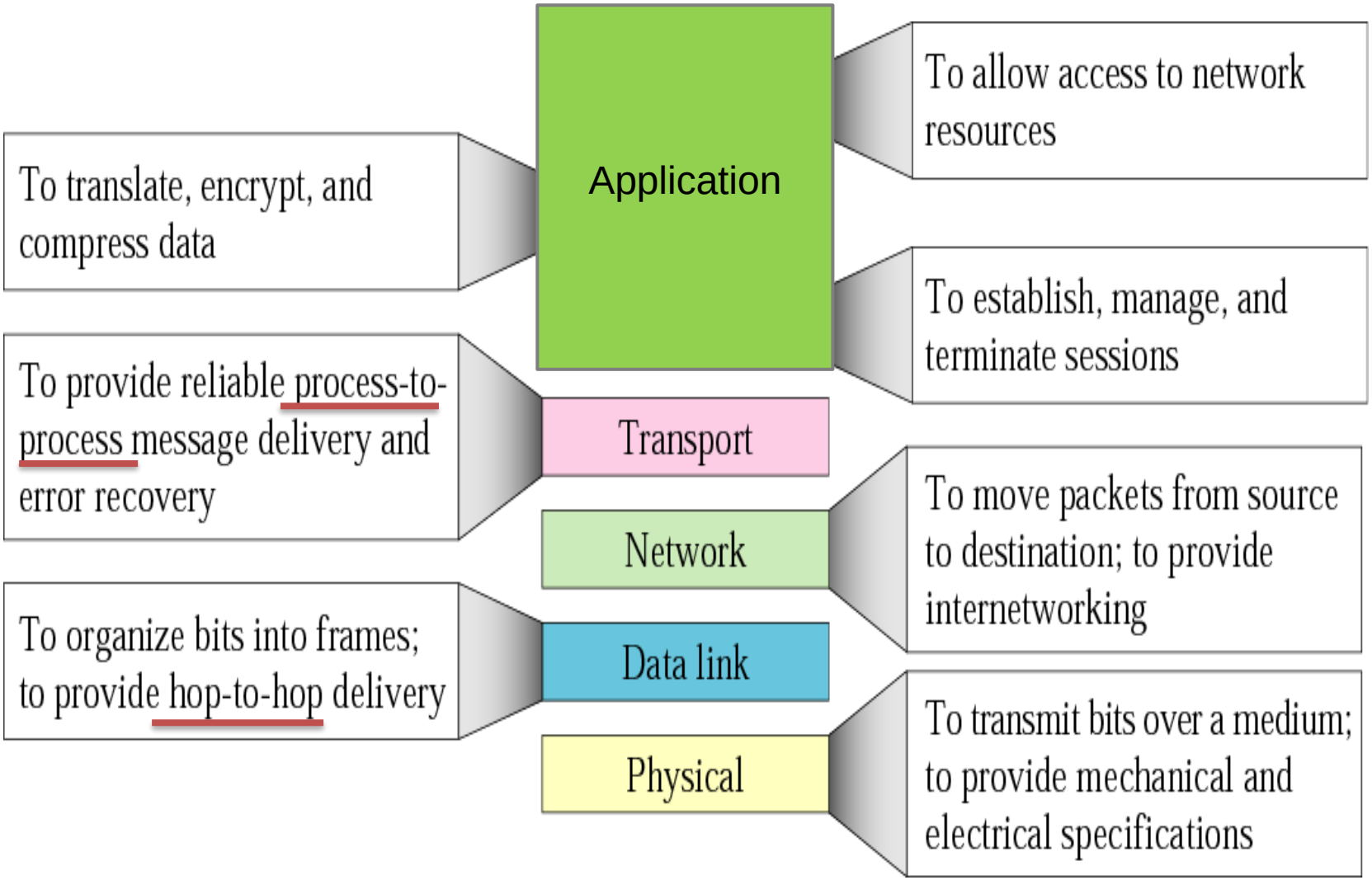
The Transport Layer

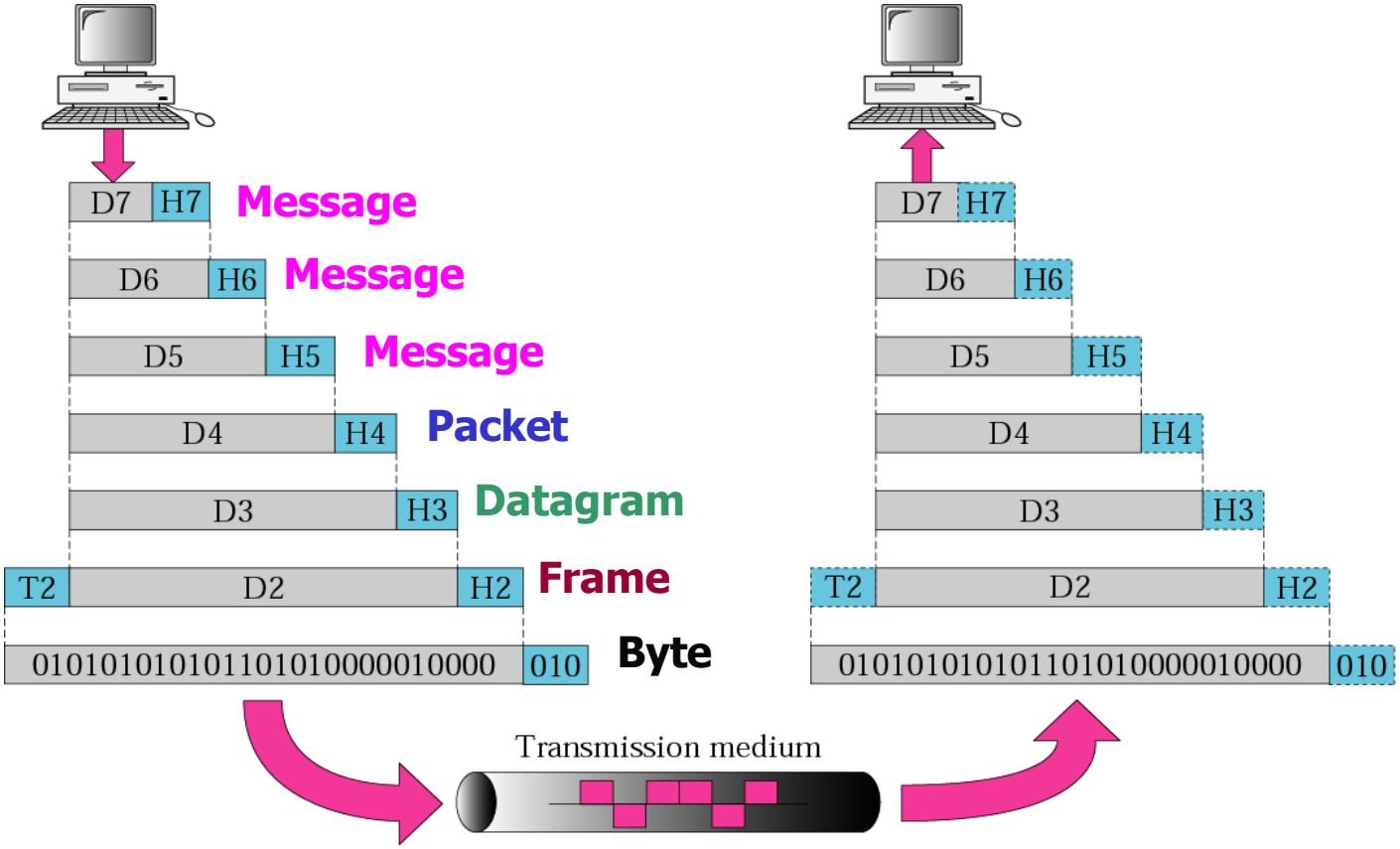
Content:

- Transport Layer Services
- Elements Of Transport Layer
- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)

Summary of Network layers



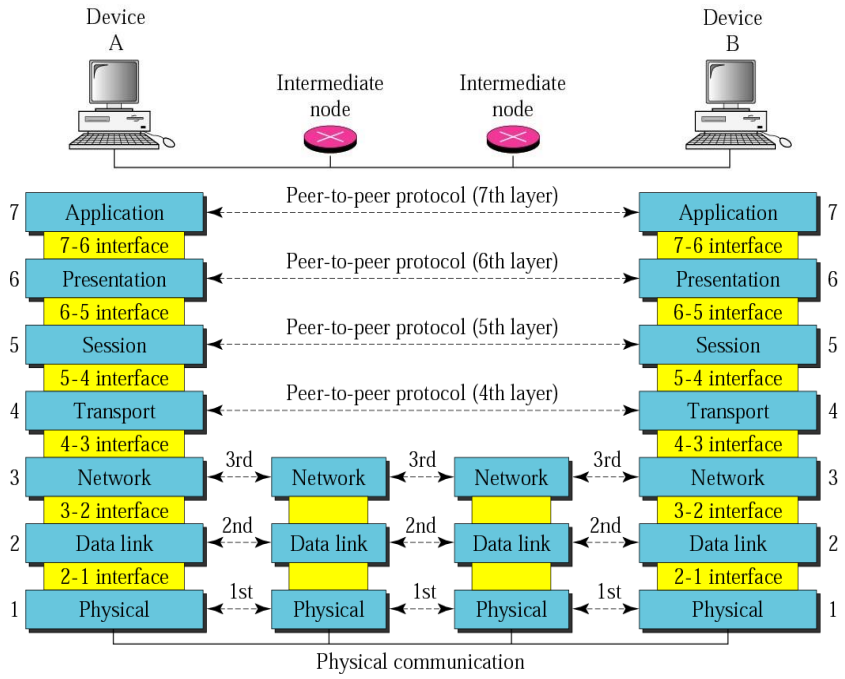
TCP/IP protocol suit

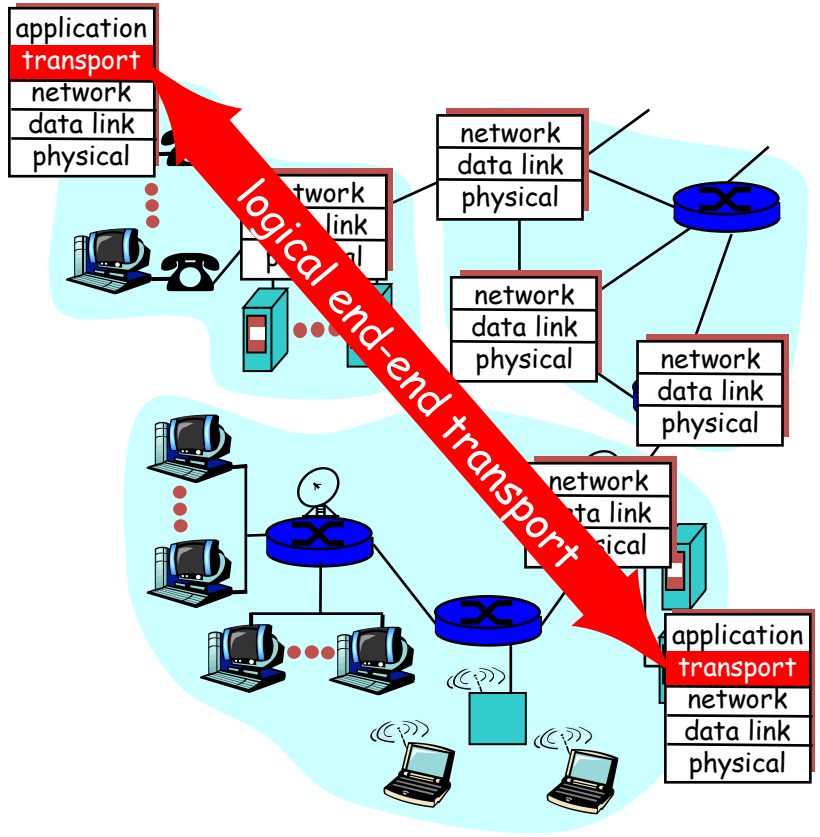


Logical end to end delivery

- The transport layer provides a logical communication between application processes running on different hosts.

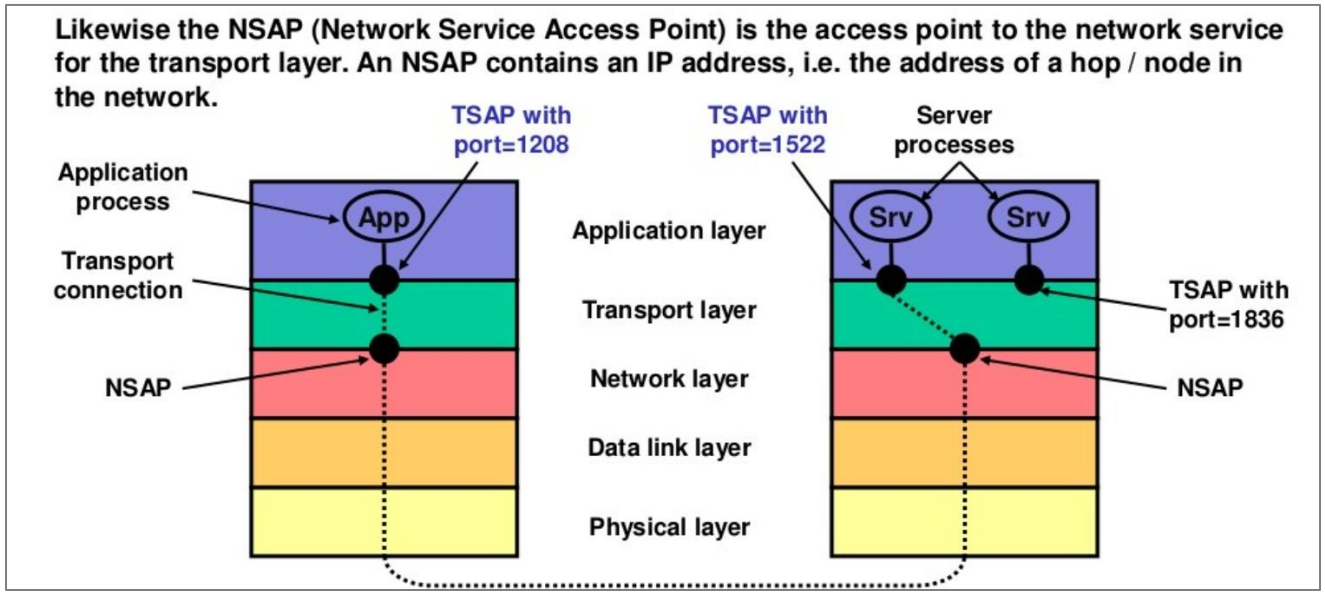
- The transport layer protocols are implemented in the end systems but not in the network routers.





Logical end to end delivery

- The *transport layer* delivers *messages* between ***Transport Service Access Points*** (TSAPs or *ports*) in different computers. Several processes running on a computer may be exchanging messages with processes running on other computers. The TSAPs appended to the messages differentiate those information streams.

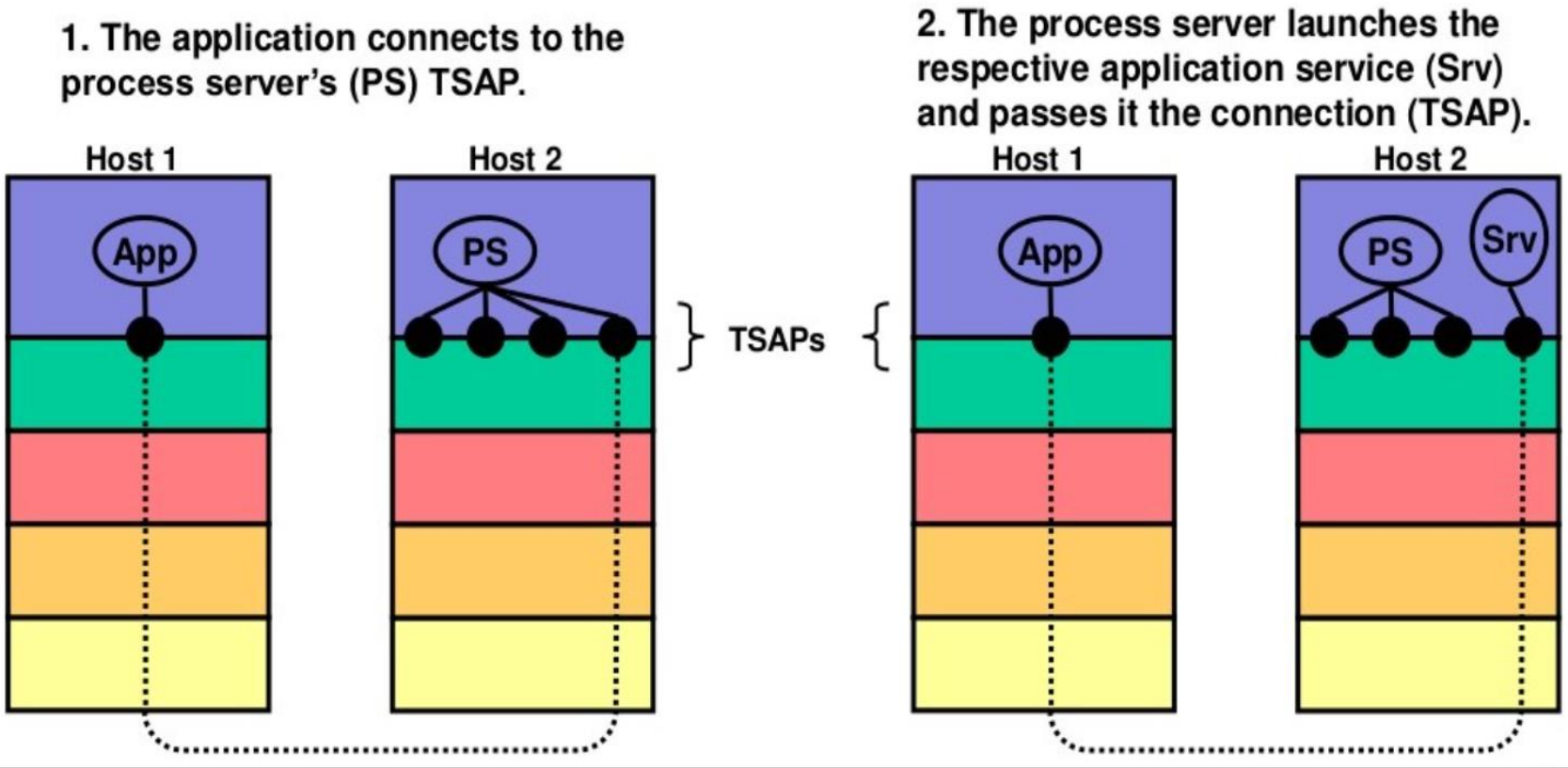


- ***Some frequently used applications such as e-mail and file transfers are allocated fixed TSAPs (also called well-known ports).***

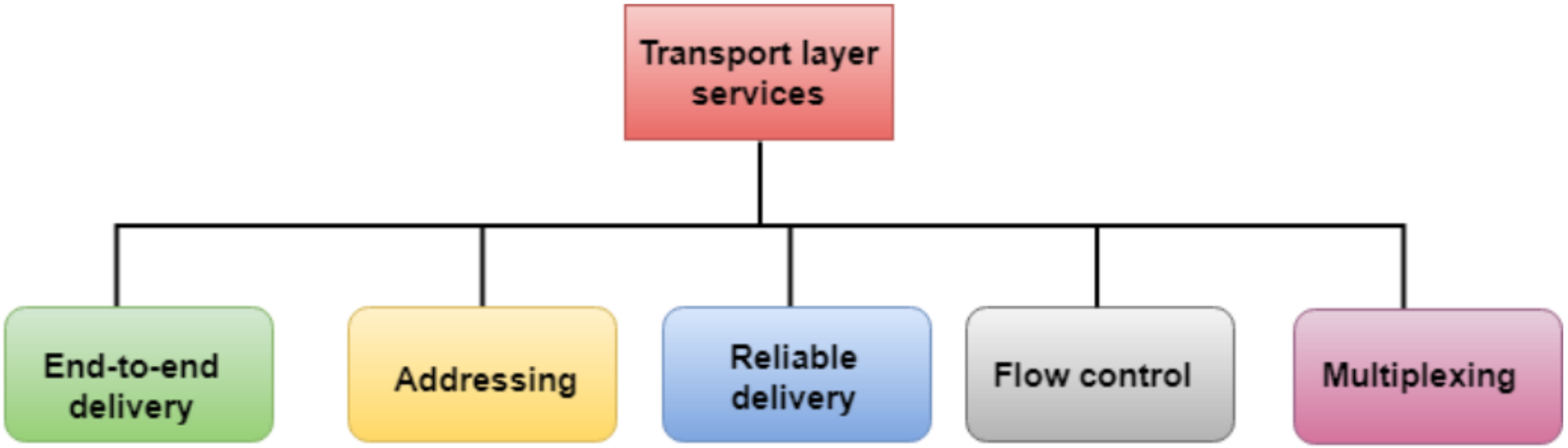
- To connect to a process with unknown TSAP, a remote process first connects to a process server attached to a fixed TSAP. The server then indicates the TSAP of the desired process.

Logical end to end delivery

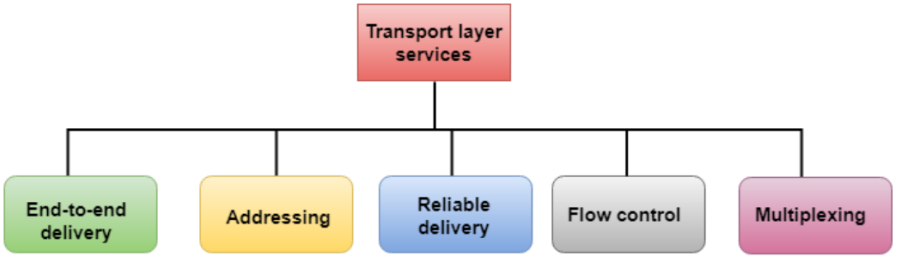
Prior to exchanging data a client and server must establish a connection (like a telephone connection).



Transport Layer Service



Transport Layer Service



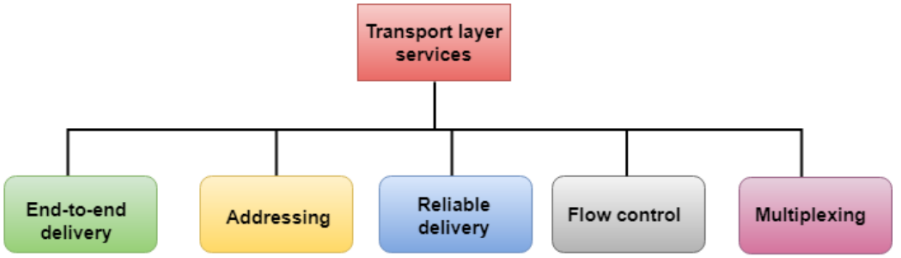
End-to-end delivery:

The transport layer transmits the entire message to the destination. Therefore, it ensures the end-to-end delivery of an entire message from a source to the destination.

Flow Control :

Flow control is used to prevent the sender from overwhelming the receiver. The transport layer is responsible for flow control. It uses the sliding window protocol that makes the data transmission more efficient.

Transport Layer Service

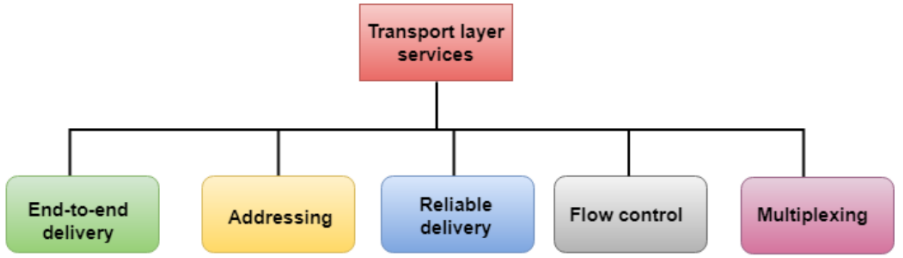


Reliable delivery:

The transport layer provides reliability services by retransmitting the lost and damaged packets. The reliable delivery has four aspects:

- Error control
- Sequence control
- Loss control
- Duplication control

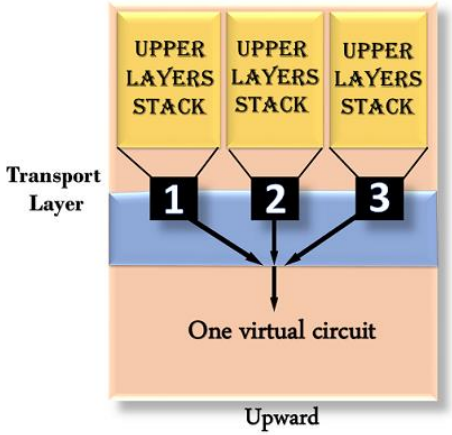
Transport Layer Service



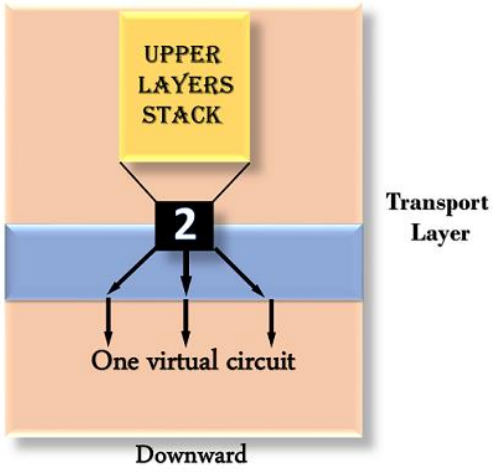
Multiplexing

The transport layer uses the multiplexing to improve transmission efficiency. Multiplexing can occur in two ways:

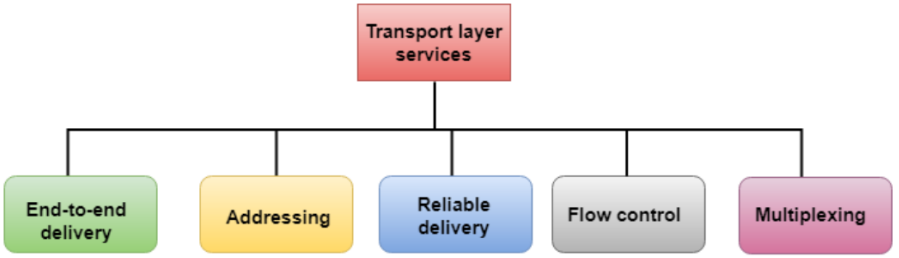
Upward multiplexing: Upward multiplexing means multiple transport layer connections use the same network connection. To make more cost-effective, the transport layer sends several transmissions bound for the same destination along the same path; this is achieved through upward multiplexing.



Downward multiplexing: Downward multiplexing means one transport layer connection uses the multiple network connections. Downward multiplexing allows the transport layer to split a connection among several paths to improve the throughput. This type of multiplexing is used when networks have a low or slow capacity.

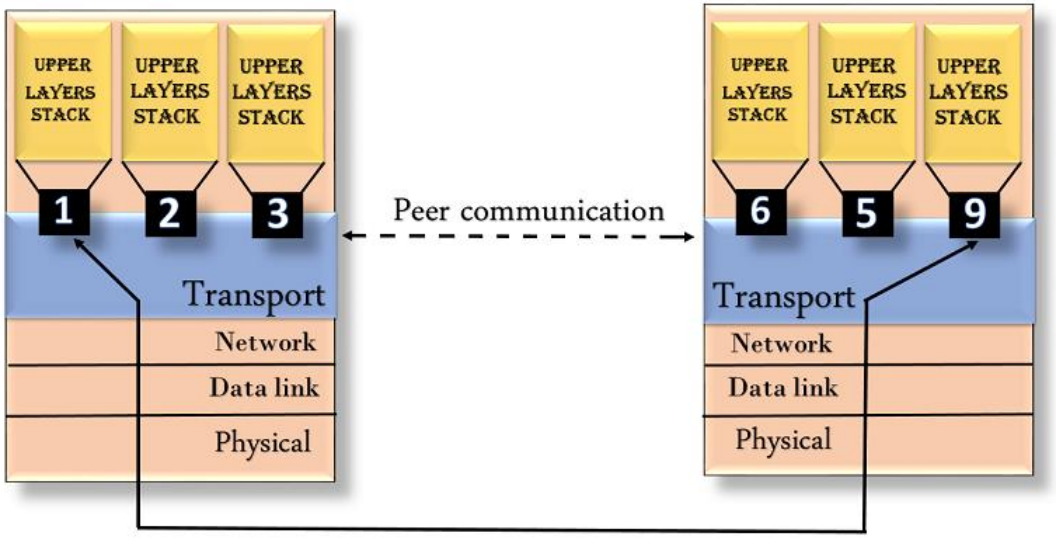


Transport Layer Service



Addressing

- According to the layered model, the transport layer interacts with the functions of the session layer. Many protocols combine session, presentation, and application layer protocols into a single layer known as the application layer. In these cases, delivery to the session layer means the delivery to the application layer. Data generated by an application on one machine must be transmitted to the correct application on another machine. In this case, addressing is provided by the transport layer.
- The transport layer provides the user address which is specified as a station or port.
- The transport layer protocols need to know which upper-layer protocols are communicating.



Elements Of Transport Protocol

- Addressing
- Connection Establishment
- Connection Release
- Flow control and Buffering

Addressing

- Method of defining transport layer addresses: **Port**

- An IP protocol is a host-to-host protocol used to deliver a packet from source host to the destination host while transport layer protocols are port-to-port protocols that work on the top of the IP protocols to deliver the packet from the originating port to the IP services, and from IP services to the destination port.

- Each port is defined by a positive integer address, and it is of 16 bits.

Elements Of Transport Protocol

- Addressing
- Connection Establishment
- Connection Release
- Flow control and Buffering

Connection Establishment

What is delayed duplicate-

In this problem, the TCP segments roam around in the network and delivered to the receiver when the duplicates retransmitted reached there already.

To solve this specific problem, Tomlinson (1975) introduced the **three-way handshake.**

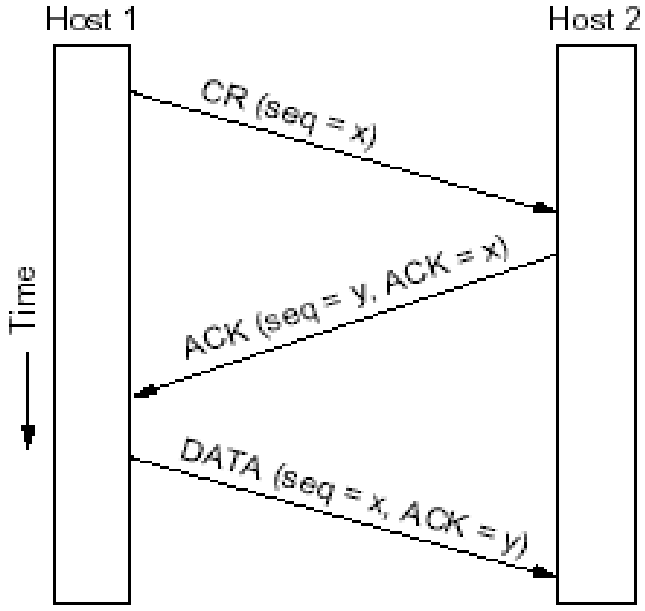
Three Way Handshaking – Normal case

This establishment protocol involves one peer checking with the other that the connection request is indeed current.

1. Host 1 chooses a sequence number, *x, and sends a CONNECTION REQUEST segment containing it to host 2.*

2. *Host 2* replies with an ACK segment acknowledging *x and announcing its own initial sequence* number, *y.*

3. *Finally, host 1 acknowledges host 2's choice of an initial sequence* number in the first data segment that it sends.

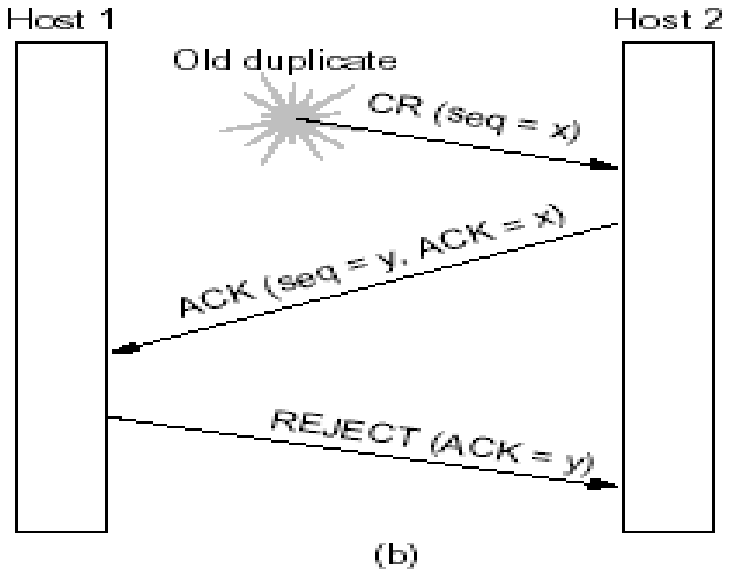


Three Way Handshaking – Scenario -1

1. The first segment is a delayed duplicate CONNECTION REQUEST from an old connection.

2. This segment arrives at host 2 without host 1's knowledge. Host 2 reacts to this segment by sending host 1 an ACK segment, in effect asking for verification that host 1 was indeed trying to set up a new connection.

3. When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.



Elements Of Transport Protocol

- Addressing
- Connection Establishment
- Connection Release
- Flow control and Buffering

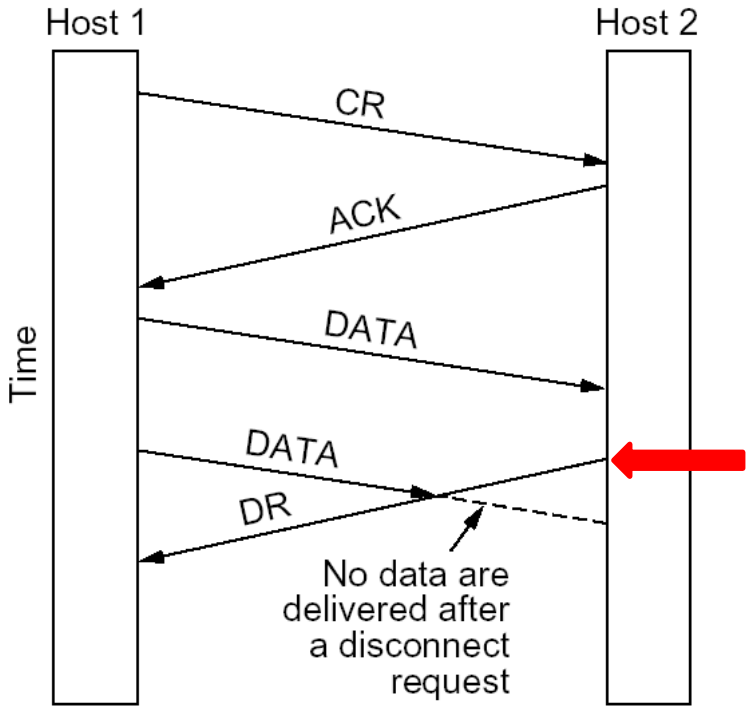
Connection Release

- Two ways:
  - Asymmetric
  - Symmetric
- Asymmetric is abrupt and may cause data loss
- Symmetric release, in which each direction is released independently of the other.
- Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it.

Connection Release – Asymmetric release

Asymmetric release is abrupt and may result in data loss. After the connection is established, host 1 sends a segment that arrives properly at host 2. Then host 1 sends another segment.

Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.



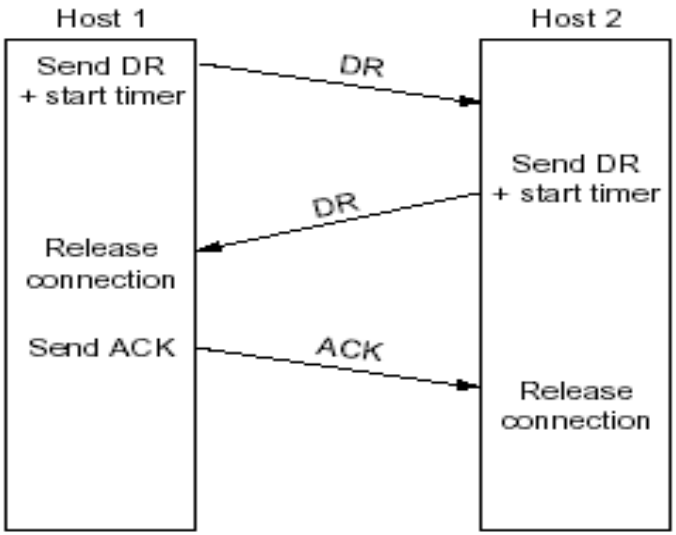
Connection Release – Symmetric release

Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it. In other situations, determining that all the work has been done and the connection should be terminated is not so obvious.

One can envision a protocol in which host 1 says ‘‘I am done. Are you done too?’’ If host 2 responds: ‘‘I am done too. Goodbye, the connection can be safely released.’’

Three Way Handshake For Disconnection – Normal case

- Users sends a DR (DISCONNECTION REQUEST) segment to initiate the connection release.
- When it arrives, the recipient sends back a DR segment and starts a timer, just in case its DR is lost.
- When this DR arrives, the original sender sends back an ACK segment and releases the connection.
- Finally, when the ACK segment arrives, the receiver also releases the connection.
- Releasing a connection means that the transport entity removes the information about the connection from its table of currently open connections and signals the connection's owner (the transport user) somehow.



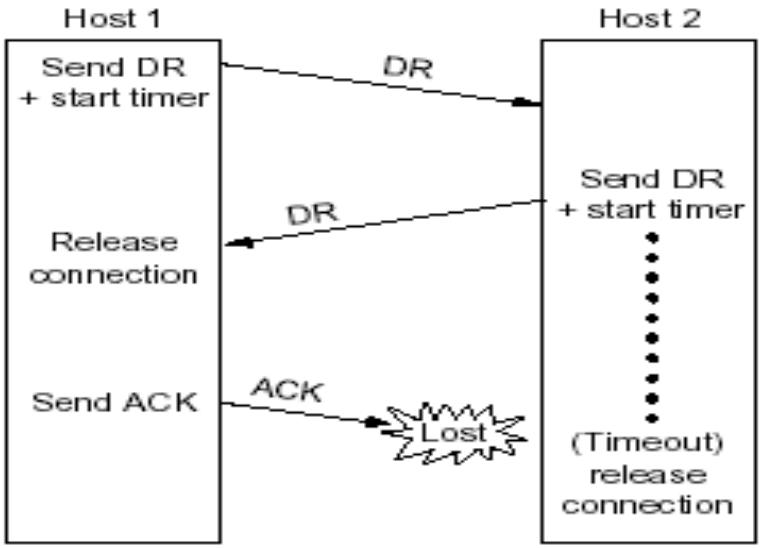
Three Way Handshake For Disconnection – Scenario-I &II

Scenario –I : If the final ACK segment is lost, the situation is saved by the timer. When the timer expires, the connection is released anyway.

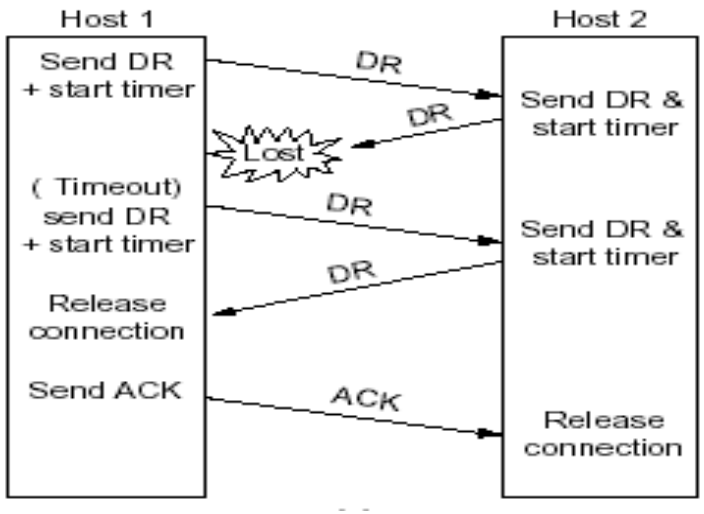
Scenario –II :

Now consider the case of the second DR being lost.

The user initiating the disconnection will not receive the expected response, will time out, and will start all over again.



Scenario –I



Scenario –II

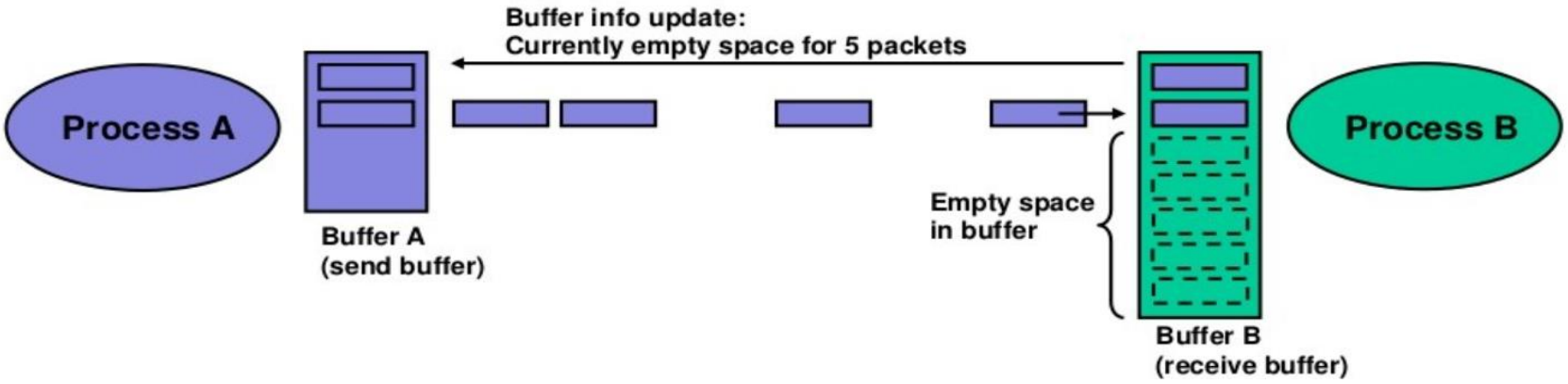
Elements Of Transport Protocol

- Addressing
- Connection Establishment
- Connection Release
- Flow control and Buffering

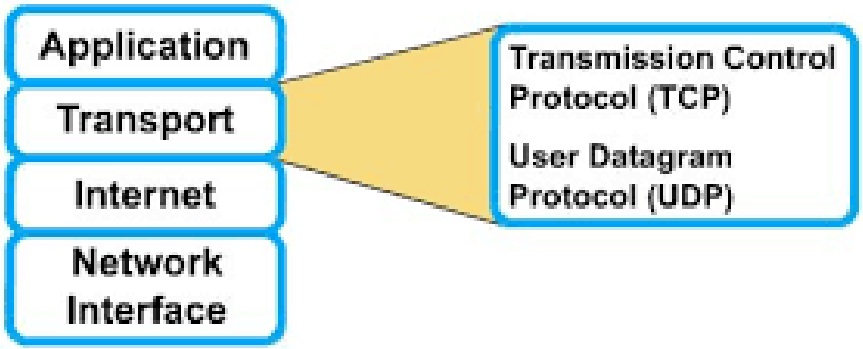
Flow Control And Buffering

- Error control is ensuring that the data is delivered with the desired level of reliability, usually that all of the data is delivered without any errors.

Problem:
The sender process may send at much higher speed than the receiver process can handle the data thus causing overflow (= packet loss).
Proposed solution:
The receiver buffers incoming packets.
A sliding window mechanism provides a "backpressure" to the sender process when the buffer is imminent to overflow (or better prevents the receive buffer from becoming full in the first place). The receiver process continuously tells the sending process how much empty space is left in its receive buffer. The sender process never sends more data than can be accommodated in the receive buffer.
More details see TCP flow control.

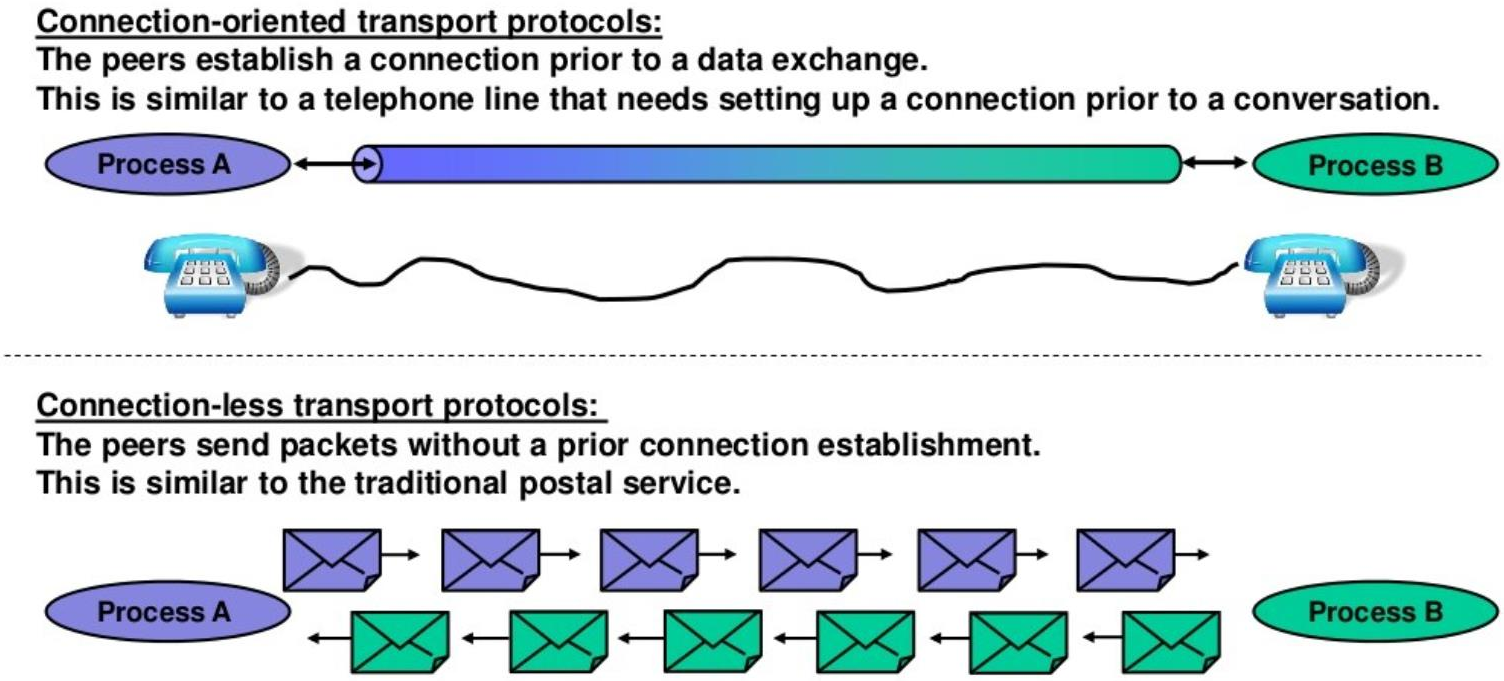


Transport Layer Protocol



Transport Layer Protocol

- Normally transport layer is represented by two protocols: TCP and UDP.
- Connection Oriented – Transmission Control Protocol
- Connectionless Oriented – User Datagram Protocol



Transport Layer Protocol

- Normally transport layer is represented by two protocols: TCP and UDP.
- Connection Oriented – Transmission Control Protocol (Reliable)
- Connectionless Oriented – User Datagram Protocol (Unreliable)

Combinations:
The characteristics connection-oriented / connection-less and reliable / unreliable can be combined. Usually connection-oriented protocols provide reliable transport service.

	Reliable	Unreliable
Connection-oriented	TCP, SCTP	-
Connection-less	RUDP	UDP

UDP: Unreliable, connection-less message (datagram) delivery protocol.
TCP: Reliable, connection-oriented stream transfer protocol.
SCTP: Reliable, connection-oriented message transfer protocol.
RUDP: Reliable UDP (mixture between TCP and UDP)

Transport Layer Protocol

- Normally transport layer is represented by two protocols: TCP and UDP.
- Connection Oriented – Transmission Control Protocol
- Connectionless Oriented – User Datagram Protocol

TCP stands for **Transmission Control Protocol.**

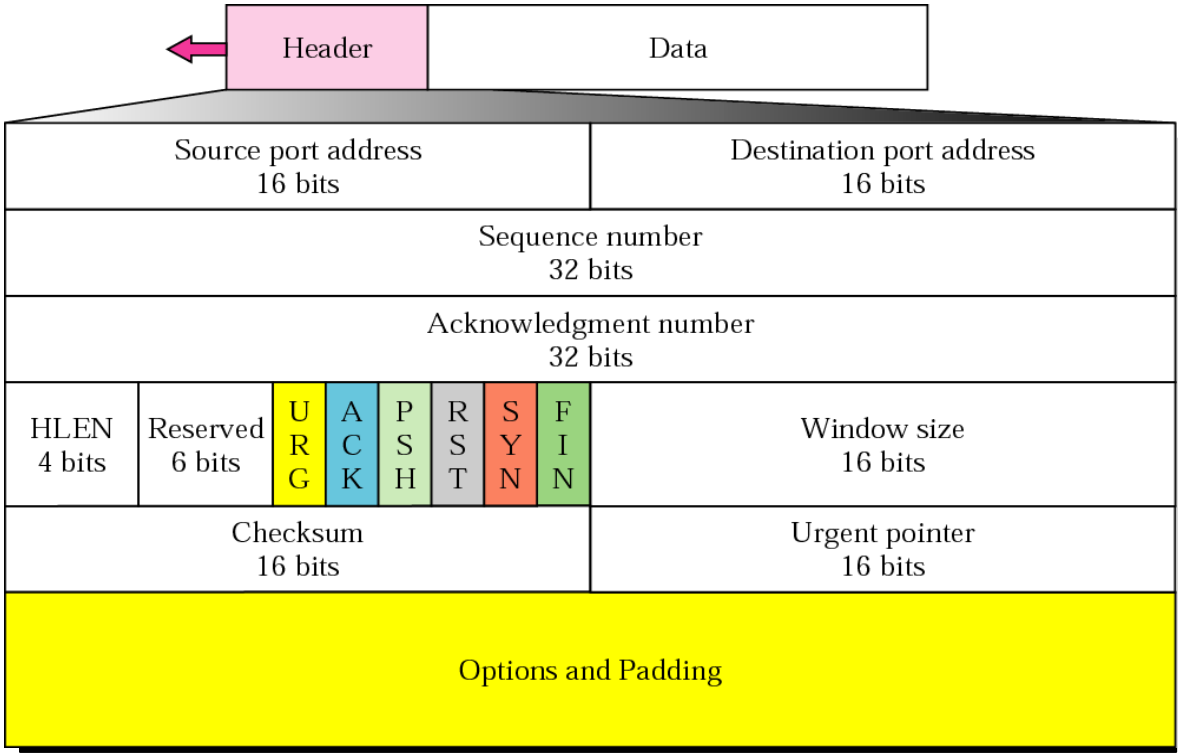
- It provides full transport layer services to applications.
- It is a connection-oriented protocol means the connection established between both the ends of the transmission.

UDP stands for **User Datagram Protocol**.

- UDP is a simple protocol and it provides nonsequenced transport functionality.
- UDP is a connectionless protocol.
- This type of protocol is used when reliability and security are less important than speed and size.
- The packet produced by the UDP protocol is known as a user datagram.

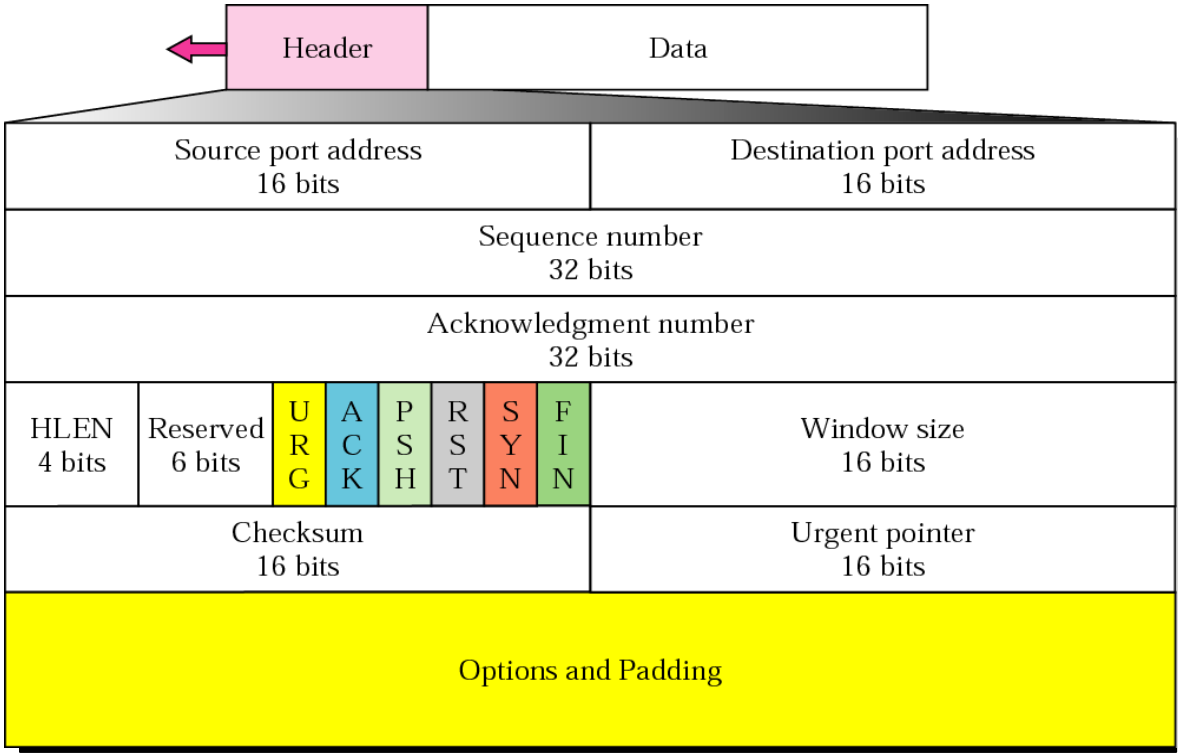
TCP segment format

- Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options.
- After the options, if any, up to 65,535 − 20 − 20 = 65,495 data bytes
- Segments without any data are legal and are commonly used for acknowledgements and control messages.

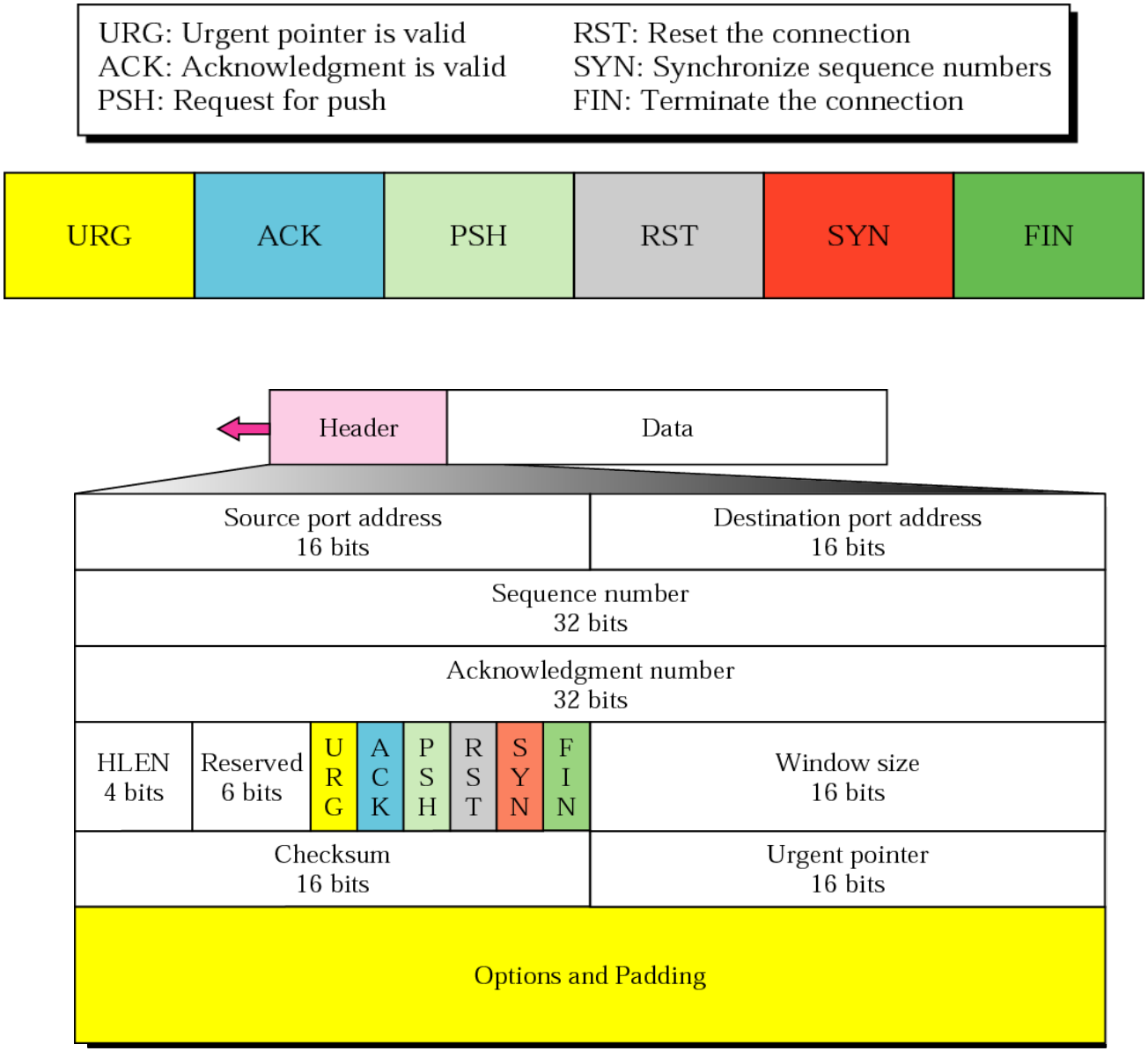


TCP segment format

- The *Source port and Destination port fields identify the local end points of the connection.*
- The *Sequence number and Acknowledgement number fields performs their usual functions.*
- The *TCP header length tells how many 32-bit words are contained in the TCP* header.

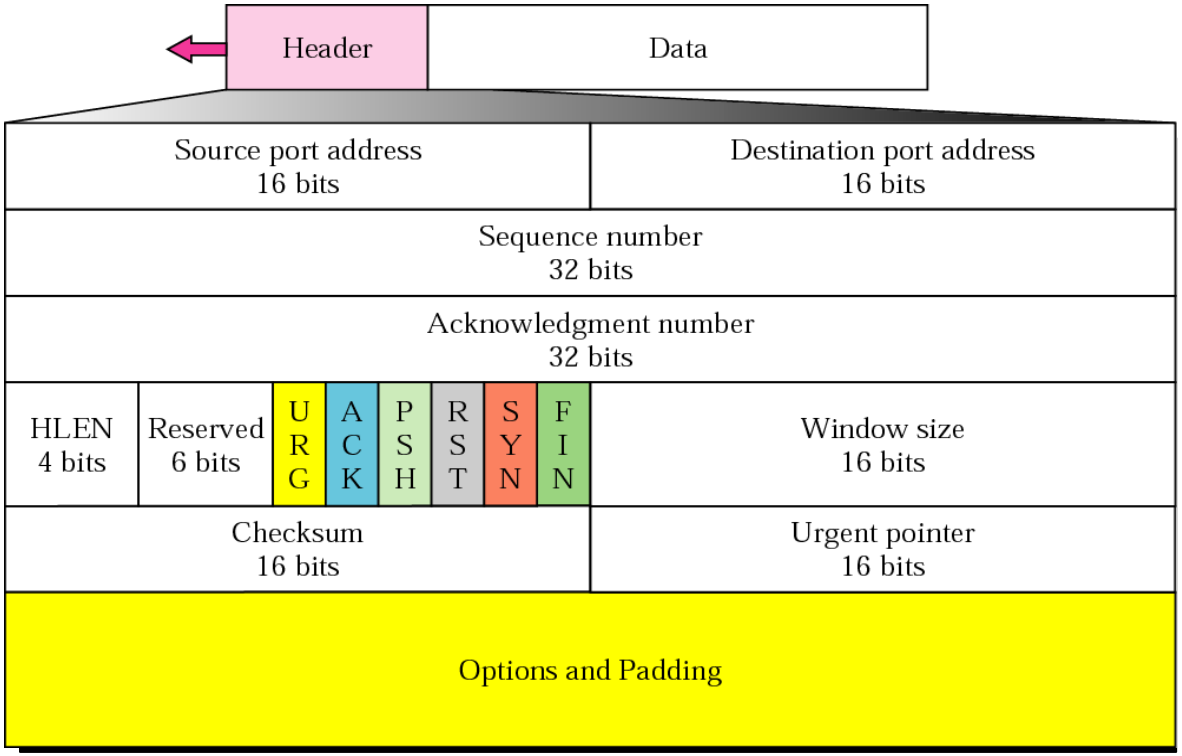


TCP segment format

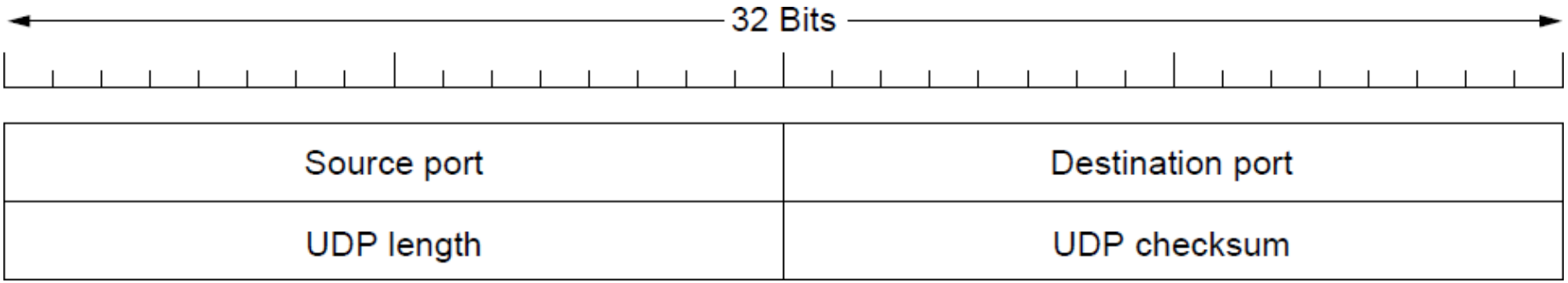


TCP segment format

Flow control in TCP is handled using a variable-sized sliding window. The ***Window size*** *field tells how many bytes may be sent starting at the byte acknowledged.*



Introduction to UDP



- The two **ports serve to identify the endpoints** within the source and destination machines. When a UDP packet arrives, its payload is handed to the process attached to the destination port.
- The *UDP length field includes the 8-byte header and the data. The minimum* length is 8 bytes, to cover the header.
- When performing this computation, the *Checksum field is set to zero and the data field is padded out with an* additional zero byte if its length is an odd number.