Most important core protocols and ports in networking

- **Name & Purpose**
- **Port numbers**
- **How they work (simple)**
- **Real life example**

🔷 1. FTP (File Transfer Protocol)

- **Purpose**: Transfer files between client and server.
- **Port**: TCP 21 (control), TCP 20 (data).
- **How it works**: Client connects to FTP server, authenticates, then uploads/downloads files.
- **Example**: A web developer uploads website files to a hosting server.

🔷 2. SSH (Secure Shell)

- **Purpose**: Secure remote login and command execution.
- **Port**: TCP 22.
- **How it works**: Encrypts all traffic (unlike Telnet). Used mainly in Linux/Unix.
- **Example**: A system admin logs into a Linux server from home securely.

🔷 3. Telnet

- **Purpose**: Remote login and command execution (insecure).
- **Port**: TCP 23.
- **How it works**: Sends data in **plain text** → easily intercepted. Rarely used today.
- **Example**: In the past, admins managed routers/switches with Telnet (now replaced by SSH).

🔷 4. SMTP (Simple Mail Transfer Protocol)

- **Purpose**: Sending email between mail servers.
- **Port**: TCP 25 (standard), TCP 587 (submission), TCP 465 (secure, legacy).
- **How it works**: When you send an email, your client uses SMTP to deliver it to the mail server.
- **Example**: Gmail sending an email to Outlook.com.

🔷 5. DNS (Domain Name System)

- **Purpose**: Translates domain names → IP addresses.
- **Port**: UDP 53 (queries), TCP 53 (zone transfers).
- **How it works**: You type `google.com`, DNS resolves it to an IP (e.g., 142.250.72.206).
- **Example**: Without DNS, you'd have to remember IPs for every website.

🔷 6. DHCP (Dynamic Host Configuration Protocol)

- **Purpose**: Automatically assigns IP addresses and network configs.
- **Port**: UDP 67 (server), UDP 68 (client).
- **How it works**: Client sends a broadcast "I need an IP," DHCP server replies with IP, subnet mask, gateway, DNS.
- **Example**: Your phone automatically gets an IP when joining Wi-Fi.

🔷 7. HTTP (HyperText Transfer Protocol)

- **Purpose**: Web browsing (not secure).
- **Port**: TCP 80.
- **How it works**: Client (browser) requests a webpage → server responds with HTML.
- **Example**: Visiting a website using `http://`.

🔷 8. HTTPS (HyperText Transfer Protocol Secure)

- **Purpose**: Secure web browsing (encrypted with SSL/TLS).
- **Port**: TCP 443.
- **How it works**: Same as HTTP but with encryption → prevents eavesdropping.
- **Example**: Online banking, shopping, secure logins (`https://`).

🔷 9. POP3 (Post Office Protocol v3)

- **Purpose**: Retrieve emails from mail server (downloads & usually deletes from server).
- **Port**: TCP 110 (POP3), TCP 995 (POP3S secure).
- **How it works**: Downloads emails to local device → not stored on server.
- **Example**: Old email clients like Outlook Express.

🔷 10. IMAP (Internet Message Access Protocol)

- **Purpose**: Retrieve emails, but keeps them stored on the server.
- **Port**: TCP 143 (IMAP), TCP 993 (IMAPS secure).
- **How it works**: Synchronizes emails across devices (server keeps master copy).
- **Example**: Gmail app syncing email across your phone, tablet, and laptop.

✅ Quick Revision Table

Protocol	Full Form	Port(s)	Purpose
FTP	File Transfer Protocol	TCP 20, 21	File transfer
SSH	Secure Shell	TCP 22	Secure remote login
Telnet	Telecommunication Network	TCP 23	Remote login (insecure)
SMTP	Simple Mail Transfer Protocol	TCP 25, 587, 465	Send emails
DNS	Domain Name System	UDP/TCP 53	Domain → IP resolution
DHCP	Dynamic Host Config Protocol	UDP 67/68	Auto IP assignment
HTTP	HyperText Transfer Protocol	TCP 80	Web browsing (not secure)
HTTPS	HyperText Transfer Protocol Secure	TCP 443	Secure web browsing
POP3	Post Office Protocol v3	TCP 110/995	Download emails
IMAP	Internet Message Access Protocol	TCP 143/993	Sync emails

---

✅ **Summary**:

- FTP, HTTP, SMTP = **transfer protocols** (files, web pages, emails).
- SSH, Telnet = **remote access protocols**.
- DNS, DHCP = **network services**.
- POP3, IMAP = **email retrieval protocols**.
- HTTPS = **secure version of HTTP**.

💠 What is NetBIOS?

- **NetBIOS** stands for **Network Basic Input/Output System**.
- It's an **API (Application Programming Interface)** created in the 1980s that allows applications on different computers to communicate over a local network.
- It's **not a protocol by itself**—it's more like a set of functions (like name resolution, session management, data transfer).

Think of it like a "language" that early Windows computers used to talk to each other.

🔷 What is NetBIOS over TCP/IP (NetBT)?

- Originally, NetBIOS worked only on small LANs using protocols like **NetBEUI**.
- But when TCP/IP became the standard, Microsoft adapted it.
- **NetBIOS over TCP/IP (NetBT)** is the method of running NetBIOS services on top of the **TCP/IP stack**.
- This is what allows NetBIOS-based applications (like Windows file sharing in older systems) to work across larger routed networks.

🔷 Ports used by NetBIOS/NetBT:

- **UDP 137** → NetBIOS Name Service (name registration and resolution)
- **UDP 138** → NetBIOS Datagram Service (connectionless communication, broadcasts)
- **TCP 139** → NetBIOS Session Service (connection-oriented communication, file/printer sharing before SMB moved to port 445)

🔷 Functions of NetBIOS:

1. **Name Service** – Computers register names (like `PC1`, `SERVER01`) and resolve them to IP addresses.
2. **Datagram Distribution** – Send small packets (like broadcasts) without a full connection.
3. **Session Service** – Establish full client-server sessions (like file sharing).

🔷 Real Life Example

Imagine a small office in the 1990s:

- Alice's computer is named **ALICE-PC**
- Bob's computer is named **BOB-PC**
- Both are on the same LAN.
- Alice opens "Network Neighborhood" in Windows 95 and sees Bob's computer.
- Behind the scenes:
  - **NetBIOS Name Service** resolved `BOB-PC` → Bob's IP address.
  - A **NetBIOS Session** was established over **TCP 139**.
  - Alice can now access shared files on Bob's machine.

Today, modern Windows systems mostly use **DNS + SMB over TCP 445** instead of NetBIOS, but NetBIOS/NetBT is often still enabled for backward compatibility.

Disable NetBIOS over TCP/IP

12

NetBIOS over TCP/IP can be disabled to enhance network security and reduce unnecessary network traffic. Here are the steps to disable it on a Windows machine:

Using Control Panel

1. **Open Control Panel**: Press the Start key, type "Control Panel," and open it.
2. **Network and Sharing Center**: Click on "Network and Sharing Center."
3. **Change Adapter Settings**: In the left pane, select "Change adapter settings."
4. **Properties**: Right-click on your network connection (e.g., Local Area Connection) and select "Properties."
5. **Internet Protocol Version 4 (TCP/IPv4)**: Select it and click the "Properties" button.
6. **Advanced Settings**: Click the "Advanced" button.
7. **WINS Tab**: Go to the WINS tab.
8. **Disable NetBIOS over TCP/IP**: Select "Disable NetBIOS over TCP/IP."
9. **Apply and Exit**: Click "Apply" and then "OK" to exit.

✅ **Quick summary:**

- NetBIOS = old naming + communication system.
- NetBIOS over TCP/IP (NetBT) = allows it to work on TCP/IP networks.
- Still seen in Windows file sharing (legacy), but mostly replaced by DNS + SMB on port 445.

🔷 What is LDAP?

- **LDAP = Lightweight Directory Access Protocol**
- It's an **open, standard protocol** used to access and manage directory services (like a phonebook for network resources).
- Originally created as a "lightweight" version of the heavier X.500 directory standard.

In plain words: LDAP is how applications talk to a **directory server** to look up info about users, computers, groups, printers, etc.

🔷 What is a Directory?

Think of a **directory** like:

- A **database** but optimized for reading/searching, not heavy transactions.
- It stores hierarchical information (like a tree):
  - Example:

-
  ```
  dc=company,dc=com
   └── ou=Users
   └── cn=Alice
   └── cn=Bob
   └── ou=Groups
   └── cn=Admins
   └── cn=HR
  ```

Here:

- `dc` = domain component
- `ou` = organizational unit
- `cn` = common name

🔷 What LDAP Does

1. **Authentication** → “Who are you?” (user login).
2. **Authorization** → “What can you access?” (permissions).
3. **Directory Lookup** → Finding user info, email addresses, printer locations, etc.

🔷 Ports used by LDAP

- **TCP/UDP 389** → Standard LDAP
- **TCP 636** → LDAPS (LDAP over SSL/TLS, secure)

🔷 Real Life Examples

1. **Corporate Login**
   - When you log into a Windows domain (Active Directory), your username/password is checked against the LDAP directory.
   - Example: Alice logs in, AD (via LDAP) confirms her password and group memberships.
2. **Email Lookup**
   - In Outlook/Thunderbird, typing a name auto-suggests emails from the corporate directory. That’s LDAP at work.
3. **Application Authentication**
   - Web apps (like a company intranet, Jira, or VPN login) often use LDAP to validate employees instead of creating separate accounts.

🔷 Key LDAP Terms

- **Entry** → One item (user, group, device).
- **Attribute** → A property of an entry (name, email, phone).
- **DN (Distinguished Name)** → Full path to an entry (like a file path).
- **Schema** → Rules for what attributes an entry can have.

✅ Quick Summary

- **LDAP** = Protocol to query & modify directory services.
- Used for authentication, authorization, and lookups.
- Runs on **389 (LDAP)** and **636 (LDAPS)**.
- Commonly used in **Active Directory** and corporate logins.

🔷 LDAP vs Active Directory

✅ What is Active Directory (AD)?

- **Active Directory** is Microsoft's **directory service** for Windows domain networks.
- It stores info about **users, computers, groups, printers, servers, policies** in a hierarchical structure.
- It's basically Microsoft's version of a **directory database**.

Think of AD as the **big phonebook of your entire organization's IT resources**.

✅ Where does LDAP fit in?

- LDAP is the **protocol** used to talk to directory services.
- AD is a **directory service** that understands LDAP (and some other protocols like Kerberos for authentication).
- So:
  - **AD = the actual database/service**
  - **LDAP = the language used to query/update that database**

Example:

- A Linux app wants to authenticate a Windows user.
- The app sends the login credentials to AD **via LDAP**.
- AD checks the credentials, then responds with "allowed" or "denied."

🔷 How LDAP Works in AD

1. **Authentication**

   - User enters `username + password`.
   - LDAP binds (connects) to AD to check credentials.
2. **Authorization**
   - Once authenticated, LDAP can query AD for user's **group memberships** (Admins, HR, Students, etc.).
   - Apps use this info to decide what the user can access.
3. **Search & Directory Lookup**
   - Example: Outlook using **Global Address List (GAL)** → powered by LDAP queries to AD.

🔷 Ports in AD (using LDAP)

- **389 (LDAP)** – Clear text queries/authentication.
- **636 (LDAPS)** – Secure LDAP over SSL/TLS.
- AD also uses **3268 (Global Catalog LDAP)** for searching across multiple domains in a forest.

🔷 Real Life Example: Login Flow

Imagine Alice logs into her work laptop (domain-joined):

1. Alice enters username + password.
2. Laptop contacts **Domain Controller (AD server)**.
3. Authentication: AD verifies her password (via Kerberos/LDAP bind).
4. Authorization: AD checks if Alice is in the **HR group**.
5. Result: Alice gets access to HR apps and files.

🔷 Quick Analogy

- **Active Directory = Bank vault full of account records**
- **LDAP = The teller counter (protocol) that lets you ask "Does this account exist?" or "What's the balance?"**

✅ **Summary**:

- LDAP is **not Active Directory** — it's the protocol AD (and other directories) use.
- AD supports LDAP for authentication, authorization, and directory lookups.
- Every time you log into Windows domain, join a printer, or search for an employee in Outlook, LDAP is working in the background.

🔷 What is SMB?

- **SMB = Server Message Block**
- A **network file sharing protocol** that allows computers to:
  - Share files and folders
  - Share printers
  - Communicate between client and server applications
- Originally developed by IBM in the 1980s, later adopted and extended by Microsoft.

📌 **Think of SMB as the protocol that makes "Windows File Sharing" work.**
When you go to `\\Server\SharedFolder`, you're using SMB.

🔷 What is CIFS?

- **CIFS = Common Internet File System**
- Microsoft's specific **implementation of SMB (version 1.0)** in the 1990s.
- It extended SMB to work over the Internet and TCP/IP (instead of only NetBIOS).
- Today, CIFS is considered **old, chatty, and inefficient** compared to newer SMB versions.

📌 So: **CIFS is just SMB v1** (legacy).

🔷 SMB Protocol Versions

1. **SMB 1.0 (CIFS)** → Old, insecure, very chatty (lots of overhead).
2. **SMB 2.0/2.1** → Introduced in Windows Vista/Server 2008 (faster, fewer commands).
3. **SMB 3.x** → Windows 8/Server 2012 onward (adds encryption, performance boosts).
   - Still widely used today.

🔷 Ports Used

- **TCP 445** → Direct SMB over TCP (modern use).
- **TCP 139** → SMB over NetBIOS Session Service (legacy).

👉 Before Windows 2000, SMB mostly ran over NetBIOS (port 139).
👉 Modern systems use **port 445** directly.

🔷 Functions of SMB/CIFS

- **File Sharing** → Access files on remote computers.
- **Printer Sharing** → Print to shared network printers.
- **Inter-process Communication (IPC)** → Messaging, named pipes.
- **Network Browsing** → See shared resources on the network (e.g., "Network Neighborhood").

🔷 Real Life Example

- You're at work and open `\\HR-SERVER\Payroll` from your laptop.
- Behind the scenes:
  1. Your PC connects to **HR-SERVER** over **TCP 445**.
  2. SMB protocol negotiates a session.
  3. You authenticate (via AD/LDAP/Kerberos).
  4. SMB grants access to the shared folder.
  5. You can now open, copy, or save files — just like they're local.

🔷 SMB vs CIFS (Quick Comparison)

Feature	SMB	CIFS
Stands for	Server Message Block	Common Internet File System
Type	General protocol (many versions)	Microsoft's SMB v1 implementation
Era	1980s → present	1990s (Windows NT, 2000, XP)
Efficiency	Newer versions (2.x, 3.x) are fast & secure	Very chatty, inefficient
Security	SMB 3 adds encryption & signing	Weak security (easy to exploit)
Current Use	Still widely used (file & printer sharing)	Deprecated (disabled in modern Windows)

✅ Summary

- **SMB = protocol for file/printer sharing**.
- **CIFS = Microsoft's old SMB v1** (now obsolete).
- Runs on **port 445 (modern)** and **139 (legacy)**.
- Still critical in Windows networks, though newer SMB versions replaced CIFS.

🔷 What is RDP?

- **RDP = Remote Desktop Protocol**
- Developed by Microsoft.
- Allows a user to connect to another computer over the network and control its desktop as if they were sitting in front of it.

📌 It's basically **screen sharing + input redirection** at the protocol level.

- Your keyboard/mouse inputs get sent to the remote machine.
- The remote computer sends back display updates.

🔷 Ports Used

- **TCP 3389** → Default RDP port.
- **UDP 3389** → Used in newer versions for performance improvements.

🔷 How RDP Works

1. You open **Remote Desktop Connection** (`mstsc.exe`) on Windows.
2. Enter the IP address or hostname of the target machine.
3. Your computer establishes an encrypted session on **port 3389**.
4. Remote machine asks for login credentials (username/password, often via Active Directory).
5. Once authenticated, you see the remote desktop, and can work on it just like it's local.

🔷 Key Features of RDP

- **Graphical Desktop Access** → Not just files, the whole desktop.
- **Clipboard Sharing** → Copy-paste text/files between local & remote machine.
- **Printer & Drive Redirection** → Print on your local printer while working remotely.
- **Session Management** → Multiple users can connect to a server simultaneously (like in Windows Server Remote Desktop Services).
- **Encryption** → RDP traffic is encrypted (TLS/SSL).

🔷 Real Life Examples

1. **IT Support** → Helpdesk connects to your PC remotely to fix issues.
2. **Work from Home** → Employees use RDP to log into their office desktop from home.

3. **Server Administration** → Sysadmins manage Windows servers in data centers using RDP.

🔷 Security Considerations

- **RDP is a common hacking target** (port 3389 often scanned).
- Vulnerabilities (like **BlueKeep**) have been exploited.
- Best practices:
  - Use **strong passwords + account lockout**.
  - Use **VPN + firewall** instead of exposing port 3389 to the internet.
  - Enable **Network Level Authentication (NLA)** for better security.
  - Keep RDP patched.

🔷 Quick Comparison (RDP vs SMB/SSH/VNC)

Protocol	Purpose	Default Port	Common Use
RDP	Remote desktop (full GUI)	TCP 3389	Windows remote access
SMB	File/printer sharing	TCP 445	Shared drives/folders
SSH	Secure command-line access	TCP 22	Linux remote management
VNC	Remote desktop (cross-platform)	TCP 5900+	GUI remote control, less efficient than RDP

✅ Summary

- **RDP** = Microsoft protocol for remote desktop access.
- Uses **TCP/UDP 3389**.
- Encrypted, supports full desktop control, file/clipboard/printer redirection.
- Very powerful but must be secured against attacks.

TCP and UDP

🔷 TCP (Transmission Control Protocol)

- **Connection-oriented** → Establishes a connection before data transfer (like a phone call).
- **Reliable** → Ensures data arrives correctly, in order, and without loss.
- **Error Checking & Retransmission** → If packets are lost/dropped, TCP resends them.
- **Flow Control & Congestion Control** → Prevents overwhelming the receiver or network.

- **Slower** than UDP (because of overhead), but reliable.

✅ **Real Life Analogy**: Sending a registered letter with tracking → You know exactly when it's delivered.

✅ **Examples of Protocols using TCP**:

- HTTP/HTTPS (web browsing) → Port 80/443
- FTP (file transfer) → Port 21
- SMTP (email sending) → Port 25
- Telnet/SSH (remote login) → Port 23/22

🔷 UDP (User Datagram Protocol)

- **Connectionless** → No handshake, just sends data (like a postcard).
- **Unreliable** → No guarantee of delivery, order, or duplication checks.
- **No Flow Control** → Just sends as fast as possible.
- **Lightweight & Fast** → Minimal overhead, best for speed over reliability.

✅ **Real Life Analogy**: Sending a regular postcard → It might arrive, might get lost, but it's fast and simple.

✅ **Examples of Protocols using UDP**:

- DNS → Port 53 (fast lookups)
- DHCP → Ports 67/68 (quick IP assignment)
- VoIP (Skype, WhatsApp calls) → Needs speed > reliability
- Online Gaming → Dropping 1 packet doesn't matter, but speed does

---

🔷 Key Differences (TCP vs UDP)

Feature	TCP	UDP
Type	Connection-oriented	Connectionless
Reliability	Reliable (acknowledgements, retransmissions)	Unreliable (no delivery guarantee)
Speed	Slower (more overhead)	Faster (less overhead)
Ordering	Packets arrive in sequence	Packets may arrive out of order
Error Checking	Yes (and fixes errors)	Basic checksum only
Use Case	When accuracy matters (web, file transfer, emails)	When speed matters (streaming, calls, games)

🔷 Quick Real World Examples

- **TCP**:
  - Loading a web page (HTTP/HTTPS)
  - Downloading a file via FTP
  - Logging into a remote server (SSH)
- **UDP**:
  - Watching YouTube or Netflix (streaming video/audio)
  - Playing online games (Fortnite, COD)
  - Making a VoIP call (Zoom, WhatsApp call)

✅ **Summary**:

- **TCP = Reliable, slower, ordered → accuracy matters**
- **UDP = Fast, lightweight, unordered → speed matters**

🔷 TCP & UDP are Transport Layer Protocols

- They sit at the **Transport Layer (Layer 4)** of the **OSI model**.
- Their job is **to move raw data** between applications on two devices.
- But… they don't define **what that data means**.

👉 Example:

- TCP can guarantee that 10 packets arrive in order.
- But TCP doesn't know if those packets contain:
  - An email 📧
  - A web page 🌐
  - A file 📂
  - Or a video 🎥

🔷 Application Layer Protocols (HTTP, SMTP, etc.)

These sit **above TCP/UDP** at the **Application Layer (Layer 7)**.
They **define the rules for how applications interpret the data**.

Examples:

- **HTTP**: Defines how browsers request web pages and how servers respond.
- **SMTP**: Defines how emails are formatted and delivered.
- **FTP**: Defines commands for uploading/downloading files.
- **DNS**: Defines how name queries work.

Without these protocols, raw TCP/UDP streams would just look like meaningless bits.

🔷 Analogy (Simple)

- **TCP/UDP = Roads/Highways** → Just deliver cars (data packets) from A to B.
- **Application Protocols = The Cars & Their Rules** →
  - HTTP = Truck carrying web pages
  - SMTP = Mail van carrying emails
  - FTP = Cargo truck carrying files
  - DNS = Taxi driver asking for directions

Roads don't care *what* the car is carrying — they just deliver it.
Cars (application protocols) define **what is being carried and how**.

🔷 Real Example Flow (Sending an Email)

1. **SMTP (Application Layer)** → Formats email and defines rules for sending it.
2. **TCP (Transport Layer)** → Ensures the email data arrives reliably.
3. **IP (Network Layer)** → Routes packets across networks.
4. **Ethernet/Wi-Fi (Data Link Layer)** → Physically sends bits over cable/wireless.

So **SMTP needs TCP** to deliver reliably, but TCP alone cannot send an email.

🔷 Quick Summary

- **TCP/UDP = Transport mechanisms** → *how data gets delivered*.
- **Application protocols = Define meaning of the data** → *what the data is for*.
- That's why we need **both**:
  - TCP/UDP provide **delivery**.
  - HTTP/SMTP/DNS/etc. provide **context and rules**.