Computer Networking fundamentals

Contents

What is Networking? 2

How the Internet works 3

Layer 1: Physical - Actual transmission of raw data bits over physical medium 4

Layer 2: Data Link - Physical addressing (MAC addresses), error detection/correction 5

- knowing the data link layer is important to AWS? 6
- Medium Access Control (MAC): 7

Layer 3: Network - Handles logical addressing and routing (e.g., IP addresses, routers) 8

- what is an IP? 9
- What is an IP Address (IPv4)? 9
- What is the difference between an internal (Private) and external (Public) address? 10
- Private IP ranges according to RFC 1918 (10.x.x.x, 172.16.x.x, 192.168.x.x) 11
- What is a broadcast address and a network address? 12
- What is Default Gateway? 13
- What is CIDR (like /24, /16) Subnetting and CIDR? 13
- Example: Subnetting a /24 network into smaller pieces 15
- How to calculate an address range using CIDR 16
- Using the Subnet Mask to Find Network, Broadcast, and Usable Addresses 18
- What is NAT (Network Address Translation)? 21
- What happens when you open Google from your home Wi-Fi?" (NAT & PAT in action) 22
- What is PAT – Port Address Translation (aka NAT Overload)? 25

Layer 4: Transport **-** Provides reliable data transfer, error recovery (e.g., TCP, UDP) 26

- What is load balancer in AWS? 27
- Relation between OSI Layer 4 (Transport Layer) and AWS 28

Layer 5: Session **-** Establishes, manages, and terminates connections (sessions) between applications .. 30

Layer 6: Presentation - Data translation, encryption, compression (makes data readable for Application layer) 31

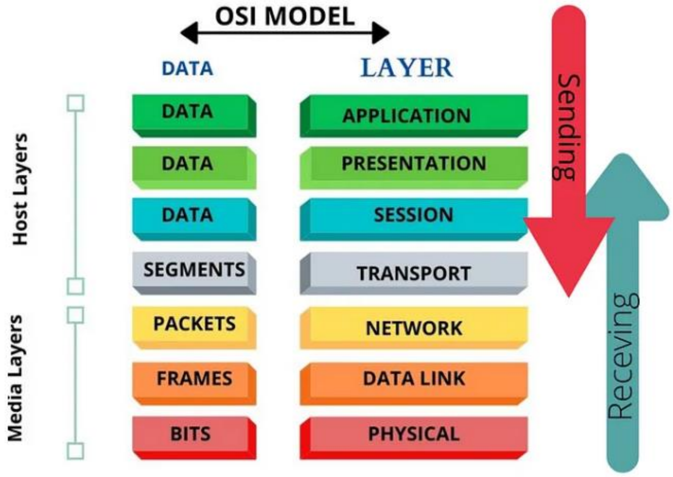
Layer 7: Application Layer - Interface for end-user processes (e.g., web browsers, email) 32

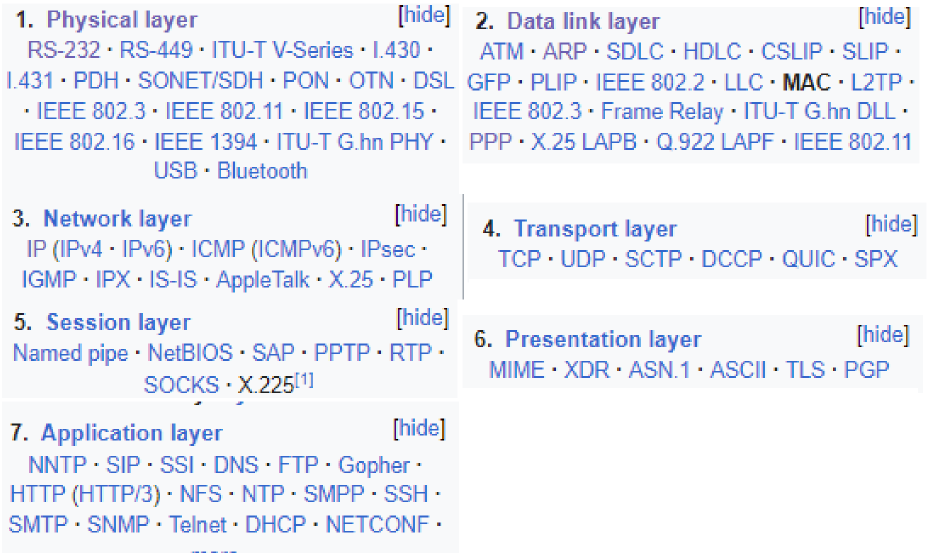
What it is DNS resolution is and how it happens........33
What SSH means?........33
HTTP and HTTPS........35
Visual mapping between all OSI layers and AWS services........37

What is Networking?

- A network is two or more devices connected such that they can communicate with one another.

OSI Model: Open Systems Interconnection





1. Physical layer [hide]
RS-232 · RS-449 · ITU-T V-Series · I.430 · I.431 · PDH · SONET/SDH · PON · OTN · DSL · IEEE 802.3 · IEEE 802.11 · IEEE 802.15 · IEEE 802.16 · IEEE 1394 · ITU-T G.hn PHY · USB · Bluetooth

2. Data link layer [hide]
ATM · ARP · SDLC · HDLC · CSLIP · SLIP · GFP · PLIP · IEEE 802.2 · LLC · MAC · L2TP · IEEE 802.3 · Frame Relay · ITU-T G.hn DLL · PPP · X.25 LAPB · Q.922 LAPF · IEEE 802.11

3. Network layer [hide]
IP (IPv4 · IPv6) · ICMP (ICMPv6) · IPsec · IGMP · IPX · IS-IS · AppleTalk · X.25 · PLP

4. Transport layer [hide]
TCP · UDP · SCTP · DCCP · QUIC · SPX

5. Session layer [hide]
Named pipe · NetBIOS · SAP · PPTP · RTP · SOCKS · X.225[1]

6. Presentation layer [hide]
MIME · XDR · ASN.1 · ASCII · TLS · PGP

7. Application layer [hide]
NNTP · SIP · SSI · DNS · FTP · Gopher · HTTP (HTTP/3) · NFS · NTP · SMPP · SSH · SMTP · SNMP · Telnet · DHCP · NETCONF ·

How the Internet works

The internet is basically a huge network that connects millions of computers all over the world. When you use it — like right now — here's roughly what happens:

1. You ask for something:
   Like when you type a website address (say, www.example.com) or click a link.

2. Your device sends a request:
   Your computer/phone sends a little digital message called a *request* out into the internet.

3. Routers guide the request:
   That message bounces through a bunch of special computers called *routers* that direct it toward the server (the computer where the website lives).

4. Servers respond:
   The server receives your request, finds the right information (like a webpage), and sends it back to you.

5. Data travels back:
   The information (the website, image, video, whatever) zips back across the internet, again hopping through routers, until it gets to your device.

6. Your device shows it:
   Your browser (Chrome, Safari, etc.) takes that data and displays it nicely for you.

Behind the scenes, the internet uses things like:

- IP addresses (like digital home addresses for computers),
- DNS servers (like phone books to find the right server),
- Protocols (rules for talking, like TCP/IP).

It's like mailing a letter, but instant — and with millions of letters flying everywhere at once.

Layer 1: Physical - Actual transmission of raw data bits over physical medium

Key Functions

1. **Bit Transmission**: Converts data into electrical, optical, or radio signals that can be transmitted over a medium. It focuses on how bits are represented (e.g., voltage levels, light pulses, or radio waves).
2. **Physical Medium**: Specifies the type of physical medium used for communication. This can include twisted pair cables, fiber optics, coaxial cables, and wireless signals.
3. **Transmission Mode**: Defines the method of data transmission, such as:
    - **Simplex**: Data flows in one direction only (e.g., keyboard to computer).
    - **Half-Duplex**: Data can flow in both directions, but not simultaneously (e.g., walkie-talkies).
    - **Full-Duplex**: Data can flow in both directions simultaneously (e.g., telephone conversations).
4. **Signal Encoding**: Determines how data is converted into signals, including modulation techniques and encoding schemes like Manchester encoding or NRZ (Non-Return-to-Zero).
5. **Data Rate**: Defines the speed of transmission, measured in bits per second (bps), which is crucial for assessing the performance of a network.
6. **Physical Topology**: Refers to the layout of the physical connections between devices (e.g., star, ring, bus).
7. **Synchronization**: Ensures that the sender and receiver are synchronized so that the data is interpreted correctly, which can involve clock signals or timing mechanisms.
8. **Line Configuration**: Determines how devices are connected (e.g., point-to-point or multipoint).

Components

- **Hardware**: Includes physical devices like cables, switches, routers, network interface cards (NICs), and repeaters.
- **Connectors**: Physical connectors such as RJ45 for Ethernet cables, LC/SC for fiber optics, or antennae for wireless signals.

Physical layer

RS-232 RS-449 ITU-T V-Series I.430 I.431 PDH SONET/SDH PON OTN DSL
IEEE 802.3 IEEE 802.11 IEEE 802.15 IEEE 802.16 IEEE 1394 ITU-T G.hn PHY
USB Bluetooth

Layer 2: Data Link - Physical addressing (MAC addresses), error detection/correction

Layer 2 of the OSI model is the Data Link layer. It plays a crucial role in ensuring reliable data transfer between directly connected devices on a network. Here are some key functions and concepts related to the Data Link layer:

Functions of the Data Link Layer:

1. **Framing**: Data is encapsulated into frames for transmission. Each frame includes a header and a trailer to help identify the data and check for errors.
2. **Physical Addressing**: The Data Link layer uses MAC (Media Access Control) addresses to identify devices on the same local network.
3. **Error Detection and Correction**: It provides mechanisms to detect and possibly correct errors that occur during data transmission. Common techniques include checksums and cyclic redundancy checks (CRC).
4. **Flow Control**: This function ensures that a sender does not overwhelm a receiver by sending data too quickly, using methods like stop-and-wait or sliding window protocols.
5. **Medium Access Control (MAC)**: The Data Link layer determines how devices on a network share the physical medium. Protocols like CSMA/CD (Carrier Sense Multiple Access with Collision Detection) and CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) are examples.

Types of Data Link Layer Protocols:

- **Ethernet**: Widely used in wired networks, specifying how frames are structured and transmitted.
- **Wi-Fi**: A standard for wireless networking, defining how data is transmitted over radio waves.
- **PPP (Point-to-Point Protocol)**: Used for direct connections between two network nodes, often in dial-up connections.

Sub-layers:

The Data Link layer can be divided into two sub-layers:

1. **Logical Link Control (LLC)**: Manages communication between the network layer and the Data Link layer, providing services like error checking and frame synchronization.
2. **Media Access Control (MAC)**: Controls how devices on the same physical network segment access the shared medium.

Data link layer

ATM ARP SDLC HDLC CSLIP SLIP GFP PLIP IEEE 802.2 LLC MAC L2TP IEEE 802.3

Frame Relay ITU-T G.hn DLL PPP X.25 LAPB Q.922 LAPF IEEE 802.11

knowing the data link layer is important to AWS?

Absolutely, understanding the data link layer can be valuable when working with AWS or any cloud environment. The data link layer is part of the OSI model, responsible for establishing and managing connections between devices on the same local network. Here's why it can matter in the AWS context:

1. **Networking Basics:** AWS relies heavily on virtualized networking. Knowing the data link layer helps you understand how data packets are transmitted over local networks, which is foundational to configuring and troubleshooting virtual private clouds (VPCs), subnets, and network interfaces in AWS.
2. **Troubleshooting Network Issues:** Issues like packet loss or faulty network configurations often have their roots in lower layers like the data link layer. Understanding this layer aids in diagnosing and resolving connectivity problems in AWS-hosted systems.
3. **Hybrid Environments:** When extending AWS services into on-premises setups or other cloud services, the data link layer becomes crucial for ensuring proper communication between systems, particularly over VPNs or Direct Connect.
4. **Security Considerations:** Network protocols at this layer, like Ethernet, can come into play when enforcing security measures, even in cloud setups. Though AWS provides robust tools, familiarity with underlying network principles is helpful for advanced configurations.

While AWS abstracts much of the networking work, having a solid grasp of the data link layer can give you an edge in optimizing performance, security, and troubleshooting.

Medium Access Control (MAC):

Medium Access Control (MAC) is a crucial component of network protocols that manages how data packets are transmitted over a shared communication medium. Its primary purpose is to regulate access to the physical transmission medium, ensuring that multiple devices can communicate without interfering with one another.

Key Functions of MAC:

1. **Channel Access Control**: Determines how devices gain access to the shared medium. Common methods include:
   - **Carrier Sense Multiple Access (CSMA)**: Listens to the channel before transmitting to avoid collisions.
   - **Time Division Multiple Access (TDMA)**: Divides the channel into time slots assigned to different devices.
   - **Frequency Division Multiple Access (FDMA)**: Allocates different frequency bands to different devices.
2. **Collision Management**: Handles situations where two devices attempt to transmit simultaneously, leading to data collisions. Strategies include:
   - **Collision Detection (CD)**: Monitors the channel for collisions and retransmits if necessary (e.g., CSMA/CD).
   - **Collision Avoidance (CA)**: Uses techniques to prevent collisions before they occur (e.g., CSMA/CA).
3. **Frame Delimitation**: Identifies the start and end of each data packet (frame) to ensure accurate transmission.
4. **Addressing**: Assigns unique addresses to devices on the network, allowing the MAC layer to direct data packets to the correct recipient.
5. **Error Detection and Correction**: Implements mechanisms to detect and possibly correct errors in transmitted data.

MAC in Different Protocols:

- **Ethernet**: Uses CSMA/CD for collision detection in wired networks.
- **Wi-Fi**: Utilizes CSMA/CA to avoid collisions in wireless networks.
- **Bluetooth**: Employs a form of time division for its connections.

Importance:

MAC is essential for efficient and reliable data communication in networks, helping to maximize throughput and minimize delays while ensuring that multiple devices can communicate simultaneously without conflict.

Layer 3: Network - Handles logical addressing and routing (e.g., IP addresses, routers)

Layer 3 of the OSI (Open Systems Interconnection) model is known as the Network Layer. This layer is responsible for the following key functions:

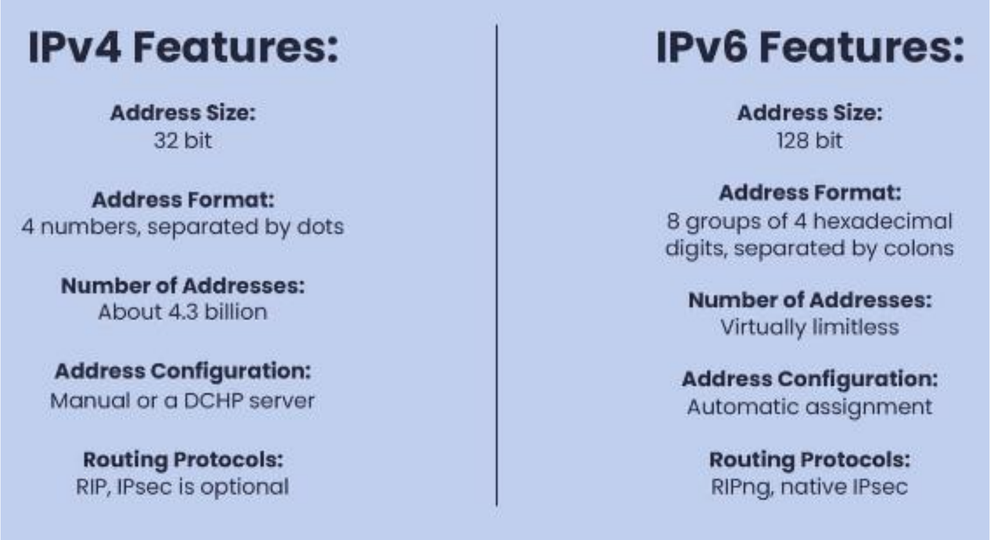
1. **Routing:** Determines the best path for data to travel across the network, ensuring it reaches the correct destination.
2. **Logical Addressing:** Assigns IP addresses to devices on the network, allowing them to communicate with each other regardless of their physical location.
3. **Packet Forwarding:** Manages the process of forwarding packets to their destination based on the routing decisions made.
4. **Fragmentation and Reassembly:** Handles breaking down larger packets into smaller ones for transmission and reassembling them at the destination.
5. **Error Handling:** Provides mechanisms for detecting and correcting errors that may occur during data transmission.

Common protocols that operate at Layer 3 include Internet Protocol (IP), Internet Control Message Protocol (ICMP), and Routing Information Protocol (RIP). Would you like to know more about a specific aspect of the Network Layer?

Network layer

IP IPv4 IPv6 ICMP (ICMPv6) IPsec IGMP IPX IS-IS AppleTalk X.25 PLP

IPv4 vs IPv6:



IPv4 Features:	IPv6 Features:
Address Size: 32 bit	**Address Size:** 128 bit
Address Format: 4 numbers, separated by dots	**Address Format:** 8 groups of 4 hexadecimal digits, separated by colons
Number of Addresses: About 4.3 billion	**Number of Addresses:** Virtually limitless
Address Configuration: Manual or a DCHP server	**Address Configuration:** Automatic assignment
Routing Protocols: RIP, IPsec is optional	**Routing Protocols:** RIPng, native IPsec

what is an IP?

"IP" usually stands for **Internet Protocol** — it's the system that allows computers and devices to communicate over a network.

More simply:
An **IP address** is like the "home address" for your device on a network or the internet. It tells other devices where to send the information you're asking for, like a website, an email, or a file.

There are two main types of IP addresses:

- **IPv4**: Looks like 192.168.1.1 (four numbers separated by dots).
- **IPv6**: Looks like 2001:0db8:85a3:0000:0000:8a2e: 0370:7334 (longer, with letters and numbers).

Were you asking about **IP addresses** or something else like **IP in business (Intellectual Property)**?

What is an IP Address (IPv4)?

Structure:

- An IPv4 address is **32 bits** long.
- These 32 bits are divided into **4 groups** called **octets**.
- Each **octet** is **8 bits**.
- In practice, we write IPv4 addresses in **decimal form**, with each octet separated by a **dot** (.). Example: 192.168.1.1

Octets:

- Each octet can range from **0 to 255** (because 8 bits can represent $2^8 = 256$ different values).
- Example breakdown of 192.168.1.1:
    - 192 → 11000000
    - 168 → 10101000
    - 1 → 00000001
    - 1 → 00000001

So, the full binary version would be:
11000000 10101000 00000001 00000001

What is the difference between an internal (Private) and external (Public) address?

Internal (Private) IP Address:

- **Used inside** a private network (like your home Wi-Fi, office LAN).
- **Not routable** on the internet (they **can't be accessed** directly from outside the network).
- Devices like your laptop, phone, printer, and smart TV usually have private IPs.
- **Ranges reserved** for private use:
    - 10.0.0.0 to 10.255.255.255
    - 172.16.0.0 to 172.31.255.255
    - 192.168.0.0 to 192.168.255.255

Example:
Your laptop at home might have an IP like 192.168.1.5.

External (Public) IP Address:

- **Used on the internet**.
- **Routable** on the internet (visible to the outside world).
- Assigned by your **Internet Service Provider (ISP)**.

- Identifies your network or device **globally**.

Example:
Your home router might be given a public IP like 203.0.113.42, which the internet sees.

Quick Analogy:

- **Private IP** = Your apartment number (inside the building).
- **Public IP** = The building's street address.

Important Detail:
At home, your router typically uses **NAT** (Network Address Translation) to let many private devices share one public IP address when accessing the internet.

Private IP ranges according to RFC 1918 (10.x.x.x, 172.16.x.x, 192.168.x.x)

Private IP Ranges (according to RFC 1918):

Address Range	**CIDR Notation**	**Number of Addresses**
10.0.0.0 – 10.255.255.255	10.0.0.0/8	16,777,216 addresses
172.16.0.0 – 172.31.255.255	172.16.0.0/12	1,048,576 addresses
192.168.0.0 – 192.168.255.255	192.168.0.0/16	65,536 addresses

Explanation:

- **10.x.x.x** → Very large private range (used a lot in enterprise networks).
- **172.16.x.x to 172.31.x.x** → Medium-size private block.
- **192.168.x.x** → Smaller, very common in-home networks (most routers default here).

Why RFC 1918?
RFC 1918 officially **reserves** these ranges so no public internet routers will ever route them. They're strictly for **internal/private use**.

If a device with a private IP tries to access the internet directly, it won't work unless it's **translated** (usually by a router with **NAT**).

What is a broadcast address and a network address?

Network Address:

- It's the **starting address** of a network.
- It **identifies the network itself**, not any specific device.
- **First address** in any IP range (all host bits are 0).
- **Cannot** be assigned to a device.

✅ Example:
If you have the network 192.168.1.0/24:

- **Network Address** = 192.168.1.0

Broadcast Address:

- It's the **ending address** of a network.
- It’s used to **send a message to all devices** in that network.
- **Last address** in any IP range (all host bits are 1).
- Also, **cannot** be assigned to a device.

✅ Example:
In the same network 192.168.1.0/24:

- **Broadcast Address** = 192.168.1.255
    - (Any message sent here will be received by **every device** in 192.168.1.x.)

Quick Summary Table:

Type	Description	Example (192.168.1.0/24)
Network Address	Identifies the whole network	192.168.1.0

Type	**Description**	**Example (192.168.1.0/24)**
Broadcast Address	Message to all network devices	192.168.1.255

Tiny Tip:

- Network = "Where you live"
- Broadcast = "Shout to everyone living there"

What is Default Gateway?

- It's the **device** (usually a **router**) that your device sends data to **when it needs to reach another network**.
- If your computer doesn't know how to reach an IP address (like a website), it sends the traffic to the **default gateway**.
- The gateway **forwards** that traffic out to the internet or other networks.

✅ **Example:**

- Your PC IP: 192.168.1.10
- Default Gateway: 192.168.1.1 (Usually the router's IP)

When you try to open google.com, your PC sends the request to 192.168.1.1, and the router takes it from there.

Quick Analogy:

- Your **PC** = Person inside a house.
- **Default Gateway** = The front door to leave the house and go anywhere else

What is CIDR (like /24, /16) Subnetting and CIDR?

What is CIDR?

- **CIDR** = **Classless Inter-Domain Routing**.
- It's a **way of writing IP addresses + their network size**.
- Instead of the old "class A, B, C" system, CIDR lets you **flexibly divide networks**.
- You see CIDR as a **slash** (/) and a **number** (like /24, /16, /8).

Example:

- 192.168.1.0/24
- /24 means **the first 24 bits** are the **network part**, and the rest are for **hosts**.

How CIDR Works:

IP address = 32 bits total
CIDR number = How many bits are **locked** for the network.

- /24 → 24 bits for network, 8 bits for host.
- /16 → 16 bits for network, 16 bits for host.
- /8 → 8 bits for network, 24 bits for host.

The fewer bits left for hosts, the **smaller** the network.

✅ Example:

CIDR	Subnet Mask	Number of Hosts
/8	255.0.0.0	~16 million hosts
/16	255.255.0.0	~65,000 hosts
/24	255.255.255.0	254 hosts

(You always subtract 2 addresses: one for the ***network address*** *and one for the* ***broadcast address****.)*

What is Subnetting?

- **Subnetting** = Breaking a big network into **smaller networks**.
- Helps organize and secure networks better.
- Each subnet will have its own **network address**, **broadcast address**, and **range of IPs**.

Quick Visual:
If you have a 192.168.0.0/16 (65,536 addresses) —
you can subnet it into **256** smaller /24 networks (each with 254 usable addresses).

Summary:

Term	**Meaning**
CIDR (/24)	How big the network is, written with a slash
Subnetting	Dividing a bigger network into smaller pieces

Example: Subnetting a /24 network into smaller pieces

192.168.1.0/24
Normally, /24 = 256 IP addresses total (254 usable).

Suppose you want **smaller subnets** — say **/26**.

- **/26** means **26 bits** are for the network, leaving **6 bits** for hosts.
- 2^6 = 64 total addresses per subnet.
- Usable hosts = 64 - 2 = **62 hosts** (subtract network and broadcast addresses).

✅ Now, **Subnetting 192.168.1.0/24 into /26 gives you 4 smaller networks:**

Subnet	**Network Address**	**Broadcast Address**	**Usable IP Range**
1st Subnet /26	192.168.1.0	192.168.1.63	192.168.1.1 – 192.168.1.62
2nd Subnet /26	192.168.1.64	192.168.1.127	192.168.1.65 – 192.168.1.126
3rd Subnet /26	192.168.1.128	192.168.1.191	192.168.1.129 – 192.168.1.190
4th Subnet /26	192.168.1.192	192.168.1.255	192.168.1.193 – 192.168.1.254

🚀 **Quick Pattern:**

- Subnets increment by 64 (0, 64, 128, 192).
- Broadcast is **one IP before** the next network.

🔥 **Quick Tips:**

- **Subnet Size** = 2^(32 - subnet bits)
- To find subnets: **Add the block size** (like 64) to the starting IP.

Summary:

- **/24** = 256 IPs → **Split into four /26** = each with **64 IPs**.
- **Each /26** = **62 usable** addresses for devices.

How to calculate an address range using CIDR

📚 **Steps to Calculate an Address Range from CIDR:**

1. **Identify the network address** (the starting point).
2. **Find the block size** (number of IPs in the subnet).
    - Block size = 2^(32 - CIDR prefix).
3. **Calculate:**
    - **First usable IP** (usually network address + 1).
    - **Last usable IP** (broadcast address - 1).
    - **Broadcast address** (last IP in the block).

✅ **Example 1:**

Network: 192.168.1.0/26

Step 1: Block Size

- 32 - 26 = 6 bits for hosts.
- 2^6 = **64 addresses** total.

Step 2: Range

- Starting (Network Address): **192.168.1.0**
- Ending (Broadcast Address): **192.168.1.63**

Step 3: Usable IPs

- First usable: **192.168.1.1**
- Last usable: **192.168.1.62**

Type	IP Address
Network Address	192.168.1.0
First usable address	192.168.1.1
Last usable address	192.168.1.62
Broadcast Address	192.168.1.63

📈 **General Formula:**

- **Total IPs** = 2^(32 - subnet mask bits)
- **Usable IPs** = Total IPs - 2 (network + broadcast)
- **Block size** = Jump to next subnet.

🔥 **Another Example:**

Network: 10.0.0.0/20

- 32 - 20 = 12 bits for hosts.
- 2^{12} = 4096 addresses.
- Network Address: **10.0.0.0**

- Broadcast Address: **10.0.15.255**
- First Usable: **10.0.0.1**
- Last Usable: **10.0.15.254**

📦 **Shortcut for Finding Block Size:**

CIDR	Block Size (IPs)
/24	256 addresses
/25	128 addresses
/26	64 addresses
/27	32 addresses
/28	16 addresses
/29	8 addresses
/30	4 addresses
/31	2 addresses (special for point-to-point links)
/32	1 address (single host)

Using the Subnet Mask to Find Network, Broadcast, and Usable Addresses

📍 **Step-by-Step:**

1. **Write down the Subnet Mask** (like 255.255.255.192 for /26).
2. **Find the interesting octet** (the octet where the subnetting happens).
3. **Calculate Block Size**:
    - Block size = 256 - subnet mask value (in that octet).
4. **Find the multiple of Block Size that is just less than or equal to the IP address.**
    - That's the **Network Address**.

5. **Add Block Size - 1** to the network address to find the **Broadcast Address**.
6. **First usable = Network + 1**
7. **Last usable = Broadcast - 1**

📚 **Example:**

IP = 192.168.1.77
Subnet Mask = 255.255.255.192

Step 1: Subnet mask is 255.255.255.192
(Third octet is 255, Fourth octet is 192 → subnetting happens in the **fourth octet**.)

Step 2: Find Block Size

- Block size = 256 - 192 = 64

Step 3: Find nearest multiple of 64 ≤ 77

- Multiples of 64 are: 0, 64, 128, 192...
- 64 ≤ 77 < 128 → So, **Network Address** = 192.168.1.64

Step 4: Broadcast Address

- Network + Block Size - 1
- 64 + 64 = 128 → 128 - 1 = **127**
- So, **Broadcast Address** = 192.168.1.127

Step 5: Usable IP Range

- First usable: 192.168.1.65
- Last usable: 192.168.1.126

✅ **Final Results:**

Type	IP Address
Network Address	192.168.1.64
First Usable Address	192.168.1.65
Last Usable Address	192.168.1.126
Broadcast Address	192.168.1.127

🎯 Summary Table:

Subnet Mask	CIDR	Block Size (addresses)
255.255.255.0	/24	256
255.255.255.128	/25	128
255.255.255.192	/26	64
255.255.255.224	/27	32
255.255.255.240	/28	16
255.255.255.248	/29	8
255.255.255.252	/30	4

🏁 Super Fast Method (Shortcut)

1. Focus on the "magic" octet (where mask isn’t 255 or 0).
2. Block size = 256 - subnet mask value.
3. Find which block the IP belongs to.
4. Done!

What is NAT (Network Address Translation)?

🌐 What is NAT (Network Address Translation)?

- **NAT** is a technique used by routers to **change IP addresses** in packets as they pass between different networks (usually from a private network to the Internet).
- It **translates private IP addresses** (like 192.168.x.x) into a **public IP address** (like 203.0.113.5) — and **vice versa**.

✅ Why?

- Private IP addresses (RFC 1918) **cannot travel across the Internet**.
- NAT allows many devices on a private network to **share one** (or a few) **public IPs**.
- It **saves public IP addresses** and **adds security** (hides internal network structure).

🛠 Types of NAT:

Type	**Description**
Static NAT	1 private IP ↔ 1 public IP (one-to-one mapping)
Dynamic NAT	Private IPs are mapped to public IPs **from a pool**
PAT (Port Address Translation)	Many private IPs share **one public IP** — **different ports** distinguish them.

🔥 What is PAT (Port Address Translation)?

- **PAT** is a special kind of NAT.
- Also called **"NAT Overload."**
- It **maps multiple private IPs** to **one public IP address**, but **uses different port numbers**.
- Helps **hundreds of devices** access the internet **with a single public IP**!

✅ Example:

- Inside IPs:

  - 192.168.1.2:12345 → Public IP:203.0.113.5:40000
  - 192.168.1.3:12346 → Public IP:203.0.113.5:40001
- Router tracks the **port numbers** to know which internal device to send the reply to.

📈 **Visual Summary:**

Private Network (192.168.x.x)

↓

Router with NAT/PAT

↓

Public IP (e.g., 203.0.113.5)

↓

Internet

🎯 **Quick Memory Trick:**

- **NAT** = Translates IP addresses.
- **PAT** = Translates IP addresses **+ ports**.

✍️ **In Simple Words:**

- **NAT**: Changes your private IP to public when going online.
- **PAT**: Allows thousands of devices to "share" a single public IP using different **ports**.

What happens when you open Google from your home Wi-Fi?" (NAT & PAT in action)

1. 🏠 **At home:**

- Your device (phone, laptop, PC) has a **private IP**, like 192.168.1.10.
- Your home router has:
  - **Private side** (LAN): 192.168.1.1
  - **Public side** (WAN): maybe 203.0.113.5 (the IP given by your Internet provider).

---

2. 📥 **You type www.google.com in your browser.**

- Your device **wants to talk to Google's public IP** (say, 142.250.190.4).
- Your device creates a packet:
    - **Source IP:** 192.168.1.10
    - **Source Port:** Random, e.g., 50000
    - **Destination IP:** 142.250.190.4
    - **Destination Port:** 443 (for HTTPS)

---

3. 🚪 **Router NAT/PAT kicks in.**

- **NAT**: Your router sees the private IP (192.168.1.10) and **translates** it to the **public IP (203.0.113.5)**.
- **PAT**: To track multiple devices, the router also **changes the source port** — for example, from 50000 to 60000.

✅ Now the packet looks like:

- **Source IP:** 203.0.113.5
- **Source Port:** 60000
- **Destination IP:** 142.250.190.4
- **Destination Port:** 443

---

4. 🌐 **Packet travels over the Internet.**

- Google's server **receives the request** from:
    - IP 203.0.113.5:60000 (your router’s public IP and port)
- Google **sends a reply** back to 203.0.113.5:60000.

---

5. 🏡 **Router receives the reply.**

- Your router **uses its NAT table** to figure out:
  - "Ah! Port 60000 is assigned to 192.168.1.10:50000!"
- It **translates the packet back**:
  - Destination IP: 192.168.1.10
  - Destination Port: 50000

✅ It **forwards the reply** to your computer!

---

6. 📬 **You see Google's website open.**

- To you, it looks instant.
- In the background, **NAT and PAT** handled **IP address** and **port translations** automatically!

📈 **Summary:**

Step	**Action**
1	Device creates packet (private IP)
2	Router NATs and PATs packet to public IP/port
3	Packet sent over Internet
4	Google responds to router's public IP/port
5	Router de-NATs and forwards to device
6	Google loads on your screen

🎯 **Key Point:**

- **NAT** hides your private IP.
- **PAT** tracks multiple devices by using **different ports** on the **same public IP**.

Your Device (192.168.1.10) → Home Router (NAT/PAT) 203.0.113.5 → Internet 🌐 → Google Server (142.250.190.4) → Reply Flow (coming back) → Google Server (142.250.190.4) → Google Server (142.250.190.4) → Google Server (142.250.190.4) → Internet 🌐 → Home Router (NAT/PAT) 203.0.113.5 → Your Device (192.168.1.10)

What is PAT – Port Address Translation (aka NAT Overload)?

PAT is a **type of NAT** where:

- **Multiple private devices** (many 192.168.x.x IPs)
- **Share ONE public IP address** (like 203.0.113.5)
- **Different port numbers** are used to **separate** each device's connection.

How?

- Each internal device's packet **gets a unique port** assigned by the router.
- The router **keeps track** of which port belongs to which private device.

✅ **Example:**

Private Device	Private IP	Translated Public IP + Port
Laptop	192.168.1.10:50000	203.0.113.5:60001
Phone	192.168.1.11:50001	203.0.113.5:60002
Smart TV	192.168.1.12:50002	203.0.113.5:60003

Result:
All these devices **look like one public IP** on the internet — but internally, the router uses **different ports** to **keep track** of who is who!

🌩 Why Use PAT in the Cloud?

✅ **In Cloud environments**, PAT is very important because:

Reason	Why it matters
IP Address	Public IPv4 addresses are limited and expensive. PAT allows many cloud

Reason	Why it matters
Conservation	servers/instances to share one or a few public IPs.
Scalability	You can easily handle thousands of internal (private) cloud VMs without needing thousands of public IPs.
Security	PAT hides internal IPs from the internet. Attackers can't directly see or attack internal cloud VMs.
Cost Saving	Fewer public IP addresses = less cost when building large cloud applications.

✅ **Example in the Cloud:**

- Imagine a Kubernetes cluster or a large app deployment in AWS/Azure/GCP:
  - Hundreds of internal services (private IPs).
  - Only a few public IPs exposed via **PAT** for outgoing traffic (API calls, updates, etc.).

📈 **Quick Visual Summary:**

Internal Cloud VMs (Private IPs)→ Cloud NAT Gateway (PAT/NAT Overload) → Internet (Public IP + different ports)

🧠 **In Simple Words:**

- **PAT** = "Let hundreds (or thousands) of machines talk to the internet using just **one** public IP by **mixing up the ports**."
- **In the cloud**, this **saves money, addresses, and improves security**.

Layer 4: Transport - Provides reliable data transfer, error recovery (e.g., TCP, UDP)

Layer 4 of the OSI model, known as the Transport Layer, is responsible for ensuring reliable data transmission between devices. It manages the flow of data and provides services such as error detection, correction, and data segmentation. Key protocols at this layer include:

1. **TCP (Transmission Control Protocol)**: A connection-oriented protocol that ensures reliable, ordered, and error-checked delivery of data. It establishes a connection before data transfer and uses acknowledgments to confirm receipt.

2. **UDP (User Datagram Protocol)**: A connectionless protocol that allows for faster data transmission without the overhead of establishing a connection or ensuring reliability. It's often used in applications where speed is crucial, such as video streaming or online gaming.
3. **SCTP (Stream Control Transmission Protocol)**: A protocol designed for transporting PSTN (Public Switched Telephone Network) signaling messages, but it also supports multiple streams of data between two endpoints, allowing for message delivery without head-of-line blocking.

What is load balancer in AWS?

In AWS (Amazon Web Services), a **load balancer** is a service that automatically distributes incoming application or network traffic across multiple targets—such as EC2 instances, containers, or IP addresses—in one or more Availability Zones. This improves fault tolerance, scalability, and application availability.

AWS offers **Elastic Load Balancing (ELB)** with several types:

1. Application Load Balancer (ALB)

- Best for **HTTP and HTTPS traffic**
- Operates at **Layer 7** (Application layer)
- Supports advanced routing: path-based, host-based, query-string, etc.
- Commonly used for **web applications** and **microservices**

2. Network Load Balancer (NLB)

- Designed for **TCP, TLS**, and **UDP traffic**
- Operates at **Layer 4** (Transport layer)
- Capable of handling **millions of requests per second** with ultra-low latency
- Suitable for **real-time applications** and **high-performance networking**

3. Gateway Load Balancer (GWLB)

- Used to **deploy, scale, and manage third-party virtual appliances**
- Supports **transparent insertion of appliances** such as firewalls or intrusion detection systems
- Operates at **Layer 3 (Network layer)**

4. Classic Load Balancer (CLB) *(Legacy)*

- Supports **basic Layer 4 and Layer 7 features**
- Generally used for older applications; newer applications are recommended to use ALB or NLB

Key Benefits:

- **High availability**
- **Automatic scaling**
- **Health checks** to route traffic only to healthy instances
- **SSL termination**
- **Integrated with Auto Scaling and AWS Certificate Manager**

Would you like an example of setting up an Application Load Balancer?

Transport layer

TCP UDP SCTP DCCP QUIC SPX

Relation between OSI Layer 4 (Transport Layer) and AWS

1. Layer 4 = **Transport Layer**

- Main responsibilities:
  - Managing **end-to-end communication**.
  - **Ports** and **protocols**: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).
  - Ensures **data delivery** (TCP is reliable, UDP is faster but unreliable).
  - Tracks **sessions** between devices.

Examples of Layer 4 stuff:

- TCP port 80 → HTTP
- TCP port 443 → HTTPS
- UDP port 53 → DNS

- Source Port, Destination Port

---

2. In AWS: How Layer 4 shows up

AWS heavily deals with Layer 4 operations, especially in **networking and security**. Here's how:

AWS Service/Feature	How it Relates to Layer 4
Security Groups	Act like a **stateful Layer 4 firewall** (allow/deny by **TCP/UDP ports** and IP addresses).
NACLs (Network ACLs)	Stateless firewall controls based on **Layer 4** information (ports, protocols).
Elastic Load Balancer (ELB) (Classic, Network Load Balancer)	**Network Load Balancer (NLB)** operates at **Layer 4** — routing based on TCP/UDP ports, not inspecting application data.
VPC Peering/Transit Gateway	Communication between VPCs uses **TCP/UDP connections** at Layer 4.
Cloud NAT Gateways	PAT/NAT operations use **TCP/UDP ports** to manage traffic for many instances.
VPN Connections	AWS VPN tunnels use **Layer 4 protocols** (IPsec relies on UDP ports for NAT-T).
AWS Global Accelerator	Works at Layer 4 to optimize TCP/UDP connections across AWS regions.

🔥 **Example: Security Group Rules = Layer 4**

Suppose you add a rule in a Security Group like:

- Allow inbound TCP on port 22 (SSH) from 0.0.0.0/0

You are directly controlling **Layer 4** behaviour:

- TCP = transport protocol (Layer 4)
- Port 22 = transport-layer port number (Layer 4)

🎯 **In short:**

- **When you define ports, protocols (TCP/UDP), and session rules in AWS, you are working with OSI Layer 4.**
- AWS services like **Security Groups**, **NLBs**, and **NAT Gateways** depend heavily on **Layer 4 concepts** to manage cloud networking.

Layer 5: Session - Establishes, manages, and terminates connections (sessions) between applications

Layer 5 of the OSI model is the Session Layer. Its primary function is to establish, manage, and terminate sessions between applications on different devices. Here are some key aspects of the Session Layer:

1. **Session Management**: It handles the opening, closing, and managing of sessions, ensuring that communication is organized and synchronized.
2. **Synchronization**: It can provide checkpoints during data transfer, allowing a session to resume from a certain point in case of interruptions.
3. **Control Dialog**: It determines how and when data is exchanged, allowing for full-duplex or half-duplex communication.
4. **Token Management**: The Session Layer can manage tokens that control access to shared resources, preventing conflicts in data transmission.
5. **Session Establishment**: It establishes connections between client and server applications, often using protocols like RPC (Remote Procedure Call) or NetBIOS.

The Session Layer is crucial for facilitating communication between applications, making it easier for them to interact in a coordinated manner.

Session layer

Named pipe **NetBIOS** **SAP** **PPTP** **RTP** **SOCKS** **X.225**[1]

Layer 6: Presentation - Data translation, encryption, compression (makes data readable for Application layer)

Layer 6, the Presentation Layer, is crucial in the OSI model as it prepares data for the application layer. It ensures that data is in a usable format by handling tasks such as:

1. **Data Translation**: Converts data from one format to another, ensuring compatibility between different systems.
2. **Data Compression**: Reduces the size of data to optimize bandwidth usage.
3. **Data Encryption**: Secures data by converting it into a coded format that can only be read by authorized users.

In essence, the Presentation Layer acts as a translator between the application layer and the lower layers, enabling proper data exchange and enhancing communication efficiency. Would you like to know more about how it interacts with other layers?

Presentation layer

MIME XDR ASN.1 ASCII TLS PGP

Layer 7: Application Layer - Interface for end-user processes (e.g., web browsers, email)

The Application Layer is the highest level of the OSI model, and it plays a crucial role in network communication by providing services directly to end users. Here’s an overview of its key features:

Key Functions:

1. **User Interface**: It provides the means for users to interact with applications, often through graphical interfaces or command-line tools.
2. **Application Services**: It supports application-level protocols that allow software applications to communicate over the network (e.g., HTTP, FTP, SMTP).
3. **Data Representation**: It ensures that data is presented in a way that the application can understand, handling aspects like data formatting and character encoding.
4. **Session Management**: It manages sessions between applications, maintaining the state and context of communication.

Common Protocols:

- **HTTP/HTTPS**: Used for web browsing.
- **FTP/SFTP**: Used for file transfers.
- **SMTP/IMAP/POP3**: Used for email transmission and retrieval.
- **DNS**: Translates domain names to IP addresses.

Examples of Applications:

- Web browsers (e.g., Chrome, Firefox)
- Email clients (e.g., Outlook, Thunderbird)
- File transfer programs (e.g., FileZilla)
- Chat applications (e.g., Slack, Zoom)

Application layer

NNTP SIP SSI DNS FTP Gopher HTTP (HTTP/3) NFS NTP SMPP SSH SMTP SNMP

Telnet DHCP NETCONF

What it is DNS resolution is and how it happens

DNS resolution is the process of translating human-friendly domain names (like **example.com**) into machine-friendly IP addresses (like **192.0.2.1**) that computers use to locate and communicate with one another on the internet. Here's how it typically happens:

1. **User's Request**: When you type a website URL into your browser, your computer checks its cache to see if it already knows the IP address for that domain. If it does, it uses that information without needing further resolution.
2. **Recursive Resolver**: If the IP address isn't cached locally, your computer forwards the query to a DNS resolver (usually provided by your Internet Service Provider). This resolver acts like a middleman to find the answer for you.
3. **Root Servers**: The resolver first contacts a root DNS server, which directs it to the appropriate top-level domain (TLD) server based on the domain extension (.com, .org, etc.).
4. **TLD Servers**: Next, the resolver queries the TLD server, which points it to the authoritative nameserver for the specific domain name.
5. **Authoritative Nameserver**: The authoritative nameserver holds the domain's DNS records and provides the IP address to the resolver.
6. **Final Response**: The resolver sends the IP address back to your computer, which uses it to connect to the website's server. The website then loads in your browser.

This process happens quickly—usually in milliseconds—and ensures you can access websites without memorizing complex numerical IP addresses! Does that make sense?

What SSH means?

Sure! Let me elaborate on SSH.

What is SSH?

SSH (**Secure Shell**) is a cryptographic network protocol designed to allow users to securely access and control a remote device over an unsecured network. It replaces older, less secure protocols like Telnet or FTP by encrypting all data exchanged between the client and the server. This ensures protection against eavesdropping, man-in-the-middle attacks, and data theft.

How SSH Works

SSH operates using a **client-server model**:

1. **Client Side**: The user (you) initiates a connection to a remote server using an SSH client. Common tools for this are:

  - OpenSSH (default on most Linux/Unix systems).
  - PuTTY (popular on Windows).

2. **Server Side**: The server must have an SSH daemon (a program listening for and managing SSH connections) running. Once a connection request is received, the server and client:
   - Authenticate each other.
   - Establish an encrypted channel for secure communication.

Key Features

1. **Authentication**:
   - SSH supports password-based authentication, but the more secure method is **public key authentication**. Users generate a key pair (public and private key). The private key stays on the client, while the public key is shared with the server. If the keys match, access is granted.
2. **Encryption**: Ensures all data sent between the client and server is unreadable to third parties.
3. **Tunneling**: SSH can create an encrypted tunnel, which allows for secure forwarding of ports or even entire network protocols.

Common Uses

1. **Remote Login**: SSH is primarily used to log in to remote servers or devices securely. For example, system administrators often use SSH to manage cloud servers.
2. **Secure File Transfer**: Tools like SCP (Secure Copy Protocol) and SFTP (SSH File Transfer Protocol) use SSH to transfer files between devices securely.
3. **Port Forwarding**: SSH allows users to redirect network traffic securely, often used for accessing internal network services from a remote location.
4. **Automation**: Developers and sysadmins use SSH in scripts for automating tasks like software deployments or server maintenance.

Advantages of SSH

- Strong encryption ensures privacy.
- Secure authentication mechanisms reduce the risk of unauthorized access.
- It’s widely supported across operating systems.

If you're interested, I can also provide guidance on setting up SSH, generating key pairs, or any specific

HTTP and HTTPS

HTTP stands for **HyperText Transfer Protocol**, and it is the foundation of the World Wide Web. It allows computers, servers, and other devices to communicate over the internet. HTTP defines how requests and responses are formatted and transmitted between clients (like your web browser) and servers (where websites are hosted).

How HTTP Works

When you type a URL in your browser or click a link:

1. Your browser (the client) sends an **HTTP request** to the server where the website is stored.
2. The server processes the request and sends back an **HTTP response**, which usually contains the data (like the web page's text, images, and code).
3. Your browser displays the content to you.

For example, when you visit http://example.com, your browser requests the website's files, and the server provides them through HTTP.

Key Features of HTTP

1. **Stateless Protocol**: HTTP doesn't retain any information about previous interactions (each request is independent), which makes it lightweight but often requires additional mechanisms (like cookies) to manage sessions.
2. **Human-Readable**: It uses plain text for requests and responses, making it simple to understand and debug.
3. **Flexible**: HTTP can handle different types of content, such as HTML pages, images, videos, and more.

HTTP Methods

These define the type of action the client wants to perform:

- **GET**: Request data from a server (e.g., loading a webpage).
- **POST**: Send data to a server (e.g., submitting a form).
- **PUT**: Update data on a server.

- **DELETE**: Remove data from a server.

HTTPS

HTTP has an improved, secure version called **HTTPS** (HyperText Transfer Protocol Secure). It uses encryption (via SSL/TLS) to protect data from being intercepted or tampered with, ensuring a safe communication channel.

If you want, I can explain how HTTP fits into the larger picture of the internet or delve into specific aspects of it! Just let me know. 😊

HTTPS stands for **HyperText Transfer Protocol Secure**. It's an enhanced, secure version of HTTP that ensures the safe exchange of information over the web by encrypting all data transferred between your browser (client) and the server. The "S" in HTTPS stands for "Secure," and it's achieved through the use of **SSL/TLS** (Secure Sockets Layer/Transport Layer Security) protocols.

How HTTPS Works

1. **Encryption**: HTTPS encrypts the data, making it unreadable to anyone attempting to intercept the communication. This is especially important for sensitive information like passwords or credit card details.
2. **Authentication**: HTTPS verifies that you're communicating with the intended website and not an impostor. This is achieved through **digital certificates** issued by trusted Certificate Authorities (CAs).
3. **Data Integrity**: HTTPS ensures that the data sent between you and the server is not altered or tampered with during transmission.

Benefits of HTTPS

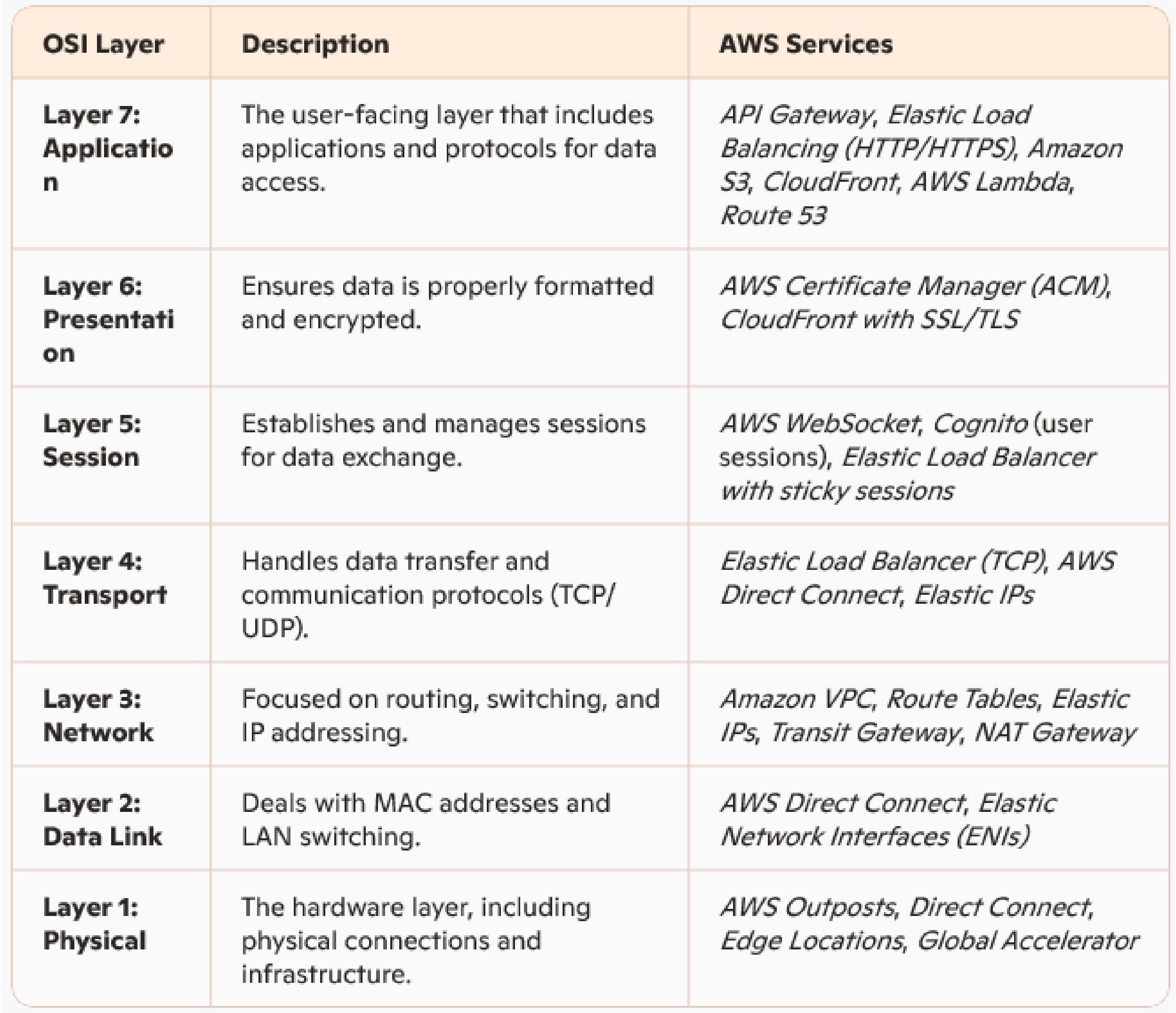
- **Security**: Protects sensitive data from being intercepted (man-in-the-middle attacks).
- **Trust**: Websites with HTTPS often display a padlock icon in the browser's address bar, signaling that the connection is secure.
- **SEO Advantage**: Search engines, like Google, prioritize HTTPS websites in search results, offering an incentive for website owners to adopt HTTPS.

Real-World Example

When you access a website like https://example.com, your browser and the server establish a secure, encrypted connection using an SSL/TLS handshake. This ensures your interaction is private and protected from eavesdropping or data theft.

Let me know if you'd like to dive deeper into SSL/TLS or learn how to secure your own website! 😊

Visual mapping between all OSI layers and AWS services



OSI Layer	**Description**	**AWS Services**
Layer 7: Application	The user-facing layer that includes applications and protocols for data access.	*API Gateway, Elastic Load Balancing (HTTP/HTTPS), Amazon S3, CloudFront, AWS Lambda, Route 53*
Layer 6: Presentation	Ensures data is properly formatted and encrypted.	*AWS Certificate Manager (ACM), CloudFront with SSL/TLS*
Layer 5: Session	Establishes and manages sessions for data exchange.	*AWS WebSocket, Cognito* (user sessions), *Elastic Load Balancer with sticky sessions*
Layer 4: Transport	Handles data transfer and communication protocols (TCP/UDP).	*Elastic Load Balancer (TCP), AWS Direct Connect, Elastic IPs*
Layer 3: Network	Focused on routing, switching, and IP addressing.	*Amazon VPC, Route Tables, Elastic IPs, Transit Gateway, NAT Gateway*
Layer 2: Data Link	Deals with MAC addresses and LAN switching.	*AWS Direct Connect, Elastic Network Interfaces (ENIs)*
Layer 1: Physical	The hardware layer, including physical connections and infrastructure.	*AWS Outposts, Direct Connect, Edge Locations, Global Accelerator*