-weight	Lightweight
Handshaking Techniques	Uses SYN ACK SYN-ACK	Connectionless no handshake
Broadcasting	Does not support broadcasting	Supports broadcasting
Protocols	Used by HTTP HTTPS FTP SMTP Telnet	Used by DNS DHCP TFTP SNMP RIP VoIP
Stream Type	Byte stream	Message stream
Overhead	Low but higher than UDP	Very low
Applications	Email file transfer web browsing	Gaming video streaming online video chats VoIP

TCP is handy where you need reliable data transfer, like with pictures and videos. Whereas, UDP is more suited for applications to avoid lag, like when engaging in an online game. Like when downloading a movie via TCI, I want every byte transferred and to be correct. However, when streaming a live sports game, I am utilizing UDP, and if a frame skips, that's fine. I am still able to watch without the buffering.