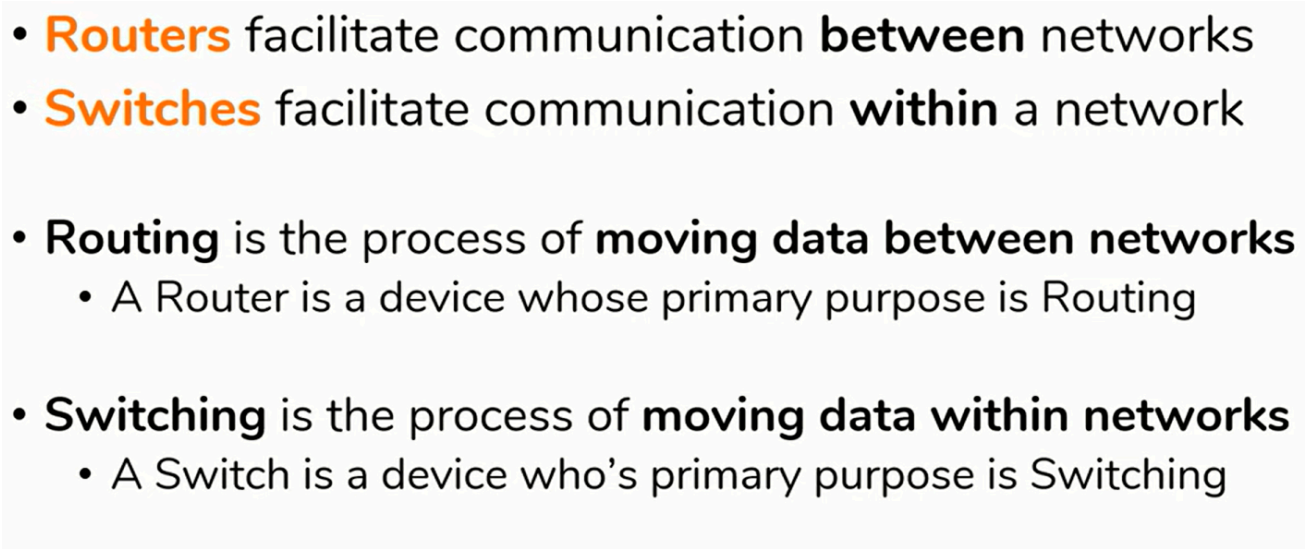
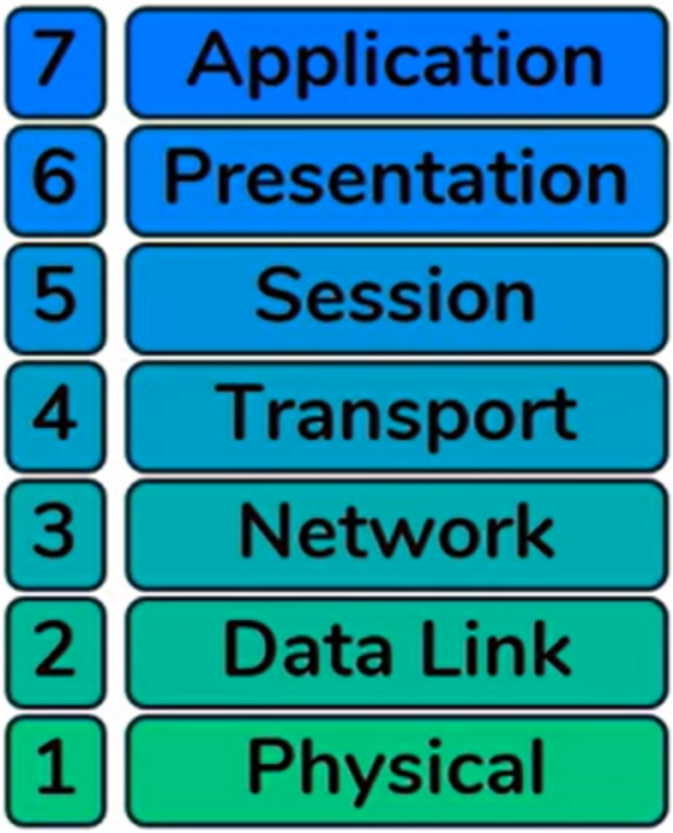
Jad_AbouNajem_summary

Network Fundamentals:

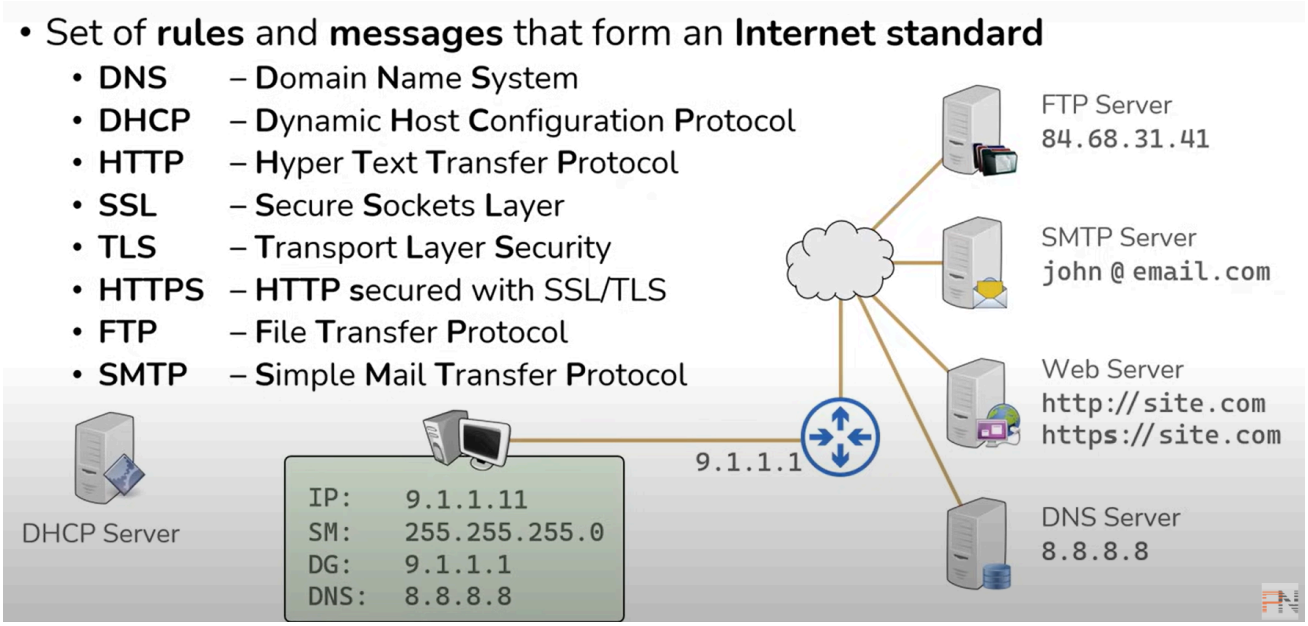
1. client requests, server responds.
2. hubs are multi-port repeaters.
3. bridges sit between Hub-connected hosts.
4. bridges know which hosts are on which side, which solves the hub issue(unwanted hosts receiving packets).
5. switches are a combination of hubs and bridges.
6. switches learn which hosts are on each port.
7. hosts on a network share the same IP address space.
8. routers facilitate communication between networks.
9. routers connect to the internet.
10. each interface in a router is given an IP address related to the network it serves(gateway).
11. 

    OSI Model:

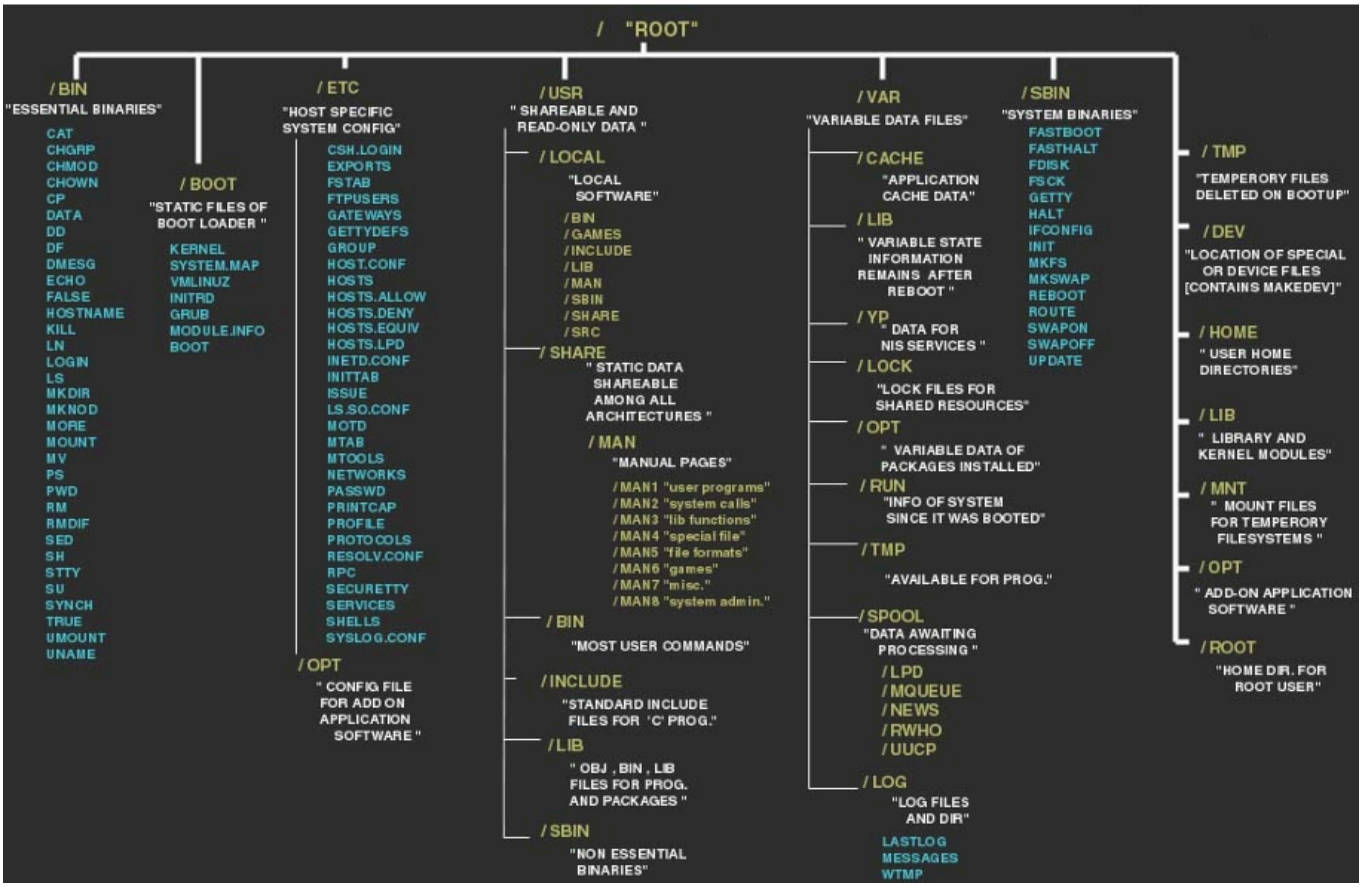


Physical:
transports bits.
Data link:
interacts with the wire (physical layer).
Hop to Hop delivery
NIC: Network Interface Cards/ WIFI- Access Cards.
every NIC has a unique MAC address..
switches are L2 tech(Hop to Hop).
Network:
everything that has IP address in L3 tech (routers, hosts).
End to End delivery.
ARP: Links a L3 address to L2 address.
Transport:
Service to Service
distinguish data streams.
makes sure the right program receives the right data.
Addressing Scheme: Ports.
TCP(reliability) - UDP(efficiency).
Encapsulation:
TCP+ Data ->segment
IP + TCP + Data ->packet
L2 + IP + TCP + Data->frame

Then De-Encapsulation happens when received.
Subnet mask identifies the size of the IP network.
Switches: Learn Flood Forward
Routing tables are populated by:
Directly Connected
Static Routes
Dynamic Routes
When 0.0.0.0/0 exists in routing table--> "for everything else, go here".
HTTPS is HTTP secured within a SSL or TLS tunnel.



Linux file system



HTTP Responses:

1. 1xx:Informational.
2. 2xx:successful.
3. 3xx:The client is redirected to different resources.
4. 4xx:The request contains an error of some kind.
5. 5xx:The server encountered an error fulfilling the request.

specific:

1. 200:successful
2. 302: Found
3. 401: Unauthorized
4. 403: Forbidden
5. 404: Not Found
6. 500: Internal Server Error

proxies:

1. forward proxies work for the client , reverse for the server(Cloudflare is a reverse proxy)

android:

1. Linux Kernel Layer: hardware abstraction, memory management..
2. Hardware abstraction layer: above Linux kernel, allows device manufacturers to create drivers for specific hardware conf while maintaining comp with android framework
3. Native libraries layer: written in c/c++, enhance Java-based Android framework's capabilities.
4. Android Runtime Layer: executing and managing bytecode.
5. Java API Framework: set of libraries, APIs, runtime env.
6. Application Layer: topmost layer , user-installed apps.

APK file struct:

- **META-INF:** Contains verification information generated when the app is signed.
- **MANIFEST.MF:** A file that lists the names and hashes (typically SHA256 in Base64) of all the files in the APK.
- **CERT.SF:** Contains a list of names and hashes corresponding to the lines in the MANIFEST.MF file.
- **CERT.RSA:** This file includes the public key and the signature for the CERT.SF file.
- **Assets:** Holds assets bundled with the application, which can be accessed by the AssetManager. These assets can include images, videos, documents, databases, and more.
- **lib:** Contains native libraries with compiled code for different device architectures.
- **res:** Contains predefined application resources, such as XML files for state lists of colors, UI layouts, fonts, values, and more.
- **AndroidManifest.xml:** A manifest file that defines the application's package name, activities, resources, version, and other essential information.
- **classes.dex:** Contains all the Java classes in the dex (Dalvik Executable) file format, ready to be executed by the Android Runtime.
- **resources.arsc:** Holds precompiled resources, linking the code to the corresponding resources.