**PROTOCOLS BASED ON THE OSI MODEL**: it categorized by layers, as per your provided structure:

**1. Application Protocol**

**Layers**: Application, Presentation, Session  
**Role**:

* Provides application-to-application communication.
* Facilitates data exchange, encryption, formatting, and session management.  
  **Examples**:
* **SMTP** (Simple Mail Transfer Protocol): Used for sending emails.
* **SNMP** (Simple Network Management Protocol): Used for network device management.
* **FTP** (File Transfer Protocol): Transfers files between devices.
* **HTTP/HTTPS** (HyperText Transfer Protocol): Used for web communication.
* **DNS** (Domain Name System): Resolves domain names to IP addresses.

**2. Transport Protocol**

**Layer**: Transport  
**Role**:

* Establishes communication sessions between computers.
* Ensures reliable and error-free data transfer.  
  **Examples**:
* **TCP** (Transmission Control Protocol): Ensures reliable, ordered delivery of data.
* **UDP** (User Datagram Protocol): Provides faster but less reliable data transfer.
* **NETBEUI** (NetBIOS Extended User Interface): A lightweight protocol for small networks.

**3. Network Protocol**

**Layers**: Network, Data Link, Physical  
**Role**:

* Manages addressing, routing, error detection, and retransmissions.
* Ensures data packets are correctly delivered between devices.  
  **Examples**:
* **IP** (Internet Protocol): Provides logical addressing and routing.
* **ARP** (Address Resolution Protocol): Resolves IP addresses to MAC addresses.
* **ICMP** (Internet Control Message Protocol): Used for error reporting and diagnostics.
* **Ethernet**: Defines how data is transmitted on LANs.
* **PPP** (Point-to-Point Protocol): Used in direct connections over serial links.

**Key Takeaways:**

1. **Application Protocols** focus on end-user services like email, file transfer, and browsing.
2. **Transport Protocols** ensure that communication is established and data is reliably exchanged.
3. **Network Protocols** manage the actual transmission of data, including addressing, routing, and error handling.

**1. TCP/IP - Transmission Control Protocol / Internet Protocol**

**Layer**:

* **TCP**: Transport Layer (OSI Layer 4)
* **IP**: Network Layer (OSI Layer 3)

**Purpose**:

* **TCP (Transmission Control Protocol)**:
  + Ensures reliable, ordered, and error-checked delivery of data between computers.
  + Manages data segmentation, acknowledgments, and retransmissions.
* **IP (Internet Protocol)**:
  + Responsible for addressing and routing data packets across networks.
  + Defines logical addressing (IP addresses) for devices.

**2. SMTP - Simple Mail Transport Protocol**

**Layer**:

* Application Layer (OSI Layer 7)

**Purpose**:

* Facilitates the routing of electronic mail between mail servers.
* Operates over TCP/IP to ensure reliable delivery of emails.
* Works with protocols like **POP3** or **IMAP** for retrieving email from servers.

**3. FTP - File Transfer Protocol**

**Layer**:

* Application Layer (OSI Layer 7)

**Purpose**:

* Transfers files between computers using the TCP/IP protocol suite.
* Provides secure and reliable file sharing by leveraging TCP for error-free communication.
* Includes features for authentication and data encryption (in modern implementations).

**4. SNMP - Simple Network Management Protocol**

**Layer**:

* Application Layer (OSI Layer 7)

**Purpose**:

* Enables monitoring and management of devices on a network (e.g., routers, switches).
* Operates over TCP/IP for communication between network management systems and devices.
* Collects and organizes information about managed devices using **Management Information Bases (MIBs)**.

**Summary Table**

| **Protocol** | **Layer** | **Purpose** |
| --- | --- | --- |
| **TCP** | Transport (OSI Layer 4) | Provides reliable, error-checked communication between computers. |
| **IP** | Network (OSI Layer 3) | Handles addressing and routing of data packets between devices. |
| **SMTP** | Application (OSI Layer 7) | Routes email between servers over TCP/IP. |
| **FTP** | Application (OSI Layer 7) | Transfers files between computers over TCP/IP. |
| **SNMP** | Application (OSI Layer 7) | Manages and monitors network communication and devices. |

These protocols form the backbone of modern networking and enable essential services like email, file sharing, device management, and reliable data transmission.

Here’s a detailed list of more widely used protocols categorized by their layers in the **OSI model**, along with their purposes:

**Application Layer Protocols (Layer 7)**

* **HTTP (HyperText Transfer Protocol)**: Facilitates web communication, enabling browsers to retrieve web pages from servers.
* **HTTPS (HTTP Secure)**: A secure version of HTTP that uses encryption (TLS/SSL) for secure communication.
* **DNS (Domain Name System)**: Resolves human-readable domain names (e.g., www.example.com) into IP addresses.
* **POP3 (Post Office Protocol v3)**: Retrieves emails from a mail server; messages are downloaded and deleted from the server.
* **IMAP (Internet Message Access Protocol)**: Retrieves emails from a server while allowing them to remain on the server for access from multiple devices.
* **Telnet**: Provides a text-based interface for remote login to another device over a network.
* **SSH (Secure Shell)**: Provides encrypted remote login and secure file transfer.
* **LDAP (Lightweight Directory Access Protocol)**: Accesses and maintains distributed directory information (e.g., user authentication).
* **NTP (Network Time Protocol)**: Synchronizes clocks across devices in a network.
* **DHCP (Dynamic Host Configuration Protocol)**: Dynamically assigns IP addresses and configuration settings to devices in a network.

**Transport Layer Protocols (Layer 4)**

* **UDP (User Datagram Protocol)**: Provides fast, connectionless communication without guarantees of reliability.
  + Used in real-time applications like video streaming, VoIP, and gaming.
* **SCTP (Stream Control Transmission Protocol)**: Combines features of TCP and UDP; supports multi-streaming and multi-homing.
* **DCCP (Datagram Congestion Control Protocol)**: Aims to provide reliable transport with congestion control, often for multimedia applications.

**Network Layer Protocols (Layer 3)**

* **ICMP (Internet Control Message Protocol)**: Used for error messages, diagnostics, and network troubleshooting (e.g., ping, traceroute).
* **IGMP (Internet Group Management Protocol)**: Manages group memberships for multicast traffic in IP networks.
* **BGP (Border Gateway Protocol)**: Manages routing between autonomous systems on the internet.
* **OSPF (Open Shortest Path First)**: A link-state routing protocol for determining the best path in IP networks.
* **RIP (Routing Information Protocol)**: A distance-vector routing protocol used in smaller networks.
* **EIGRP (Enhanced Interior Gateway Routing Protocol)**: A Cisco proprietary protocol for advanced routing.

**Data Link Layer Protocols (Layer 2)**

* **Ethernet (IEEE 802.3)**: Defines how devices on a LAN communicate over a wired connection.
* **Wi-Fi (IEEE 802.11)**: Defines wireless communication standards for LANs.
* **PPP (Point-to-Point Protocol)**: Provides a direct connection between two nodes, often used in dial-up internet connections.
* **ARP (Address Resolution Protocol)**: Resolves IP addresses to MAC addresses in a local network.
* **RARP (Reverse Address Resolution Protocol)**: Resolves MAC addresses to IP addresses.

**Physical Layer Protocols (Layer 1)**

* **Bluetooth**: Short-range wireless communication standard for personal devices.
* **USB (Universal Serial Bus)**: Defines physical interfaces for data transfer between devices.
* **DSL (Digital Subscriber Line)**: Provides high-speed internet over telephone lines.
* **RS-232**: A standard for serial communication used in older devices.

**Other Notable Protocols Across Layers**

* **VoIP (Voice over IP)**: Enables voice communication over IP networks (e.g., SIP, RTP).
* **TLS/SSL (Transport Layer Security / Secure Sockets Layer)**: Provides encryption for secure communication, often used with HTTPS.
* **MPLS (Multiprotocol Label Switching)**: Accelerates data flow by routing packets based on labels instead of IP addresses.
* **GRE (Generic Routing Encapsulation)**: Encapsulates packets for tunneling over networks.
* **IPsec (Internet Protocol Security)**: Secures IP communication by encrypting and authenticating packets.

**Summary Table**

| **Protocol** | **OSI Layer** | **Purpose** |
| --- | --- | --- |
| **HTTP/HTTPS** | Application | Web communication (secure with HTTPS). |
| **FTP** | Application | File transfer. |
| **SMTP/IMAP/POP3** | Application | Email communication (sending, retrieving). |
| **DNS** | Application | Resolves domain names to IP addresses. |
| **TCP/UDP** | Transport | Reliable (TCP) or fast (UDP) data transmission. |
| **IP** | Network | Routing and addressing packets. |
| **ICMP** | Network | Diagnostics and error messaging (e.g., ping). |
| **Ethernet** | Data Link | LAN communication standard. |
| **Wi-Fi** | Data Link | Wireless LAN communication. |
| **Bluetooth** | Physical | Short-range wireless communication. |

**IP ADDRESS OVERVIEW:**

An **IP address** (Internet Protocol address) is a unique identifier assigned to devices on a network to facilitate communication. It is essential for routing data packets between devices over the internet or other networks.

**Key Features**

1. **32-bit Address (IPv4)**:
   * Divided into four octets (bytes), separated by dots (e.g., 192.168.1.1).
   * Each octet ranges from 0 to 255 (8 bits per octet).
2. **Components**:
   * **Network ID**: Identifies the network the device belongs to.
   * **Host ID**: Identifies the specific device within the network.
3. **Categorization**:
   * IP addresses are divided into **5 classes**: A, B, C, D, and E, based on their **first octet**.

**IP Address Classes**

| **Class** | **Range of First Octet** | **Purpose** | **Network and Host Bits** |
| --- | --- | --- | --- |
| **Class A** | 0 to 127 | Large networks (e.g., ISPs, large companies). | Network: 8 bits, Host: 24 bits |
| **Class B** | 128 to 191 | Medium-sized networks (e.g., universities). | Network: 16 bits, Host: 16 bits |
| **Class C** | 192 to 223 | Small networks (e.g., private businesses). | Network: 24 bits, Host: 8 bits |
| **Class D** | 224 to 239 | Reserved for multicast communications. | Not divided into network/host. |
| **Class E** | 240 to 255 | Experimental, reserved for future use. | Not divided into network/host. |

**Detailed Explanation of Classes**

1. **Class A**:
   * **First Octet Range**: 0.0.0.0 to 127.255.255.255.
   * **Network ID**: First 8 bits.
   * **Host ID**: Remaining 24 bits.
   * **Example**: 10.0.0.1.
2. **Class B**:
   * **First Octet Range**: 128.0.0.0 to 191.255.255.255.
   * **Network ID**: First 16 bits.
   * **Host ID**: Remaining 16 bits.
   * **Example**: 172.16.0.1.
3. **Class C**:
   * **First Octet Range**: 192.0.0.0 to 223.255.255.255.
   * **Network ID**: First 24 bits.
   * **Host ID**: Last 8 bits.
   * **Example**: 192.168.1.1.
4. **Class D**:
   * **First Octet Range**: 224.0.0.0 to 239.255.255.255.
   * Used for **multicast** communication (one-to-many).
   * Does not follow the network/host division.
5. **Class E**:
   * **First Octet Range**: 240.0.0.0 to 255.255.255.255.
   * Reserved for **experimental purposes** and future use.

**Special IP Address Ranges**

* **Private IP Addresses** (Not routable over the internet):
  + Class A: 10.0.0.0 to 10.255.255.255.
  + Class B: 172.16.0.0 to 172.31.255.255.
  + Class C: 192.168.0.0 to 192.168.255.255.
* **Loopback Address**: 127.0.0.1 is reserved for testing and diagnostics within the host device.

**Summary**

* **IP Addresses** uniquely identify devices on a network.
* They are structured into classes to manage network sizes and purposes.
* Classes **A, B, and C** are used for host addressing, while **D and E** are reserved for specialized purposes.
* Private and special addresses cater to non-internet and testing environments

THE **HARVARD ARCHITECTURE** AND **VON NEUMANN ARCHITECTURE** :

These two fundamental designs for computer systems that describe how the CPU interacts with memory. Both architectures have distinct ways of handling instructions and data, and each has its strengths and weaknesses.

**1. von Neumann Architecture**

**Overview:**

* Proposed by **John von Neumann** in 1945.
* Also known as the **Princeton Architecture**.
* Most modern computers are based on this architecture.

**Key Characteristics:**

1. **Single Memory for Instructions and Data**:
   * Instructions (program code) and data are stored in the same memory.
2. **Single Data Bus**:
   * A single bus is used for transferring both data and instructions.
   * This can create a bottleneck called the **von Neumann bottleneck**.
3. **Sequential Execution**:
   * Instructions are fetched, decoded, and executed sequentially.
4. **Simpler Design**:
   * The design is simpler and cost-effective compared to the Harvard Architecture.

**Advantages:**

* Simplifies memory design since both code and data share the same memory.
* Easier to implement and program.
* Suitable for general-purpose computing tasks.

**Disadvantages:**

* The **von Neumann bottleneck** can slow down processing because only one operation (instruction or data access) can occur at a time.
* Susceptible to accidental overwriting of program instructions since they share the same memory.

**2. Harvard Architecture**

**Overview:**

* Named after the **Harvard Mark I**, an early computer that used this design.
* Primarily used in **embedded systems**, **microcontrollers**, and **digital signal processors (DSPs)**.

**Key Characteristics:**

1. **Separate Memories for Instructions and Data**:
   * Instructions and data are stored in different memory units.
2. **Separate Data Buses**:
   * Independent buses for transferring instructions and data allow simultaneous access.
3. **Parallel Processing**:
   * Instructions can be fetched while data is being accessed, improving performance.
4. **More Complex Design**:
   * Requires additional hardware for separate memory units and buses.

**Advantages:**

* Eliminates the von Neumann bottleneck since instructions and data are accessed separately.
* Improved performance, especially in systems requiring fast and frequent data access.
* More secure, as program instructions and data are isolated from one another.

**Disadvantages:**

* More complex and expensive to implement due to separate memory units and buses.
* Difficult to modify programs dynamically (since instructions are stored in separate memory).

**Comparison Table**

| **Feature** | **von Neumann Architecture** | **Harvard Architecture** |
| --- | --- | --- |
| **Memory** | Single memory for data and instructions | Separate memories for data and instructions |
| **Data Bus** | Single bus for data and instructions | Separate buses for data and instructions |
| **Performance** | Slower due to the von Neumann bottleneck | Faster due to simultaneous access to data and instructions |
| **Complexity** | Simpler and cost-effective | More complex and costly |
| **Usage** | General-purpose computers (PCs, laptops) | Embedded systems, DSPs, microcontrollers |
| **Parallelism** | Fetches instructions and data sequentially | Can fetch instructions and data simultaneously |

**Examples of Usage:**

1. **von Neumann Architecture**:
   * Desktop computers, laptops, servers.
   * Most general-purpose operating systems and software run on this architecture.
2. **Harvard Architecture**:
   * Microcontrollers like **AVR** (used in Arduino), **PIC**, and **ARM Cortex-M**.
   * Digital Signal Processors (DSPs) for audio and video processing.

**SDLC - SOFTWARE DEVELOPMENT LIFE CYCLE: 7 PHASES**

**1. Requirement Gathering and Analysis**

* **Stakeholders Involved**: End users, IT specialists (System Analyst).
* **Primary Activities**:
  1. **Gathering Business Requirements**: Collecting detailed information about user needs and expectations.
  2. **Prioritizing Requirements**: Identifying and ranking the most critical functionalities for the system.
* **Documents Produced**:
  1. **SRS (Software Requirement Specification)**: Comprehensive document detailing the functional and non-functional requirements.
  2. **FRS (Functional Requirement Specification)**: Focused on specific functionalities the system must provide.
  3. **NFRS (Non-Functional Requirement Specification)**: Defines performance, security, scalability, and other system constraints.

**2. Planning Phase**

* **Stakeholders Involved**:
  + Business Analyst
  + Customer

**Primary Activities:**

1. **Define the System to Be Developed**:
   * Identify the system that aligns with the business goals.
2. **Set the Project Scope**:
   * **Project Scope**: Clearly outlines the boundaries of what the system will and will not do.
   * **Scope Creep**: Uncontrolled changes or continuous growth in the project scope.
   * **Feature Creep**: Adding excessive features or functionalities beyond the original scope.
   * **Project Scope Document**: Official documentation capturing the agreed-upon scope.
3. **Develop the Project Plan**:
   * Break down the project into tasks, allocate resources, and set timeframes.
   * **Project Plan**: A roadmap created by the **Project Manager**, detailing milestones, deliverables, and deadlines.
   * **Project Milestones**: Key checkpoints to measure progress and ensure timely delivery.

**Documents Produced:**

* **BRS (Business Requirement Specification)**: Captures high-level business needs and goals.
* **CRS (Customer Requirement Specification)**: Focuses on specific customer expectations.
* **URS (User Requirement Specification)**: Details user-level requirements for the system.

Access memory

Fetch Instruction

Update Register file

ALU Operation

Update PC

Decode Instruction