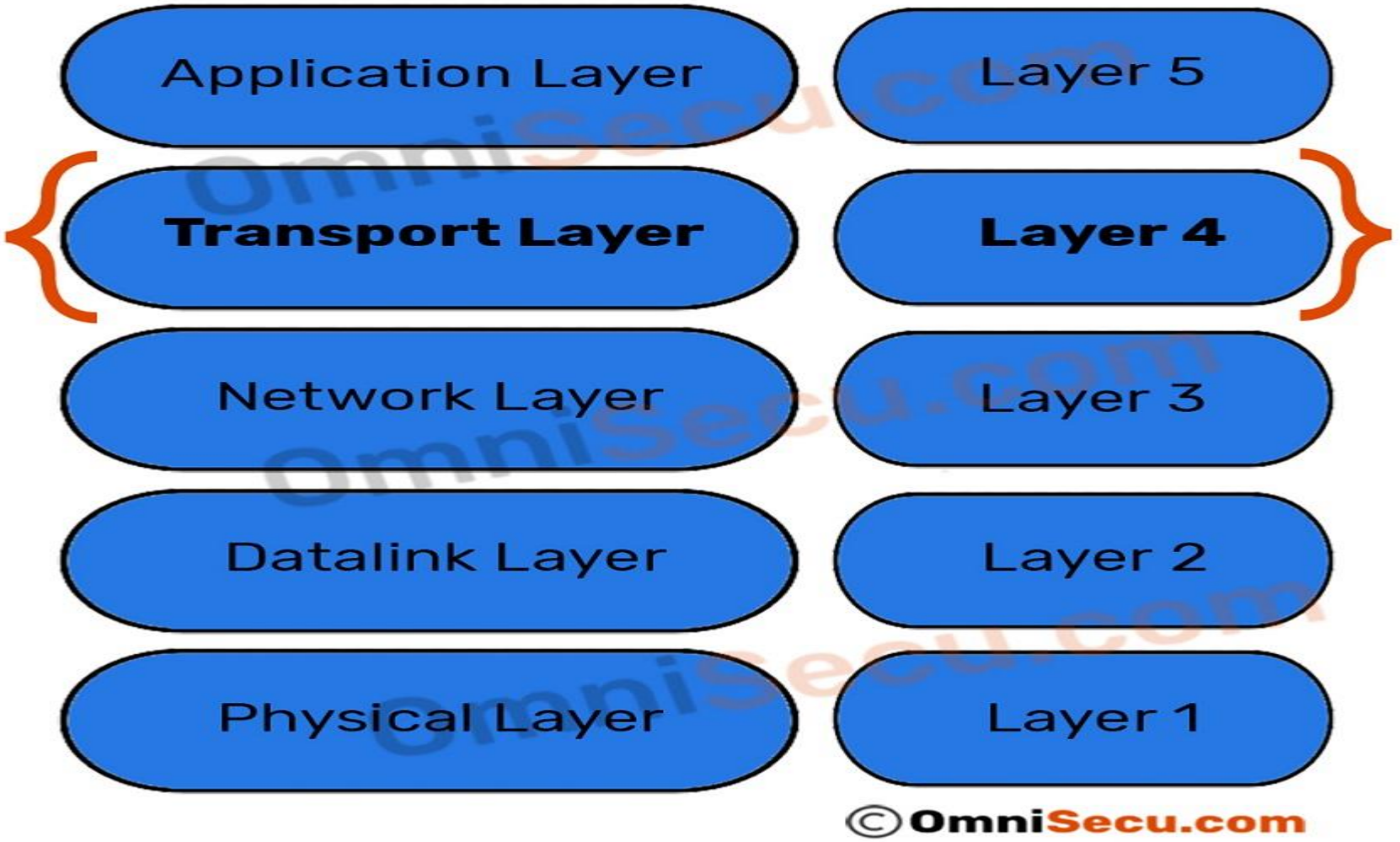
# TCP and UDP Protocols

Prof SRN Reddy, IGDTUW

# Functions of Transport layer

- To provide an interface for network applications to access the network
- To provide an  interface for accepting data from different applications on the source computer and sending that data to the recipient applications on different destination computers (**Multiplexing)**.
- Similarly on the destination computer, the incoming data from different remote computers need to be directed to the correct applications for that data was meant for (**De-multiplexing**).
- It treats each packet independently for final delivery, because each packet for final delivery belongs to different Applications on different destination computers.
- It has protocol/mechanisms for loss-free delivery of data to destination applications.
- Transport layer provides mechanisms for error checking, flow control, and re-transmission of lost data.
- Two major protocols at the Transport layer,
  - **Transmission Control Protocol (TCP)**
  - **User Datagram Protocol (UDP)**

# TCP Layered Architecture



# Transmission Control Protocol (TCP)

- **It is a connection oriented protocol, which means the devices should open a connection before transmitting data and should close the connection gracefully after transmitting the data.**
- **It assure reliable delivery of data to the destination.**
- **It provides extensive error checking mechanisms such as flow control and acknowledgment of data.**
- **Sequencing of data**
- **Delivery of data is guaranteed**
- **It is comparatively slow because of these extensive error checking mechanisms**
- **Multiplexing and Demultiplexing is supporting**
- **Retransmission of lost packets is supported**

# User Datagram Protocol (UDP)

- It is a  Datagram oriented protocol with no overhead for opening a connection (using three-way handshake), maintaining a connection, and closing (terminating) a connection.
- It is efficient for broadcast/multicast type of network transmission.
- It has only the basic error checking mechanism using checksums.
- There is no sequencing of data in User Datagram Protocol (UDP).
- The delivery of data cannot be guaranteed in User Datagram Protocol (UDP).
- User Datagram Protocol (UDP) is faster, simpler and more efficient than TCP. It
- It is less robust then TCP
- Multiplexing and Demultiplexing is possible in User Datagram Protocol (UDP) using UDP port numbers.

# TCP connection.

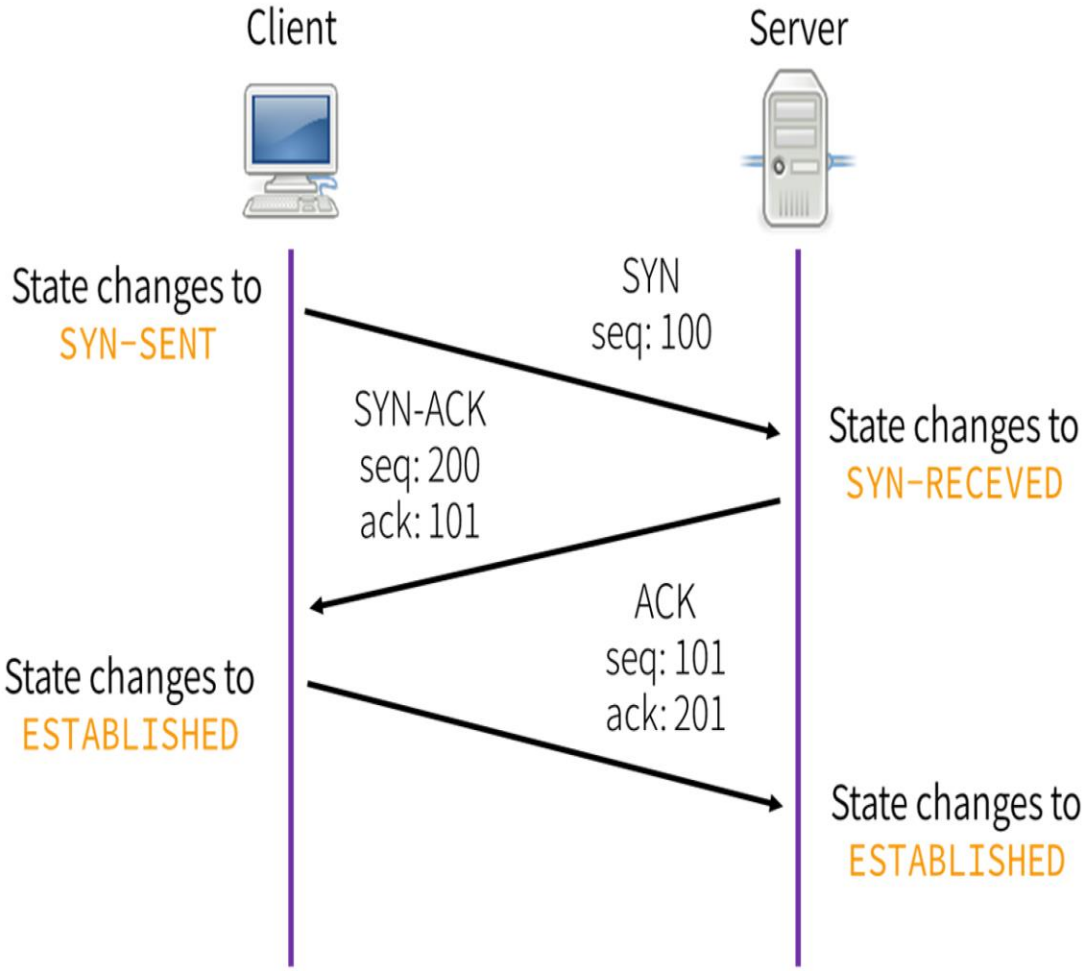
- A connection oriented protocol establishes connection between virtual path between source and destination. All segment belonging to the message are sent over this virtual path.
- TCP connection are virtual , not physical. It operates at higher level.
- It use the services of IP to deliver individual segment to the receiver, but it controls the connection itself.
- If a segment is lost or corrupted, it retransmits.
- If a segment arrives out of order, TCP holds it until the missing segments arrive
- Connection-oriented transmission requires three phases:
  - Connection establishment,
  - data transfer,
  - connection termination.

# Connection Establishment

- TCP transmits data in full duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.
- This implies that each party must initialize communication and get approval from the other party before any data are transferred.
- This connection establishment in TCP is called three way hand shaking

# 3 way Handshaking

- The client sends a SYN (synchronize) packet to the server, which has a random sequence number.
- The server sends back a SYN-ACK packet, containing a random sequence number and an ACK number acknowledging the client's sequence number.
- The client sends an ACK number to the server, acknowledging the server's sequence number.
- The sequence numbers on both ends are synchronized. Both ends can now send and receive data independently



# Data transfer

- After connection establishment the data transfer in TCP takes place.
- The data transfer takes place in bidirectional.
- Where the client and server can both send data and acknowledgments .
- The acknowledgments are piggybacked with data
- It uses buffer to store the stream of data coming from the sending application.

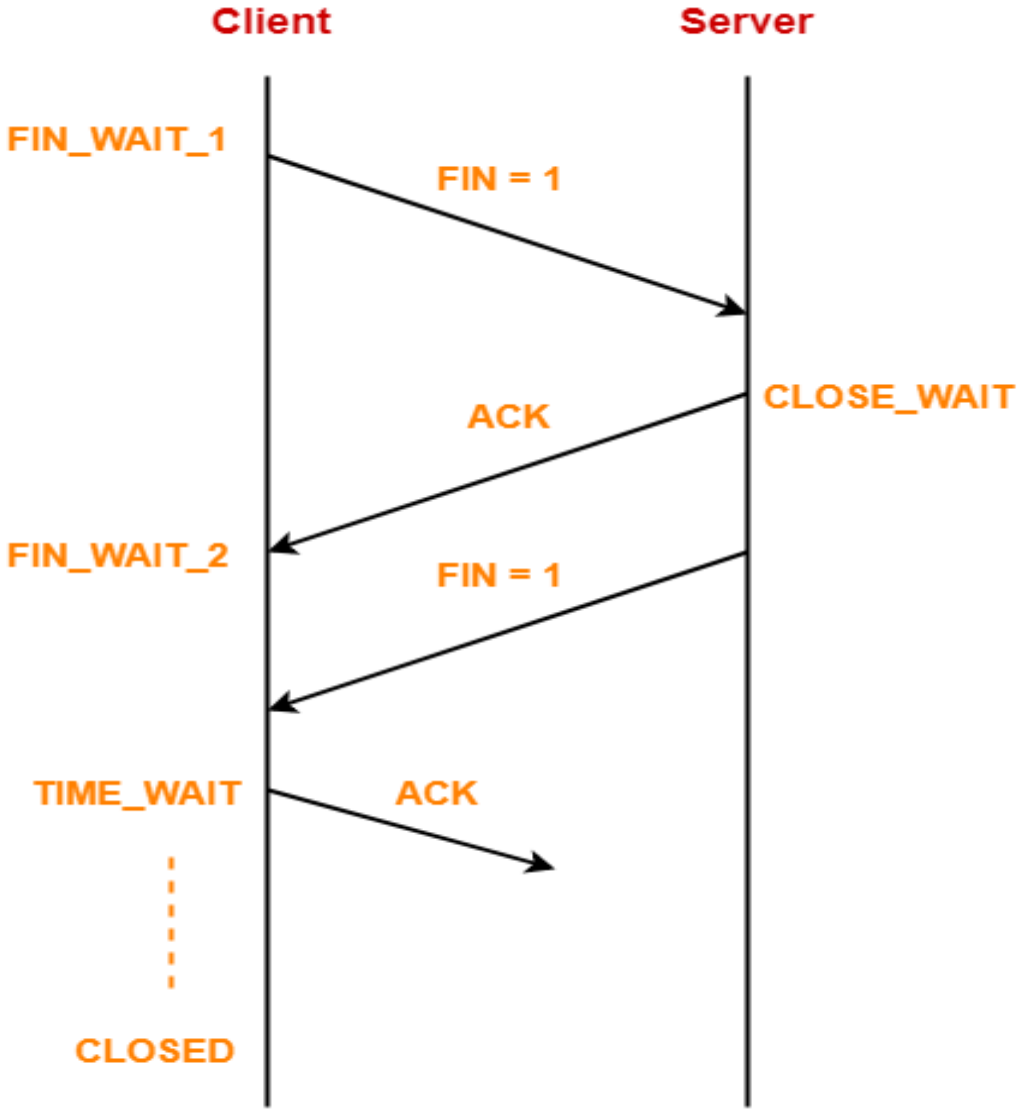
# Connection termination.

- Any of the two parties involved in exchanging data can close the connection, although it is usually initiated by the client.

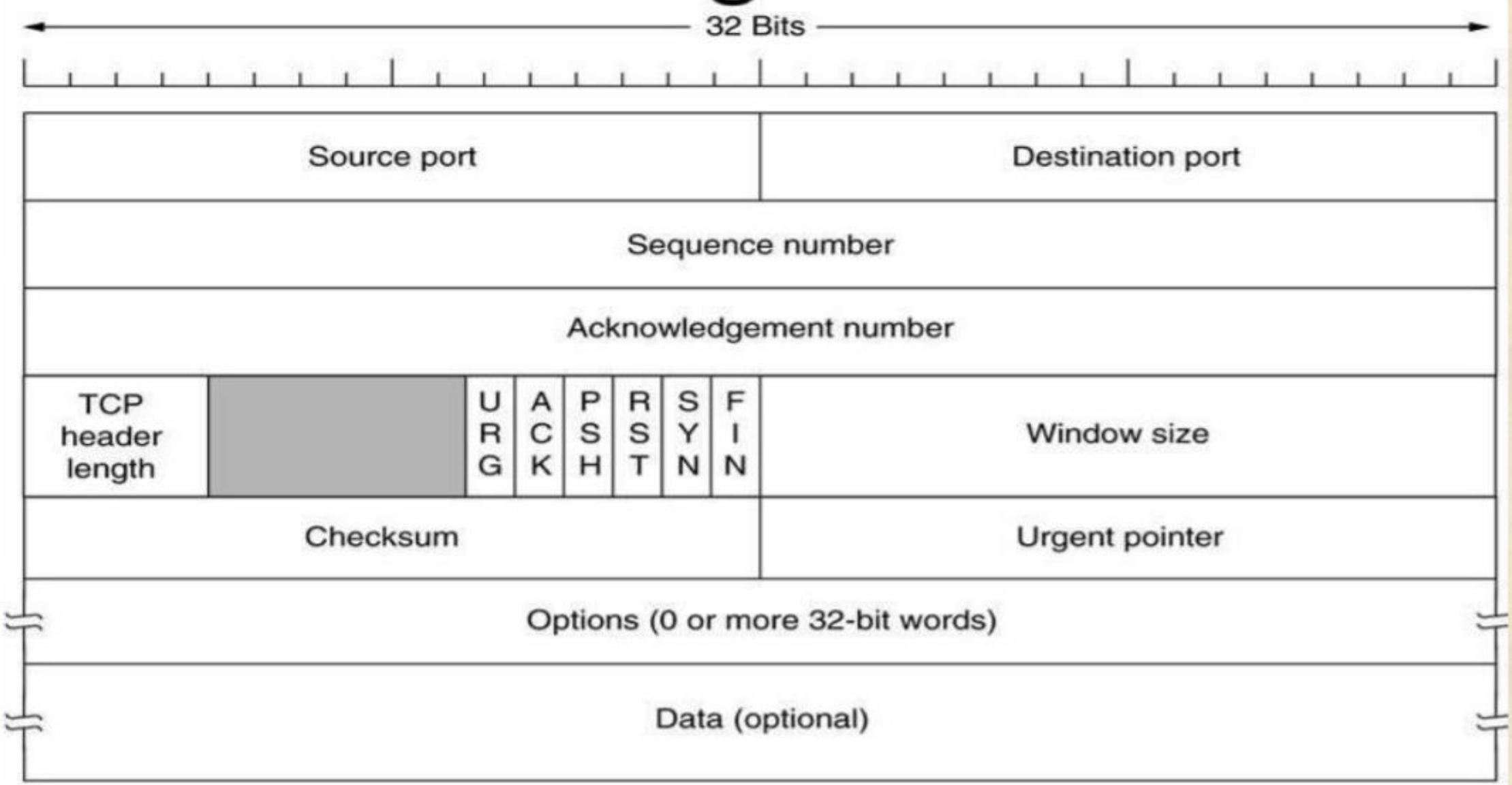
- Most of the implementation today allow two options for connection termination they are:

  1.way handshaking

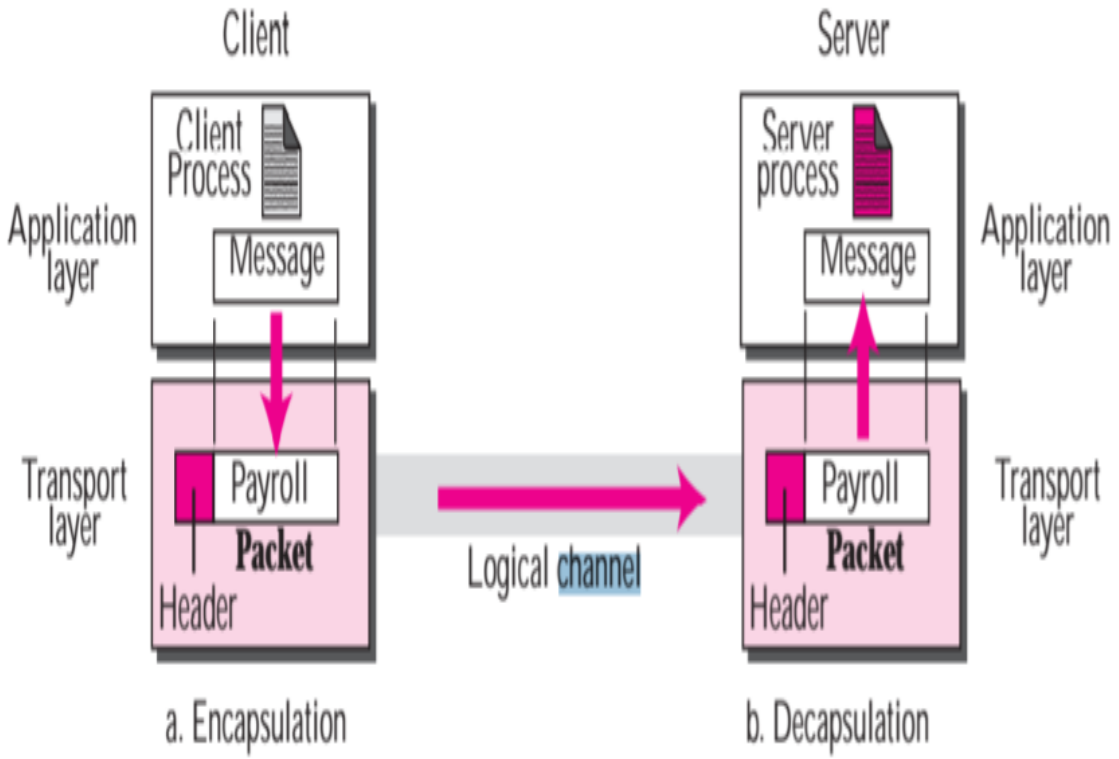
  2. Four way handshaking with a half -close option.



# The TCP Segment Header



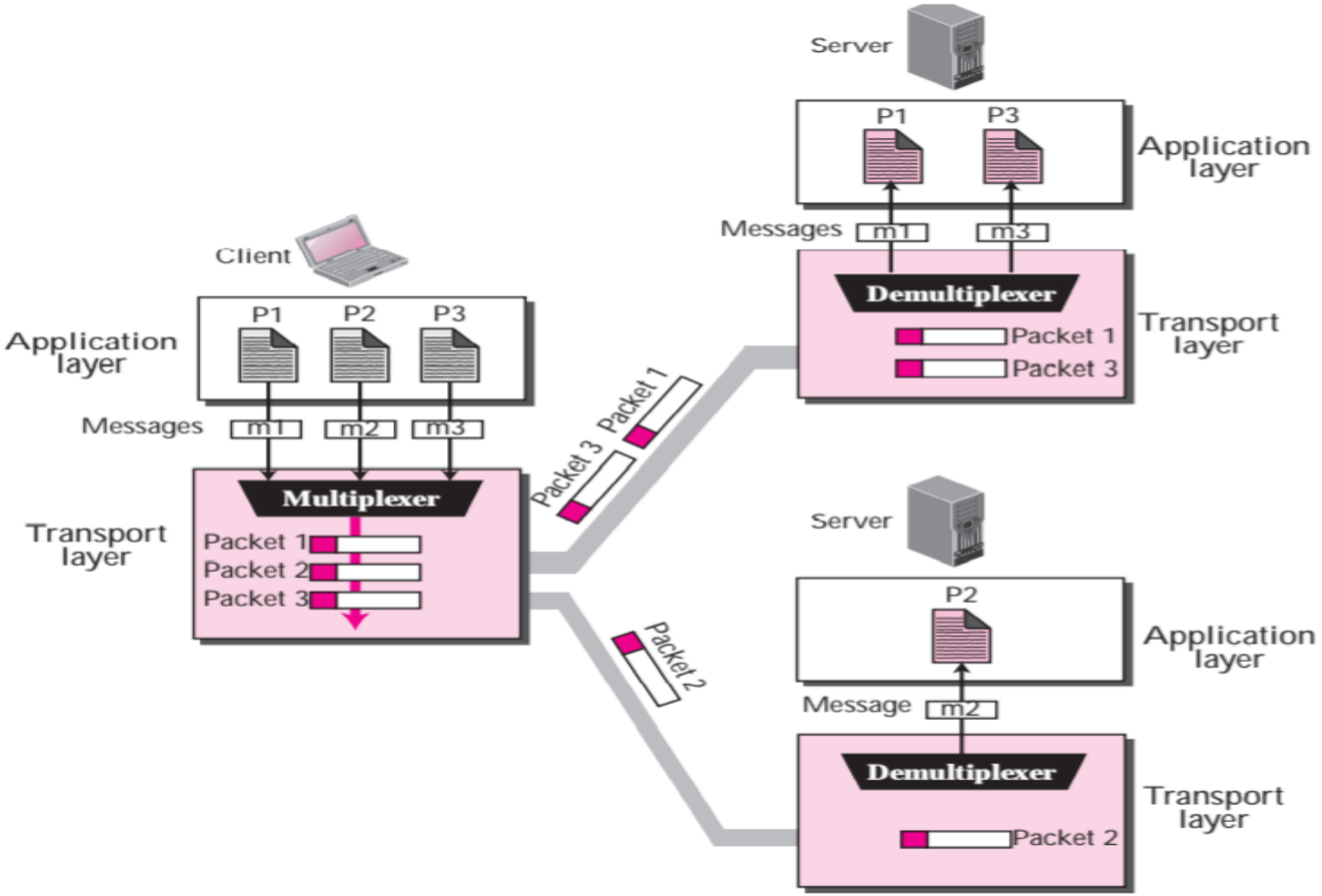
# Encapsulation and Decapsulation




**Encapsulation: Encapsulate certain data**
**Happen at Source node**
**Adds the header to the payload/ message and transmit the packet**

**Decapsulation: opening up encapsulated data back**
**Done at destination**
**Removes the header from the packet and creates the original message back**

# Multiplexing and Demultiplexing

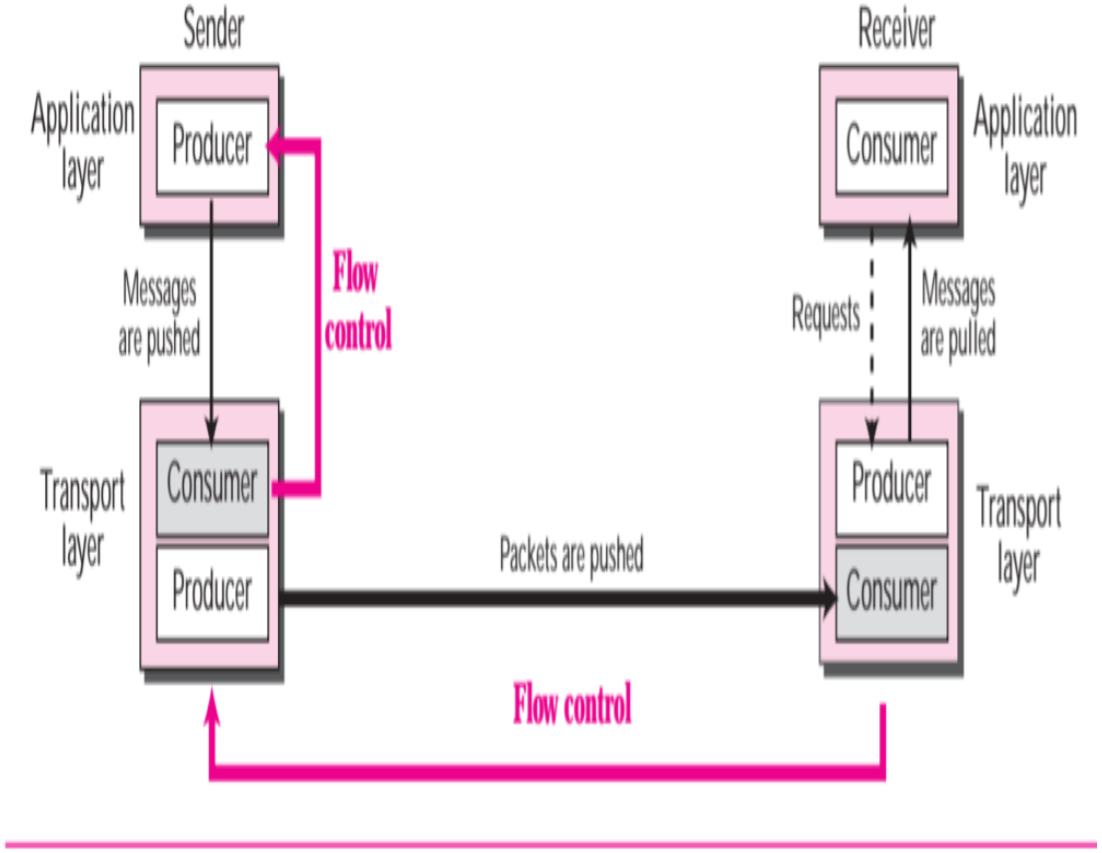



Multiplexing: Many to one
Happens at Source Node

De-Multiplexing: one to Many
Happens at Destination Node

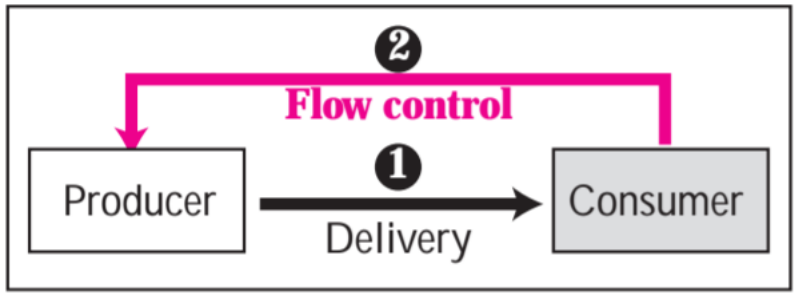
# Flow Control

- Used to maintain the balance between production and consumption rates.
- If the items are produced faster than they can be consumed, the consumer can be overwhelmed and needs to discard some items.
- If the items are produced slower than they can be consumed, the consumer should wait; the system becomes less efficient.
- Flow control is used to prevent losing the data items at the consumer site

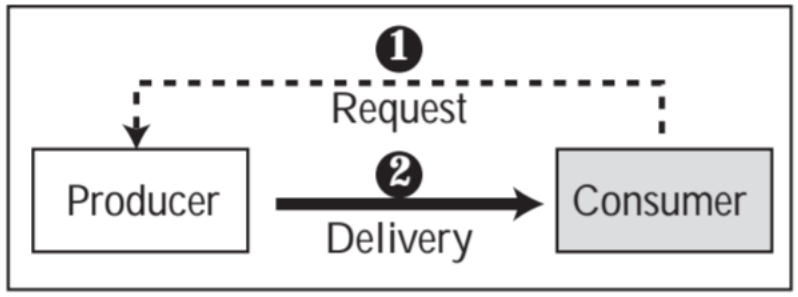


# Pushing or Pulling

- Delivery of items from a producer to a consumer can occur in one of the two ways: pushing or pulling.
- Pushing: If the sender delivers items whenever they are produced without the prior request from the consumer
- pulling : If the producer delivers the items after the consumer has requested them



a. Pushing



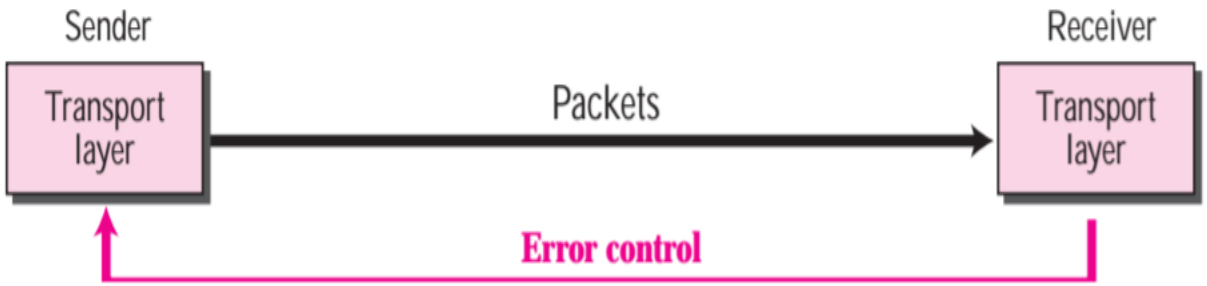
b. Pulling

# TCP Error Control Methods

- **Checksum**
- **Acknowledgment**
- **Time out**

# Error control and its Functions

- Detect and discard corrupted packets.
- Keep track of lost and discarded packets and resend them.
- Recognize duplicate packets and discard them.
- Buffer out-of-order packets until the missing packets arrive.



# Sequence Numbers

- **Error control requires that the sending transport layer knows which packet is to be resent**
- **Receiving transport layer knows which packet is a duplicate, or which packet has arrived out of order.**
- **This can be done if the packets are numbered.**
- **Sequence Number field is added to packet by the transport layer**
- **When a packet is corrupted or lost, the receiving transport layer can inform the sending transport layer to resend that packet using the sequence number.**
- **The receiving transport layer can also detect duplicate packets if two received packets have the same sequence number.**
- **The out-of-order packets can be recognized by observing gaps in the sequence numbers.**

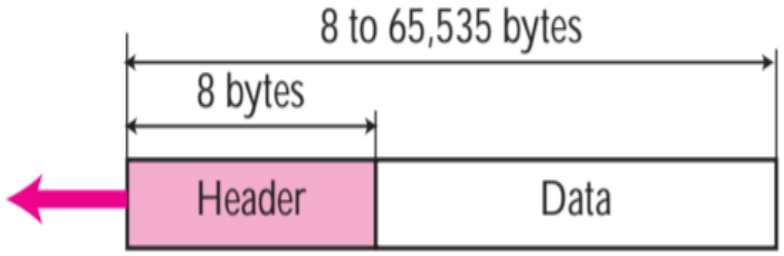
# Acknowledgment

- The receiver side can send an acknowledgement (ACK) for each or a collection of packets that have arrived safe
- The receiver can simply discard the corrupted packets.
- The sender can detect lost packets if it uses a timer.
- When a packet is sent, the sender starts a timer; when the timer expires, if an ACK does not arrive before the timer expires, the sender resends the packet.
- Duplicate packets can be silently discarded by the receiver. Out-of- order packets can be either discarded or stored until the missing ones arrives.

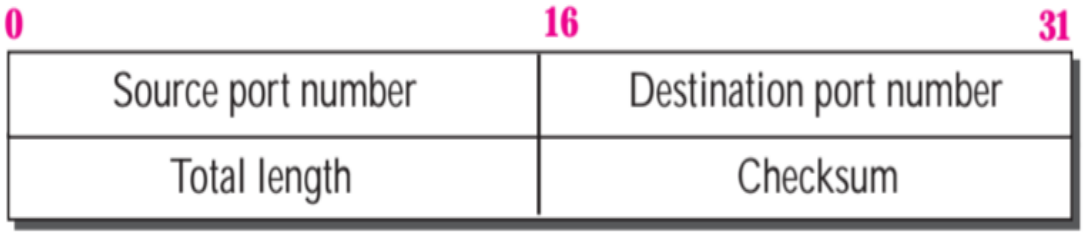
# UDP

- **UDP is a transport protocol that creates a process-to-process communication.**
- **UDP is a (mostly) unreliable and connectionless protocol that requires little overhead and offers fast delivery.**
- **The UDP packet is called a user datagram.**
- **UDP's only attempt at error control is the checksum.**
- **UDP has no flow-control mechanism.**
- **A user datagram is encapsulated in the data field of an IP datagram.**
- **UDP uses multiplexing to handle outgoing user datagrams from multiple processes on one host.**
- **UDP uses demultiplexing to handle incoming user datagrams that go to different processes on the same host.**

# UDP Header




a. UDP user datagram

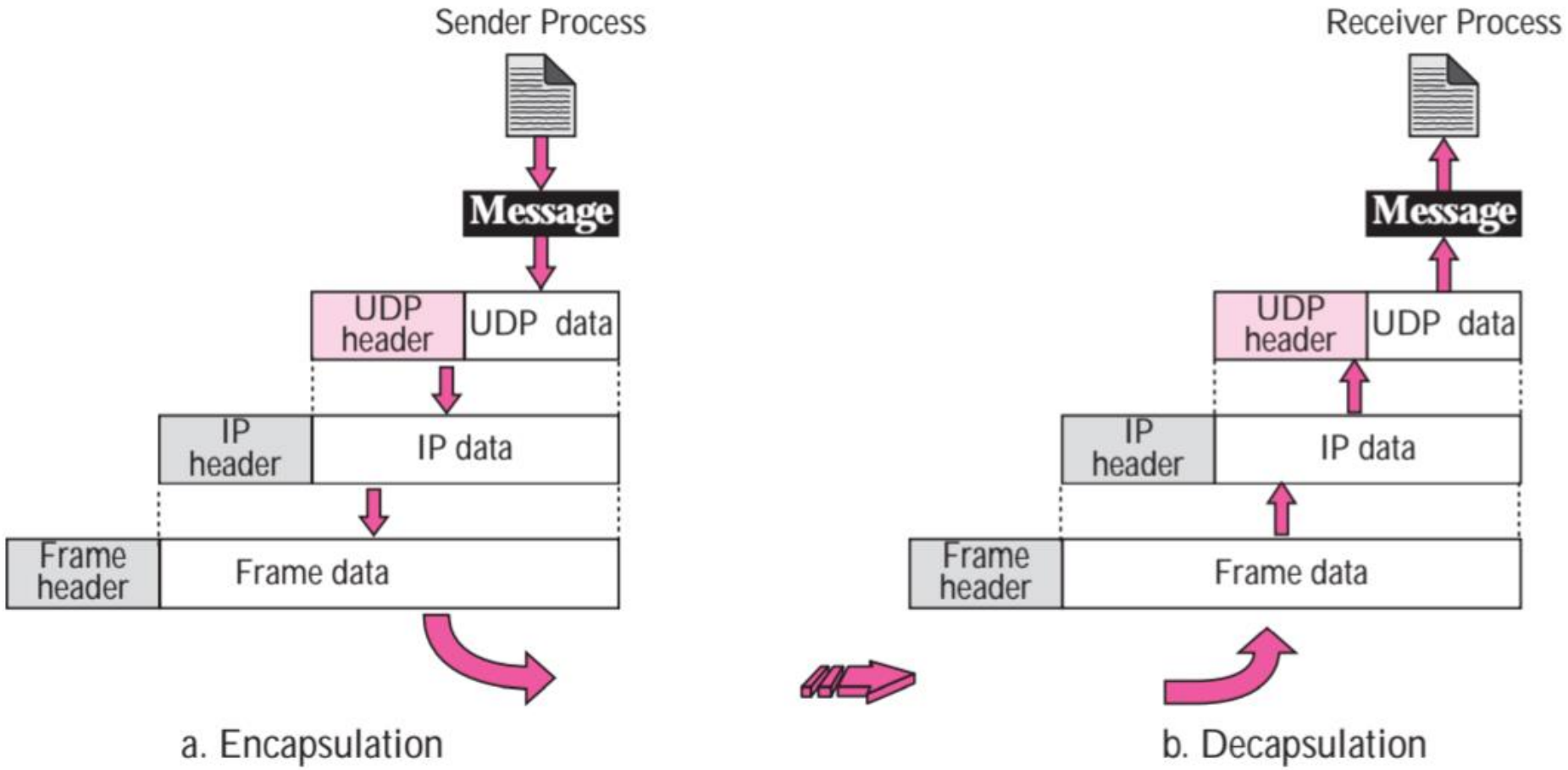



b. Header format

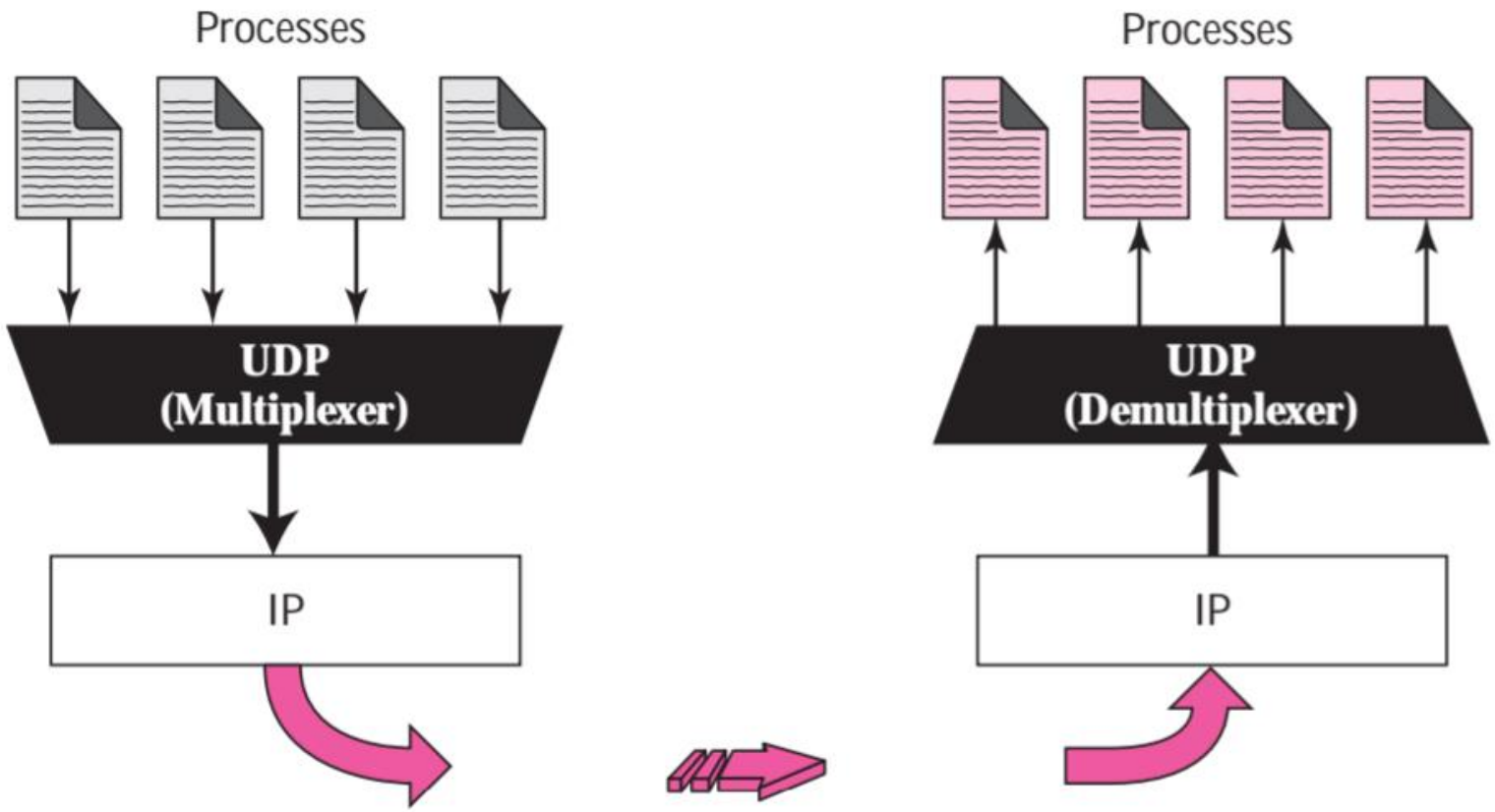
**Source port number:** This is the port number used by the process running on the source host. It is 16 bits long.

**Destination port number:** This is the port number used by the process running on the destination host. It is also 16 bits long.

# Encapsulation and Decapsulation



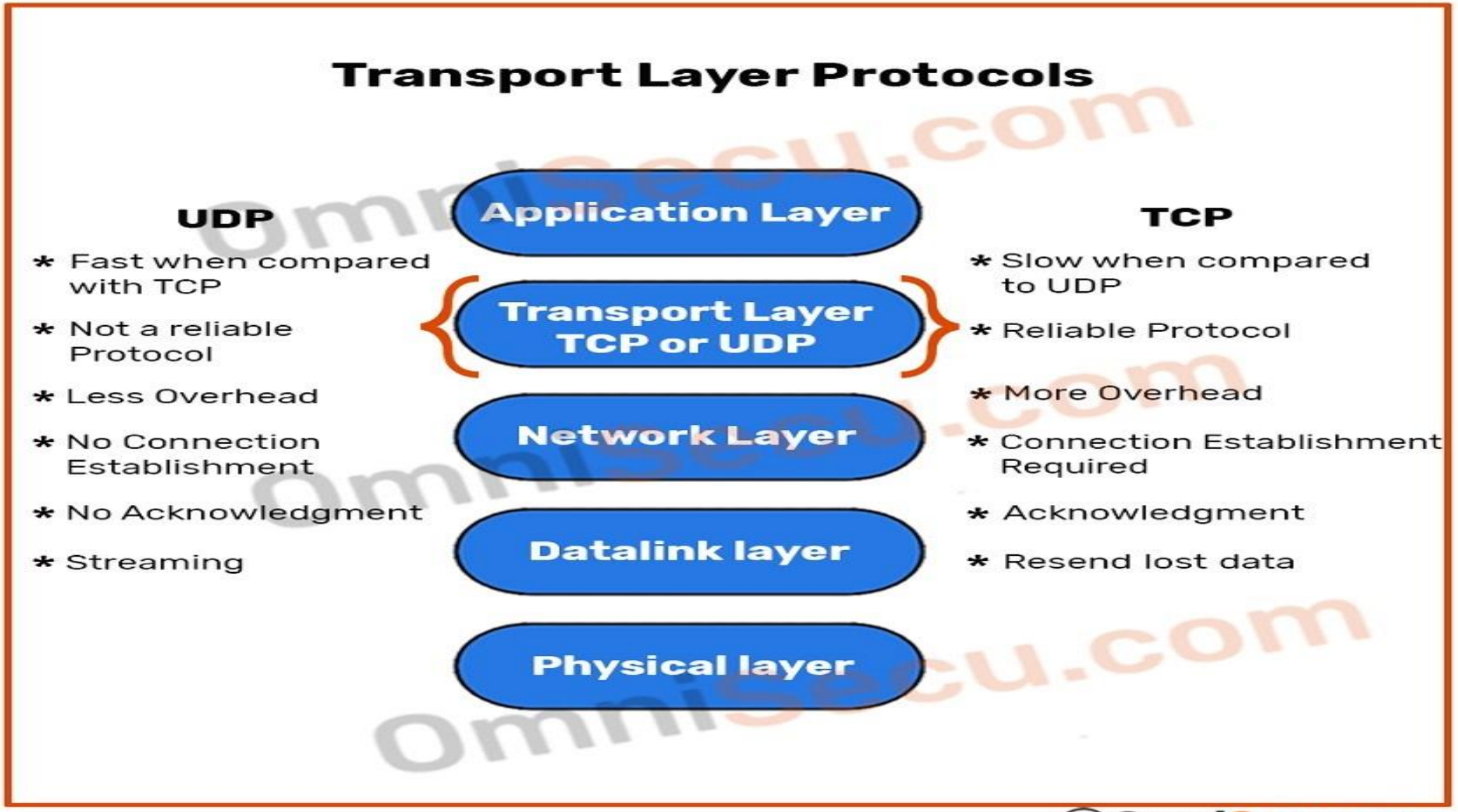
# Multiplexing and demultiplexing



# Differences are-

| Properties | TCP | UDP |
|---|---|---|
| Header | Dynamic header ( 20 – 60 B) | Static header of 8 Bytes |
| Max segment | any size or 2^30 B | short message 65536 Bytes |
| Flow Control | Yes, Window and seq. no. | NO |
| Checksum | Compulsory | Optional |
| Connection nature | TCP+ IP = connection oriented | UDP+ IP= connection less |
| Error control | Own mechanism | Depends on ICMP<br>(No self feature) |
| Support multicast | NO | YES |
| Support broadcast | NO | Yes |
| Examples service | HTTP,SMTP,FTP,TELNET | TFTP,DNS,SNMP |

# TCP vs UDP


Transport Layer Protocols
UDP
* Fast when compared with TCP
* Not a reliable Protocol
* Less Overhead
* No Connection Establishment
* No Acknowledgment
* Streaming
Application Layer
Transport Layer TCP or UDP
Network Layer
Datalink layer
Physical layer
TCP
* Slow when compared to UDP
* Reliable Protocol
* More Overhead
* Connection Establishment Required
* Acknowledgment
* Resend lost data
©OmniSecu.com

# References


- https://www.omnisecu.com/tcpip/transmission-control-protocol-tcp.php
- file:///C:/Users/S%20R%20N%20REDDY/Downloads/tcp-transmissioncontrolprotocol-copy-150311124141-conversion-gate01.pdf
- B.A. Forouzan, "TCP/ IP Protocol Suit", 4th Edition [ Online Edition]

Q&A

Thanks