CS321: Computer Networks

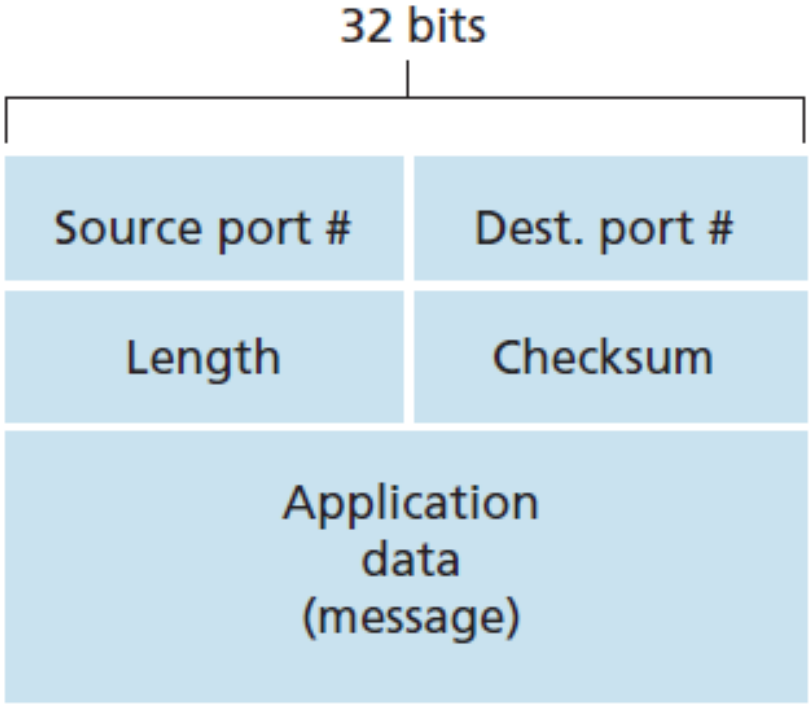
UDP and TCP

Dr. Manas Khatua
Assistant Professor
Dept. of CSE
IIT Jodhpur
E-mail: manaskhatua@iitj.ac.in

Introduction to UDP

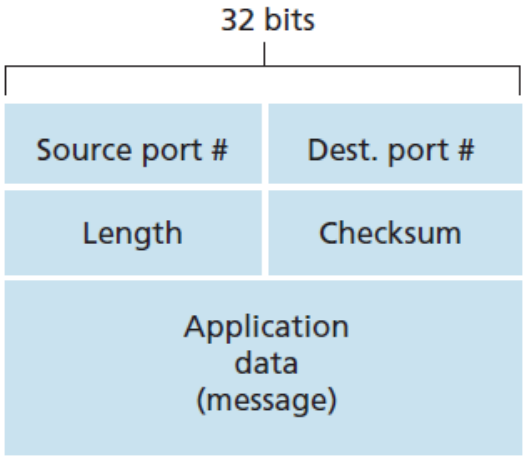
- UDP (User Datagram Protocol)
  - Transport-layer protocol
  - Connectionless
  - Simple
  - Efficient
  - Unreliable
  - Finer application-level control

  over what is sent and when



UDP Services

- Process-to-process communication
  - Need socket address (IP + Port)



- Connectionless service
  - No sequence number
  - No relation between UDP datagrams
  - No connection establishment
  - Datagram can travel through different path
  - No segmentation (message size < (65535 - 8))
- Encapsulation and decapsulation
  - Needs to send message from one process to another
- Multiplexing and demultiplexing
  - One UDP, but several process in application layer wants to use its services

Cont…

- No flow control
  - No window mechanism
- No error control
  - Error detection through checksum but no control
- No congestion control
  - Assumption is that congestion will not occur as UDP datagrams are small in size
- Queuing
  - Queues are associated with port

UDP Checksum

- Checksum
  - It is optional
  - Consider three parts:
    - Pseudoheader of IP header
    - UDP header,
    - message from the upper layer
  - Datalink layer has error detection mechanism. Why do we need checksum in transport layer?
    - neither link-by-link reliability nor in-memory error detection is guaranteed w.r.t. end-to-end service. So we need error detection in Transport layer
  - What is pseudoheader?
    - pseudo header contains some of the same information from the real IPv4 header.
    - it is not the real IPv4 header used to send an IP packet
  - Why do we need pseudoheader?
    - Socket-address need to be correct.
    - To ensure intended receiver (avoids in-memory error)

Cont…

- What value is sent for the checksum in each one of the following hypothetical situations?
  - The sender decides not to include the checksum.
  - The sender decides to include the checksum, but the value of the sum is all 1s.
  - The sender decides to include the checksum, but the value of the sum is all 0s.

- Solution:
  - All 0s
  - When the sender complements the sum, the result is all 0s; the sender complements the result again before sending. The value sent for the checksum is all 1s. The second complement operation is needed to avoid confusion with the previous case. Note that this does not create confusion because the value of the checksum is never all 1s in a normal situation

  - This situation never happens because it implies that the value of every term included in the calculation of the sum is all 0s, which is impossible.

UDP Applications

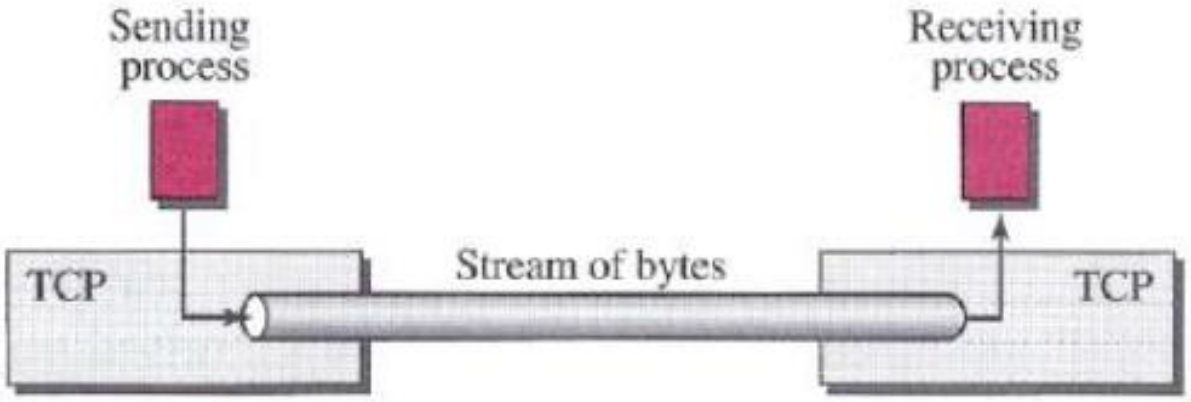
- If the request and response can each fit in a single user datagram, a connectionless service may be preferable.
  - E.g. DNS request and response; But, not suitable in SMTP as e-mail size could be large
- Lack of error control is advantageous sometimes
  - E.g. real-time communication through Skype, Voice over IP, online games, live streaming
  - But, not suitable for file download, Video-On-demand
- Lack of congestion control
  - Advantageous in error-prone network
- It is simple
  - suitable for bootstrapping or other purposes without a full protocol stack, such as the DHCP
- It is stateless
  - suitable for very large numbers of clients, such as in streaming media applications such as IPTV.
- It works well in unidirectional communication
  - suitable for broadcast information such as in many kinds of service discovery

Introduction to TCP

- **TCP** (Transmission Control Protocol)
  - Connection-oriented (but not virtual-circuit)
  - Create connection, do data transfer, tear down connection
  - Reliable (using ACK, Timeout and Retransmission)
  - Uses ARQ strategy (GBN and SR)
  - Cumulative ACK
  - Checksum for error detection
  - Commonly used in Internet
- Connection-oriented service
  - two TCP's establish a logical connection
  - This is not a virtual-circuit switching
  - Connections are between two end systems only
  - Data are exchanged in both directions
  - The connection is terminated at the end of transmission

TCP Services

- Process-to-process communication
  - Needs socket address
- Reliable service
  - Using ACK, Timeout, Retransmission
- Stream Delivery Service



- Multiplexing and Demultiplexing
  - Need of connection establishment

TCP Demultiplexing

- UDP socket is fully identified by a two-tuple: destination IP address, destination port number.
- TCP socket is identified by a four-tuple: source IP address, source port number, destination IP address, destination port number.

- If two **UDP** segments have different source IP addresses and/or source port numbers, but have the same *destination* IP address and *destination* port number, then the two segments will be directed to the same destination process via the same destination socket.
- In contrast with UDP, two arriving **TCP** segments with different source IP addresses or source port numbers will be directed to two different sockets.

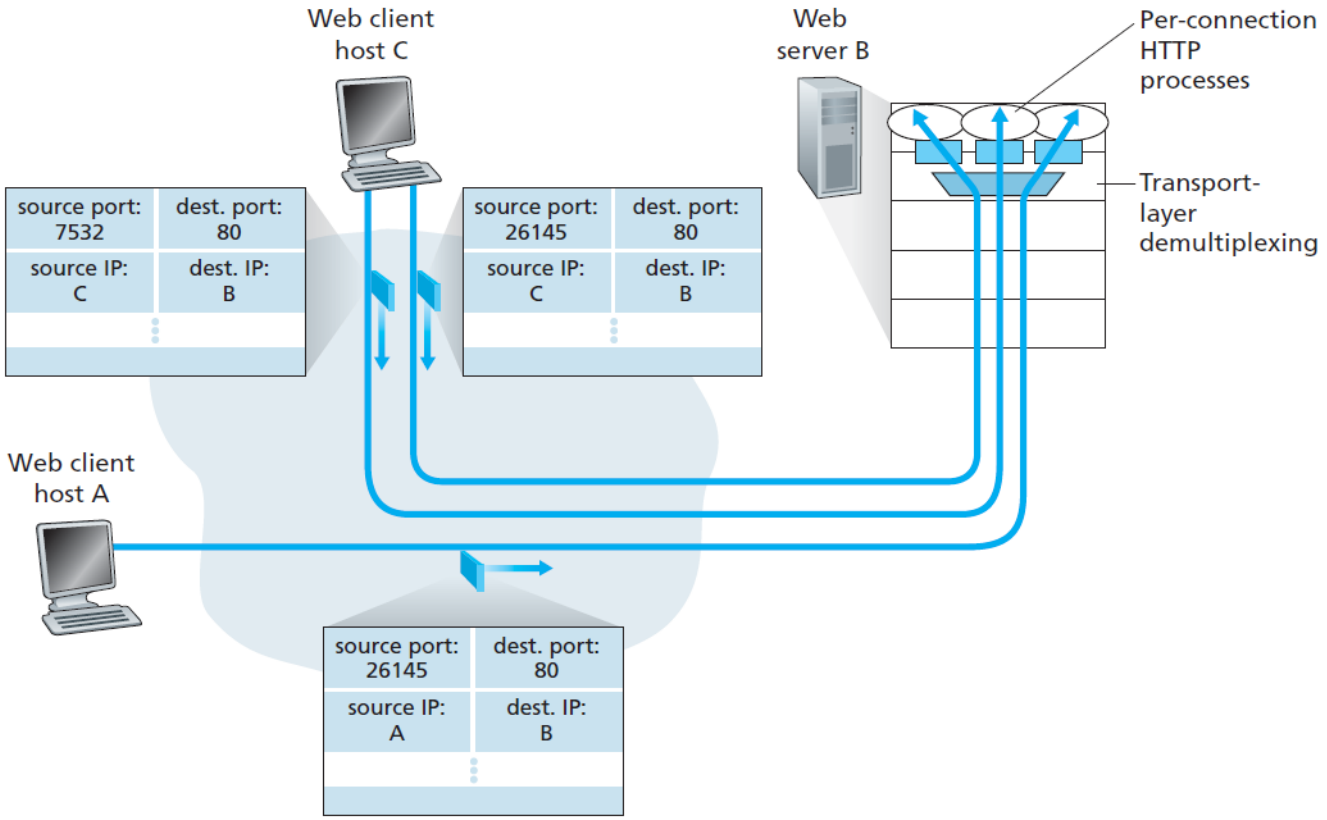
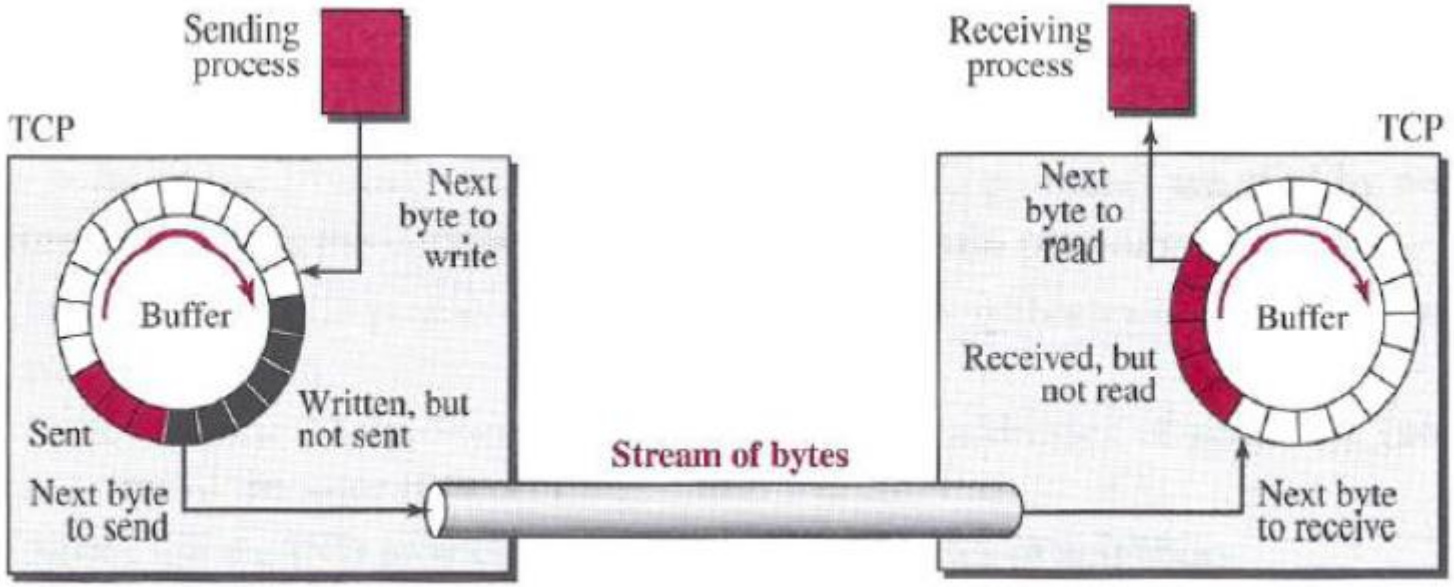


Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

TCP Services

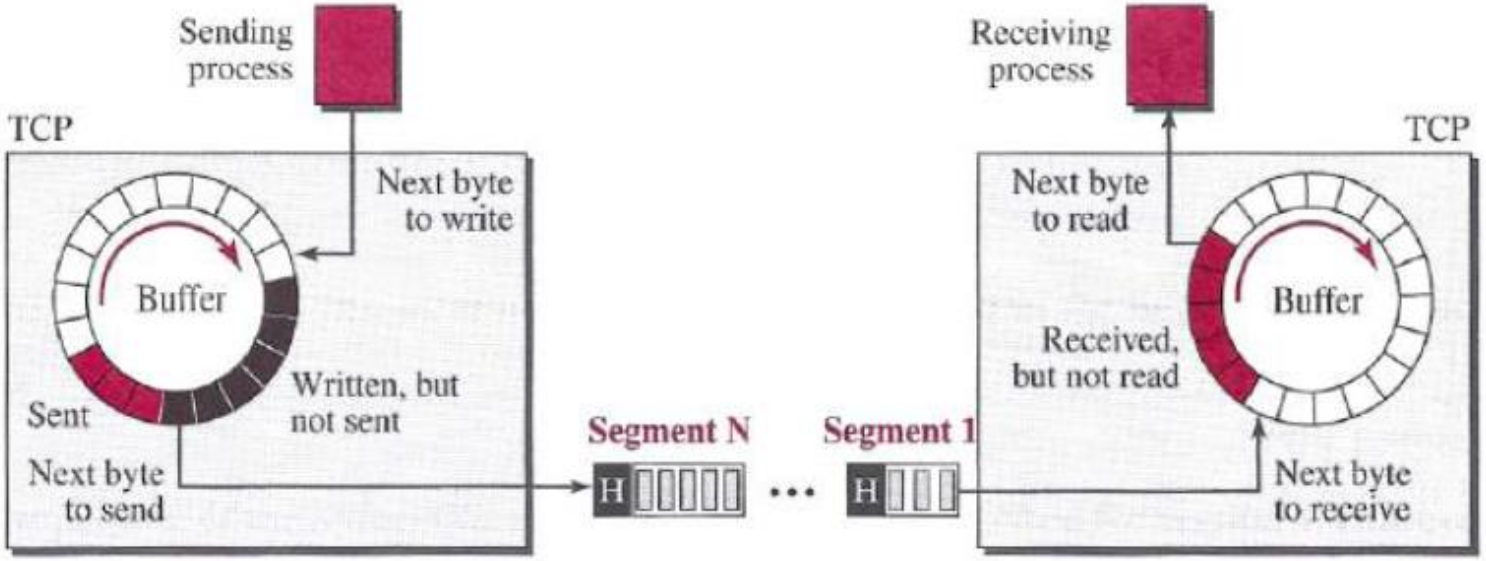
- Flow control
  - Sending and Receiving buffer
  - may not necessarily write or read data at the same rate

- Full-Duplex Communication
  - Each TCP endpoint has its own sending and receiving buffer



Cont…

- Segmentation and Reassembly
  - TCP usually determines the maximum segment size (MSS) based on the maximum transmission unit (MTU) of Layer 3 (IP layer).
  - Note: for UDP, the assumption is that the UDP segments are small in size



Numbering in TCP

- No segment number
- But, TCP uses sequence (SEQ) number and ACK number
  - These are byte numbers, but not segment number

- Number is independent in each direction

- TCP numbers all data bytes transmitted in a connection

- TCP chooses an arbitrary number between 0 and 2^{32}- 1 for numbering the first byte

SEQ and ACK numbers

- After the bytes have been numbered,
  - TCP assigns a SEQ number to each segment that is being sent
  - For 1st segment: SEQ number is random
  - For any other segment: SEQ number of previous segment + number of byte in the previous segment

- Each party also uses an ACK number to confirm the bytes it has received.

- The ACK number defines the number of the next byte that the party expects to receive.

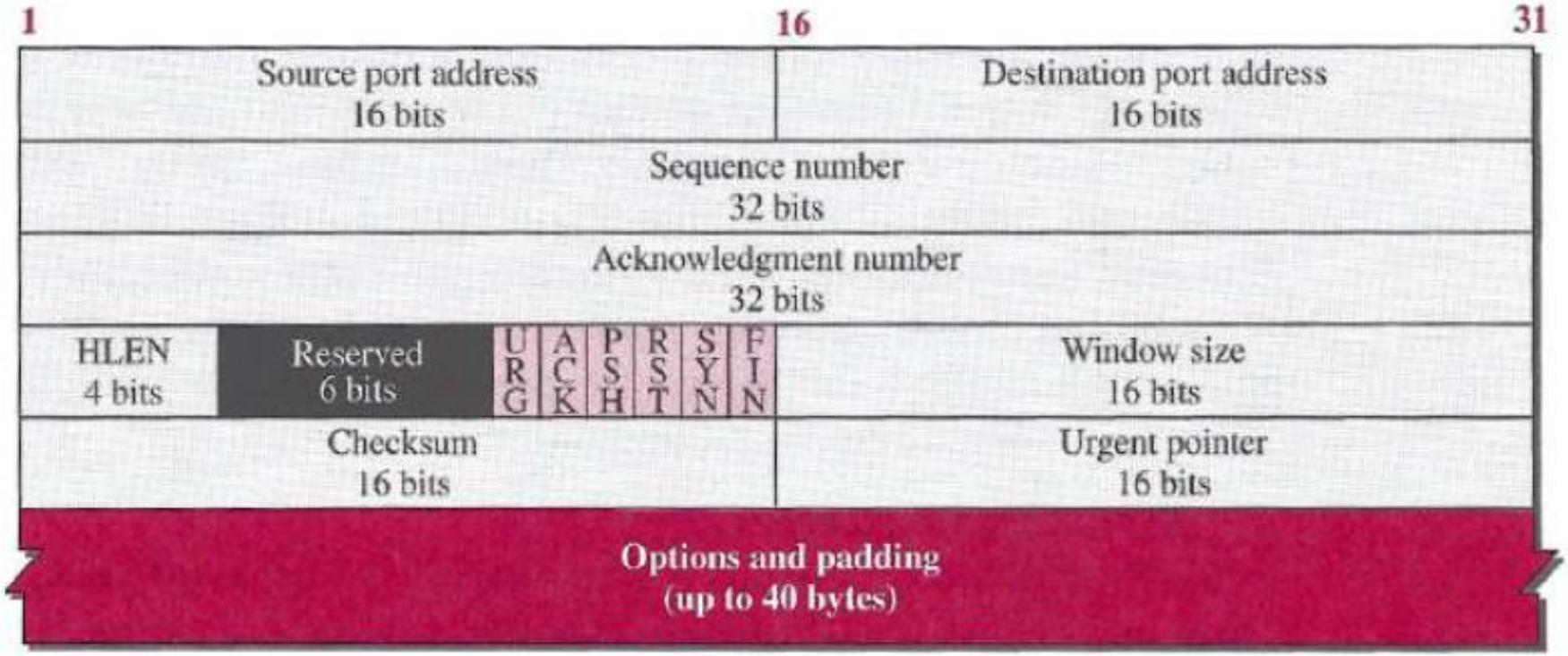
- The ACK number is cumulative

Fields in a TCP Segment

- Source & Destination port numbers.
- Sequence number & Acknowledgement number.
- Header length.
- Control. (6 control flags: URG, ACK, PSH, RST, SYN, FIN)
- Window size (window size of the sending TCP in bytes)
- Checksum. (pseudo header + TCP header + data)
- Urgent pointer (valid only if urgent flag is set)
- Options (optional information in the TCP header)

TCP Segment Format

a. Segment



b. Header

TCP Connection

- TCP is connection-oriented although IP is connectionless
- TCP connection is logical, not physical.

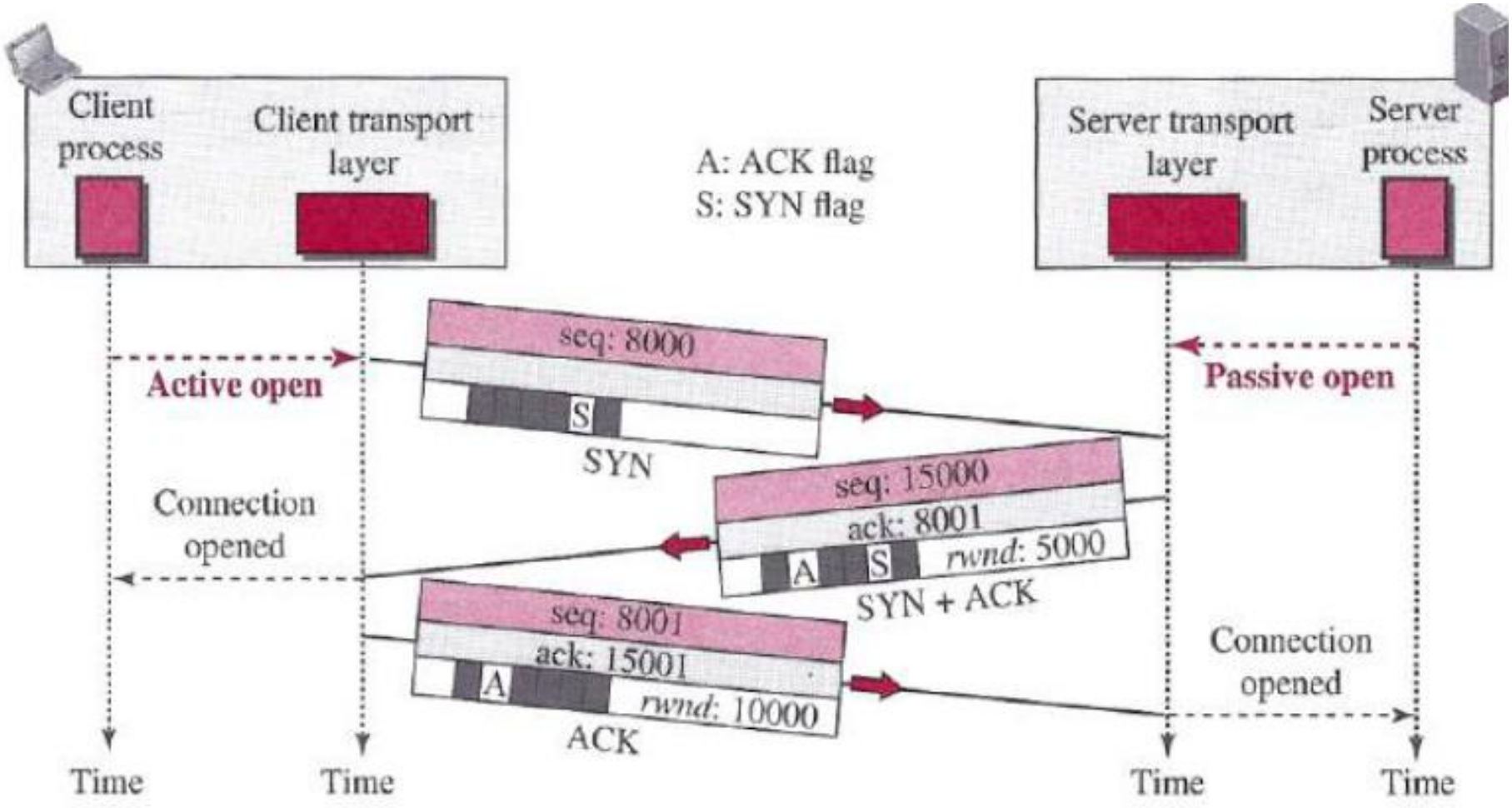
- TCP operates in full-duplex mode
- TCP uses three-way-handshaking
  - SYN
  - ACK+SYN
  - ACK

- Let, an application program, called the client, wants to make a connection with another application program, called the server, using TCP

- The process starts with the server.
  - Passive open (server process informs transport layer of server that it is ready)
  - Active open (client process issues request to client transport layer)
  - Now Client transport layer starts three-way-handshaking

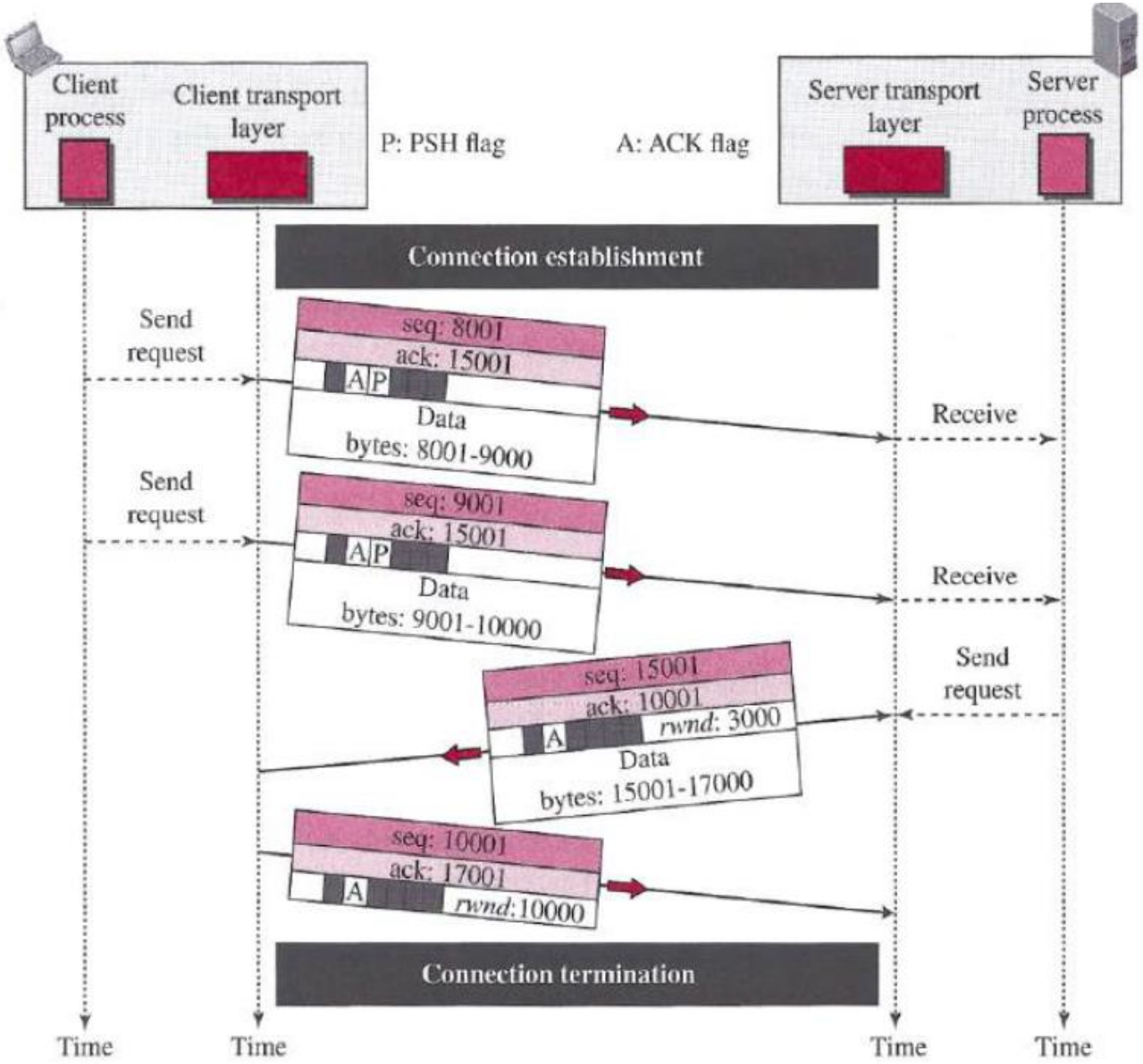
Connection Creation : step 1

- Using three-way-handshaking
- ACK and SYN flags are used in Step-1



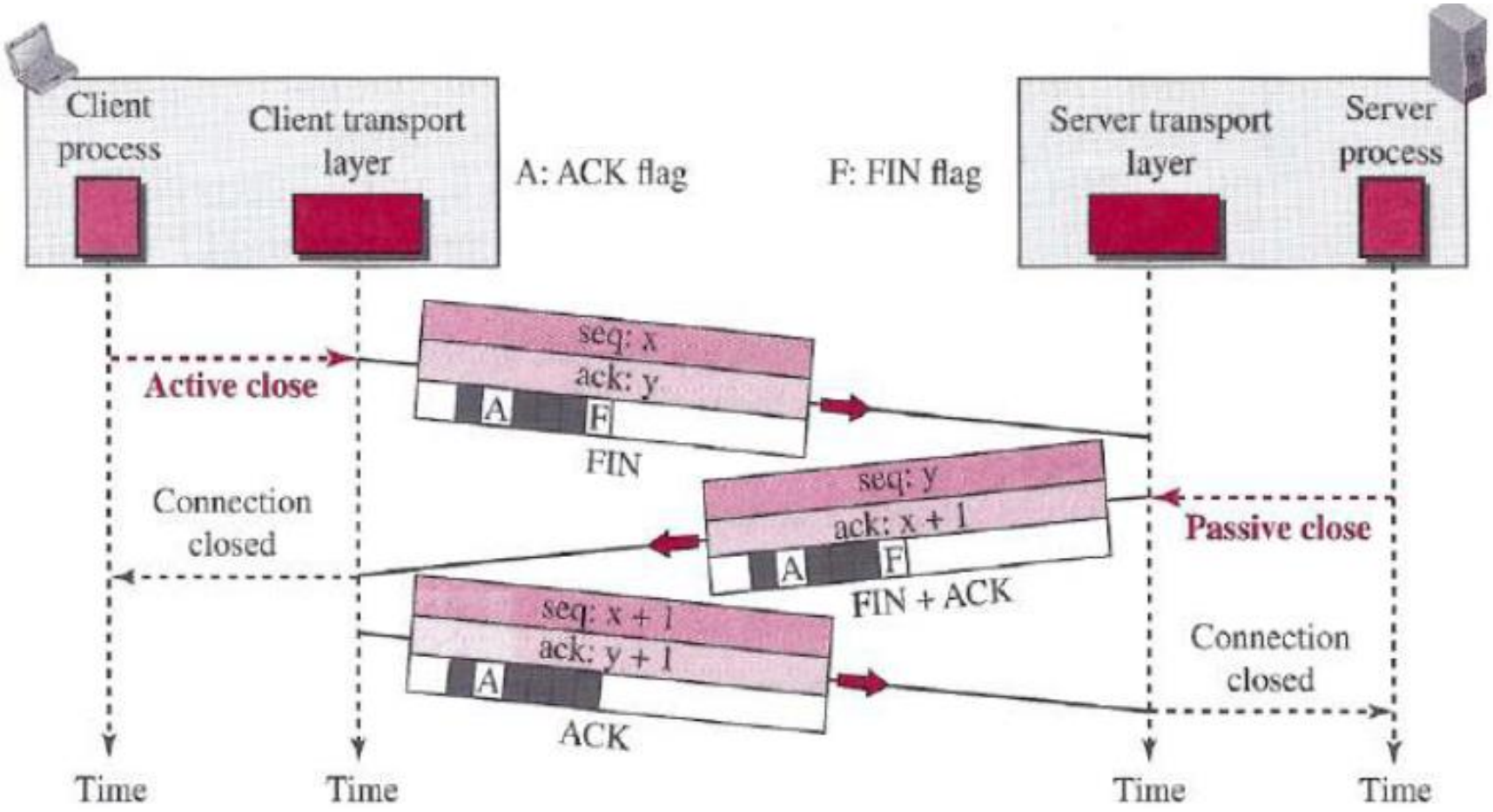
Data Transfer : step 2

ACK and/or PSH flags are used in Step-2



Connection Termination : step 3

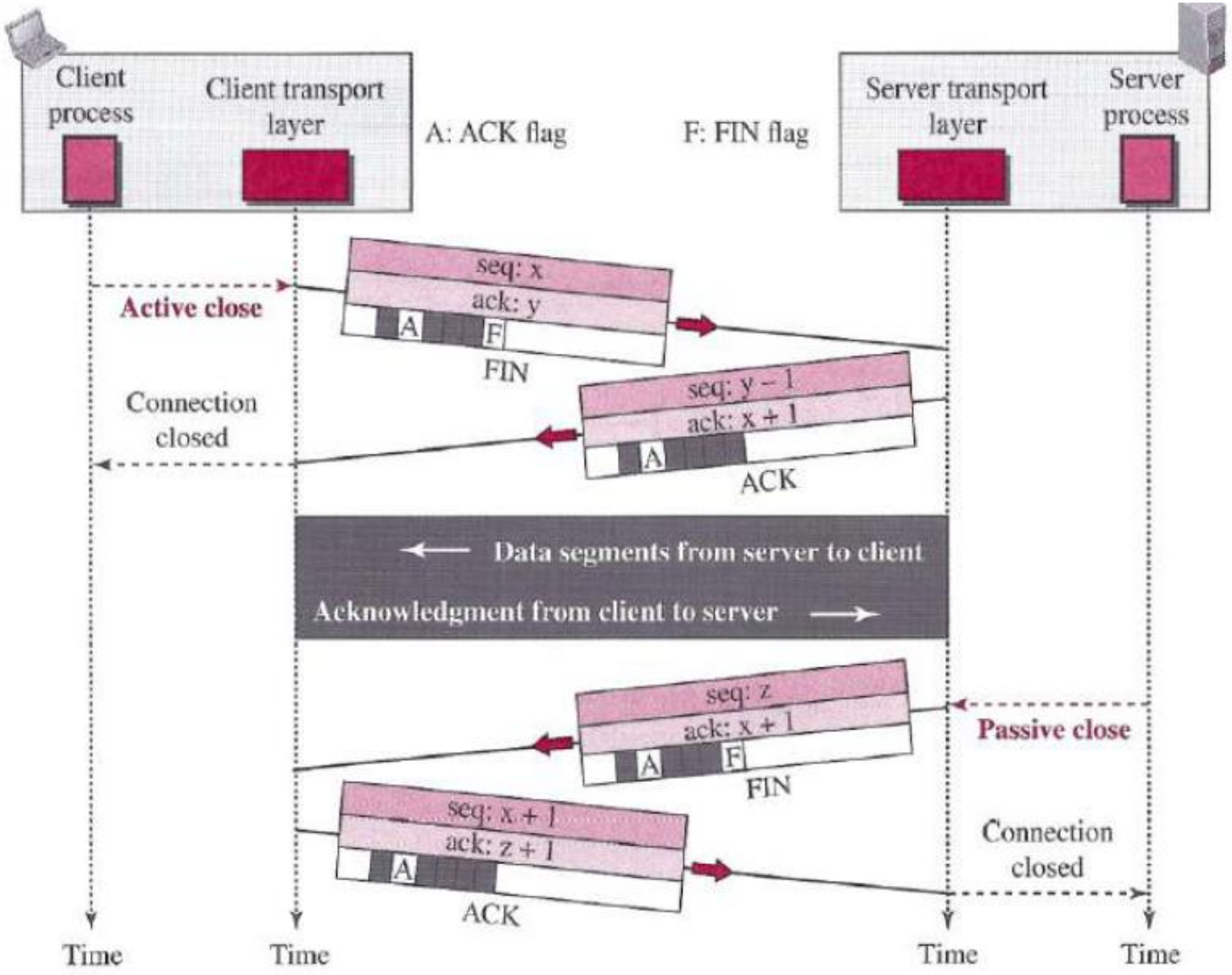
- ACK and FIN flags are used in Step-3



Half-close Connection

Example:

Sorting at server



Scenario

Create Connection

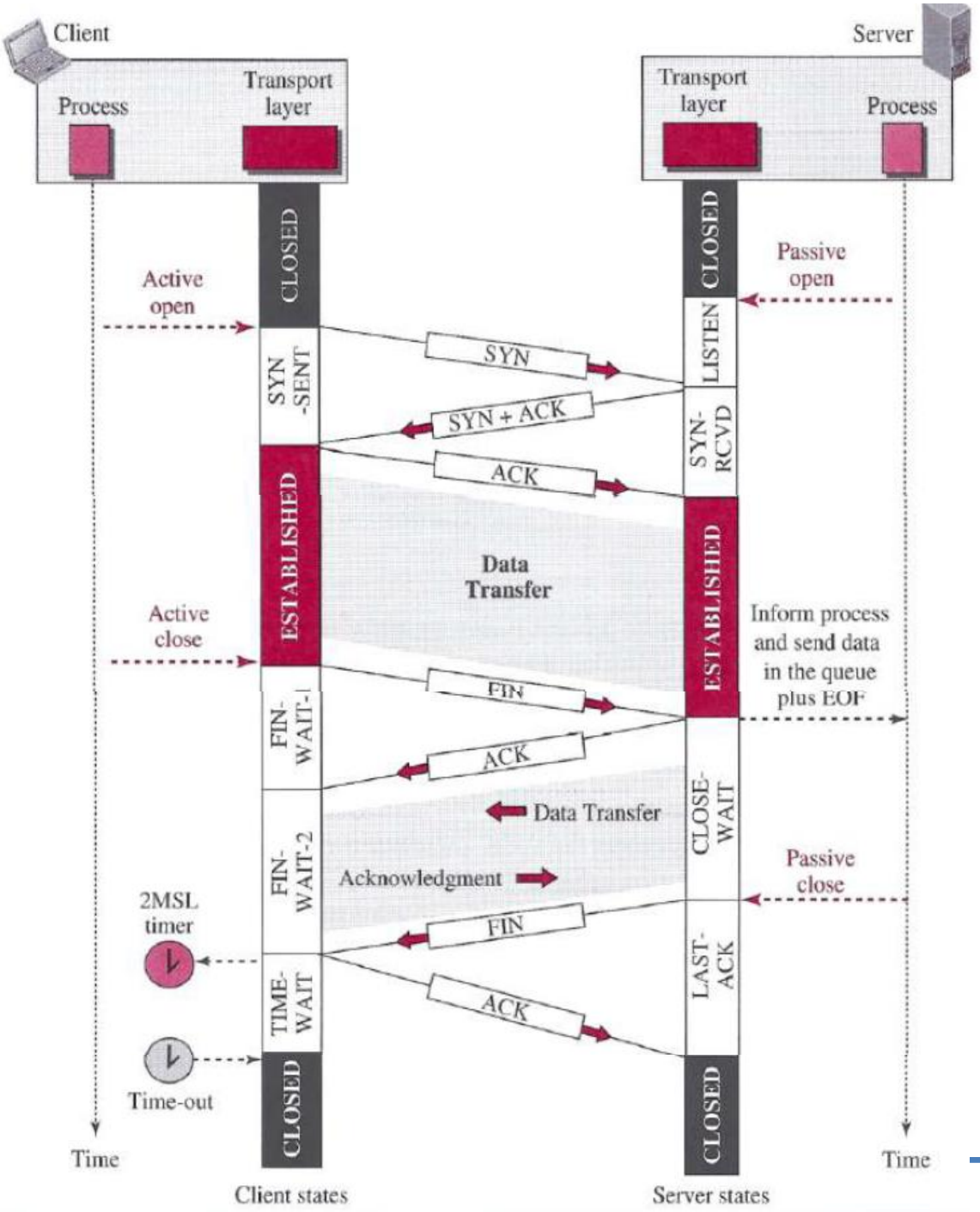
Data Transfer
in both direction

Half-close

Receive Response

Close other half

Closed status



PSH, RST, URG flags

- PUSH (PSH) flag means sending TCP must not wait for the window to be filled, and then send the segment
- PUSH flag informs the receiving TCP to deliver the received segment immediately to application program

- RESET (RST) flag means it is telling the sender "I don't have socket for that segment. Please don't resend the segment"
- It is required when a host receives TCP SYN segment with a destination port (say 80) but the destination is not accepting any connection on that port (may be Web server is not running at port 80)

- When the URGENT (URG) bit is set the Urgent Pointer is also set (in the TCP header Options field: 16 bit).
- The URG pointer tell how many bytes of the data is urgent in the segment that has arrived.
- Example if the data size is 100 bytes and only firs 50 bytes is urgent, the urgent pointer will have a value of 50

TCP Applications

- Major Internet applications rely on TCP
  - World Wide Web (HTTP)
  - E-mail (SMTP, IMAP, POP)
  - File Transfer Protocol (FTP)
  - Secure Shell (SSH)
  - Telnet

Thanks!