🚀 1) Introduction to Transport Layer

- It is **Layer 4 of OSI model (Layer 3 in TCP/IP)**.
- Provides **end-to-end communication between applications** on different computers.
- Ensures data is **delivered accurately, in order, and without errors.**
- Uses **port numbers** to deliver data to the correct application (e.g. web uses port 80).
- **Example protocols: TCP (reliable), UDP (unreliable)**

🧩 2) Multiplexing and Demultiplexing

- **Multiplexing:** Combining messages from different applications into segments for sending.
- **Demultiplexing:** Delivering received segments to correct applications.
- Uses **IP addresses + port numbers** to identify correct destination.
- **UDP:** Uses only destination port for demux.
- **TCP:** Uses 4-tuple → source IP, source port, destination IP, destination port.

📦 3) Connectionless Transport – UDP

- UDP = User Datagram Protocol
- **No connection setup** between sender and receiver.
- Adds **source/destination port numbers** and passes to network layer.
- **Unreliable & unordered:** Packets may get lost, duplicated, or arrive out of order.
- **Used in:** DNS, SNMP, online gaming, live video/audio streaming.
- **Header fields:** Source port, Destination port, Length, Checksum
- **Checksum:** Used to detect errors in the segment.

✅ 4) Principles of Reliable Data Transfer (RDT)

- Reliable protocols are built over unreliable channels.
- Use **ACK (Acknowledgement), NAK (Negative ACK)**, **sequence numbers**, and **timers** to ensure reliability.
- **rdt 1.0:** Assumes perfect channel (no errors).
- **rdt 2.0:** Handles bit errors using checksum + ACK/NAK (stop-and-wait).
- **rdt 2.1:** Adds sequence numbers (0,1) to detect duplicate packets.
- **rdt 2.2:** Uses only ACKs, no NAK. Duplicate ACK acts like NAK.

- **rdt 3.0:** Handles errors + packet loss using **timeout + retransmission.**

🧬 5) Pipelined Protocols

- Allows **multiple unacknowledged packets in transit at the same time**.
- Improves efficiency by using network bandwidth fully.
- Two types:

a) Go-Back-N (GBN)

- Sender sends up to **N unACKed packets**.
- If one packet is lost, **all after it are retransmitted**.
- Receiver only accepts packets in order and sends **cumulative ACK.**
- Uses **one timer for oldest unACKed packet.**

b) Selective Repeat (SR)

- Sender sends up to **N unACKed packets**.
- Receiver **accepts out-of-order packets** and buffers them.
- Sender **retransmits only lost packets.**
- Uses **separate timer for each packet.**
- More efficient than GBN.

📡 6) Connection-Oriented Transport – TCP

- TCP = Transmission Control Protocol.
- **Reliable, connection-oriented, byte-stream protocol.**
- Breaks data into segments and ensures **ordered, error-free delivery.**
- **Example:** Like courier service with tracking number.
- **Features:**
    - Reliable delivery
    - Ordered data transfer
    - Error checking (checksum)
    - Retransmission of lost data
- **3-Way Handshake:**

1. SYN – client says "I want to connect"
2. SYN+ACK – server replies "I’m ready"

3. ACK – client confirms "Let's start"

- **TCP Segment:** includes source port, dest port, sequence number, ack number, flags (SYN, ACK, FIN), checksum, window size.

⚖️ 7) Flow Control

- Prevents **fast sender from overwhelming slow receiver.**
- Receiver advertises a **window size (rwnd)** showing how much data it can handle.
- Sender sends only within this window until it gets ACKs.
- Avoids receiver buffer overflow.

🚦 8) Congestion Control

- Prevents **too much data entering the network** which causes packet loss/delay.
- Causes:
  - Too many senders sending fast
  - Full router buffers
  - Retransmissions increasing load
- Two approaches:
  - **End-to-End (TCP):** Detect congestion via packet loss/delay.
  - **Network-assisted:** Routers give feedback to senders (rare).
- **TCP Slow Start:**
  - Starts with small congestion window.
  - Gradually increases window size until packet loss occurs.
  - Helps find network's capacity safely.

🖥️ 9) RPC (Remote Procedure Call)

- Allows a program to **call a function on another computer as if it is local.**
- Programmer just writes result = add(5,7) even if add() is on remote system.
- Uses transport layer to:
  - Send function call request as a message.
  - Wait for result and return it.
- Uses **TCP for reliable RPC** (e.g. gRPC) or **UDP for fast RPC** (e.g. ONC RPC).

- RPC itself is **not a protocol**, it **uses transport protocols.**