Transport Layer in Computer Networks

Key Viva Exam Questions to Prepare

Transport Layer

1. Why is the Transport Layer called the End-to-End layer? (Core conceptual understanding)
2. Compare and contrast TCP and UDP with suitable use cases. (Protocol knowledge and application)
3. What is a socket and what components are needed to create one? (Implementation concept)
4. How does the Transport Layer provide multiplexing/demultiplexing? (Fundamental function)
5. Explain Transport Layer flow control mechanisms. (Performance management)

Data Link Layer

1. What are the primary functions of the Data Link Layer? (Core concept)
2. Compare Go-Back-N and Selective Repeat protocols. (Protocol knowledge)
3. Explain how framing works and why it's necessary. (Fundamental concept)
4. How does the sliding window protocol improve efficiency? (Performance concept)
5. Describe the differences between HDLC and PPP protocols and their applications. (Protocol implementation knowledge)

These questions cover the key concepts that demonstrate both theoretical understanding and practical knowledge of these network layers.

1. Why Transport Layer is Called End-to-End Layer

The Transport Layer is called the End-to-End layer because it:

- Provides direct communication between end applications on source and destination hosts
- Establishes logical connections between endpoints, not intermediate devices
- Handles end-to-end error control and flow management
- Operates between the two endpoints regardless of the physical network path
- Acts as the first layer to address communication between applications rather than between machines

2. Job of Transport Layer

The Transport Layer's key responsibilities include:

- Segmenting data received from application layer

- Providing end-to-end communication services
- Establishing, maintaining, and terminating virtual circuits
- Flow control to prevent overwhelming receivers
- Error detection and recovery
- Multiplexing/demultiplexing data to/from multiple applications
- Congestion control to prevent network overload
- Ensuring reliable data delivery (for TCP)

3. Ports and Their Characteristics

A port is a logical endpoint identifier used to distinguish different services or processes:

- **Logical**, not physical (software construct, not hardware)
- 16-bit number (0-65535) with different ranges:
    - Well-known ports: 0-1023 (reserved for standard services)
    - Registered ports: 1024-49151 (registered with IANA)
    - Dynamic/private ports: 49152-65535 (for temporary connections)
- Maximum ports: 65,536 (2^16) per IP address

4. TCP vs. UDP

TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Connection-oriented	Connectionless
Reliable delivery with acknowledgments	Unreliable (best-effort) delivery
Ordered packet delivery	No guaranteed packet order
Flow control via sliding window	No flow control
Congestion control	No congestion control
Error detection and recovery	Basic error detection, no recovery
Higher overhead	Lower overhead
Example uses: Web (HTTP), Email, File Transfer	Example uses: Streaming, DNS, VoIP

Which is better? Neither is universally "better" - the choice depends on application requirements:

- TCP for applications requiring reliability and ordered delivery
- UDP for applications prioritizing speed and low overhead

5. Socket

A socket is a software endpoint for communication:

- Combination of IP address and port number
- Provides API for network communication
- Acts as interface between application and transport layers
- Enables inter-process communication across networks
- Implemented by operating systems for program use

6. Creating a Socket

To create a socket, you need:

- IP address (identifies the host machine)
- Port number (identifies the specific service/application)
- Protocol (typically TCP or UDP)
- Socket library/API functions (e.g., socket() in C)

In programming terms, socket creation typically involves:

1. Specifying address family (IPv4/IPv6)
2. Socket type (stream for TCP, datagram for UDP)
3. Protocol (TCP/UDP)

7. Socket Full Duplex Capability

Yes, sockets created between client and server are typically full-duplex:

- Data can flow in both directions simultaneously
- Both client and server can send and receive data independently
- Each endpoint maintains separate send and receive buffers
- This enables interactive communication without waiting for responses
- Particularly true for TCP sockets; UDP sockets are inherently bidirectional

8. Transport Layer Protocols

Key Transport Layer protocols include:

- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)
- SCTP (Stream Control Transmission Protocol)
- DCCP (Datagram Congestion Control Protocol)
- SPX (Sequenced Packet Exchange) - Novell NetWare
- ATP (AppleTalk Transaction Protocol)
- QUIC (Quick UDP Internet Connections)

Data Link Layer in Computer Networks

1. Job of Data Link Layer

The Data Link Layer's primary responsibilities include:

- Framing of data packets from network layer
- Physical addressing (MAC addressing)
- Error detection and correction
- Media access control (determining when to transmit)
- Flow control between adjacent nodes

- Managing connections between directly connected devices
- Ensuring reliable transit across physical links

2. Framing and Its Implementation

Framing is the process of dividing data streams into manageable units (frames):

How framing is done:

- **Character count**: Indicates number of characters in frame
- **Flag bytes with byte stuffing**: Special delimiter characters mark frame boundaries; byte stuffing used to escape flag bytes in data
- **Flag bits with bit stuffing**: Similar to flag bytes but works at bit level
- **Physical layer coding violations**: Uses special signaling patterns not used in regular data

3. Flow Control and Its Implementation

Flow control prevents a sender from overwhelming a receiver:

Flow control methods:

- **Stop-and-Wait**: Sender waits for acknowledgment before sending next frame
- **Sliding Window**: Allows multiple frames to be in transit before acknowledgment
- **Rate-based**: Sender adjusts transmission rate based on receiver feedback
- **XON/XOFF**: Receiver sends special characters to pause/resume transmission

4. Sliding Window Protocol

Sliding Window Protocol allows multiple frames to be in transit simultaneously:

- Sender maintains a "window" of sequential frames it can send
- Receiver has a window of frames it can accept
- Window "slides" as frames are acknowledged
- Improves efficiency by allowing transmission while waiting for ACKs
- Window size determines maximum unacknowledged frames
- Balances throughput with receiver's processing capabilities

5. Stop-and-Wait Protocol

Stop-and-Wait is the simplest flow control mechanism:

- Sender transmits a single frame
- Sender waits for acknowledgment
- Only after receiving ACK does sender transmit next frame
- Uses timeout to handle lost frames or ACKs
- Simple but inefficient, especially over high-latency links
- Throughput limited by round-trip time

6. Go-Back-N Protocol

Go-Back-N is a sliding window protocol variant:

- Allows multiple unacknowledged frames (N)
- Receiver acknowledges only in-order frames
- If frame k is lost, receiver discards all subsequent frames
- After timeout, sender retransmits frame k and all following frames
- More efficient than Stop-and-Wait but may waste bandwidth on retransmissions
- Suitable for environments with low error rates

7. Selective Repeat Protocol

Selective Repeat improves on Go-Back-N:

- Allows multiple unacknowledged frames
- Receiver accepts and buffers out-of-order frames
- Receiver sends selective acknowledgments for each frame
- Sender retransmits only specific lost frames
- More complex buffering requirements at receiver
- More efficient use of bandwidth than Go-Back-N
- Better for high-error-rate environments

8. HDLC Protocol

HDLC (High-level Data Link Control) is a bit-oriented protocol:

- ISO standard for point-to-point and multipoint communications
- Uses bit stuffing for transparency
- Provides three operational modes:
    - Normal Response Mode (NRM): Primary controls communication
    - Asynchronous Response Mode (ARM): Secondary can transmit without permission
    - Asynchronous Balanced Mode (ABM): Peer-to-peer communication
- Frame types: Information (I), Supervisory (S), Unnumbered (U)
- Error and flow control through sequence numbers and windowing

9. PPP Protocol

PPP (Point-to-Point Protocol) is designed for direct connections:

- Derived from HDLC but character-oriented
- Three main components:
    - Link Control Protocol (LCP): Establishes, configures, and tests connection
    - Authentication protocols (PAP, CHAP): Verify user identity
    - Network Control Protocols (NCPs): Configure different network layer protocols
- Frame format: Flag, Address, Control, Protocol, Data, FCS, Flag

- Supports multiple network layer protocols simultaneously
- Provides error detection (not correction)

10. PPP Protocol Applications

PPP is used in various scenarios:

- Dial-up internet connections
- DSL connections (PPPoE - PPP over Ethernet)
- Direct serial connections between routers
- ISDN connections
- Some cellular data services
- VPN tunneling protocols (as basis for L2TP)
- Anywhere a simple, reliable point-to-point connection is needed