UNIT 5

Transport layer

Unit Covered

- Process to process delivery
- UDP – User datagram protocol
- TCP – Transmission Control protocol

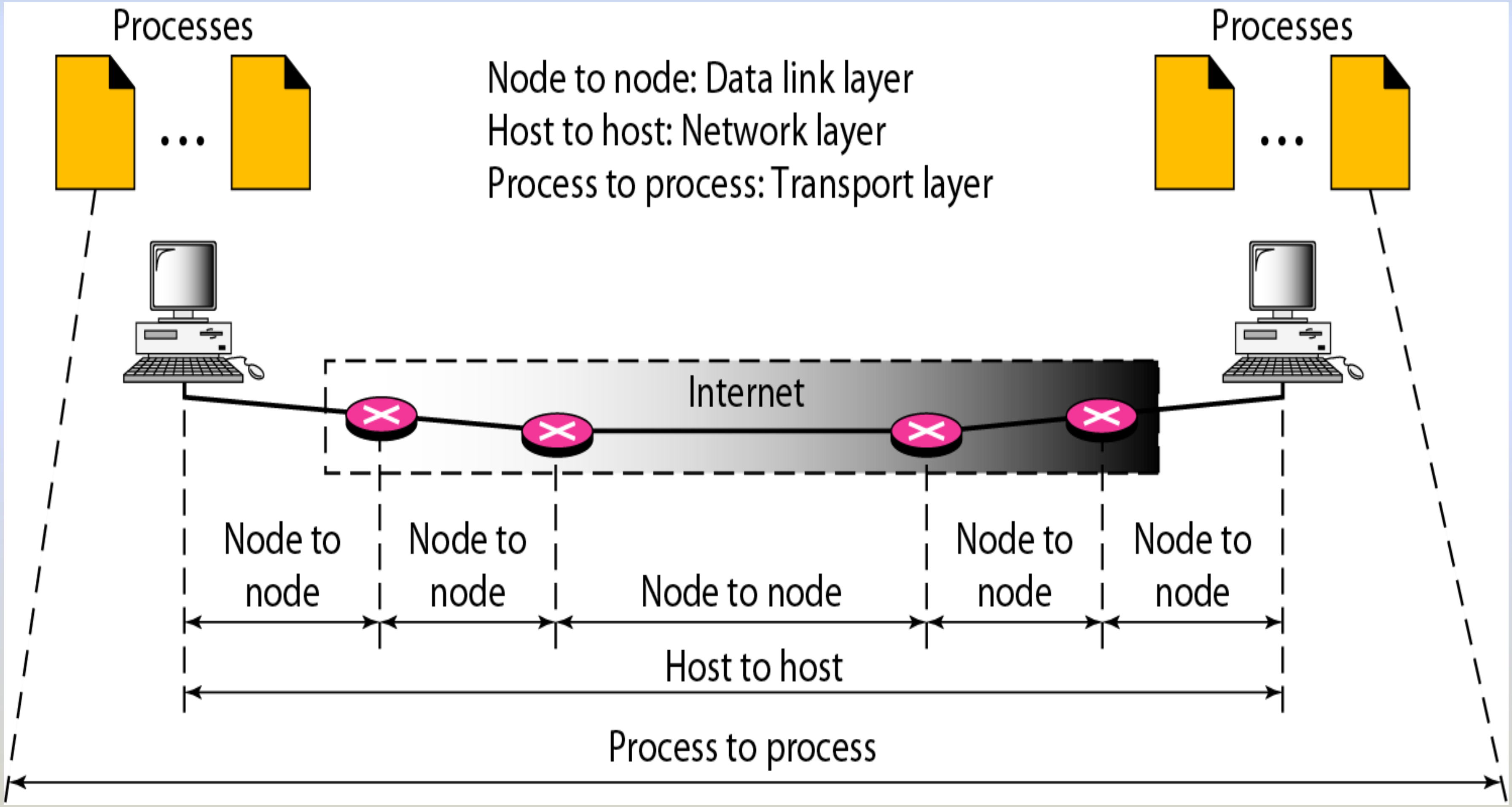
Process To Process Delivery

- Datalink Layer : node-to-node delivery (frame) (delivery of frame between two neighboring nodes over a link.
- Network Layer : host-to-host delivery (datagram)
- Transport Layer is responsible for process-to-process delivery (application Program)
  - The delivery of a packet, part of a message, from one process to another.
- Two processes communicate in a client/server relationship.

Note

The transport layer is responsible for process-to-process delivery.

Types of Data deliveries



Client /Server Paradigm

- To achieve process-to-process communication
  - Client/Server paradigm is used.
- Process on local host called **Client**, needs services from process usually on the remote host, called **Server**.
- Both processes have same name.
- Remote computer can run several server programs at the same time.
- Local host can run one or more client programs at the same time.

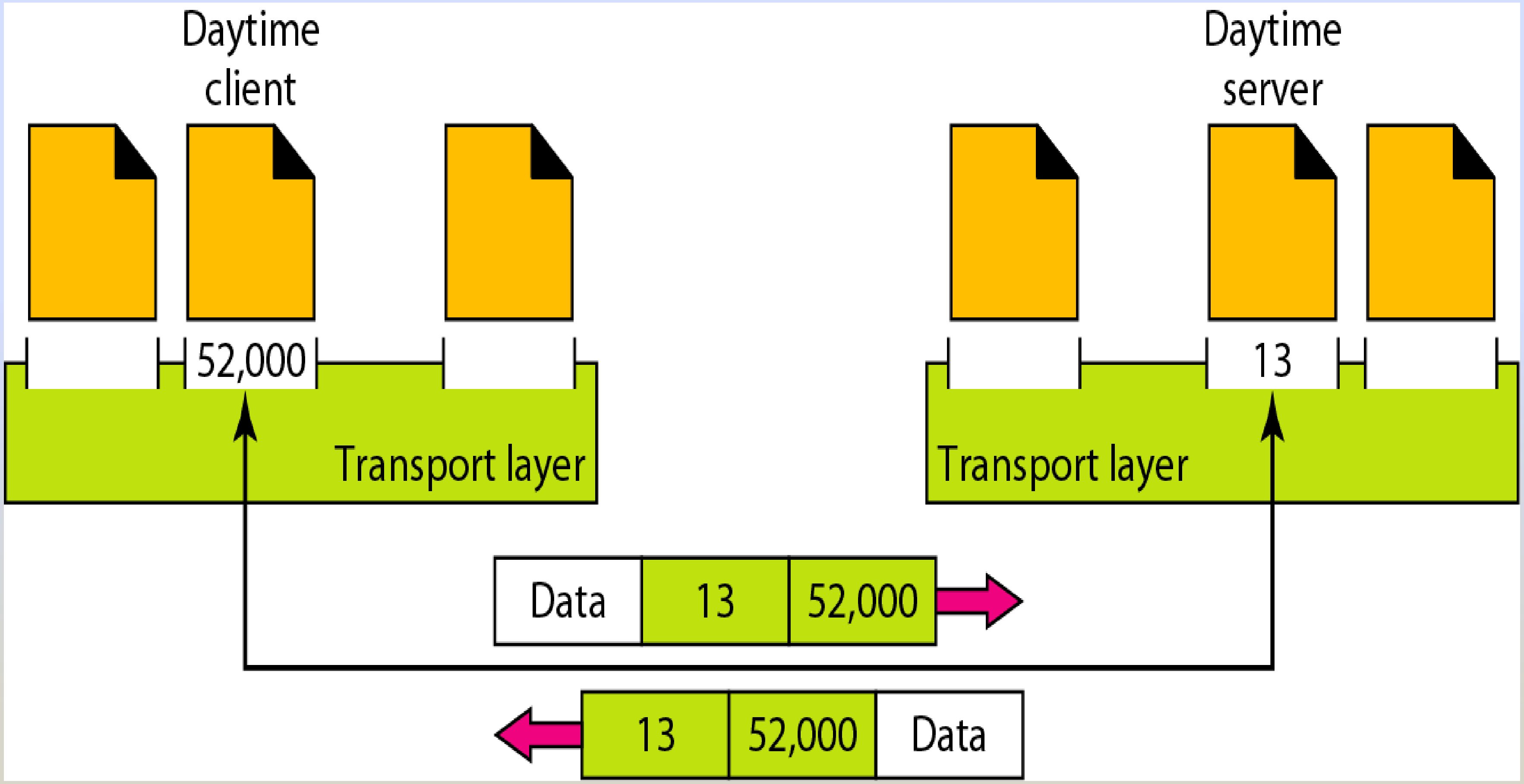
Client /Server Paradigm

- For communication we must define following :
  - Local Host
  - Local Process
  - Remote Host
  - Remote Process

Addressing

- At data link layer we need MAC address.
- At network address we need IP address.
- Transport layer address is called **port number,** to choose among multiple processes running on the destination host.
- Port number are 16-bit integer between 0-65535.
- The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host called **ephemeral port number**.
- The server port number define by itself and can not chosen randomly.
- The Internet has decided to use universal port number for servers: these are called **well-known port number**.

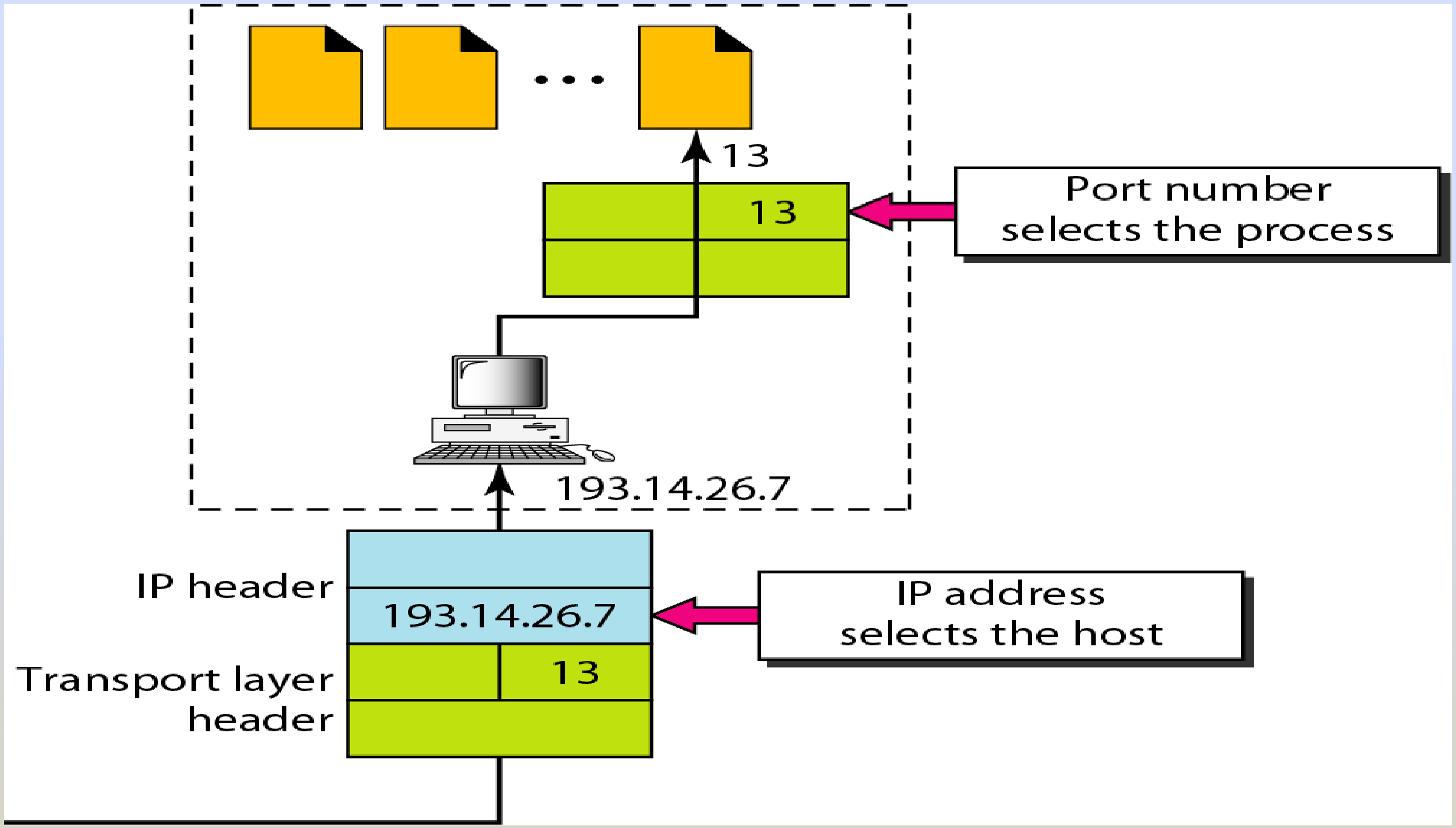
Port numbers



IANA Ranges

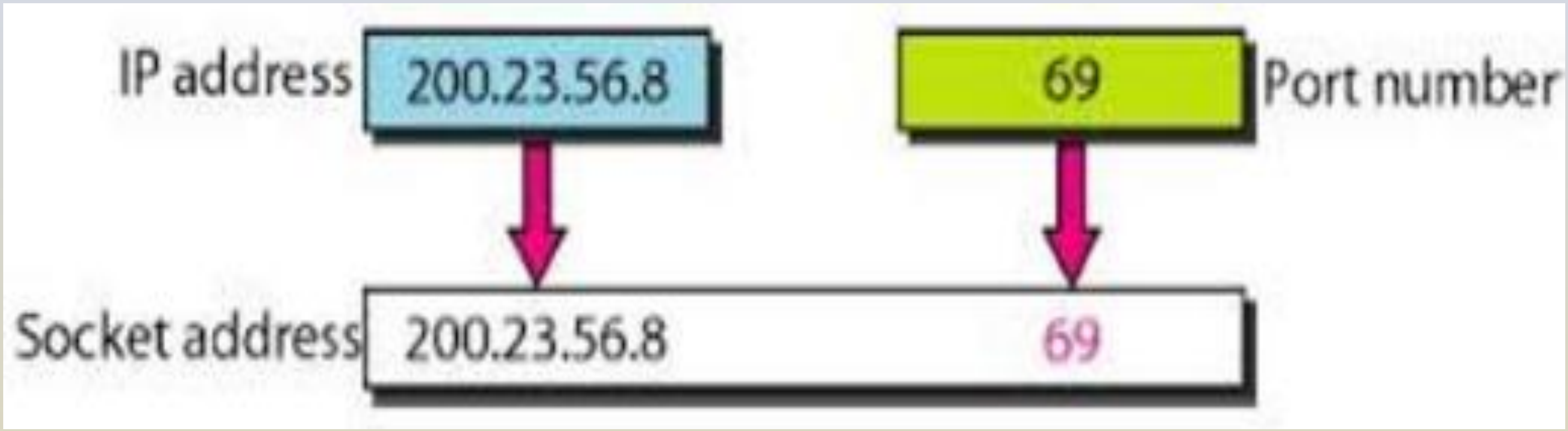
- The IANA (Internet Assigned Number Authority) has divide port number as below:
  - **Well-known ports**: the ports ranging from 0 to 1023 are assigned and controlled by IANA.
  - **Registered Ports**: the ports ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication.
  - **Dynamic ports**: the ports ranging from 49,152 to 65,535 are neither controlled not registered . They can be used by any process. These are the ephemeral ports.

IP addresses versus port numbers

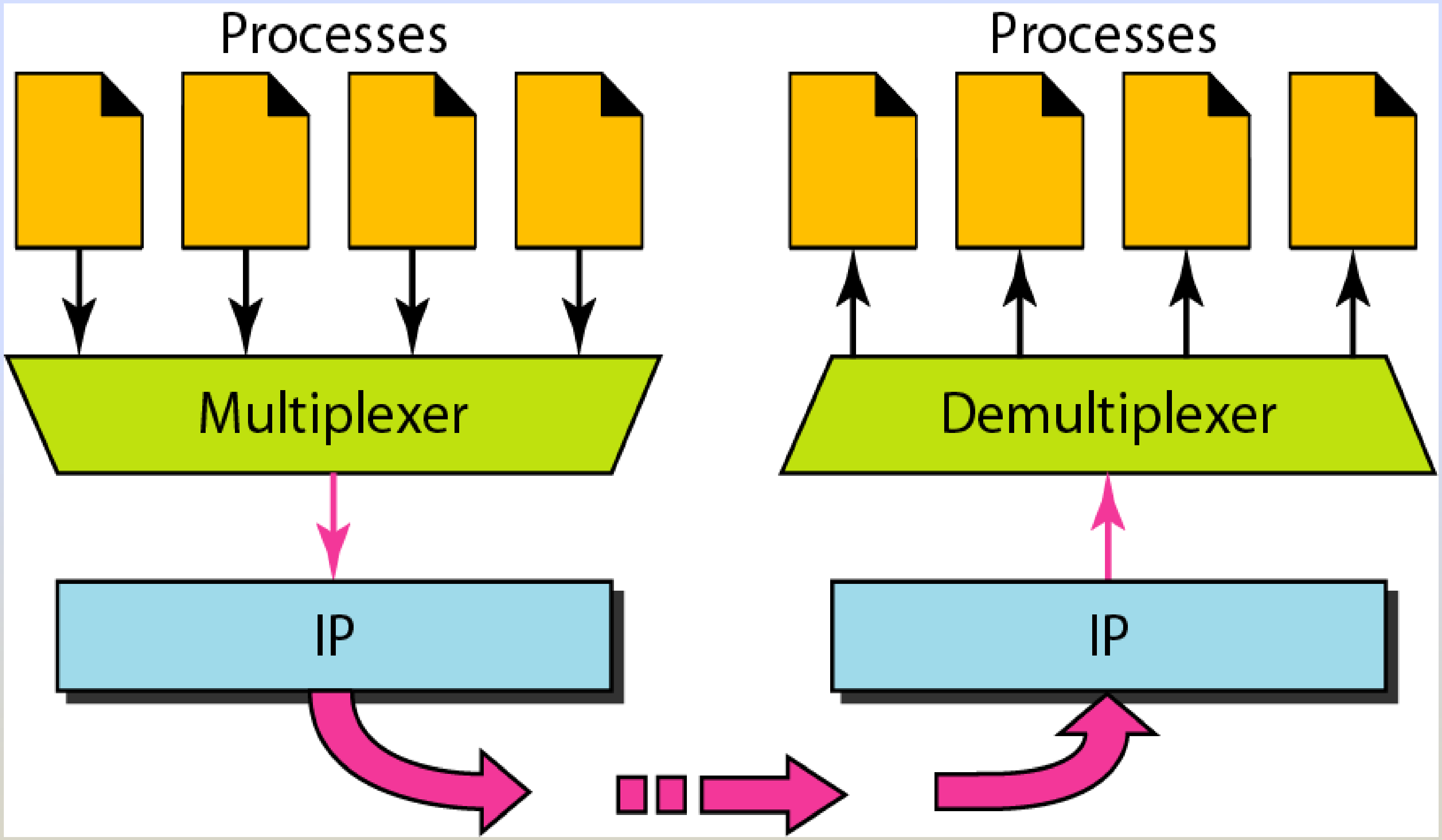


Socket addresses

- Process to process delivery needs two identifier: IP address and Port number.
- The combination of an IP address and port number is called a **socket address**.
- Transport layer protocol needs a pair of socket addresses: the client socket address and server socket address.



Multiplexing and Demultiplexing



Multiplexing and Demultiplexing

- At sender site, there may be several processes that need to send packets and there is only one transport layer protocol at any time.
- This is many-to-one relationship and requires multiplexing.
- The protocol accepts messages from different processes, differentiated by their port numbers. After adding the header, the transport layer passes the packet to the network layer.
- **At receiver side, the relationship is one-to-many and requires demultiplexing.**
- **The transport layer receives datagrams from the network layer. After error checking and dropping of header, the transport layer delivers each message to appropriate process based on the port number.**

Connectionless vs. Connection-Oriented Service

- **Connectionless Service**
  - Packet are sent from one party to another with no need for connection establishment or connection release.
  - Packet are not numbered: they may be delayed or lost or may arrive out of sequence.
  - There is no acknowledgement either.
  - UDP is connectionless protocol.

Connectionless vs. Connection-Oriented Service

- **Connection-Oriented Service**
  - In this service connection is first established between the sender and receiver.
  - Data are transferred.
  - At the end, the connection is released.
  - TCP and SCTP are connection Oriented Protocol.

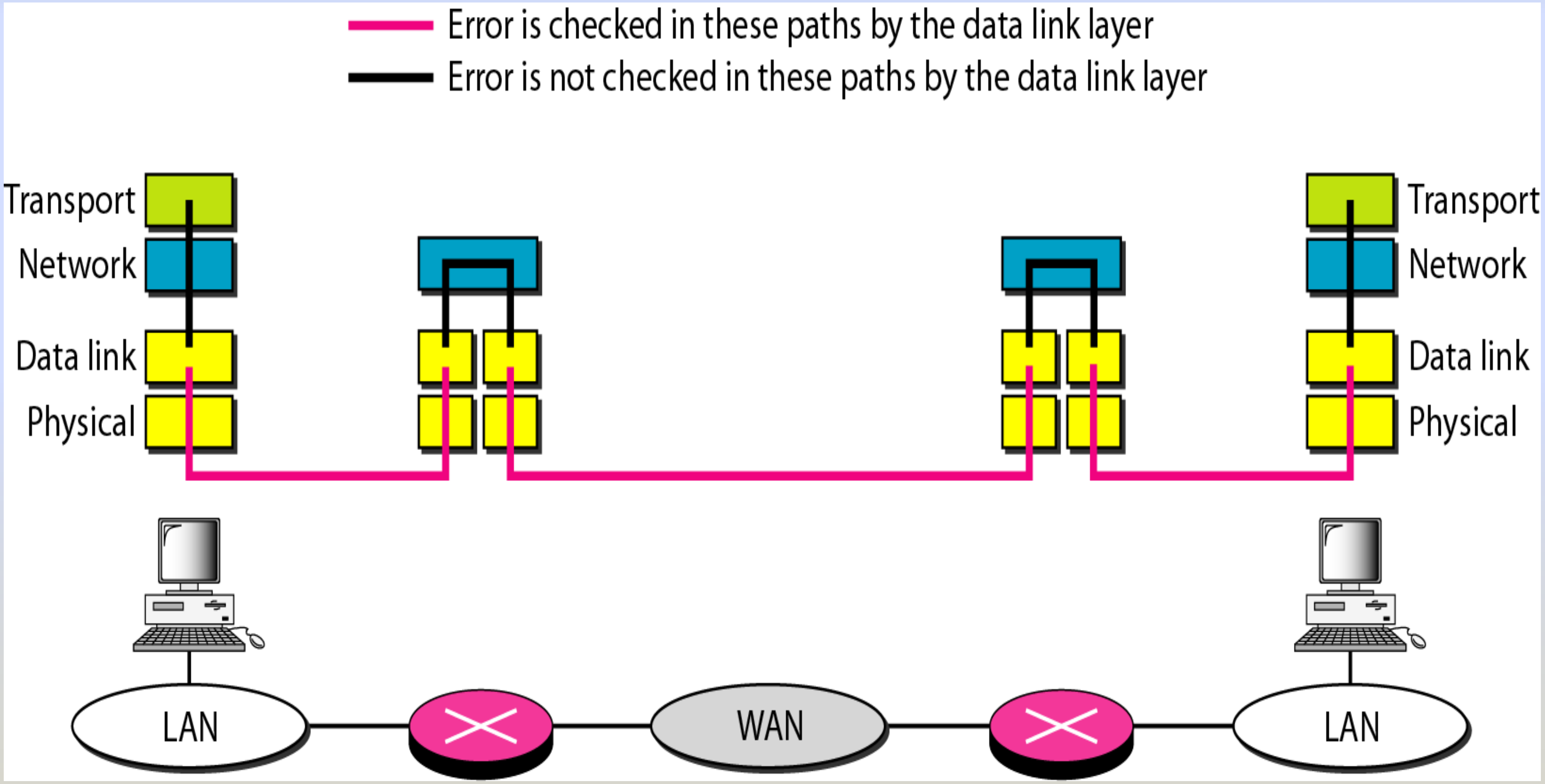
Reliable vs. Unreliable

- **Reliable**
  - If the application layer program needs reliability, reliable transport layer protocol is used by implementing flow and error control.
  - This is slower and more complex service.
  - TCP is connection oriented and reliable.

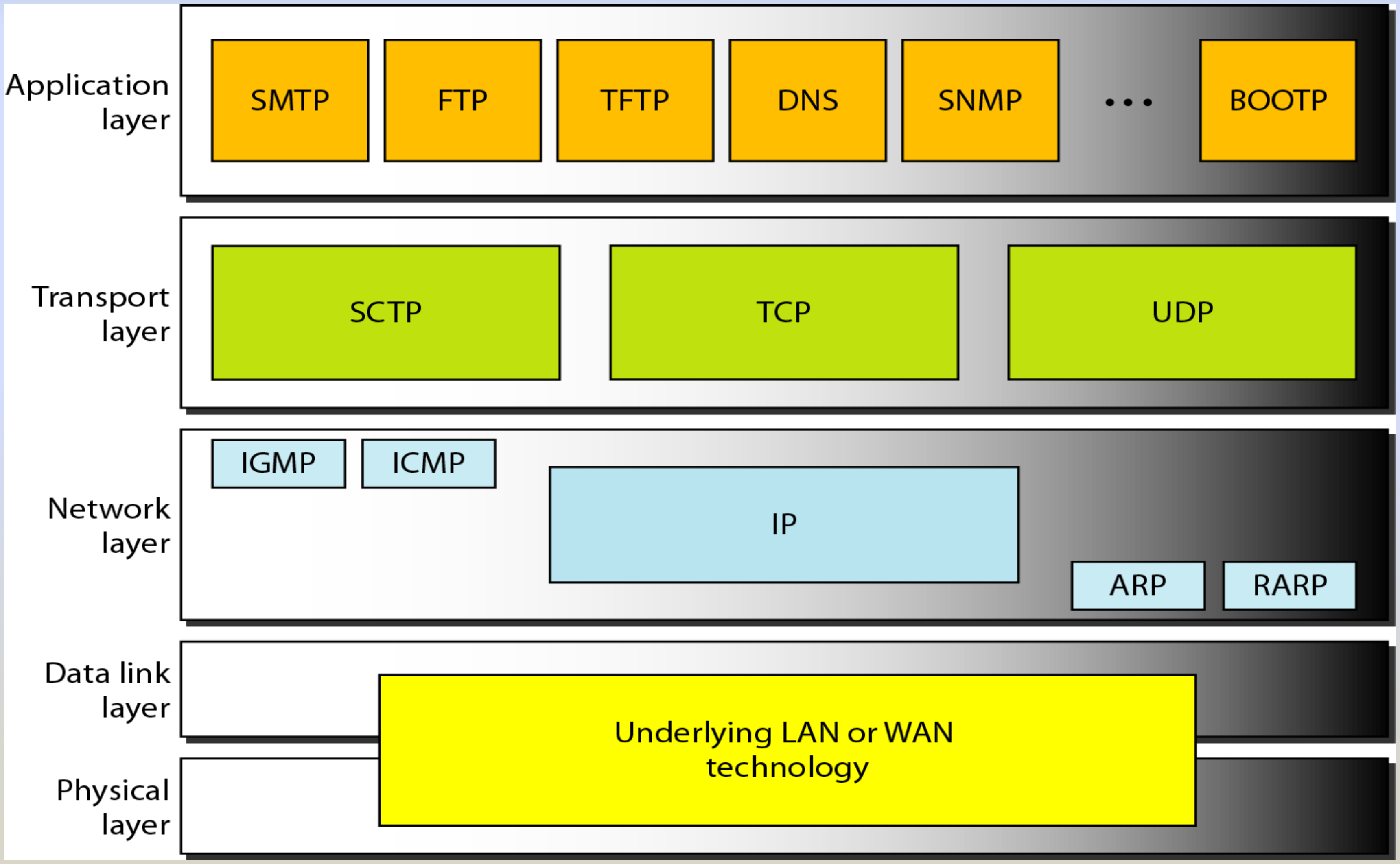
Reliable vs. Unreliable

- **Unreliable**
  - If the application layer program does not needs reliability, because it uses its own flow and error control mechanism.
  - It needs fast service or the nature of the service does not demand flow and error control, then unreliable service can be used.
  - UDP is connection less and unreliable.

Error control


Error is checked in these paths by the data link layer
Error is not checked in these paths by the data link layer
Transport
Network
Data link
Physical
Transport
Network
Data link
Physical
LAN
WAN
LAN

Position of UDP, TCP, and SCTP in TCP/IP suite



UDP : USER DATAGRAM PROTOCOL

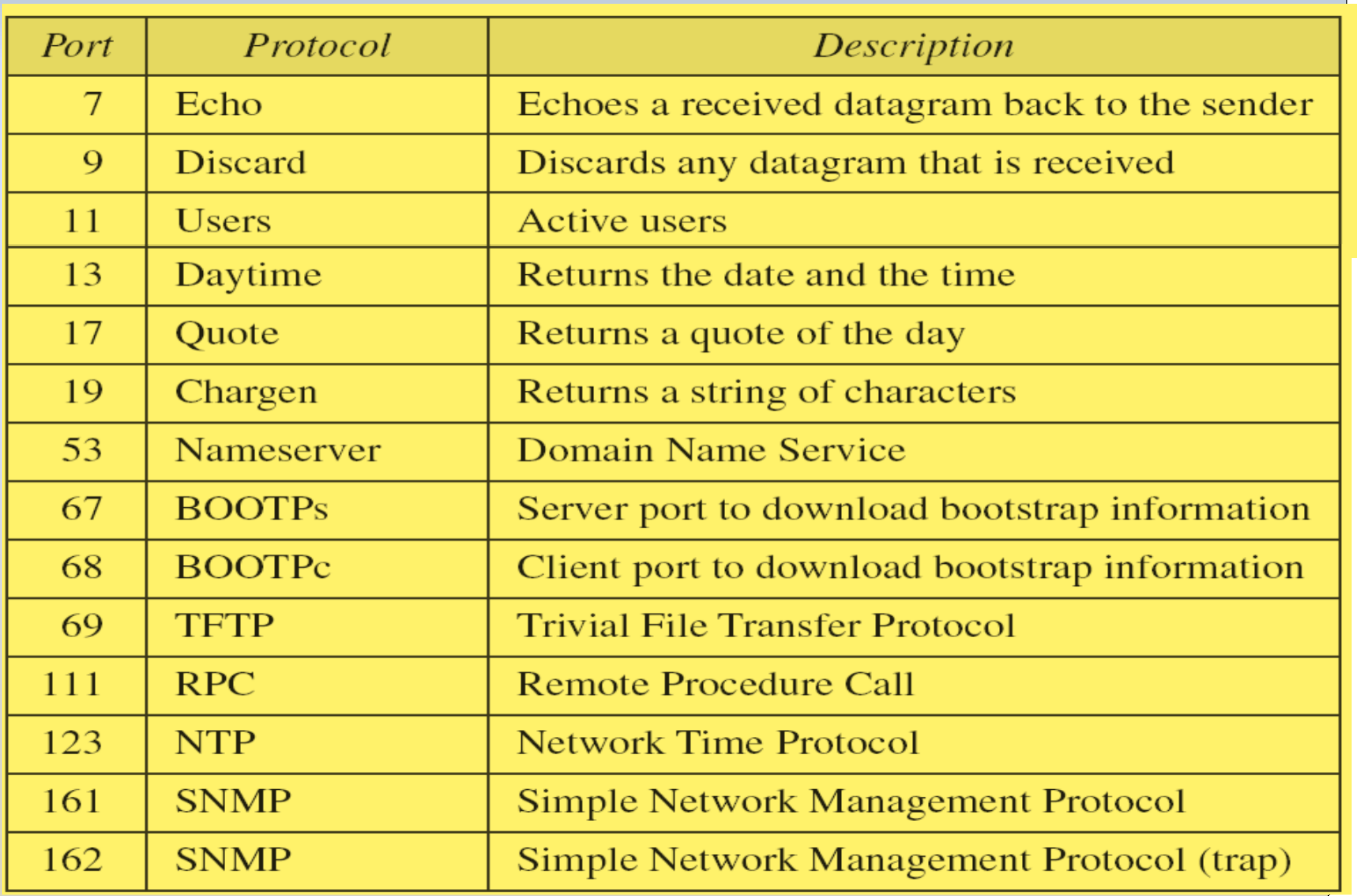
- UDP is called **connectionless** and **unreliable** transport protocol.
- It does not add anything to services of IP except to provide process-to-process communication instead of host-to-host communication.
- It perform very limited checking.
- **Disadvantage :**
  - UDP is so powerless.

Advantages of UDP

Why would a Process want to use UDP?

- UDP is a very simple protocol using very minimum overhead.
- If process want send very small message and does not care about reliability, it better to use UDP.
- Sending a small message by using UDP takes less interaction between the sender and receiver than using TCP.

Well-Known Port for UDP



Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

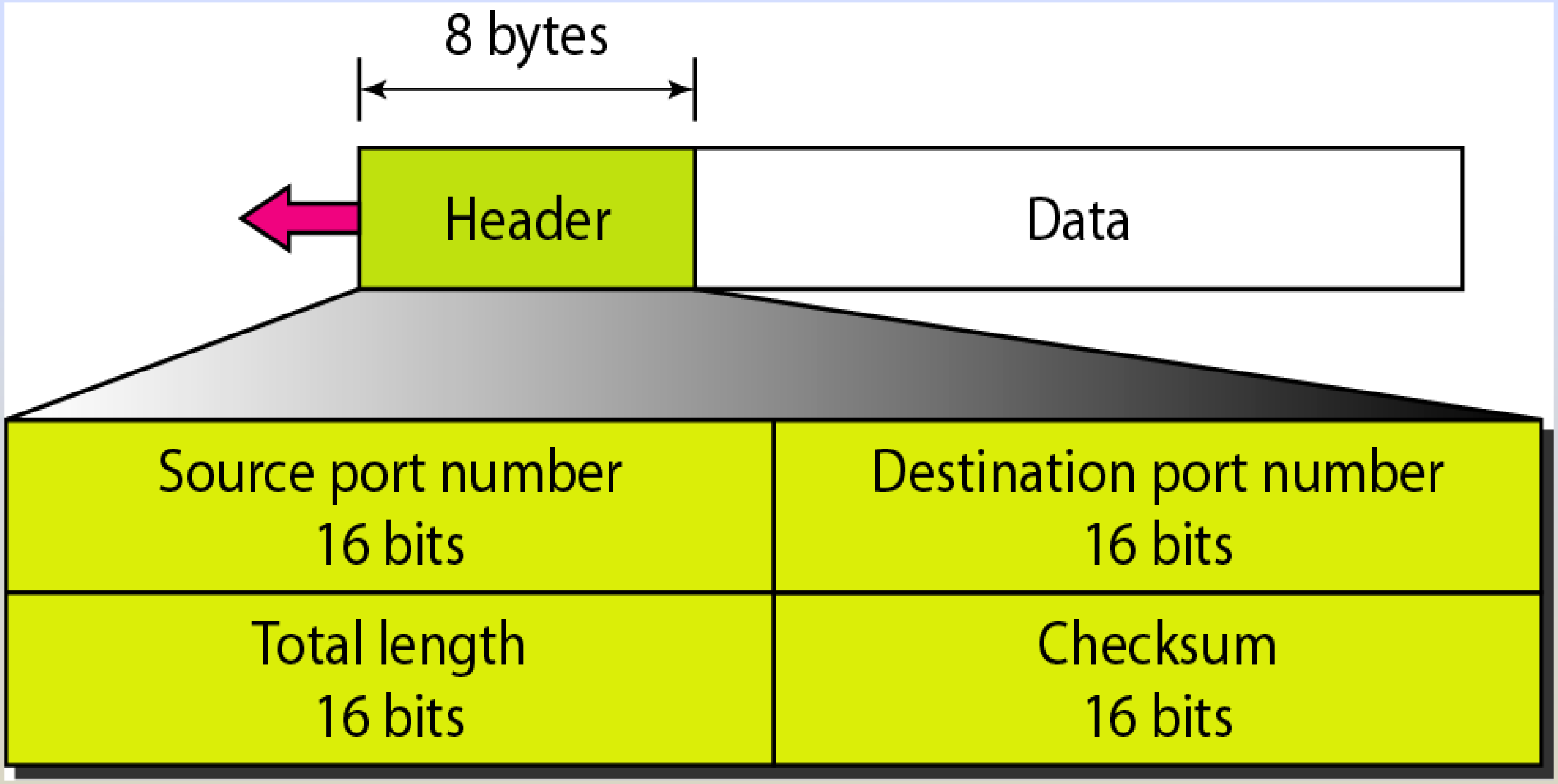
Example

```
$ grep      ftp    /etc/services
ftp              21/tcp
ftp              21/udp
```

- In UNIX, the well-known ports are stored in a file called /etc/services.
- Each line in this file gives the name of the server and the well-known port number.
- We can use the grep utility to extract the line corresponding to the desired application.
- The Example shows the port for FTP.
- Note that FTP can use port 21 with either UDP or TCP.

User Datagram

- UDP packet called User datagram, have a fixed size header of 8 bytes.
- Figure shows the format of user datagram.



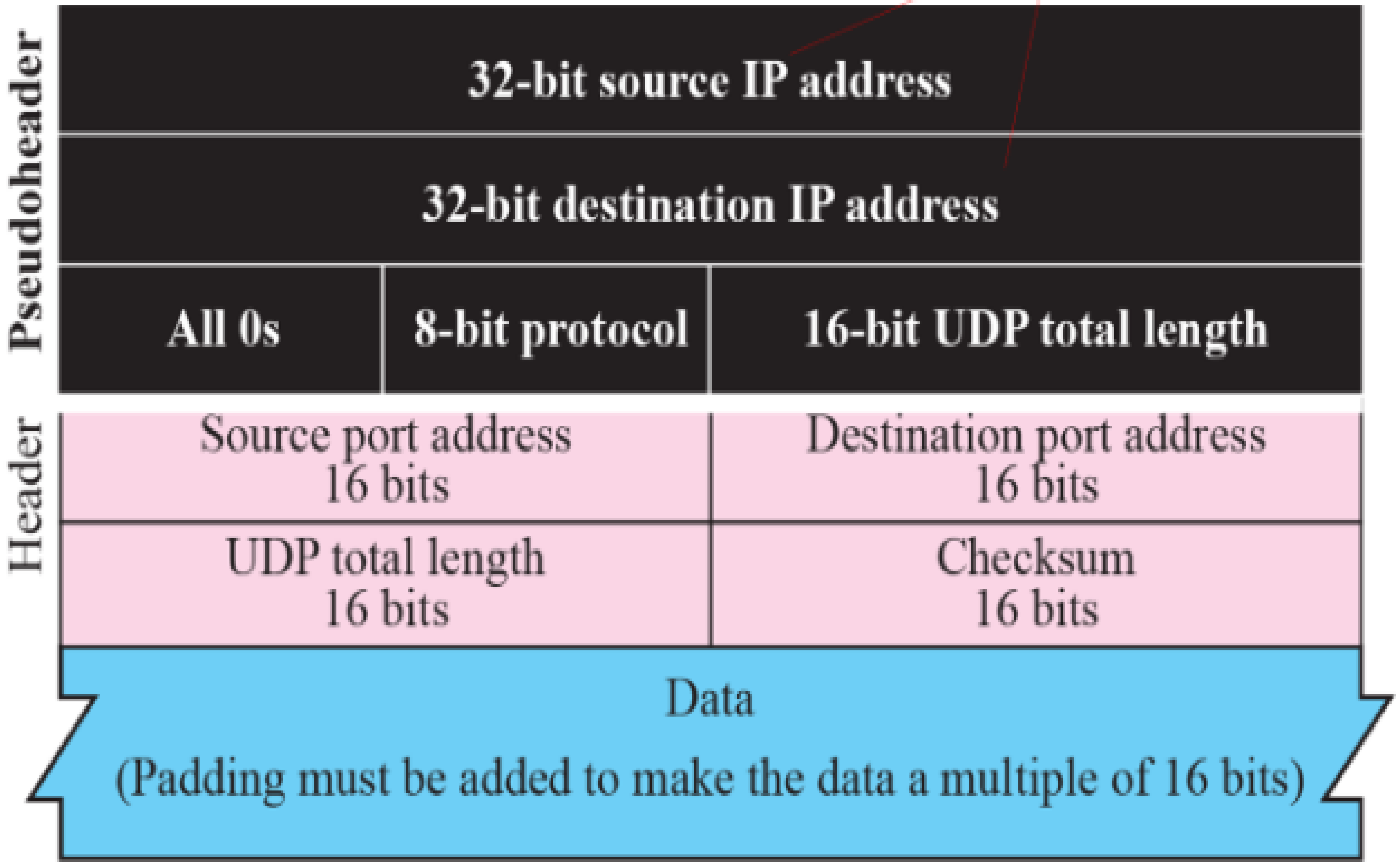
User Datagram Format

- **Source port number :**
  - Port number used by the process running on the source port.
  - Range :0 to 65535.
  - Source is client : ephemeral port, source host : well-known port.
- **Destination port number**
  - Port number used by the process running on the destination port.
  - If server : client sending request.
  - If client : server sending a response.
- **Length**
  - 16 bit length : total length of user datagram, header + Data.

UDP length = IP length – IP header's length

- **Checksum**
  - This field used to detect error.

Pseudoheader for checksum calculation


Pseudoheader
32-bit source IP address
32-bit destination IP address
All 0s
8-bit protocol
16-bit UDP total length
Header
Source port address 16 bits
Destination port address 16 bits
UDP total length 16 bits
Checksum 16 bits
Data
(Padding must be added to make the data a multiple of 16 bits)

UDP Operation

- **Connectionless service**
  - Each user datagram sent by UDP is an independent datagram.
  - No connection established.
- **Flow control and Error control.**
  - No error and flow control mechanism except checksum.
  - Sender does not know if a message has been lost or duplicated, When receiver detect the error using checksum, user datagram discarded.
- **Encapsulation and Decapsulation**
  - To send a message from one process to another, the UDP encapsulate and Decapsulation message in IP.
- **Queuing**
  - If process want to communicate with multiple processes, it obtain only one port number one outgoing and one incoming queue.

Use of UDP

- UDP suitable for process that required simple request-response communication.
- UDP suitable for process with internal flow and error control
- UDP used for management process
- Used for multicasting.(multicasting capability is embedded in the UDP.
- Used for some route updating protocols.(RIP)

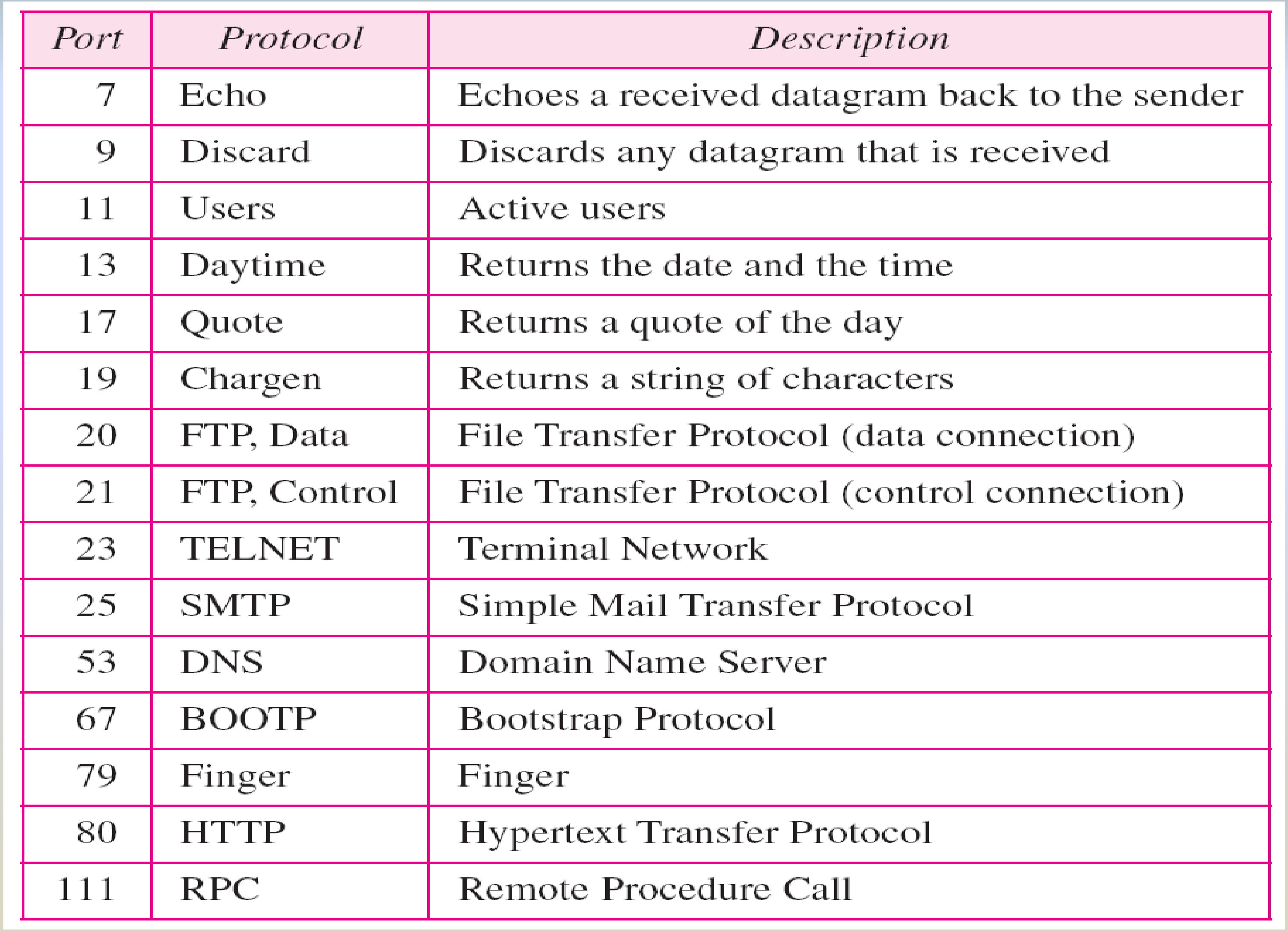
TCP : Transmission control protocol

- TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data.
- In addition, TCP uses flow and error control mechanisms at the transport level.
- It is a connection-oriented, reliable transport protocol.

Topics discussed in this section:

TCP Services
TCP Features
Segment
A TCP Connection
Flow Control
Error Control

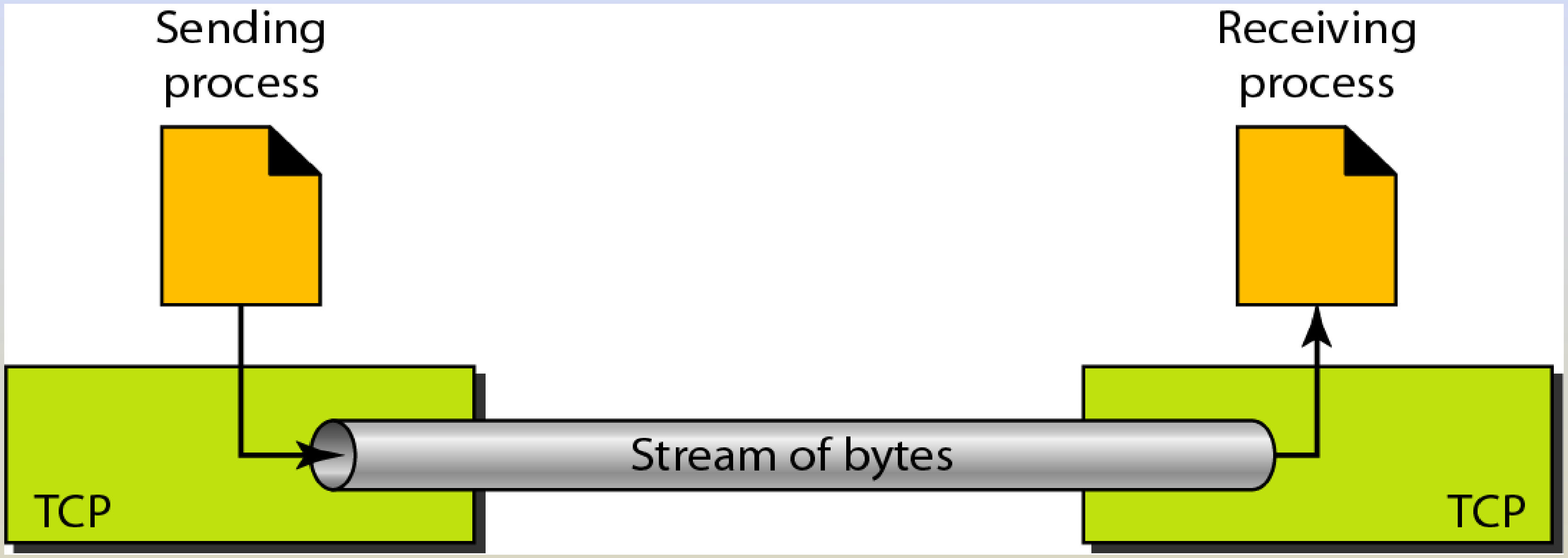
Process to Process Communication



Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Stream Delivery Service

- Allow sending process to deliver data as a stream of bytes and allow receiving process to obtain data as a stream of bytes.
- TCP creates an environment in which 2 process are seem to be connected by imaginary tube that carries the data.
- Sending process produces the stream of bytes and the receiving process consume them.

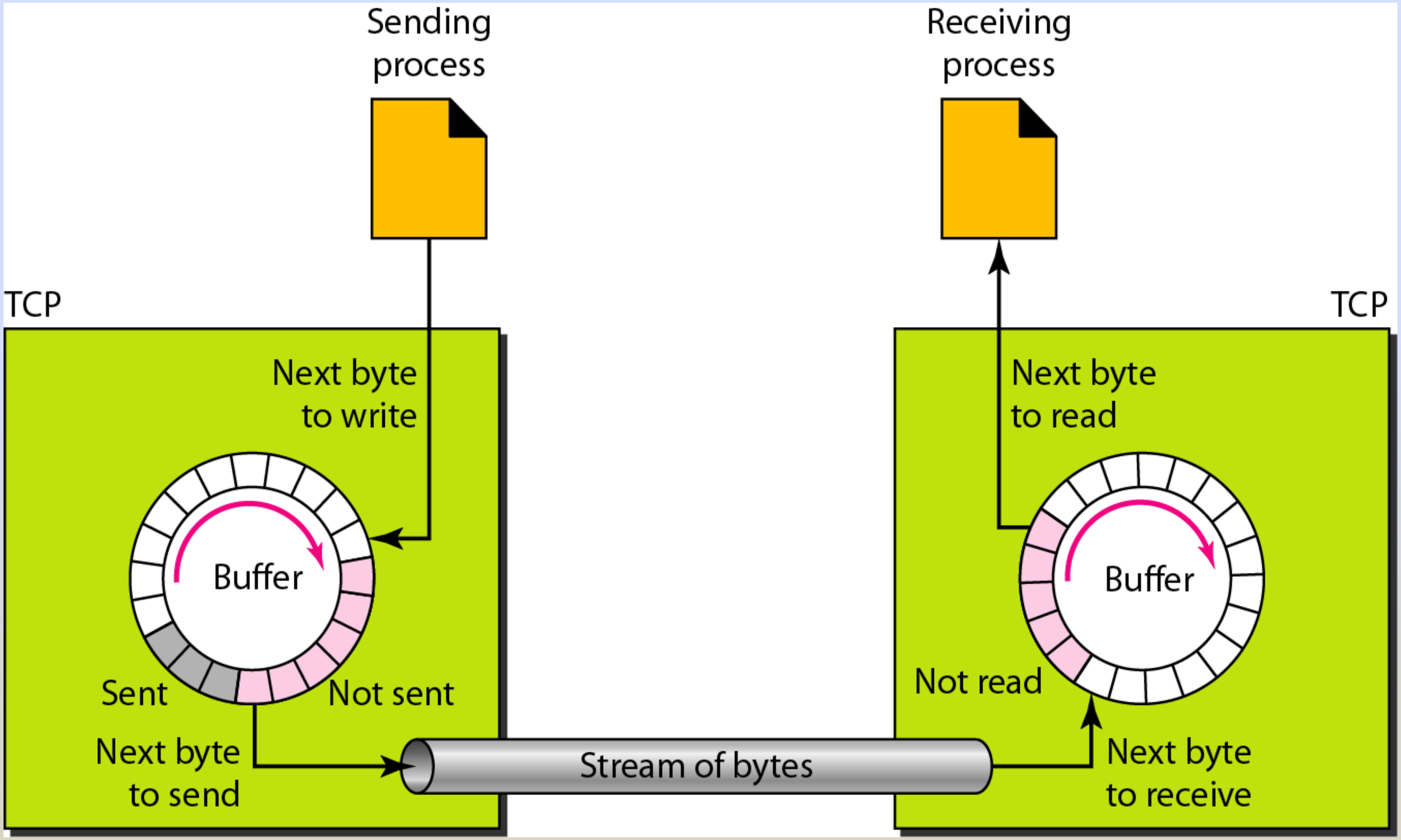


Sending and Receiving Buffer

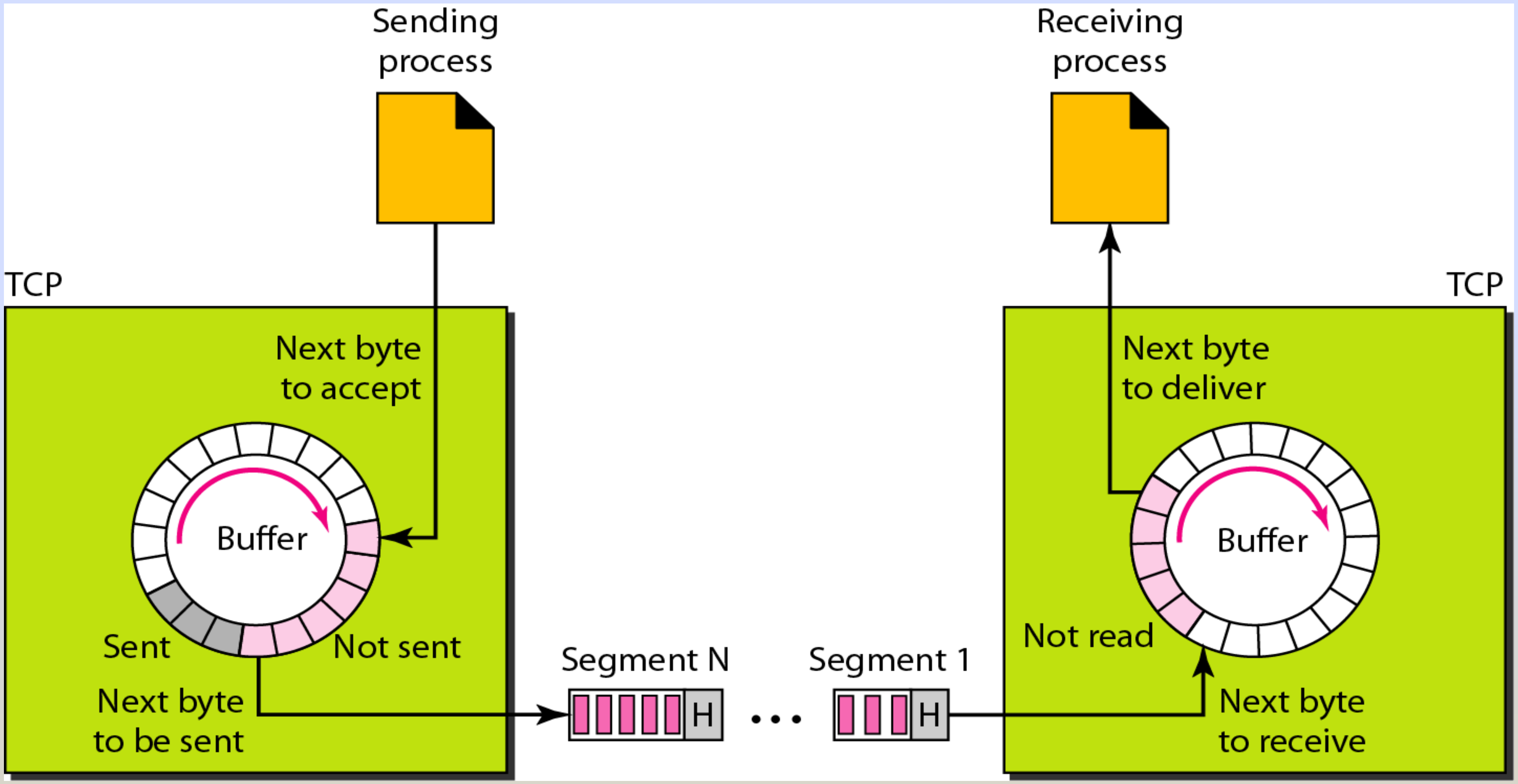
- Sending and receiving process may not write and read data at the same speed, TCP needs buffer for storage.
- There are two buffer, the sending and receiving buffer.
- Implementation of buffer is to use a circular array of 1-byte.
- Sending Side Three types of Chambers :
  - **White Section :** contains empty chambers that can be filled by the sending process.
  - **Gray Section :** Holds the byte that have been send but not acknowledged.
    - TCP keeps these byte in buffer until acknowledgement is receive.
  - **Color Section :** It contains byte to be sent by sending TCP.

- Receiving Side 2 types of Chambers :
  - **White Section :** contains empty chambers that can be filled by the bytes receive from network.
  - **Color Section :** contains received bytes that can be read by the receiving process.
    - When byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

Sending and Receiving Buffer



Segments


Sending process
Receiving process
TCP
TCP
Next byte to accept
Next byte to deliver
Buffer
Buffer
Sent
Not sent
Not read
Next byte to be sent
Next byte to receive
Segment N
Segment 1
H
H
• • •

Full Duplex Communication

- Connection Oriented Service
  - 2 TCPs established a connection between them
  - Data are exchange in both direction.
  - Connection is terminated.
- Reliable Service
  - It use acknowledgement mechanism to check the safe and sound arrival of data.

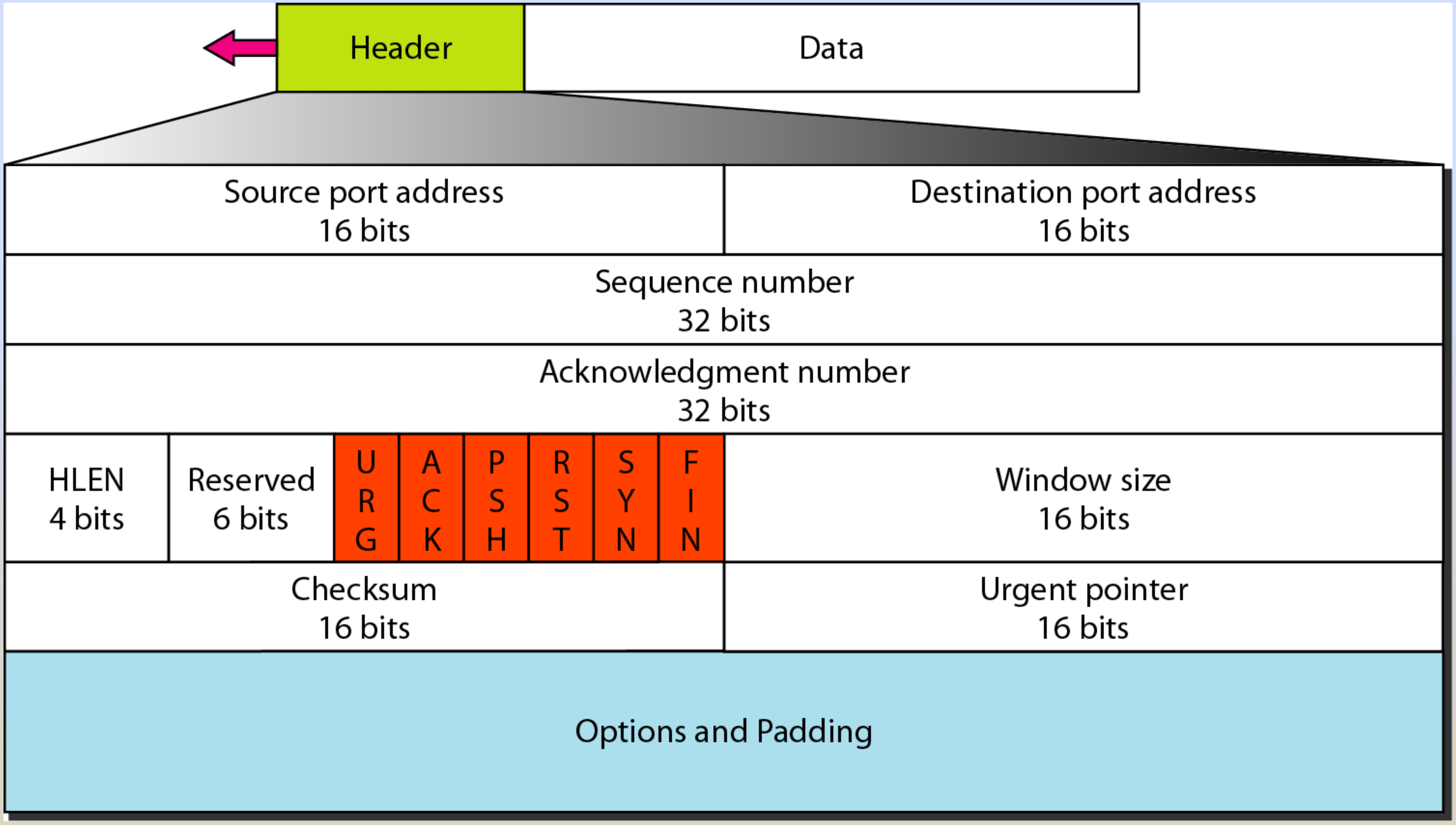
TCP Features : Numbering System :

- There are 2 number Sequence number and Acknowledgement number :
- **Byte Number:**
  - The bytes of data being transferred in each connection are numbered by TCP.
  - The numbering starts with a randomly generated number.
- **Sequence Number :**
  - After bytes have been numbered, TCP assign sequence number to each segment that being sent.
  - The sequence number for each segment is the number of first byte carried in segment.
- **Acknowledgment Number :**
  - The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receiver

- Flow Control
- Error Control

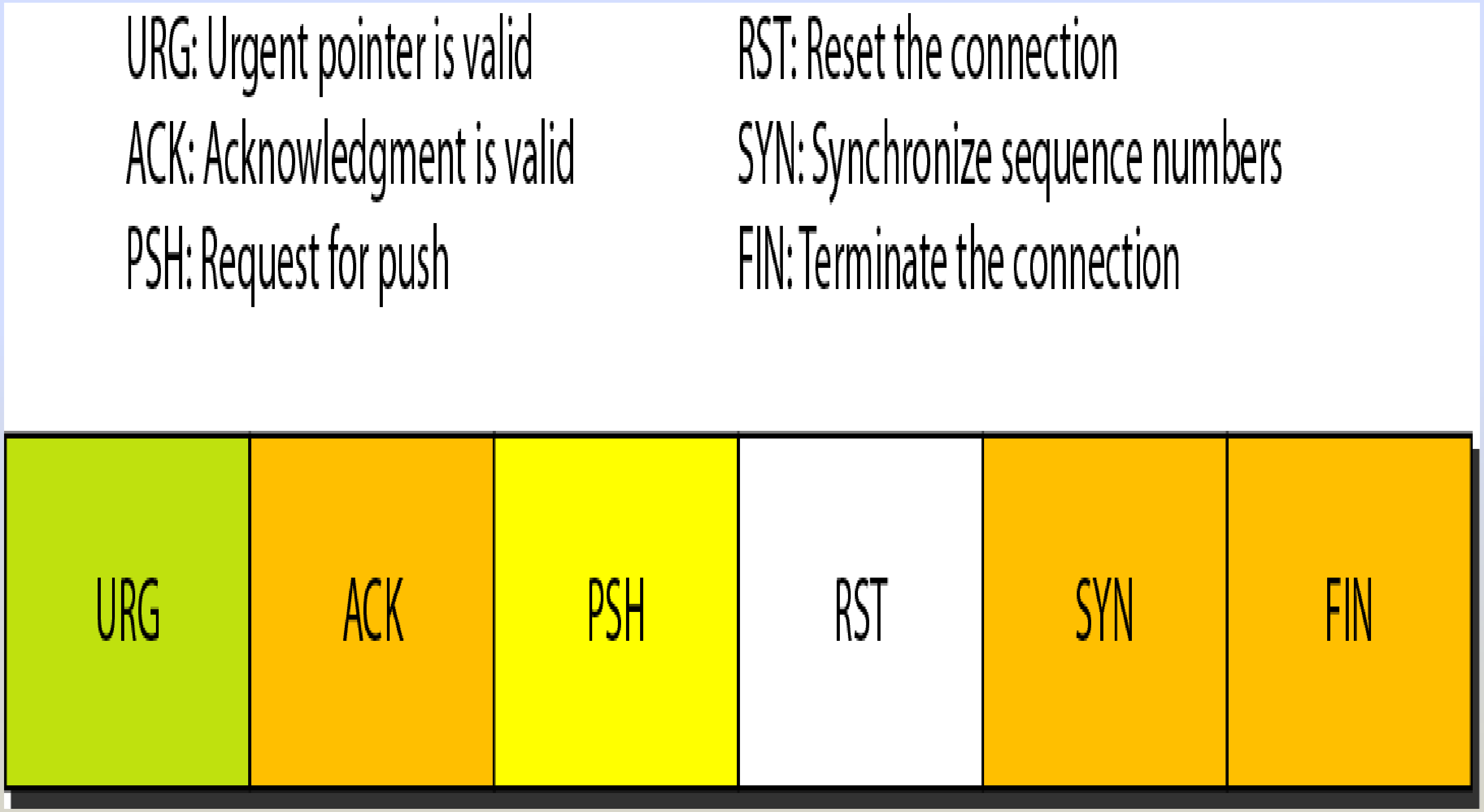
Segment: Format

- Packet in TCP is called **Segment**.



- **20 to 60 bytes of header**
- **Source Port Address :** port number of sender segment.
- **Destination Port Address:** port number of receiving segment.
- **Sequence Number :** number assign to first byte of data in segment.
- **Acknowledgment Number :**byte number that the receiver of the segment is expecting to receive from other party.
- **Header Length :** Number of 4-bit word in the TCP header.
- **Reserved :** reserved for future use.
- **Control :** 6 different control bit or flag.
- **Window size :** define size of the window.
- **Checksum :** contain checksum : calculation same as UDP calculation.

Control Field


URG: Urgent pointer is valid

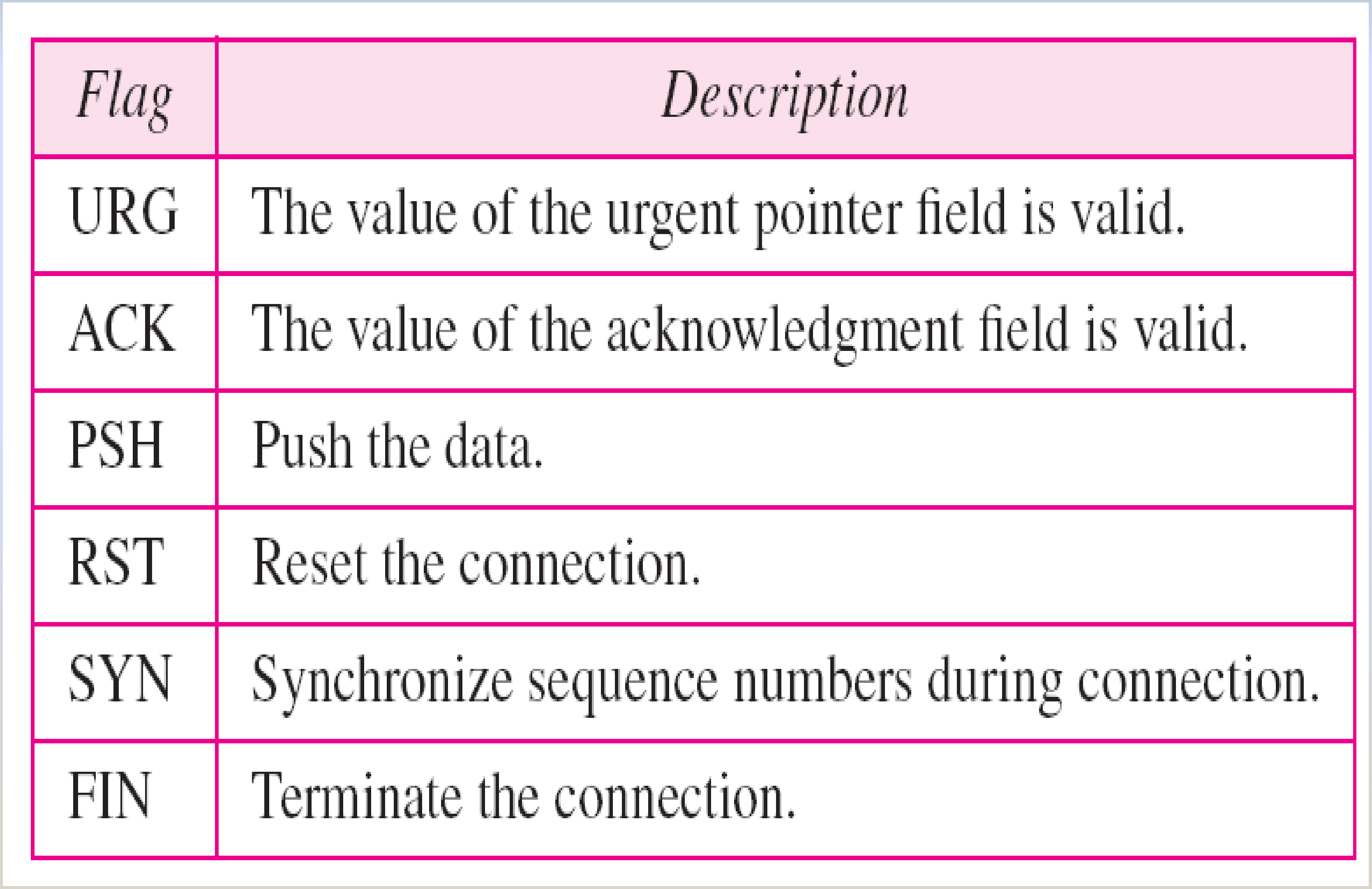
ACK: Acknowledgment is valid

PSH: Request for push

RST: Reset the connection

SYN: Synchronize sequence numbers

FIN: Terminate the connection

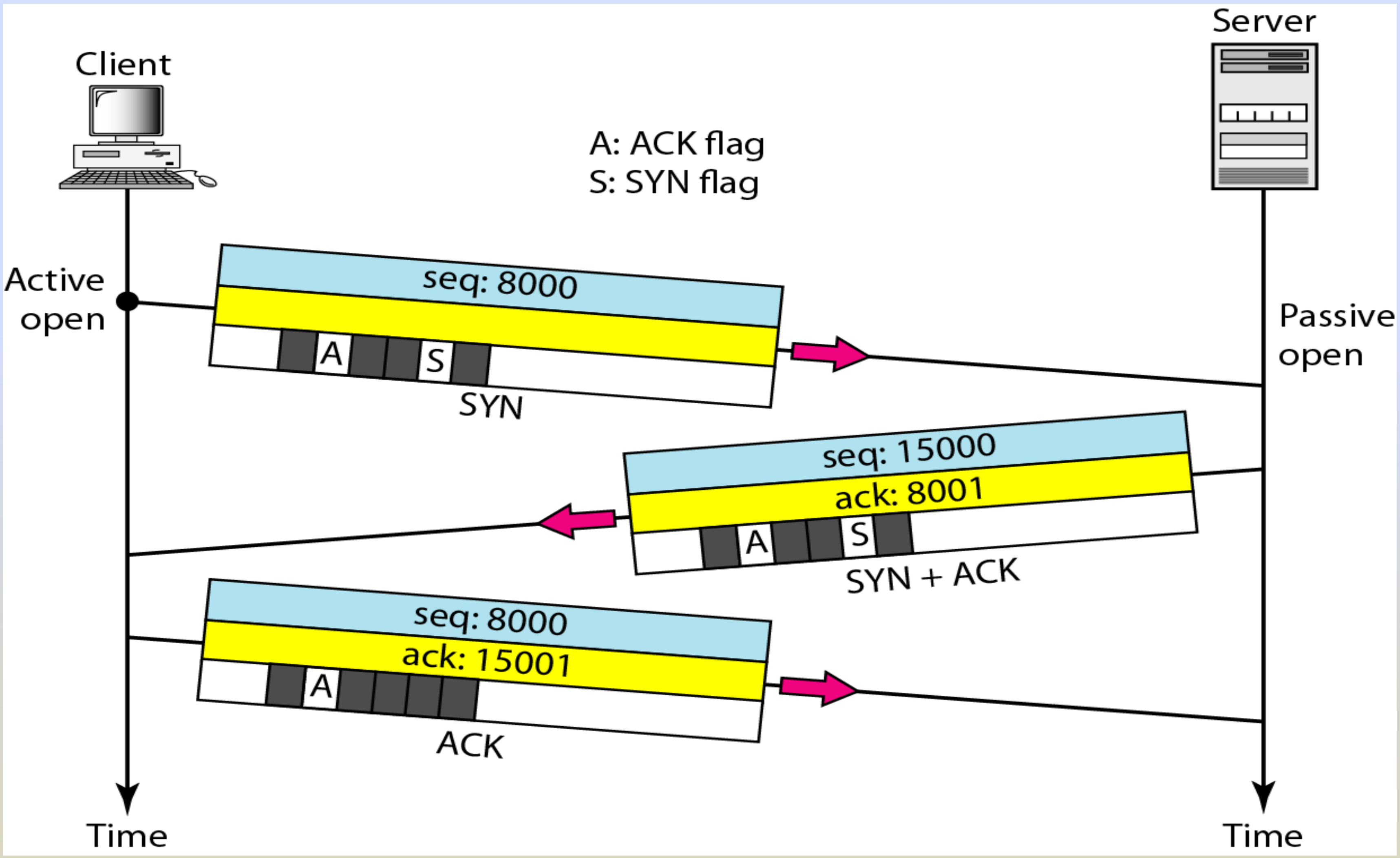


Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

TCP Connection

- Connection oriented require three phases
  - **Connection Establishment**
    - Three Way Handshaking
  - **Data transfer**
    - After connection established,
  - **Connection Termination**
    - Three way Handshaking

Connection establishment using three-way handshaking



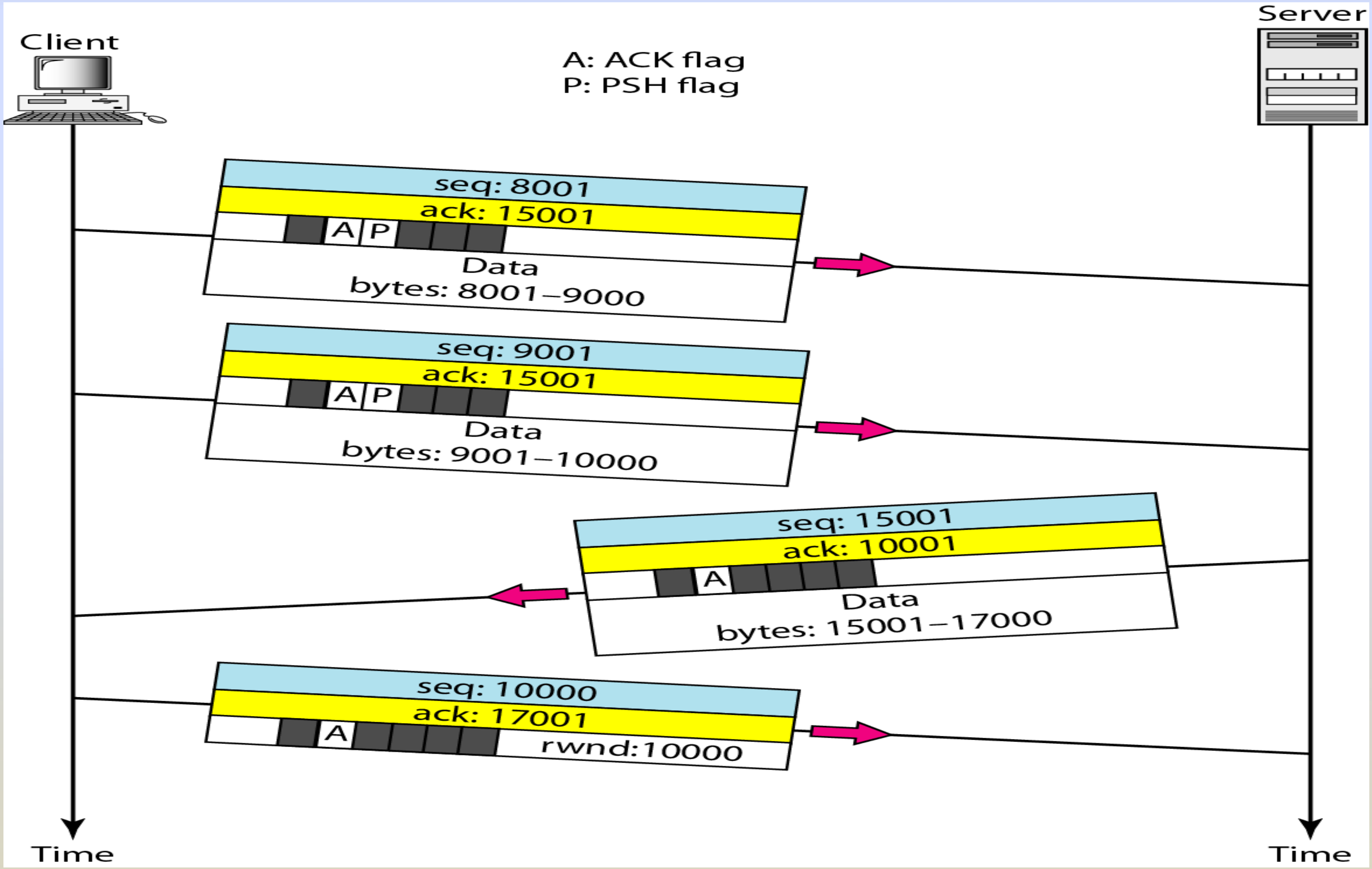
Note

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

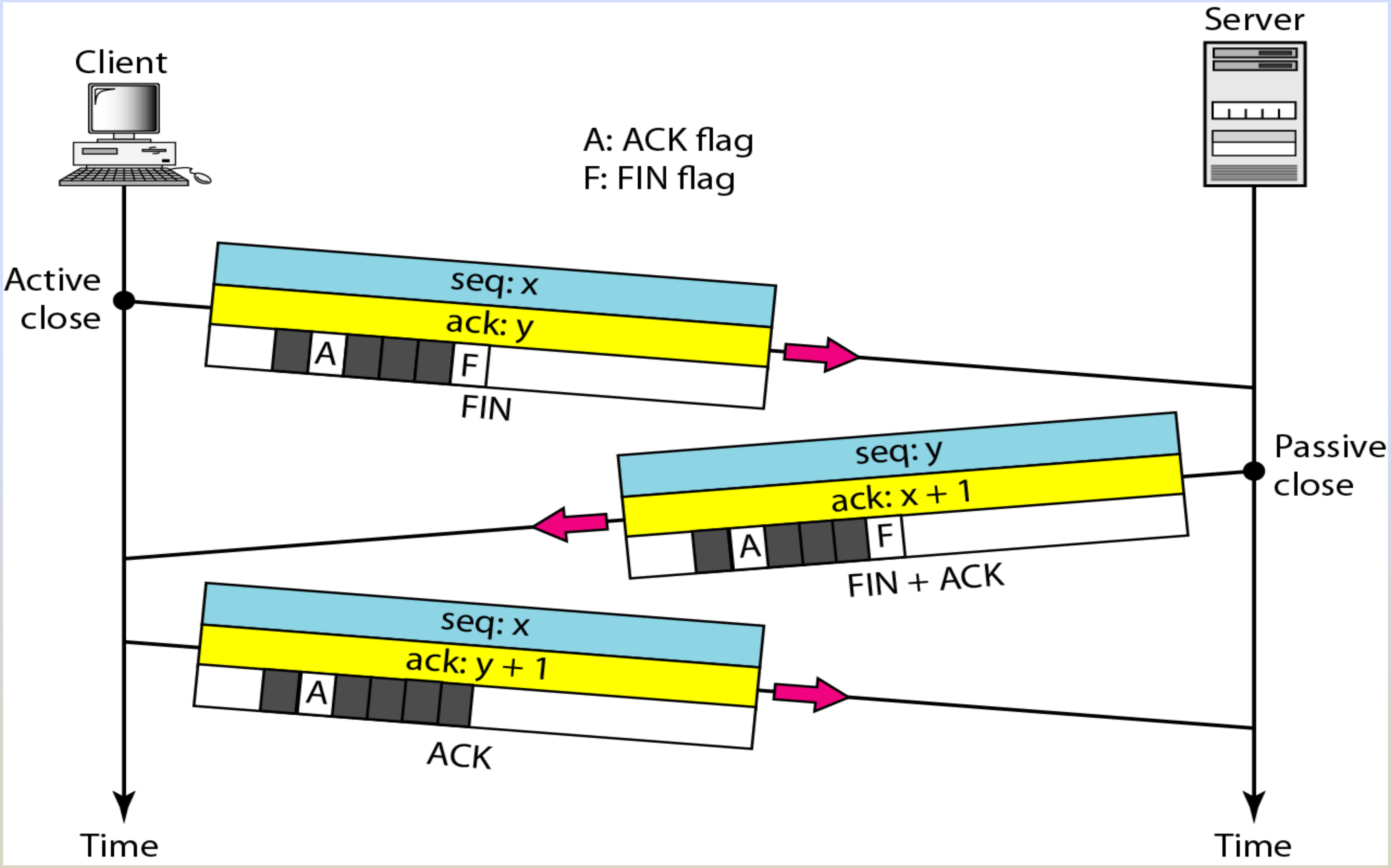
A SYN + ACK segment cannot carry data, but does consume one sequence number.

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

Data Transfer



Connection termination using three-way handshaking

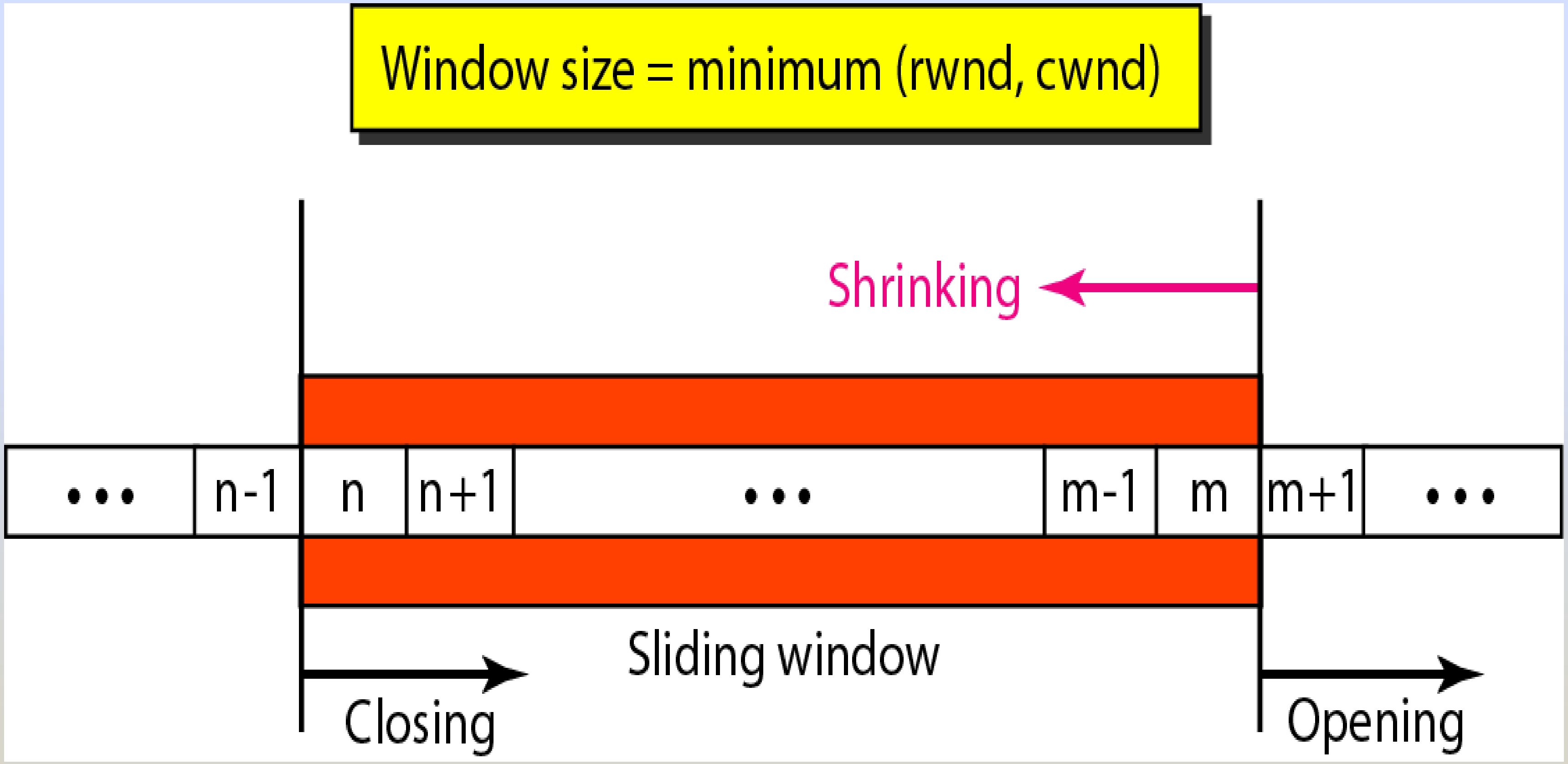


Note

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.

Flow Control



Note

A sliding window is used to make transmission more efficient as well as
to control the flow of data so that the destination does not become
overwhelmed with data.
TCP sliding windows are byte-oriented.

Note

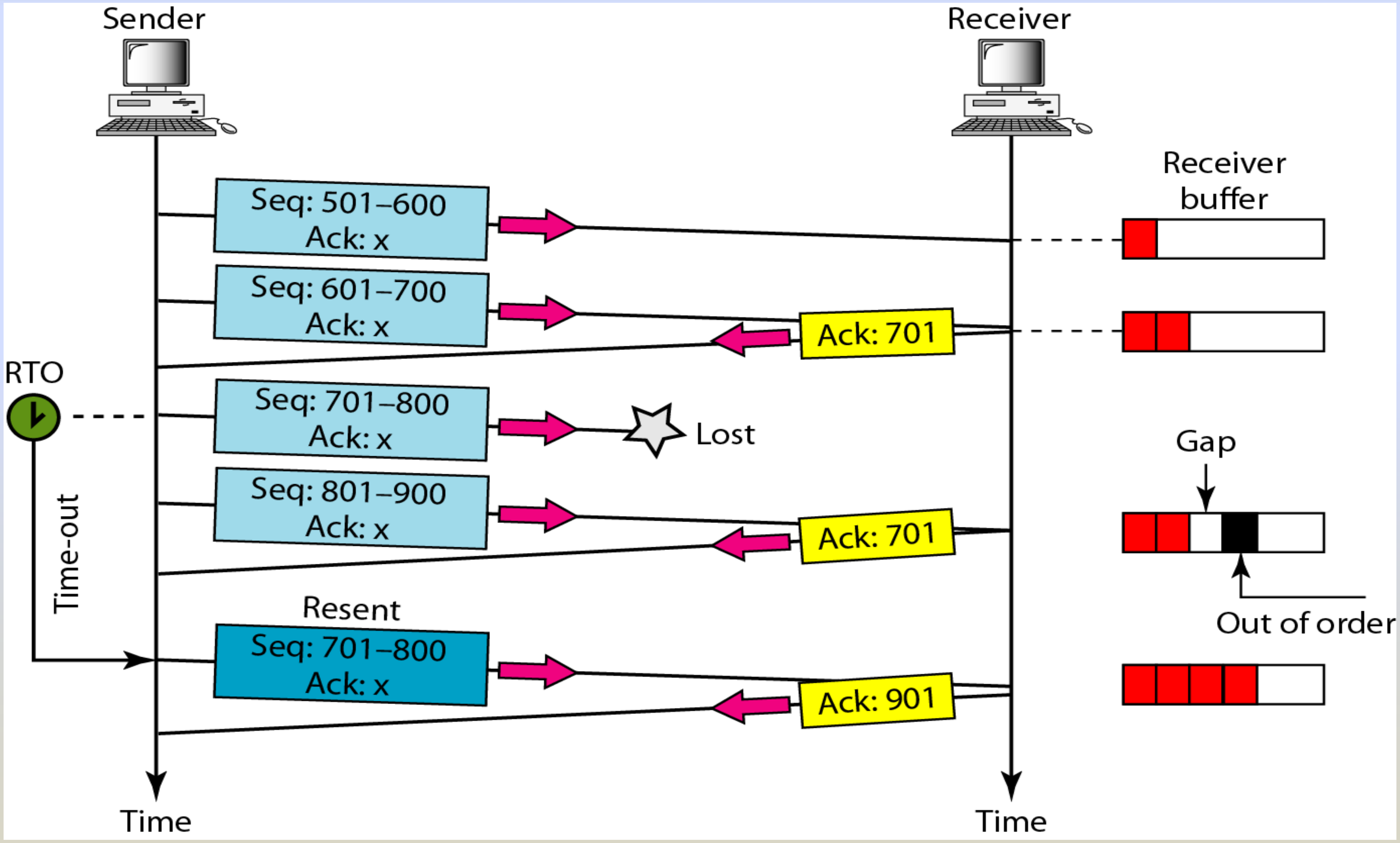
Some points about TCP sliding windows:

- The size of the window is the lesser of rwnd and cwnd.
- The source does not have to send a full window’s worth of data.
- The window can be opened or closed by the receiver, but should not be shrunk.

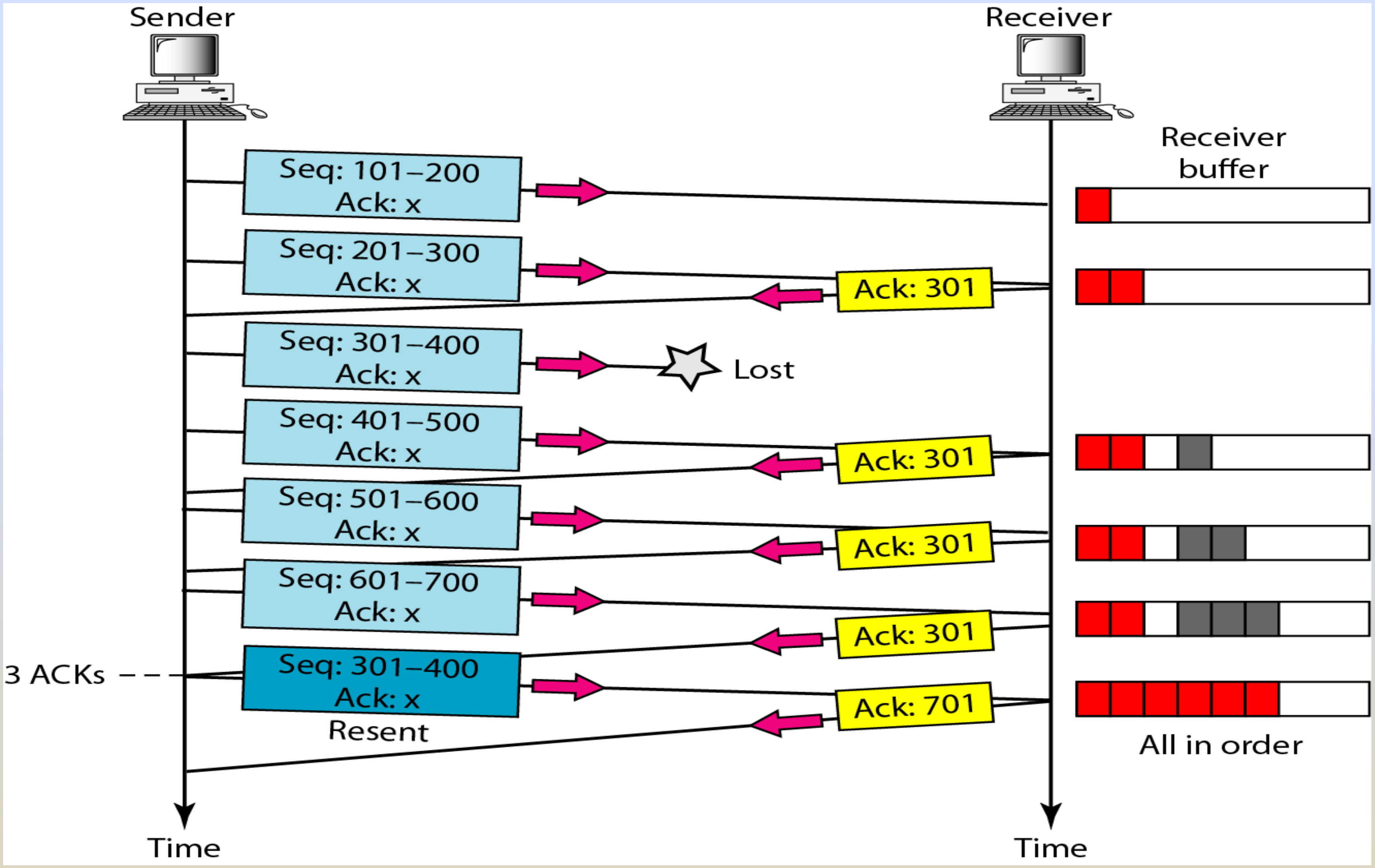
Error Control

- Retransmission : In modern implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.
- Retransmission after RTO : (Retransmission Time Out).
  - Lost Segment
  - Out of Order segment
  - Fast transmission

Lost Segment



Fast Transmission


Sender
Receiver
Receiver buffer
Seq: 101–200 Ack: x
Seq: 201–300 Ack: x
Ack: 301
Seq: 301–400 Ack: x
Lost
Seq: 401–500 Ack: x
Ack: 301
Seq: 501–600 Ack: x
Ack: 301
Seq: 601–700 Ack: x
Ack: 301
3 ACKs
Seq: 301–400 Ack: x
Resent
Ack: 701
All in order
Time
Time