

Transport Layer

- Main Tasks
- TCP
- UDP

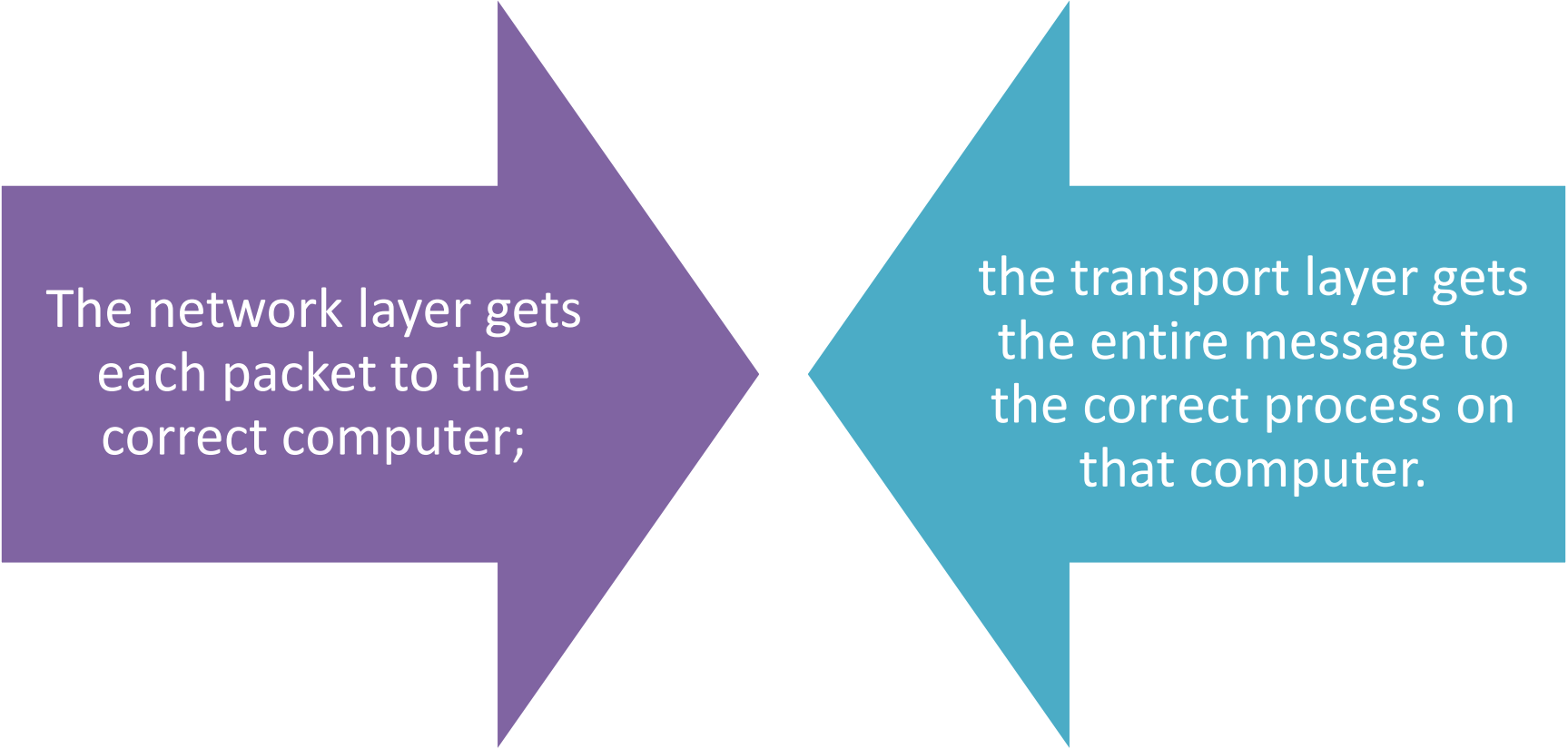
Transport Layer (4th OSI layer)

- It is responsible for **process-to-process delivery** of the entire message.
 - A process is an application program running on a host.
 - Network layer oversees source-to-destination delivery of individual packets, it does not recognize any relationship between those packets.

Process-to-Process Delivery

- The **data link layer** is responsible for delivery of frames between two neighboring nodes over a link. This is **called node-to-node delivery**.
- The **network layer** is responsible for delivery of datagrams between two hosts. This is **called host-to-host delivery**.
- The **transport layer** is responsible **for process-to-process delivery**-the delivery of a packet, part of a message, from one process to another.

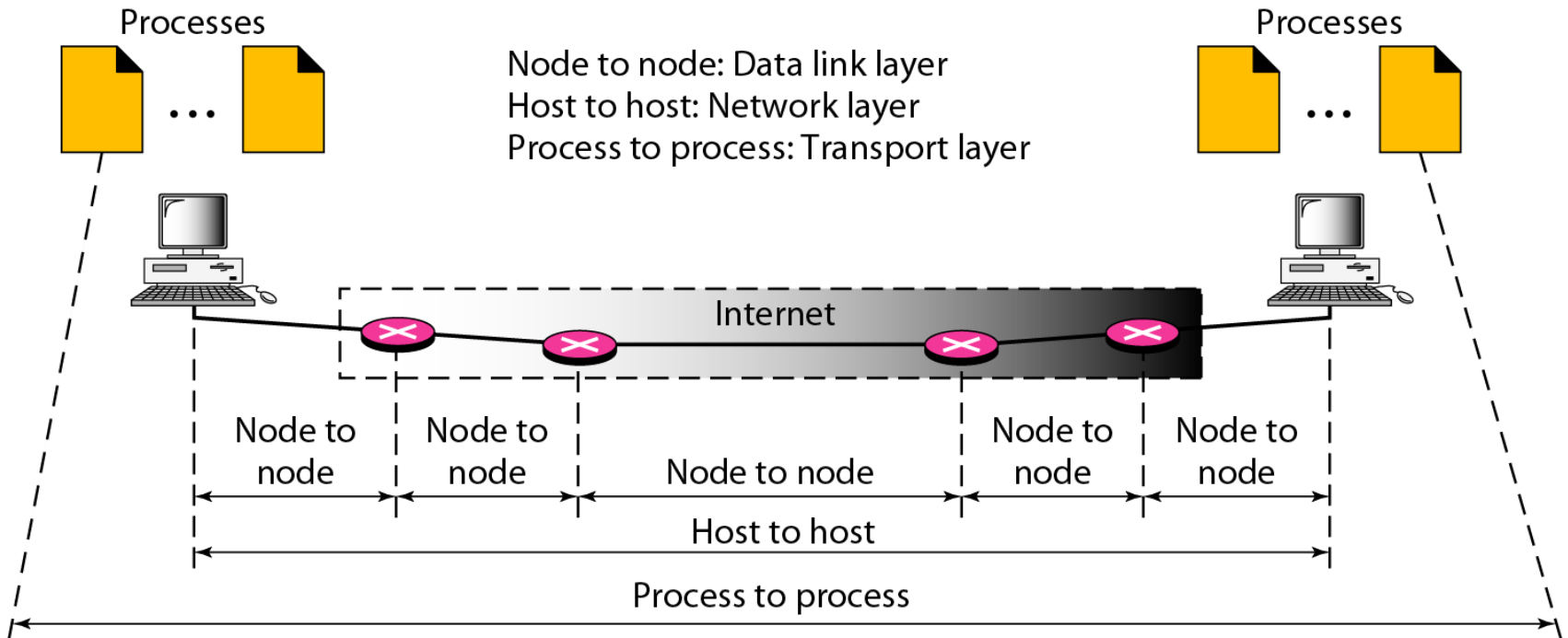
Cont'd



The network layer gets
each packet to the
correct computer;

the transport layer gets
the entire message to
the correct process on
that computer.

Figure :Types of data deliveries



Responsibilities

1. Service-point addressing

- Computers often run several programs at the same time.
- source-to-destination delivery means delivery not only from one computer to the next but also from a specific process (running program) on one computer to a specific process on the other.
- The transport layer header must therefore include a type of address called a service-point address (or port address).

Transport layer addressing

- to deliver something to one specific destination among many, we need an address.
 - At the data link layer (MAC address)
 - to choose one node among several nodes if the connection is not point-to-point.
 - At the network layer (IP address)
 - to choose one host among millions.
 - At the transport layer(port number)
 - to choose among multiple processes running on the destination host.

Cont'd

2. Segmentation and Reassembly

- A message is divided into transmittable segments, with each segment containing a sequence number.
- These numbers enable the transport layer to reassemble the message correctly upon arriving at the destination and to identify and replace packets that were lost in transmission.

Cont'd

3. Flow Control

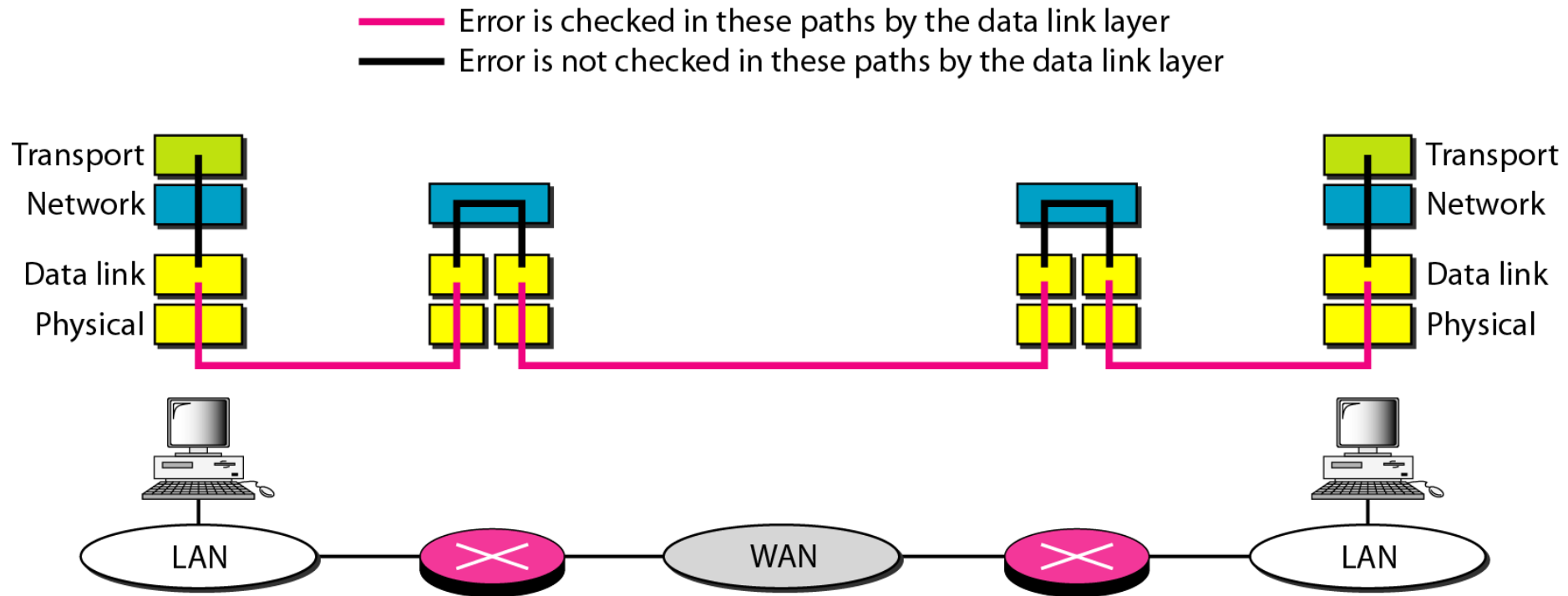
- Like the data link layer, the transport layer is responsible for flow control.
- However, flow control at this layer is performed end to end rather than across a single link.

Cont'd

4. Error control

- error control at this layer is performed process-to-process rather than across a single link.
- The sending transport layer makes sure that the entire message arrives at the receiving transport layer without error (damage, loss, or duplication).
- Error correction is usually achieved through retransmission.

Figure : Error control

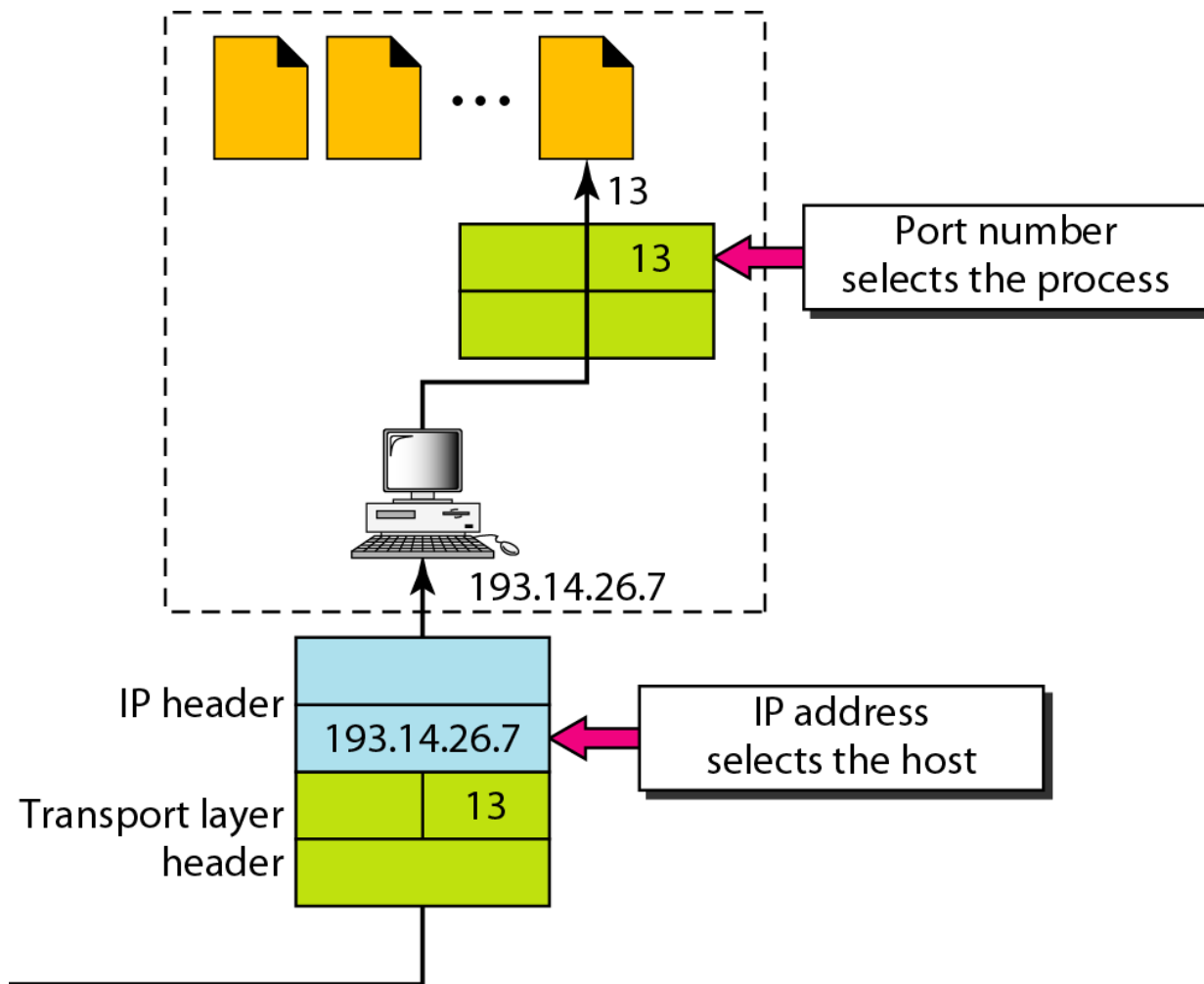


Contd.

5. Connection control

- The transport layer can be either connectionless or connection-oriented.
- Connectionless transport layer
 - treats each segment as an independent packet and delivers it to the transport layer at the destination machine.
- Connection-oriented transport layer
 - makes a connection with the transport layer at the destination machine first before delivering the packets.

Figure *IP addresses versus port numbers*



Port Number

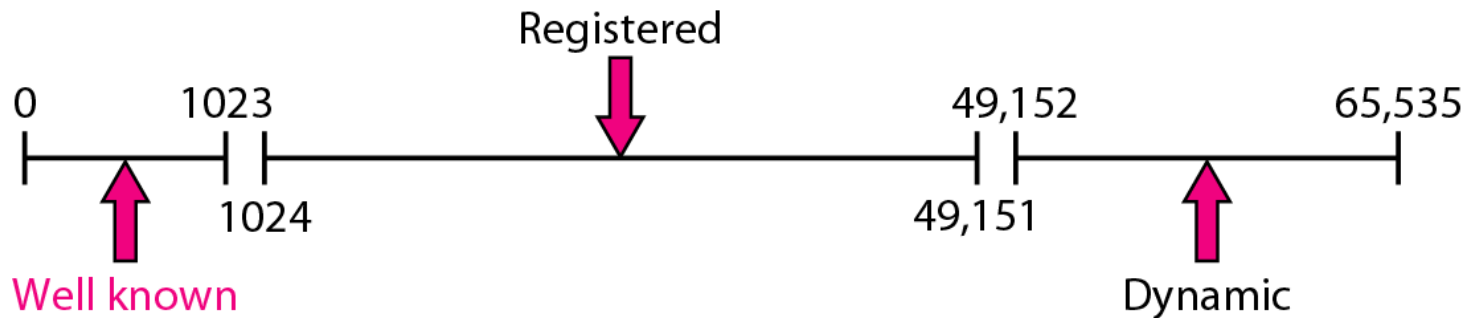
- The destination port number is needed for delivery.
- The source port number is needed for the reply.
- The port numbers are 16-bit integers between 0 and 65,535.
- **The client program**
 - defines itself with a port number
 - chosen randomly by the transport layer software running on the client host.
 - This is the **ephemeral (temporal)** port number.
- **The server process**
 - must also define itself with a port number.
 - Use well known ports and allows client applications to easily locate the corresponding server application processes on other hosts.

Figure: *IANA ranges*

Well known :
assigned and
controlled by IANA

Registered :
used by vendors for
their own server
applications

Dynamic:
private or temporary ports
which are neither controlled nor
registered. They can be used by
any process.
Called **ephemeral ports**

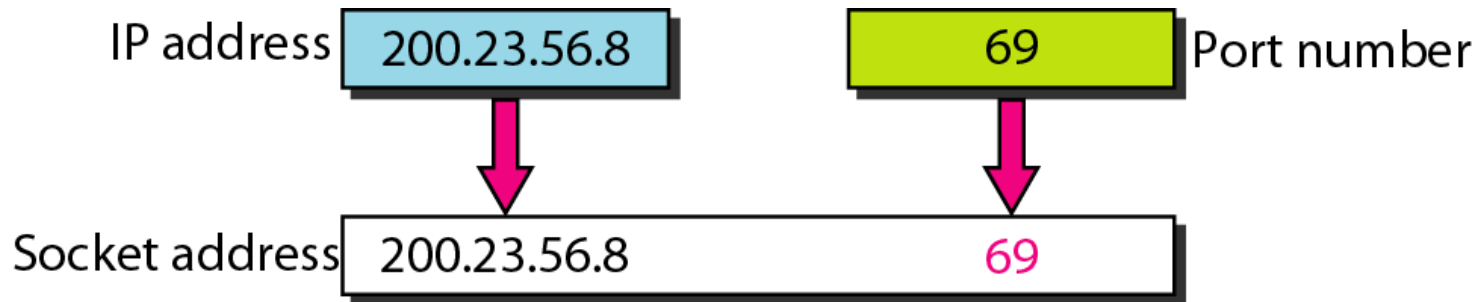


Example of well known port numbers:

53- DNS ,21-FTP, 23-telnet, 25-SMTP, 80-HTTP, etc

Socket Addresses

- Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection.
- The combination of an IP address and a port number is called a socket address.



Transport Layer Services

**Connectionless
Vs.
Connection-Oriented
Services**

**Reliable
Vs.
Unreliable
Services**

Connectionless Service

- the packets are sent from one party to another with no need for connection establishment or connection release.
- The packets are not numbered;
- they may be delayed or lost or may arrive out of sequence.
- There is no acknowledgment either.
- UDP is a connectionless protocol

Connection-Oriented Service

- a connection is first established between the sender and the receiver.
- Data are transferred.
- At the end, the connection is released.
- TCP and SCTP are connection-oriented protocols.

Reliable Vs. Unreliable

- **Reliable**

- Implementing flow and error control in Transport layer
- slower and more complex service.

- **Unreliable**

- if the application program uses its own flow and error control mechanism
- When application program needs fast service
- the nature of the service does not demand flow and error control (real-time applications)

Transport Layer Protocols

- UDP
- TCP
- SCTP (*Stream Control Transmission Protocol*)

UDP

- User Datagram Protocol
- UDP is connectionless and unreliable
- It does not add anything to the services of IP except to provide process-to-process communication.
- it performs very limited error checking.
- Does provide a checksum to verify individual packet integrity.

UDP issues

- UDP is not capable of segmenting and reassembling frames
- Does not implement sequence numbers
- No protection against duplicate packets.
- No guaranteed ordering of packets.
- No verification of the readiness of the computer receiving the message.
- No guarantee the destination will receive all transmitted bytes.

UDP issues

- UDP can transmit only **small portions of data** at a time because it is not capable of segmenting and reassembling frames and does not implement sequence numbers

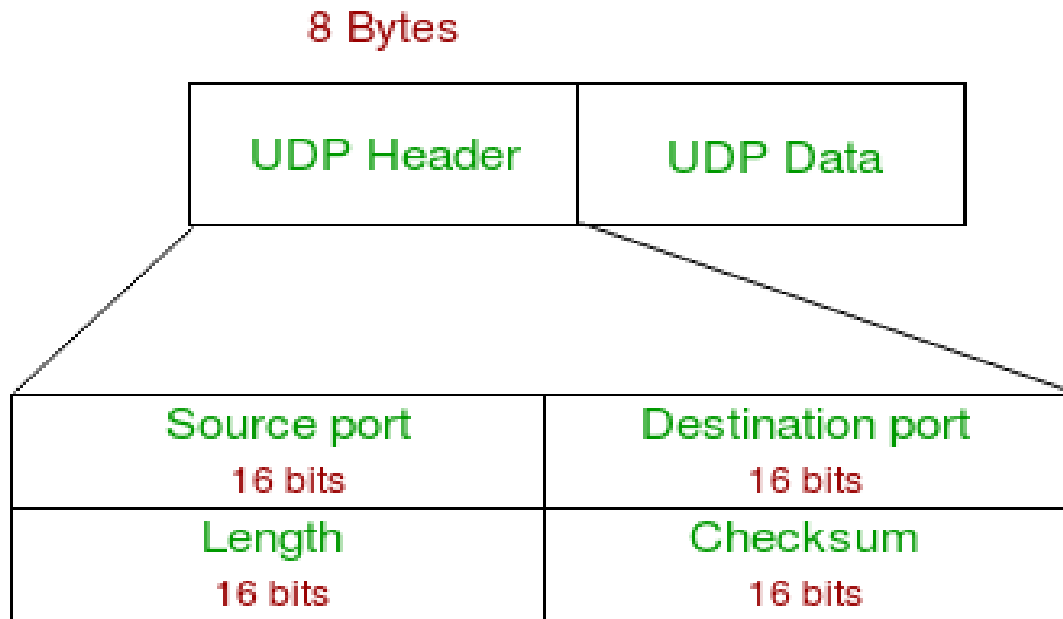
UDP Benefits

- No retransmission delay
- Speed
- Suitable for broadcasts and multicasts
- UDP is suitable for a process with internal flow and error control mechanisms.
 - For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP

UDP header packet structure

- UDP packets, called user **datagrams**
- contains four fields totaling 8 bytes.
- The fields in a UDP header are:
 - Source port number
 - Destination port number
 - Length
 - Check Sum

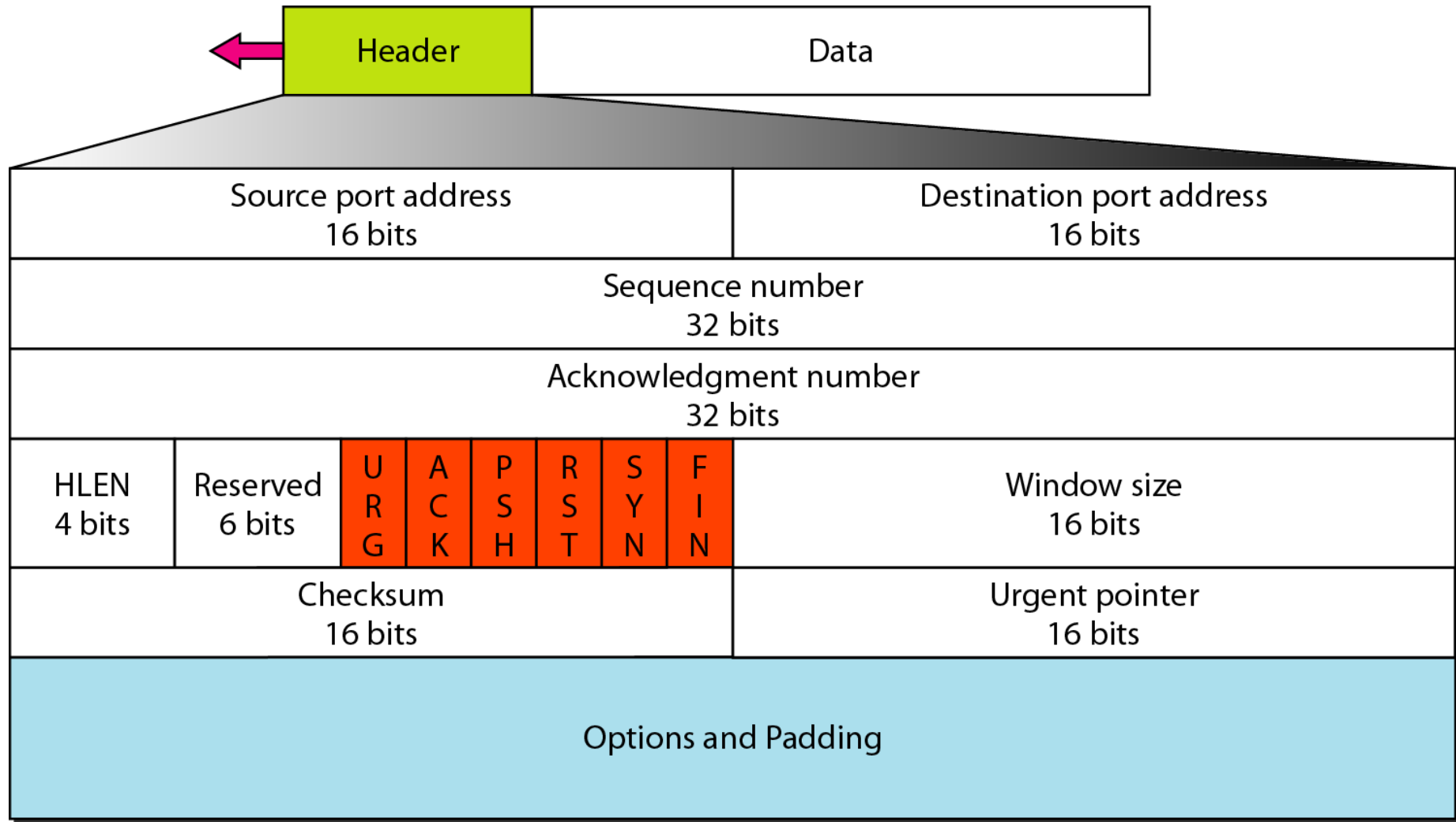
Contd.



TCP

- **Transmission Control Protocol**
- It is a connection-oriented and reliable protocol.
- TCP, like UDP, is a process-to-process (program-to-program) protocol.
- uses flow and error control mechanisms at the transport level

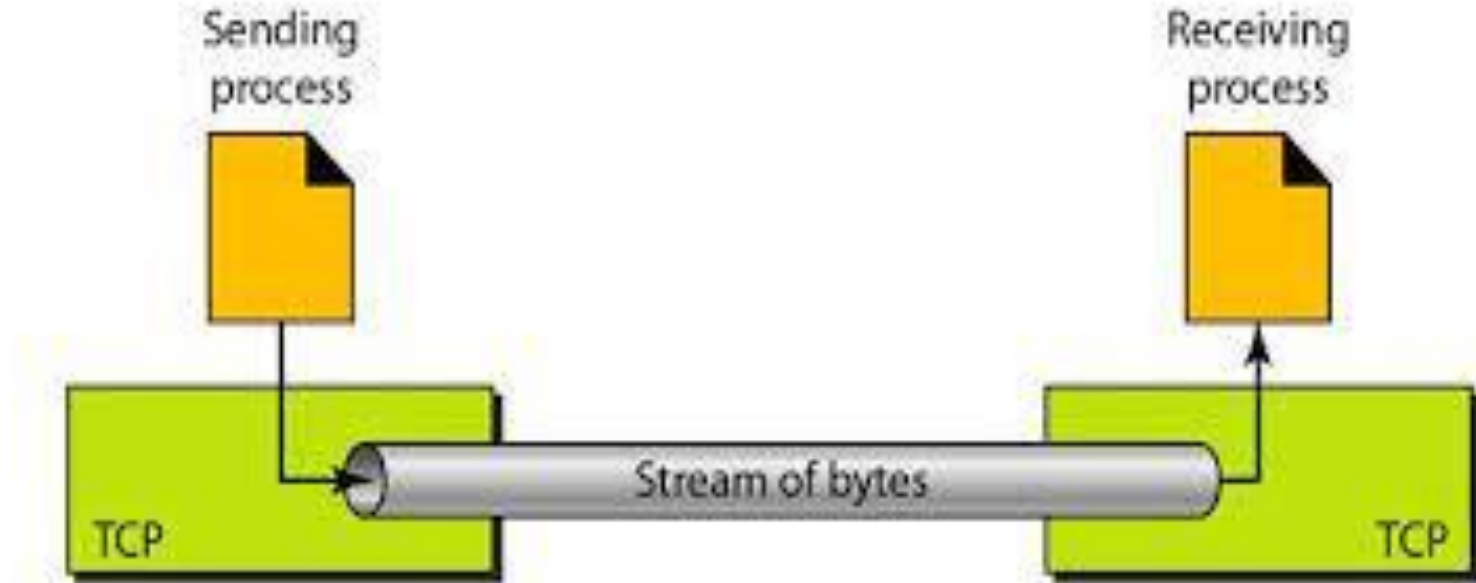
Figure : TCP segment format



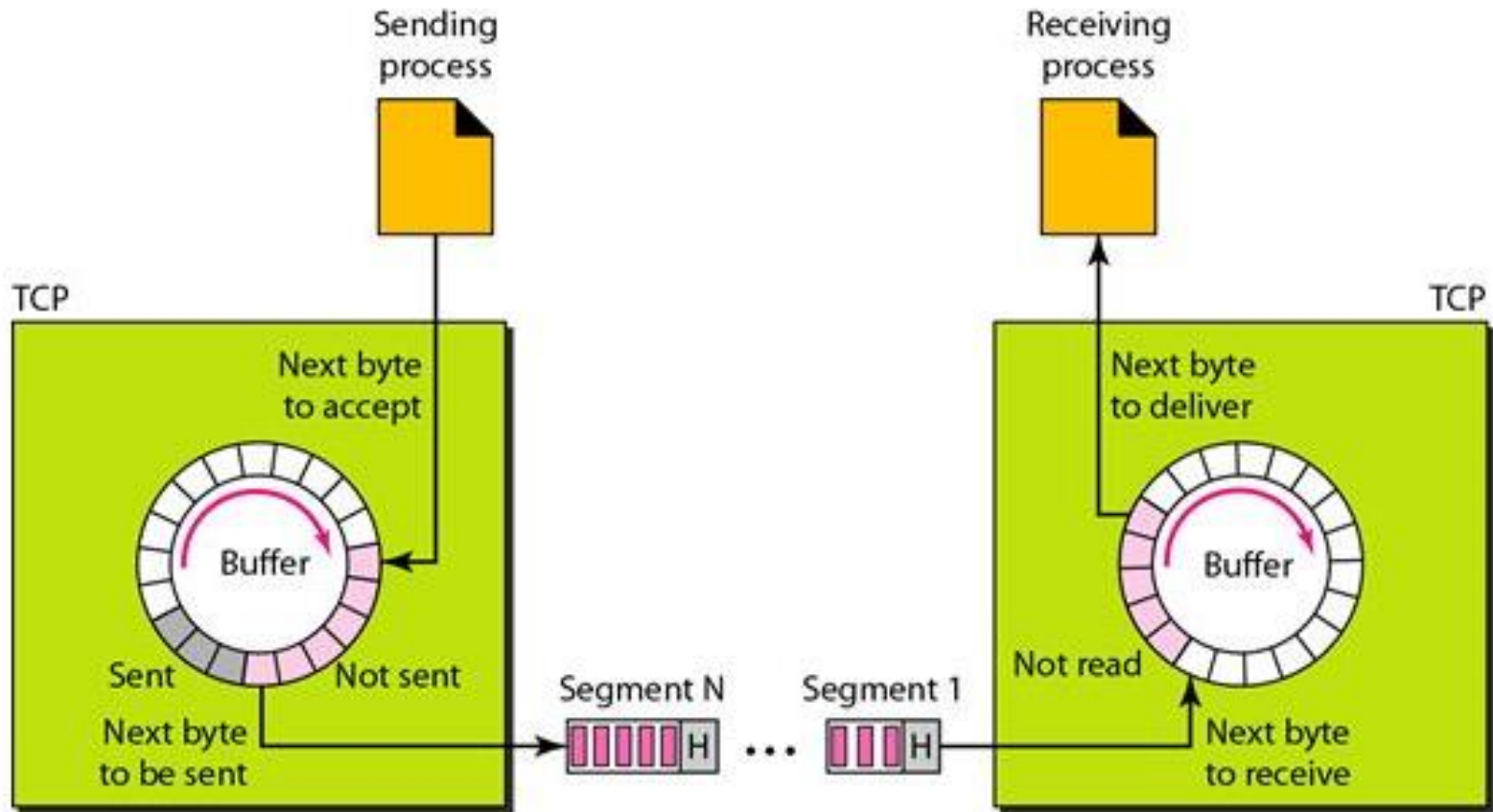
TCP Services

- Process-to-Process Communication
 - Like UDP, TCP provides process-to-process communication using port numbers.
- Stream Delivery Service
- Full-Duplex Communication
- Connection-Oriented Service
- Reliable Service

TCP data delivery



Sending and receiving buffers



TCP Segment Numbering System

- **Byte Number**

- TCP numbers all data bytes that are transmitted in a connection.
- Numbering is independent in each direction.
- When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them.
- The numbering does not necessarily start from 0.

TCP Segment Numbering System

- Byte Number

- For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056.
- Byte numbering is used for flow and error control too.

TCP Segment Numbering System

- **Sequence Number**

- After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent.
- The sequence number for each segment is the number of the first byte carried in that segment.

Quiz

- Suppose a TCP connection is transferring a file of 8000 bytes. The first byte is numbered 10,001.
- What are the sequence numbers for each segment if data are sent in four segments, each carrying 2000 bytes?

Quiz :Solution

- Segment 1 Sequence Number:

10,001 (range: 10,001 to 12,000)

- Segment 2 Sequence Number:

12,001 (range: 12,001 to 14,000)

- Segment 3 Sequence Number:

14,001 (range: 14,001 to 16,000)

- Segment 4 Sequence Number:

16,001 (range: 16,001 to 18,000)

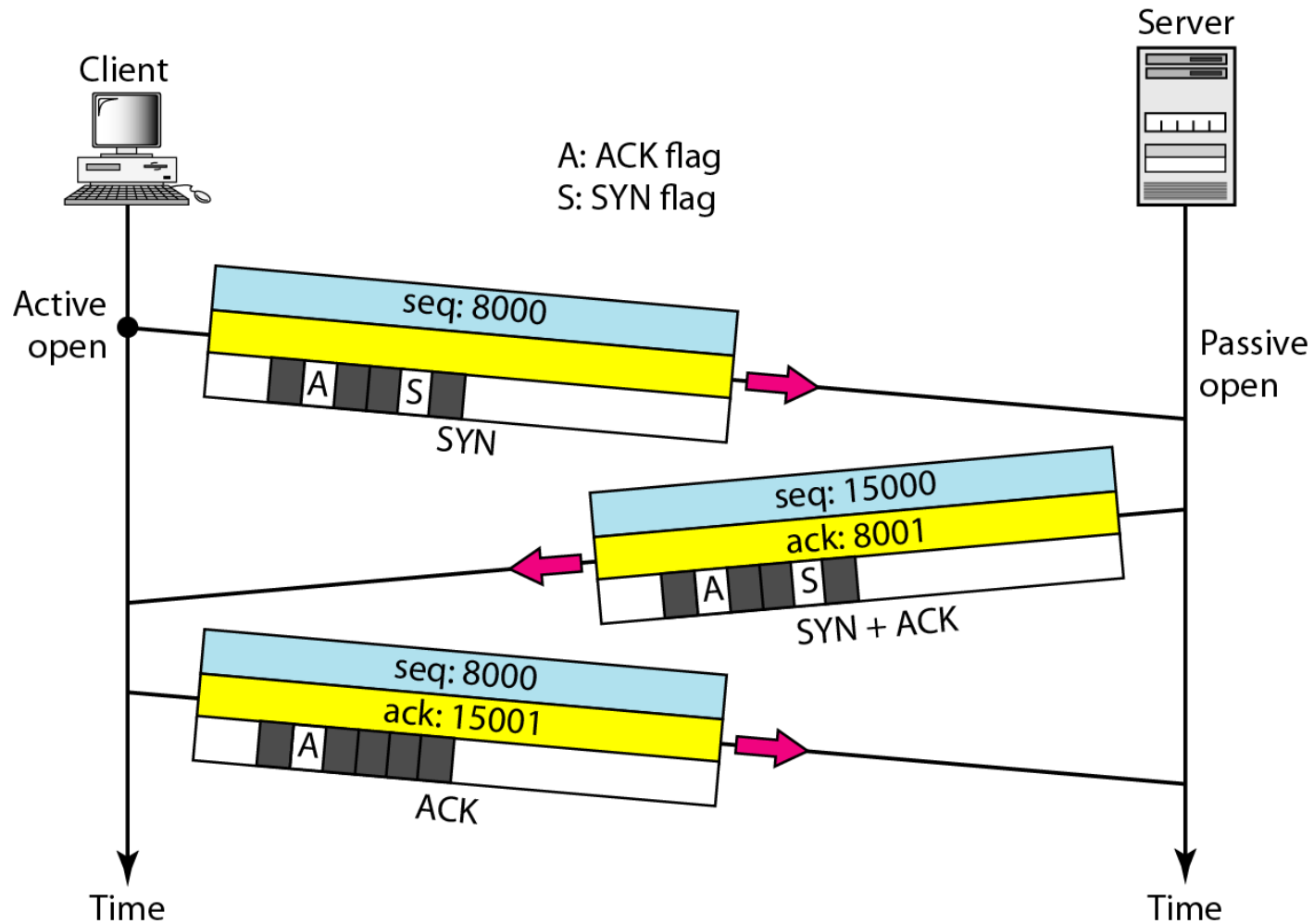
Acknowledgment Number

- It is cumulative i.e. the party takes the number of the last byte that it has received, safe and sound, adds 1 to it, and announces this sum as the acknowledgment number.
- if a party uses 5643 as an acknowledgment number, it has received all bytes from the beginning up to 5642.
 - Note that this does not mean that the party has received 5642 bytes because the first byte number might not have to start from 0.

Three-Way Handshaking

- The process starts with the server.
 - The server program tells its TCP that it is ready to accept a connection. This is called a request for a passive open.
- The client program issues a request for an active open.
- A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server.
- TCP can now start the three-way handshaking process

Figure : *Connection establishment using three-way handshaking*



A SYN segment cannot carry data, but it consumes one sequence number.

A SYN + ACK segment cannot carry data, but does consume one sequence number.

An ACK segment, if carrying no data, consumes no sequence number.

Figure: Data transfer

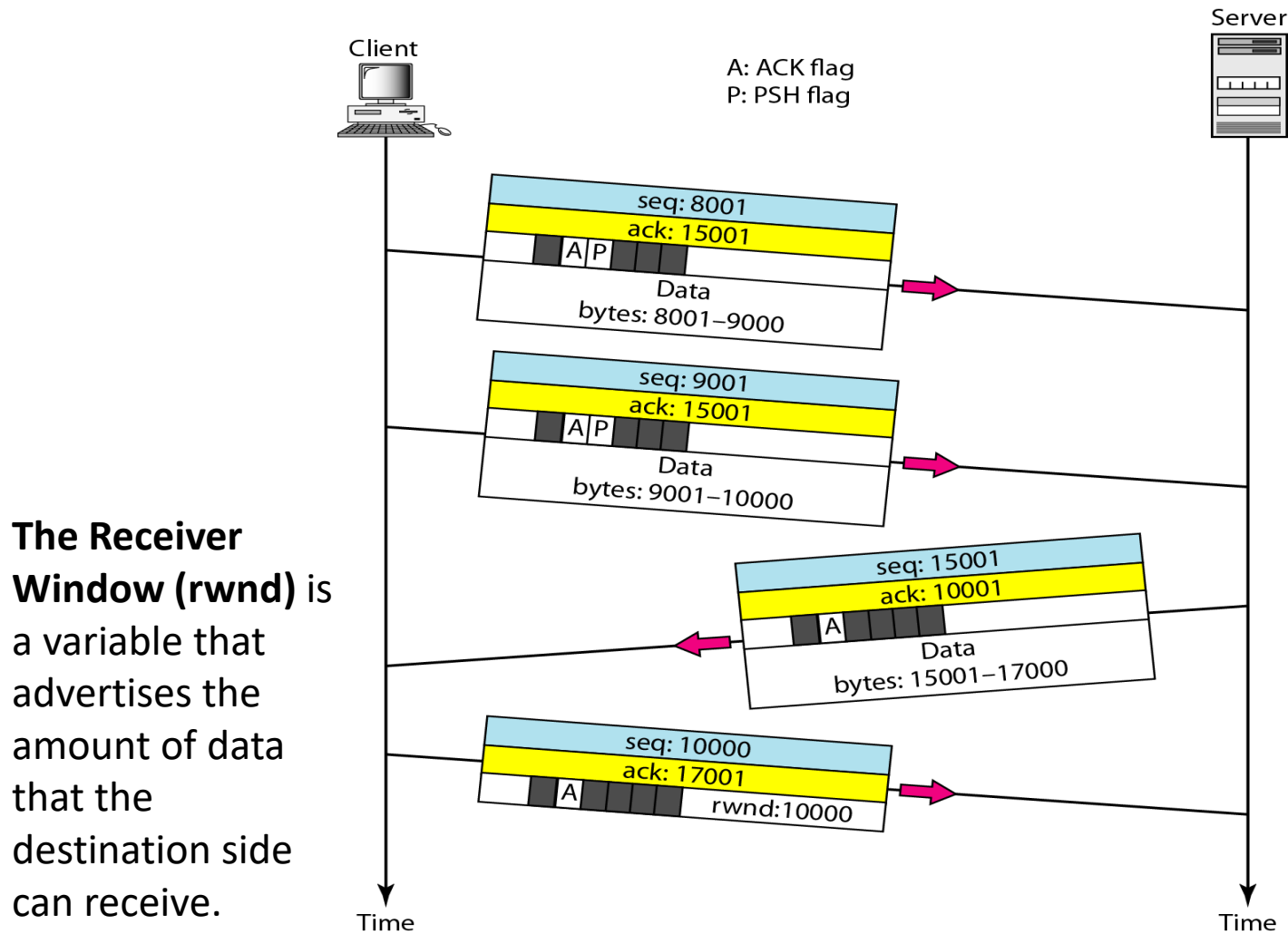
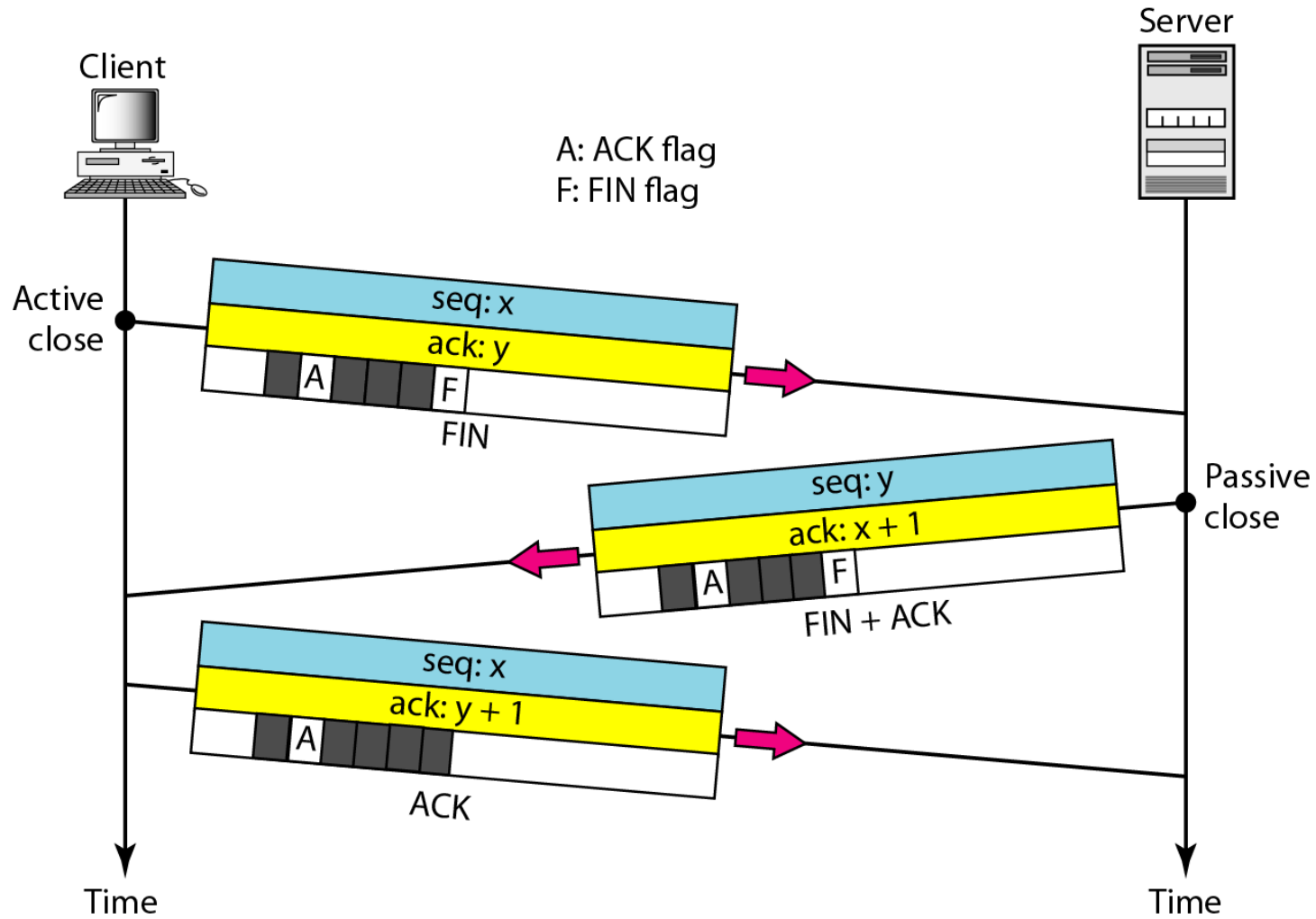


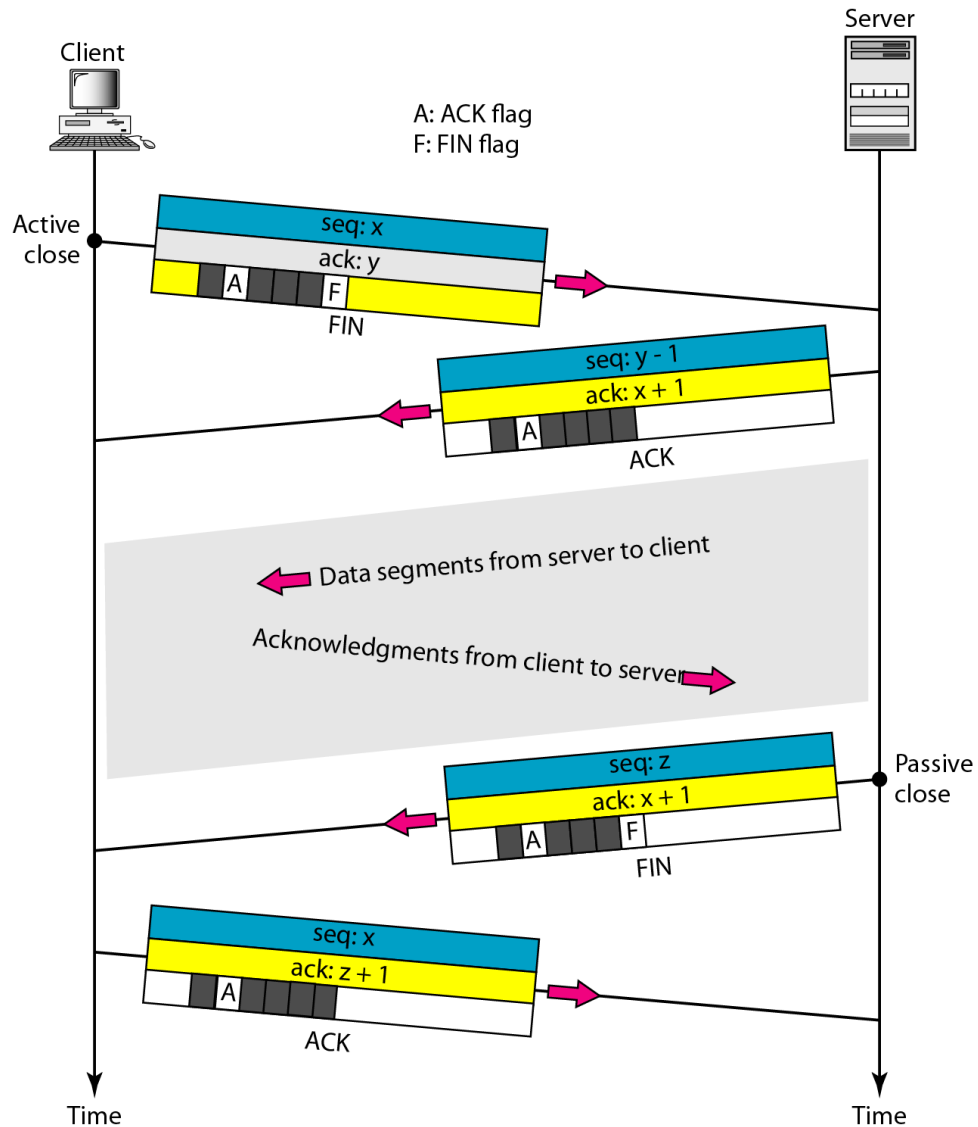
Figure : *Connection termination using three-way handshaking*



The FIN segment consumes one sequence number if it does not carry data.

The FIN + ACK segment consumes one sequence number if it does not carry data.

Figure :Half-close



Congestion

- **Congestion:**

- Congestion in a network may occur if the load on the network-the number of packets sent to the network-is greater than the capacity of the network-the number of packets a network can handle.

- **Congestion control**

- refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

Congestion control	Flow Control
Traffics are controlled entering to the network.	Traffics are controlled which are flow from sender to a receiver.
Network layer and Transport layer handle it.	Data link layer and Transport layer handle it.
Network is prevented from congestion.	Receiver's data is prevented from being overwhelmed.
Transport layer is responsible for the traffic.	Only sender is responsible for the traffic.
Traffic is prevented by slowly transmitting by the transport layer.	Traffic is prevented by slowly sending by the sender.
buffer overrun is restrained in the intermediate systems in the network.	buffer overrun is restrained in the receiver.

CONGESTION CONTROL

- Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.
- In general, we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal) as shown in Figure below.

Congestion Control

Open Loop

- Retransmission Policy
- Window Policy
- Acknowledgment Policy
- Discarding Policy
- Admission Policy

Closed Loop

- Back Pressure
- Choke packet
- Implicit signaling
- Explicit signaling

Thank You