**Complete and Exhaustive Notes: Kunal Kushwahah’s Computer Networking Full Course – OSI Model Deep Dive**

**📘 Chapter 1: Introduction to Networking & Internet Basics**

**✅ What is a Network?**

* Simple Definition: Jab do ya do se zyada computers ek dusre se connect hote hain taaki woh data share kar sakein.
* Real-Life Analogy: College mein 5 friends ek LAN cable se connect hain – ek ne movie chalayi toh sab dekh rahe hain.
* Technical Term: This is called a **Computer Network**.

**✅ What is the Internet?**

* Definition: Internet = "Network of Networks"
* Real-Life Analogy: Ghar ke Wi-Fi se connected devices + neighbor ke devices + worldwide networks = Internet
* Diagram: [Imagine: Computer A <--> Wi-Fi Router <--> Internet <--> Google Server]

**✅ Full Form of COMPUTER:**

* **C**ommonly
* **O**riented
* **M**achine
* **P**articularly
* **U**sed for
* **T**echnical
* **E**ducation and
* **R**esearch

**📘 Chapter 2: Development of the Internet (ARPANET to TCP/IP)**

**✅ Cold War & ARPA**

* 1957: USSR launched **Sputnik** satellite
* US created **ARPA (Advanced Research Projects Agency)**
* ARPA developed **ARPANET** to enable scientific communication among distant locations

**✅ First Nodes of ARPANET:**

1. MIT
2. Stanford
3. UCLA
4. University of Utah

**✅ TCP/IP Introduction**

* **TCP (Transmission Control Protocol)**:
  + Ensures data is delivered **completely and in order**
  + Used in emails, file transfer
* **IP (Internet Protocol)**:
  + Assigns unique addresses to each device (like home address)

**✅ Protocols Definition:**

* Set of rules that define how data is communicated over the internet
* Example:
  + Video Call? ✉✨ Use UDP (Speed > Accuracy)
  + Secure File Transfer? ✉✨ Use TCP (Accuracy > Speed)

**📘 Chapter 3: World Wide Web (WWW)**

**✅ Tim Berners-Lee**

* Invented the **World Wide Web**
* First website: [info.cern.ch](http://info.cern.ch)

**✅ Key Concepts:**

* **URL (Uniform Resource Locator)**: Address of a web page
* **Hyperlink**: Clickable link to navigate between pages
* **Search Engines**: Started with Yahoo; later Google etc.

**✅ Web Server**

* Stores & serves website files to users
* You can host your own using Node.js, Apache, etc.

**📘 Chapter 4: Why Protocols are Needed**

**✅ Why Rules are Important:**

* Everyone speaking different formats? ✖️
* Protocols = Standard rules to ensure smooth communication

**✅ Internet Society**

* Maintains internet rules and protocols
* Submissions called **RFC (Request for Comments)**
* Website: <https://www.internetsociety.org/>

**✅ Common Protocols:**

|  |  |  |
| --- | --- | --- |
| **Protocol** | **Purpose** | **Use-Case** |
| TCP | Reliable Transmission | Email, FTP |
| UDP | Fast, Unreliable | Video Calls, Gaming |
| HTTP | Web Pages | Browser to Server |
| HTTPS | Secure Web Pages | Online Banking |

**📘 Chapter 5: Client-Server Model & IP Addressing**

**✅ What is a Server?**

* A computer that serves resources
* Example: Google Server

**✅ What is a Client?**

* A user/device that sends a request to the server
* Example: Your laptop

**✅ Flow:**

1. You type google.com
2. Request sent to Google Server
3. Server responds with the web page

**✅ IP Address:**

* Unique address of a device on the internet
* Format: 192.168.0.1
* Like your home address for deliveries

**📘 Chapter 6: Routers, NAT, Ports, DHCP**

**✅ Router vs Modem:**

|  |  |
| --- | --- |
| Device | Function |
| **Modem** | Converts digital <--> analog |
| **Router** | Routes data to the correct device |

**✅ NAT (Network Address Translation):**

* Translates global IP to local IP
* Enables multiple devices to use one global IP

**✅ DHCP:**

* Assigns dynamic local IPs to devices
* Protocol used by the router

**✅ Ports:**

* Used to identify applications on a device
* Port 80: HTTP
* Port 443: HTTPS

**📘 Chapter 7: Bandwidth & Data Transmission**

**✅ Bandwidth Units:**

|  |  |
| --- | --- |
| Unit | Meaning |
| Kbps | Kilobits/sec |
| Mbps | Megabits/sec |
| Gbps | Gigabits/sec |

**✅ Upload vs Download:**

* **Upload**: Sending data
* **Download**: Receiving data

**✅ Guided vs Unguided Transmission:**

|  |  |
| --- | --- |
| Type | Example |
| Guided | Ethernet cables |
| Unguided | Wi-Fi, Bluetooth |

**📘 Chapter 8: Undersea Cables & Internet Infrastructure**

**✅ Submarine Cables:**

* Cables under oceans connecting continents
* Maintained by big players (e.g. Google)
* Faster than satellite communication

**✅ ISPs (Internet Service Providers):**

|  |  |
| --- | --- |
| Tier | Example |
| Tier 1 | Tata (India) |
| Tier 2 | Airtel, Jio |

**📘 Chapter 9: LAN, MAN, WAN & Network Topologies**

**✅ Network Types:**

|  |  |
| --- | --- |
| Type | Coverage |
| LAN | Small Area |
| MAN | City Level |
| WAN | Country/World |

**✅ Topologies:**

|  |  |  |
| --- | --- | --- |
| Topology | Description | Limitation |
| Bus | One long cable | One break = all fail |
| Ring | Ring shape | Break = no transmission |
| Star | Central hub | Hub failure = all fail |
| Tree | Bus + Star | Complex wiring |
| Mesh | All to all | Expensive, hard to scale |

**📘 Chapter ➊: OSI Model Deep Dive**

**✅ Seven Layers:**

1. Application Layer
2. Presentation Layer
3. Session Layer
4. Transport Layer
5. Network Layer
6. Data Link Layer
7. Physical Layer

Each layer is explained in detail (will continue in next expansion...)

**🧠 Chapter 🔟 Continued: OSI Model – Layer-by-Layer Explanation**

**1️⃣ Application Layer**

* **User-facing layer** — yahi layer hoti hai jahan se user data bhejta hai
* **Examples**: WhatsApp, Chrome, Email clients
* **Protocols used**: HTTP, HTTPS, FTP, SMTP, DNS
* **Function**: Sends user-generated data to Presentation Layer

**2️⃣ Presentation Layer**

* **Function**: Data ko machine-understandable format mein convert karta hai (Encoding/Decoding)
* **Tasks Performed**:
  + **Translation**: Characters ko binary format mein badalna
  + **Encryption/Decryption**: Secure communication ke liye (e.g., SSL/TLS)
  + **Compression/Decompression**: Speed ke liye data size kam karna
* **Example**: SSL for secure browser connections

**3️⃣ Session Layer**

* **Function**: Communication sessions establish, maintain, terminate karta hai
* **Session Example**: Jab Flipkart/Amazon mein login karte ho, ek session create hota hai
* **Tasks**:
  + **Authentication** (username/password)
  + **Authorization** (permission check)
  + **Session Control**: Timeout, sync, resume

**4️⃣ Transport Layer**

* **Function**: Reliable ya unreliable delivery decide karta hai
* **Protocols**:
  + **TCP**: Reliable, ordered, error-checked delivery
  + **UDP**: Fast, but no guarantee of order
* **Concepts**:
  + **Segmentation**: Large data is broken into segments
  + **Port Numbers**: Identify which app to send data to (e.g., Port 80 → Browser)
  + **Flow Control**: Sender-Receiver speed coordination
  + **Error Control**: Packet corruption detect karna (via checksum)

**5️⃣ Network Layer**

* **Function**: Decide karta hai ki data kis raste se destination tak pahunchega
* **Key Terms**:
  + **IP Addressing**: Logical addressing for sender & receiver
  + **Routing**: Best path select karna
* **Devices Involved**: Routers
* **Data Unit**: Packets

**6️⃣ Data Link Layer**

* **Function**: Device-to-device data transfer
* **Key Concepts**:
  + **MAC Addressing**: Physical address of device NIC (Network Interface Card)
  + **Framing**: Converts packets → frames
  + **Error Detection**: Parity checks, CRC
* **Devices**: Switches
* **Example**: Ghar ka Wi-Fi router assign karta hai MAC-based communication

**7️⃣ Physical Layer**

* **Function**: Bits ko physical signal mein convert karta hai (0s and 1s)
* **Mediums**:
  + **Wired**: Copper cable, fiber optics
  + **Wireless**: Wi-Fi, Bluetooth
* **Signals**:
  + Electrical (Ethernet)
  + Light (Fiber Optic)
  + Radio (Wi-Fi)
* **Data Unit**: Bits

**🔁 OSI Layer Communication Flow (Sender to Receiver)**

Sender Side:

App → Presentation → Session → Transport → Network → Data Link → Physical →

[Transmission Medium]

Receiver Side:

← Physical ← Data Link ← Network ← Transport ← Session ← Presentation ← App

**🧠 Key Takeaways:**

* **Each layer talks to its peer layer on the other side**
* Layer below handles the technical transport
* Layer above assumes lower layers will deliver the job

**Chapter 11: TCP/IP Model vs OSI Model**

**✅ Introduction:**

* **OSI Model** = Theoretical model (7 layers)
* **TCP/IP Model** = Practical model used on the internet (4 or 5 layers)

**✅ TCP/IP Model Layers:**

1. **Application Layer**
2. **Transport Layer**
3. **Internet Layer**
4. **Network Access Layer** (a.k.a. Link Layer)

**🧵 Comparison Table:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **OSI Model** | **TCP/IP Model** |
| Layers | 7 | 4 or 5 |
| Usage | Theoretical | Practical (Real Internet) |
| Layers Split | Presentation & Session separate | Combined into Application |
| Developed by | ISO | DARPA (U.S. Defense) |
| Protocols Example | FTP, HTTP, SMTP | TCP, IP, ARP, HTTP, DNS |

**🏠 Mapped View:**

|  |  |
| --- | --- |
| OSI Layer | TCP/IP Equivalent |
| Application | Application |
| Presentation | Application |
| Session | Application |
| Transport | Transport |
| Network | Internet |
| Data Link | Network Access |
| Physical | Network Access |

**🔹 Key Differences:**

* **Concept vs Reality**: OSI = teaching model, TCP/IP = actual usage
* **Strictness**: OSI is layered strictly; TCP/IP is more flexible
* **Defined Layers**: OSI has 7 clear layers; TCP/IP merges some

**🔍 Real-World Analogy:**

OSI is like a textbook structure; TCP/IP is your actual working computer.

**🔧 Bonus: Commonly Used Networking Tools**

**1. ipconfig / ifconfig**

* **Use**: Check your IP address (local)
* **Command (Windows)**:

ipconfig

* **Command (Linux/Mac)**:

ifconfig

**2. ping**

* **Use**: Check connection to server (e.g., Google)

ping google.com

**3. tracert / traceroute**

* **Use**: Trace path to a server

tracert google.com # Windows

traceroute google.com # Linux/Mac

**4. curl**

* **Use**: Fetch a URL's response

curl https://example.com

**5. Browser Dev Tools (Inspect Element → Network Tab)**

* **Use**: See HTTP requests, status codes, response times, headers
* **Shortcut**: Ctrl + Shift + I or F12

**Final Summary Tables + Cheatsheets**

| **Topic** | **Key Points** | **Common Ports / Numbers** | **Use Case / Notes** |
| --- | --- | --- | --- |
| **HTTP/HTTPS** | HyperText Transfer Protocol, stateless, HTTPS adds SSL/TLS encryption | HTTP: 80, HTTPS: 443 | Web browsing, secure data transfer |
| **TCP** | Connection-oriented, reliable, uses three-way handshake | Ports: 0–65535 | Ensures ordered data delivery |
| **UDP** | Connectionless, unreliable, faster, no handshake | Ports: 0–65535 | Streaming, gaming, DNS queries |
| **DNS** | Domain Name System, translates domain names to IP addresses | Port 53 | Internet address resolution |
| **DHCP** | Dynamic Host Configuration Protocol, assigns IP addresses automatically | Port 67 (server), 68 (client) | Network device IP configuration |
| **IP** | Internet Protocol, routes packets between hosts | IPv4, IPv6 | Addressing and routing on the internet |
| **MAC Address** | Unique hardware address for network interfaces | — | Layer 2 addressing on local networks |
| **FTP** | File Transfer Protocol, transfers files over TCP | Ports 20 (data), 21 (control) | File upload/download |
| **SMTP** | Simple Mail Transfer Protocol, sends email | Port 25 | Outgoing email |
| **IMAP/POP3** | Email retrieval protocols | IMAP: 143, POP3: 110 | Accessing email from server |
| **SSL/TLS** | Secure Sockets Layer / Transport Layer Security, encrypts data | Often on HTTPS (443) | Secure communication |
| **ARP** | Address Resolution Protocol, maps IP addresses to MAC addresses | — | Local network address resolution |
| **ICMP** | Internet Control Message Protocol, used for diagnostic tools like ping | — | Network diagnostics |

**Protocol Glossary (Simple, Detailed)**

* **HTTP (HyperText Transfer Protocol):**  
  Protocol used to transfer web pages over the internet. It’s stateless, meaning each request is independent.
* **HTTPS (HTTP Secure):**  
  HTTP with encryption using SSL/TLS to ensure secure data transfer.
* **TCP (Transmission Control Protocol):**  
  A connection-oriented protocol that ensures data is delivered reliably and in order. Uses a three-way handshake to establish connections.
* **UDP (User Datagram Protocol):**  
  A connectionless protocol that sends data without guaranteeing delivery, order, or error checking, but is faster than TCP.
* **DNS (Domain Name System):**  
  Converts human-readable domain names (like google.com) into IP addresses that computers use to communicate.
* **DHCP (Dynamic Host Configuration Protocol):**  
  Automatically assigns IP addresses and other network configuration parameters to devices on a network.
* **IP (Internet Protocol):**  
  Routes packets from the source to destination IP addresses across networks.
* **MAC Address (Media Access Control Address):**  
  A unique identifier assigned to a network interface card (NIC) used for communication on a local network.
* **FTP (File Transfer Protocol):**  
  Used to transfer files between client and server on a TCP/IP network.
* **SMTP (Simple Mail Transfer Protocol):**  
  Protocol for sending email messages between servers.
* **IMAP (Internet Message Access Protocol) & POP3 (Post Office Protocol 3):**  
  Protocols used to retrieve email from mail servers.
* **SSL/TLS (Secure Sockets Layer / Transport Layer Security):**  
  Protocols for encrypting data to provide secure communication over a network.
* **ARP (Address Resolution Protocol):**  
  Maps IP addresses to MAC addresses within a local network.
* **ICMP (Internet Control Message Protocol):**  
  Used by network devices to send error messages and operational information (e.g., ping).

**Official References (Reliable Sources)**

| **Topic** | **Reference Link** | **Notes** |
| --- | --- | --- |
| HTTP/HTTPS | [MDN HTTP](https://developer.mozilla.org/en-US/docs/Web/HTTP) | Comprehensive docs on HTTP methods, status codes, headers |
| TCP/UDP | [MDN TCP](https://developer.mozilla.org/en-US/docs/Glossary/TCP) | Explanation of TCP and UDP protocols |
| DNS | Internet Society DNS | Detailed overview and workings of DNS |
| DHCP | W3Schools DHCP | Easy explanation and examples |
| IP Addressing | [MDN IP](https://developer.mozilla.org/en-US/docs/Web/Networking/Internet_Protocol) | IP addressing, IPv4 & IPv6 basics |
| SSL/TLS | SSL Labs | SSL/TLS security best practices |
| FTP/SMTP/Email Protocols | [MDN Email Protocols](https://developer.mozilla.org/en-US/docs/Web/Internet/Mail) | Email sending and retrieval protocols |
| ARP/ICMP | RFC 826 ARP, RFC 792 ICMP | Official protocol specs |

**📍 1. Translation: ASCII ↔ EBCDIC (Presentation Layer)**

**🔹 What is ASCII?**

* **ASCII (American Standard Code for Information Interchange)**
* Each English character (like A, 1, @) mapped to a unique 7-bit binary value.  
  Example: 'A' = 65 = 1000001

**🔹 What is EBCDIC?**

* **Extended Binary Coded Decimal Interchange Code**
* Developed by IBM for mainframes.

**🔁 Why Conversion?**

When data is transferred across systems (e.g., browser to server), format may differ. So:

* **Presentation Layer** handles conversion like:  
  ASCII ↔ EBCDIC (platform independent communication)

**📍 2. Lossy vs Lossless Compression**

| **Compression Type** | **Description** | **Example** |
| --- | --- | --- |
| **Lossless** | No data is lost; reversible | ZIP files, PNG |
| **Lossy** | Some data is lost; irreversible | JPEG, MP3 |

**Presentation Layer** may compress data to reduce size:

* Lossless for text
* Lossy for audio/video

**📍 3. SSL/TLS at Presentation Layer**

* SSL = **Secure Sockets Layer**
* TLS = **Transport Layer Security** (modern version)

🔐 Purpose:

* Encrypts sensitive data (e.g., passwords, credit cards)
* Runs at **Presentation Layer**
* Powers **HTTPS**

🌐 Example:

* Browsing a banking website → Data is encrypted before sending

**📍 4. Abstraction Between Layers**

🧠 Concept:

* Each OSI layer trusts the layer below it to do its job.

✅ Example:

* **Session Layer** assumes **Transport Layer** will handle delivery
* Developer doesn't manage every step manually → **abstraction simplifies networking**

**📍 5. Authentication vs Authorization**

| **Term** | **Meaning** | **Example** |
| --- | --- | --- |
| **Authentication** | Verifying identity | Logging in with email/password |
| **Authorization** | Verifying permission | Can this user access admin panel? |

🛂 **Session Layer** handles this as part of session setup.

**📍 6. Sequence Numbers in Transport Layer**

🔹 Problem:

* Data is broken into chunks (segments) → May arrive **out of order**

🔢 Solution:

* Each segment has a **sequence number**
* **Receiver reassembles** segments in correct order

🧪 Example:

text

CopyEdit

Segment 1: Seq #100

Segment 2: Seq #200

Transport Layer uses these to re-order and reassemble the data.

**📍 7. Reserved Port Ranges & Port Theory**

| **Port Range** | **Purpose** | **Example** |
| --- | --- | --- |
| 0–1023 | **Well-known / Reserved** | 80 = HTTP, 443 = HTTPS |
| 1024–49151 | **Registered** | MongoDB, MySQL |
| 49152–65535 | **Dynamic / Private** | Temporary client ports |

🧠 Behind the scenes:

* Port number is a **16-bit unsigned int**
* 2¹⁶ = 65536 total ports

**📍 8. SONET & Frame Relay**

**🔸 SONET – Synchronous Optical Networking**

* High-speed fiber optic network standard
* Used for **long-distance WANs**

**🔸 Frame Relay**

* Protocol for sending data frames across WAN
* Efficient for connecting **LANs to WAN**

**📍 9. Cookies (Application Layer)**

🍪 **Cookies = Small files stored in browser**  
Used for:

* Remembering login sessions
* Shopping cart memory
* Personalization / analytics

🌐 Example:

js

CopyEdit

Set-Cookie: userId=abc123; expires=Wed, 12 Jun 2025

📍 **Part of HTTP** headers (Application Layer)

**📍 10. Sockets**

🔌 A **Socket = IP Address + Port Number**

| **Role** | **Description** |
| --- | --- |
| **IP Address** | Identifies the host |
| **Port** | Identifies the application |

🧠 Example:

* Client socket: 192.168.0.2:5050
* Server socket: 142.250.183.206:443 (Google HTTPS)

**📍 11. Packets**

📦 Internet transfers data in **packets**, not full files.

**A packet contains:**

* Source IP + Port
* Destination IP + Port
* Sequence Number
* Payload (data)
* Checksum

🧪 Example:

YouTube video streamed = multiple packets sent one after another

🔧 Layers involved:

* **Transport Layer**: Breaks into segments
* **Network Layer**: Adds IP addresses
* **Data Link Layer**: Adds MAC address + converts to frames

**📍 12. HTTP Methods**

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| **GET** | Retrieve data | Load homepage |
| **POST** | Submit data | Form submission |
| **PUT** | Update existing data | Edit profile |
| **DELETE** | Remove data | Delete account |

🌐 These are defined by the **HTTP protocol** at Application Layer

**📍 13. HTTP Status Codes**

| **Code** | **Meaning** | **Explanation** |
| --- | --- | --- |
| 200 | OK | Successful request |
| 404 | Not Found | Page doesn’t exist |
| 403 | Forbidden | No permission |
| 500 | Server Error | Internal crash |
| 301 | Moved Permanently | New URL given |

**📍 14. Real-Time DevTools Demo (Inspect)**

📍 Chrome → Right Click → Inspect → Network Tab  
Use it to:

* View GET/POST requests
* Status codes
* Headers, Payloads
* Timings

🧪 Example:

plaintext

CopyEdit

GET https://google.com/

Status: 200 OK

Type: document

Size: 15KB

**📍 15. Live Commands**

| **Command** | **Platform** | **Purpose** |
| --- | --- | --- |
| ipconfig | Windows | Show local IP |
| ifconfig | Linux/macOS | Show interface info |
| ping google.com | All | Test connection |
| curl url | All | Fetch page data |
| tracert / traceroute | Windows / Linux | Trace packet route |

**📍 16. ISP Hierarchy (Deep Dive)**

🗺 How internet flows:

1. **Tier 1 ISP** (e.g., Tata)
2. Provides bandwidth to **Tier 2** (e.g., Airtel, Jio)
3. Then to your **local ISP**
4. Reaches your **home router**

🧠 Submarine cables connect continents. Check: [SubmarineCableMap.com](https://www.submarinecablemap.com/)

**📍 17. Real-World Analogies**

| **Concept** | **Analogy** |
| --- | --- |
| Client-Server | You → Order Pizza → Restaurant |
| Packets | Sending broken parts of a message |
| DNS | Phonebook: google.com = 142.250.183.206 |
| NAT | Receptionist: knows who sent which message |