Unit :- 3 TCP/IP Protocol - III

* TCP Services :-

(1) Process to process communication :-

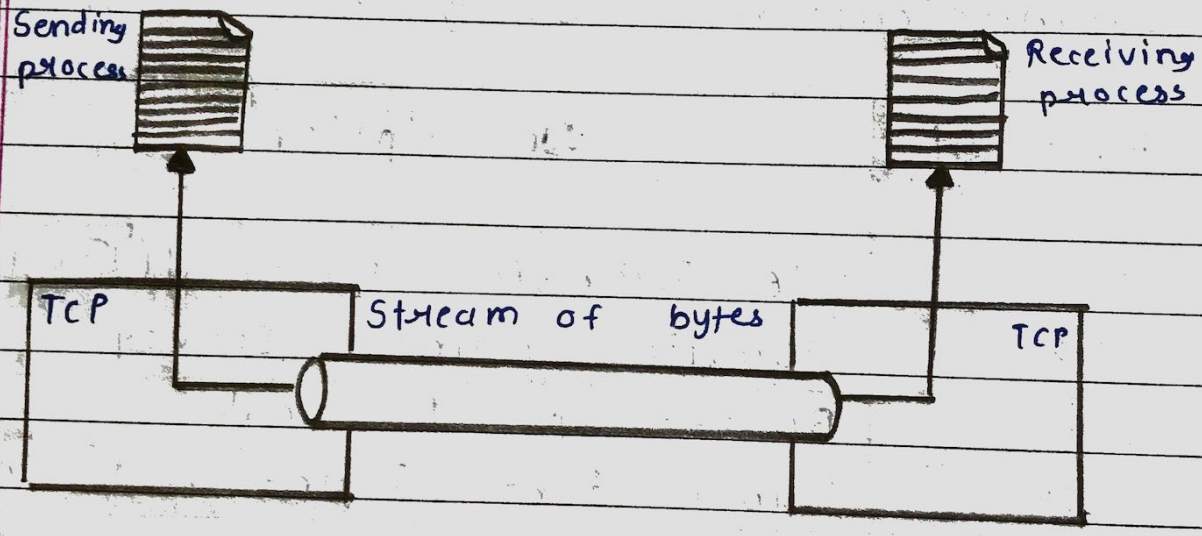
→ TCP provides process to process communication using port numbers.

Port	Protocol	Description
20 and 21	FTP	File Transfer Protocol (Data & control)
53	DNS	Domain name server
67	BOOTP	Bootstrap Protocol
80	HTTP	Hyper text Transfer Protocol

(2) Stream Delivery Service :-

→ TCP allowed the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a

stream of bytes. TCP creates an environment in which the two processes seem to be connected by an imaginary. The sending process produces (writes to) the stream of bytes and the receiving processes consumes (reads from) them.



- **Sending and Receiving buffers:**

→ Because the sending and the receiving processes may not necessarily write or read data at the same rate, TCP needs buffers of storage. There are two buffers, the sending

buffer and receiving buffer, one for each direction.

→ here, we have implemented buffer of 20 bytes with one chamber of one byte.

* At the sender the buffers has 3 types of chambers.

* The white section contains empty chambers, will be filled by sending process.

* The shaded (grey) area contains bytes area contains bytes to be sent by sending TCP.

* The coloured area (pink) holds bytes that have been sent but not acknowledge.

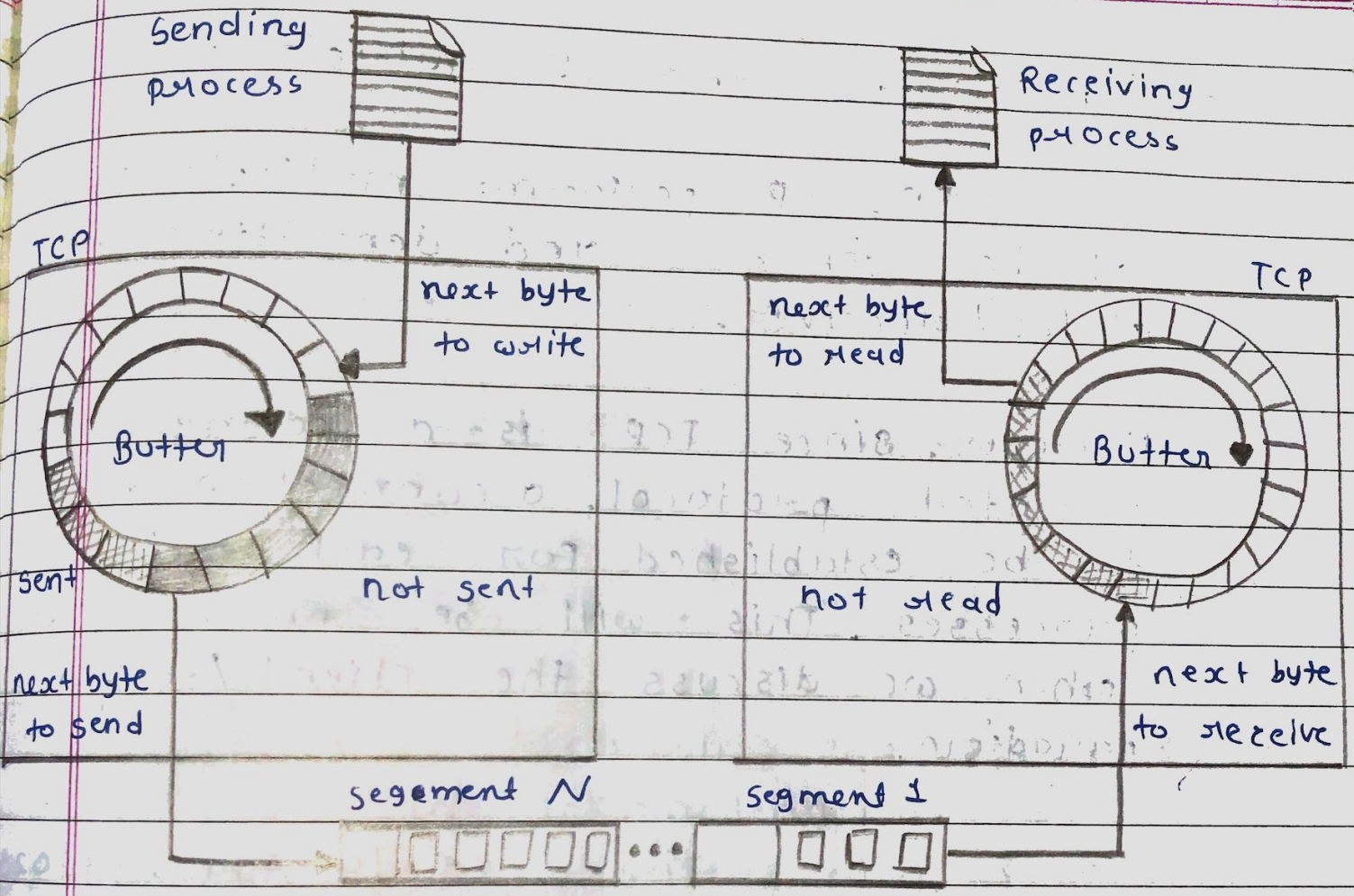
* At receiver the buffer is divided into 2 areas.

- the white area contains empty chambers, will be filled by sending process

- the coloured section contains received bytes which can be read by receiving process.

* IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes.

* At the transport layer, TCP groups a number of bytes together into packet called a segement.



- **Full-Duplex communication :-**

TCP offers full-duplex service, where data can flow in both direction at the same time. Each TCP endpoint then has its own sending and receiving buffer, and segment move in both directions.

- Multiplexing and Demultiplexing :

Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver.

However, since TCP is a connection-oriented protocol, a connection needs to be established for each pair of processes. This will be more clear when we discuss the client/server paradigm.

- 14, August, 2023
Monday

* connection-oriented service :-

When a process at site A want to send to and receive data from another process at site B, the following phases occur.

1. The two TCPs establish a virtual connection between them.

2. Data are exchanged in both directions.

3. The connection is terminated.

* Reliable service :

TCP is a Reliable transport protocol. It uses an acknowledgement mechanism to check the safe and sound arrival of data.

* TCP Features :-

(1) Numbering System
~~(1) Acknowledgement number~~.
(2) Flow control
(3) Error control
(4) congestion control

(1) Numbering System:

- Byte number :-

- TCP number all data bytes that are transmitted in a connection numbering is independent in each direction.

- For example, if the random byte number happens to be 1057 and the total data to be is 6000 bytes, the bytes are numbered from 1,057 to 7056.

- Sequence number :-

- The sequence number for each segment is a number of the first byte of data carried in that segment.

- For example, suppose a TCP connection is transfering a file of 5000bytes.

The first byte is numbered 10,001 what are sequence number for each segement if data are sent in five segements, each carrying 100 bytes?

Solution: The following shows the sequence number for each segment

Segment	Sequence Number	Range
1	10,001	10,001 to 11,000
2	11,001	11,001 to 12,000
3	12,001	12,001 to 13,000
4	13,001	13,001 to 14,000
5	14,001	14,001 to 15,000

- Acknowledgement Number :-

- Each party also uses an acknowledgement number to confirm the bytes it has received, However the acknowledgement numbers defines the number of the next byte that the party expects to receive..

- For example, if a party uses 5643 as an acknowledgement number, it has received all bytes from the beginning up to 5642.

(2) Flow control :-

- TCP, Unlike UDP, provides flow control. The sending TCP controls how much data can be accepted from the sending process; the receiving TCP controls. how much data can to be sent by the sending TCP.

(3) Error control :-

- To provide reliable service, TCP implements an error control mechanism Although error control consider a segment as the unit of data for error detection.

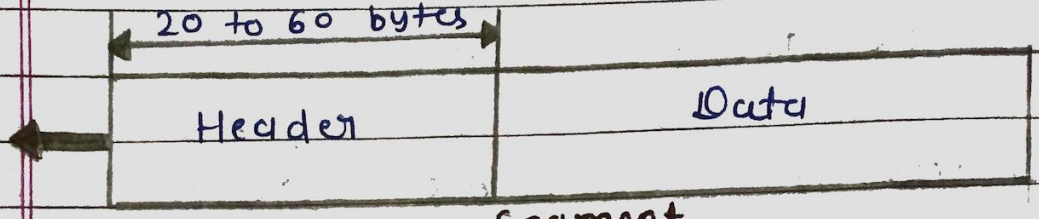
(4) Congestion control :-

- TCP, unlike UDP, takes into account congestion in the network. The amount only controlled by the receiver, but is also determined by the level of congetion, if any, in the network.

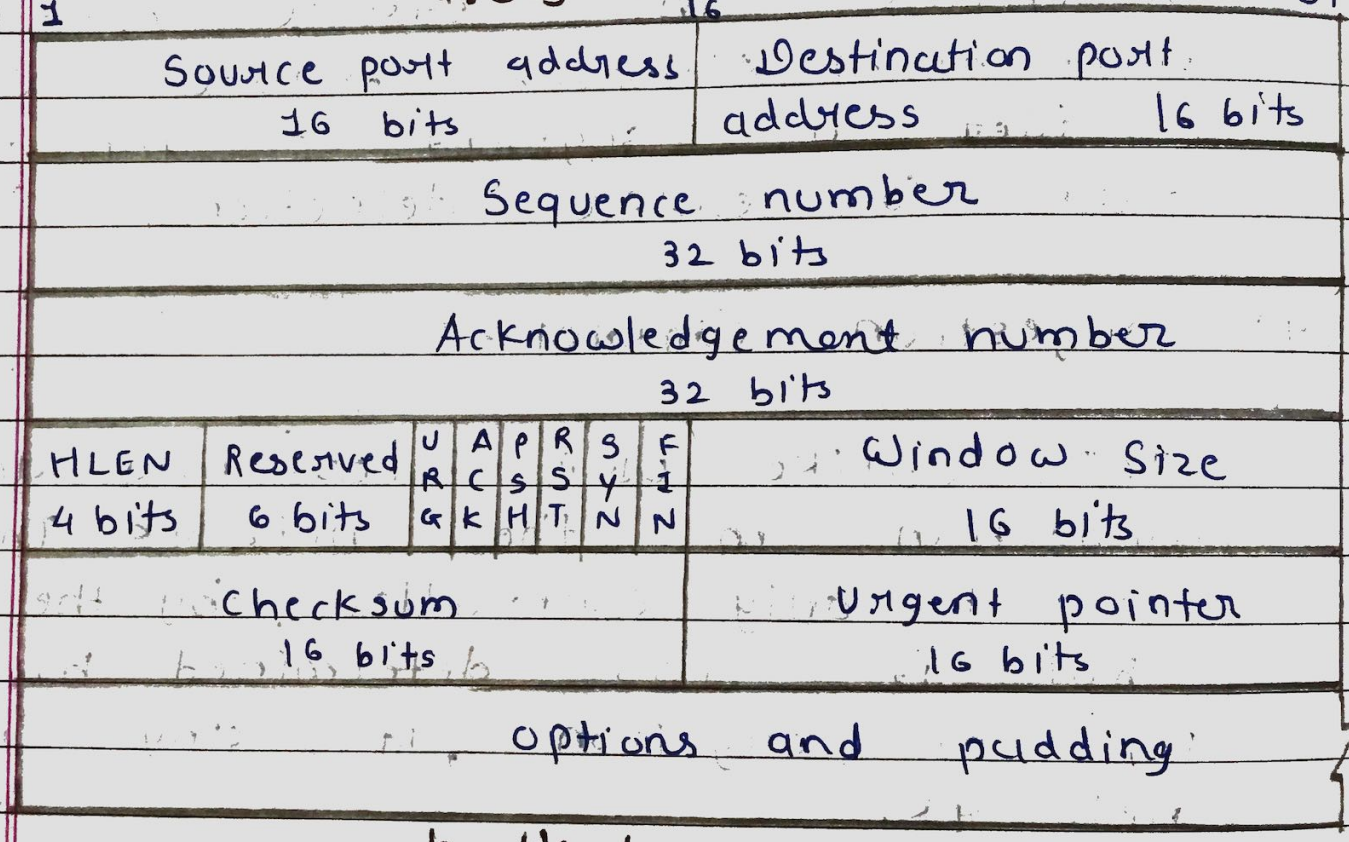
* Segment (TCP format) :-

- packet in TCP is called a segment

→



a. Segment



b. Header

* Source port address :-
This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.

* Destination port address :-
This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

* Sequence number :-
This 32-bit field defines the number assigned to the first byte of data contained in this segment.

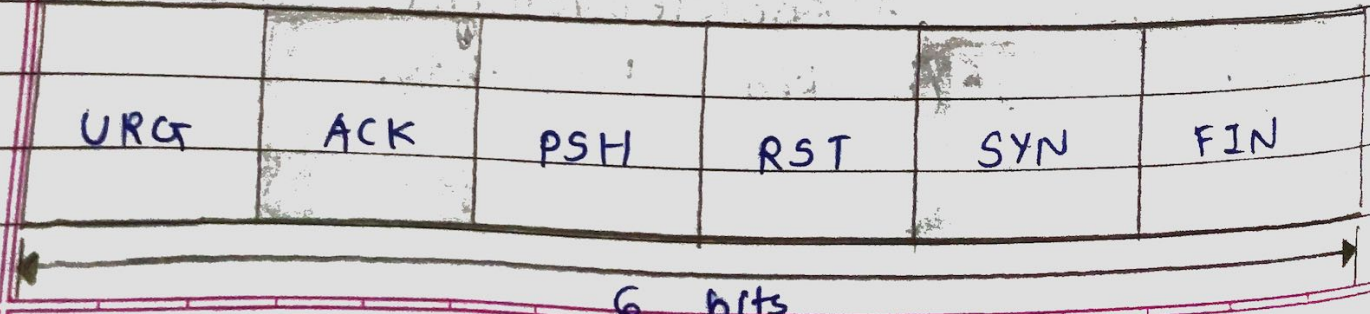
* Acknowledgement number :-
This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x from the other party, it returns x+1 as the acknowledgement number. Acknowledgement and data can be piggybacked together.

* **Header length :-**
This 4-bit field indicates the number of 4-byte word in the TCP header. The length of the header can be between 20 and 60 bytes.

* **Reserved :**
This is a 6-bit field reserved for future use.

* **Control :-**
This field defines 6 different bits or flags.

URG : Urgent pointer is valid
ACK : Acknowledgement is valid
PSH : Request for push
RST : Reset the connection
SYN : Synchronize sequence number
FIN : Terminate the connection.



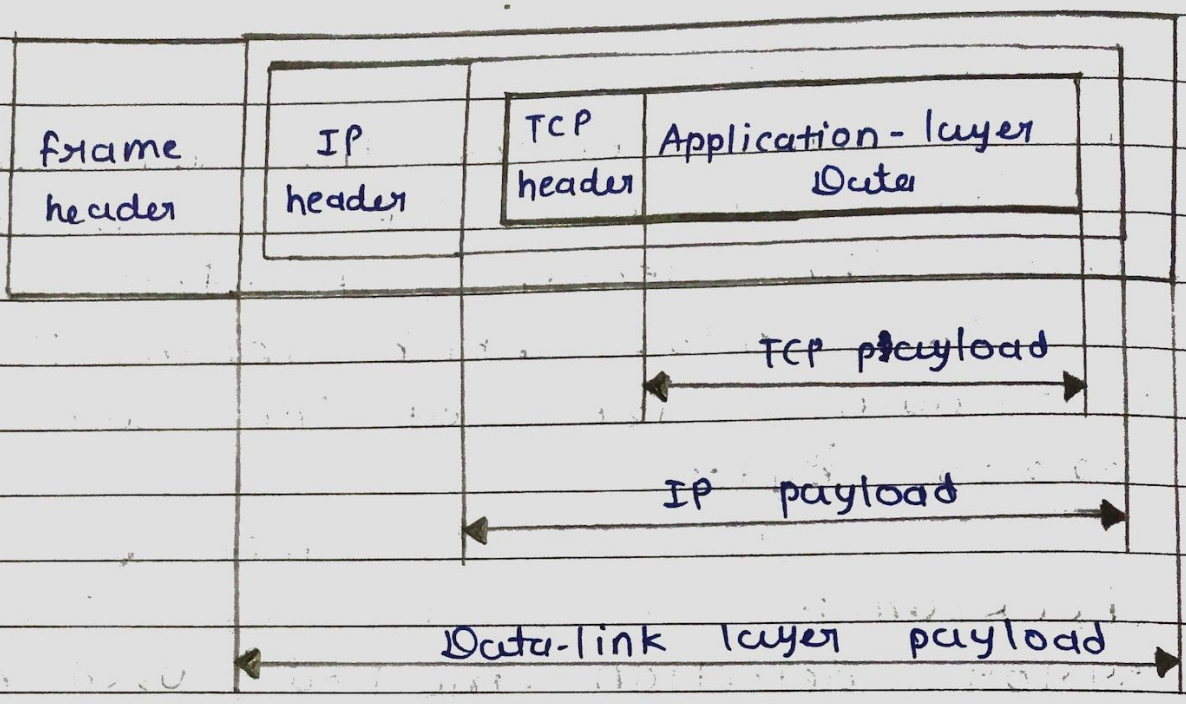
* Window size :-
This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 63,535 bytes.

* Checksum :-
error detection method used by TCP.

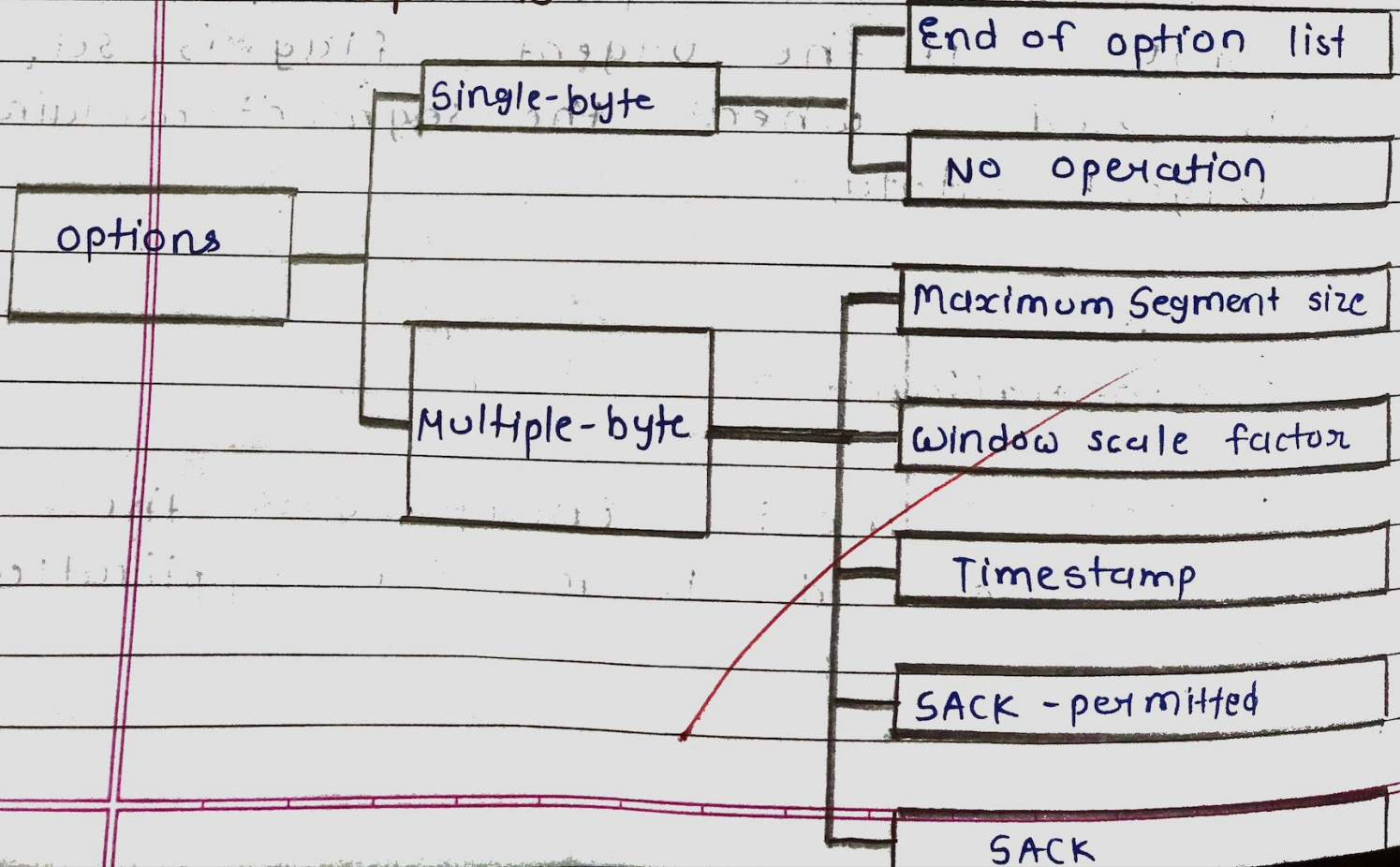
* Urgent pointer :-
This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.

* Encapsulation :-

A TCP segment encapsulates the data received from the application layer.


frame header
IP header
TCP header
Application-layer Data
TCP payload
IP payload
Data-link layer payload

* TCP Options :-


options
Single-byte
End of option list
No operation
Multiple-byte
Maximum Segment size
Window scale factor
Timestamp
SACK-permitted
SACK

- **End of option (EOP) :-**

The end of option is a 1-byte option used for padding at the end of the option section. It can only be used as the last option.

- **No operation (NOP) :-**

The no-operation (NOP) option is also a 1-byte option used as a filler.

- **Maximum segment size (MSS) :-**

The maximum segment size option defines the size of the biggest unit of data that can be received by the destination of the TCP segment.

- **Window scale factor :-**

The window size field in the header defines the size of sliding window.

This field is 16 bits long, which means that the window can range from 0 to 65,9535 bytes.

- **Timestamp :-**

The time-stamp option has two applications it measures the round trip time & prevents wrap-around sequence number.

- **SACK-permitted (Selective Acknowledgement):-**

Selective acknowledgement allows the sender to have a better idea of which segments are actually lost and which have arrived out of order.

- **SACK options :-**

The SACK options, of variable length, is used during data transfer only if both ends agree (if they have exchanged SACK-permitted options during connection establishment)

* TCP connections :-

→ TCP is connection-oriented protocol.

→ In TCP we requires three phases:
(1) connection establishment
(2) Data Transfer
(3) connection Termination

(1) Connection establishment (three-way handshaking):-

→ The connection establishment in TCP is called three-way handshaking.

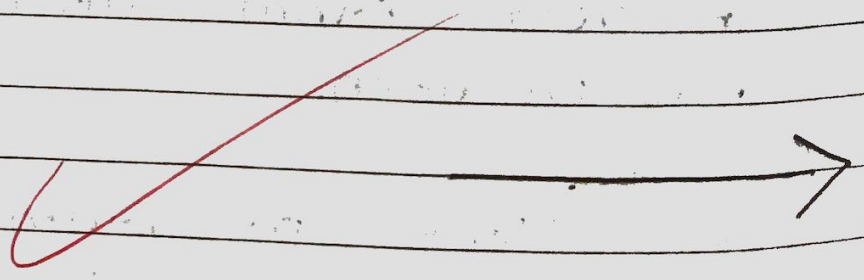
→ The server opens itself with the help of TCP program, to accept request from clients, its called passive open.

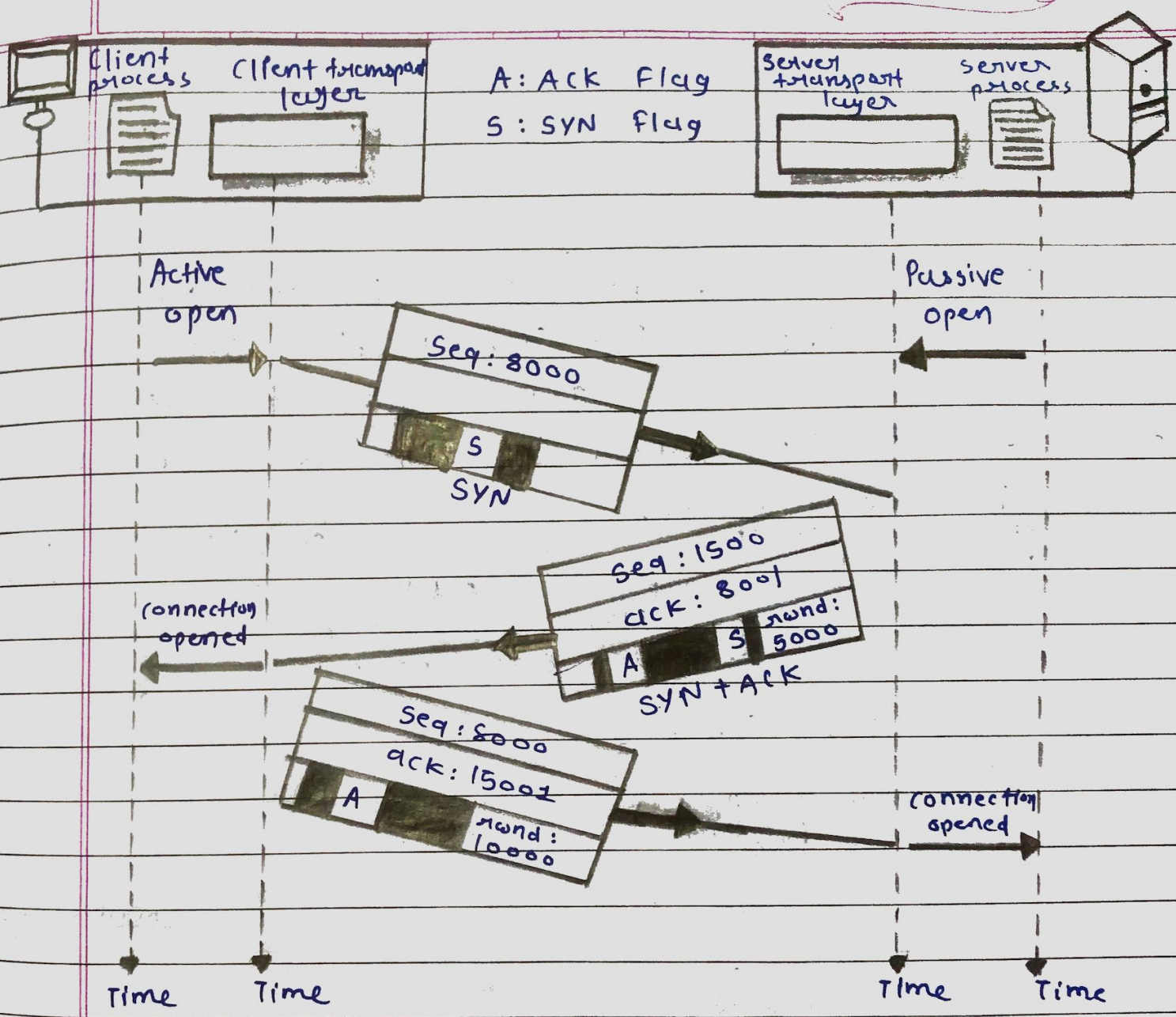
→ The client program issues a request for an active open (SYN segment)

→ The client sends the first segment, a SYN segment, in which only the SYN flag is set.

→ The server sends the second segment, a SYN + ACK segment with two flag bits set; SYN and ACK.

→ The client sends the third segment. This is just an ACK segment.



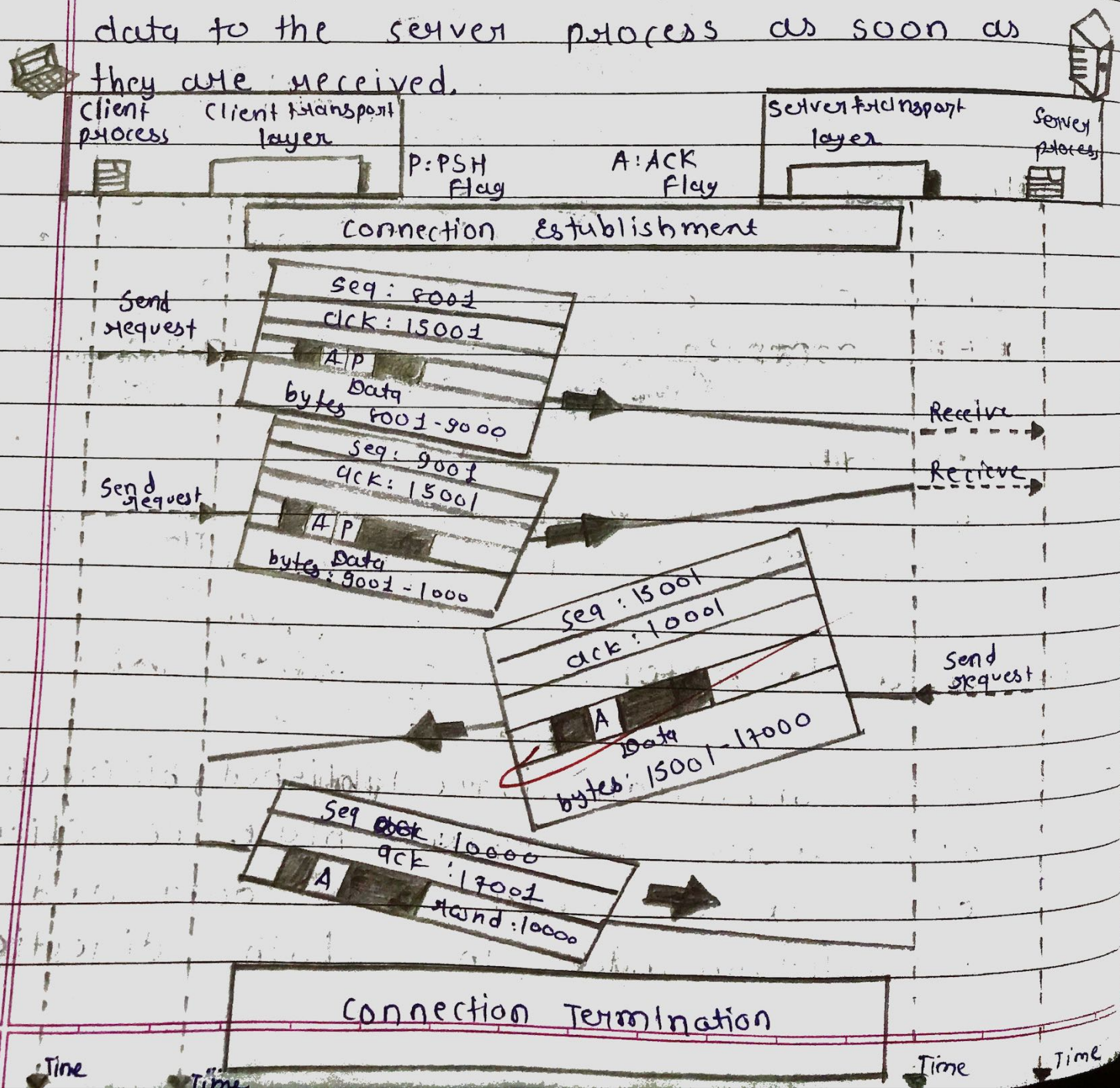


(2) Data Transfer :-

→ After connection is established, bidirectional data transfer can take place. The client and server can send data and acknowledgements in both directions.

→ The acknowledgement is piggybacked with the data.

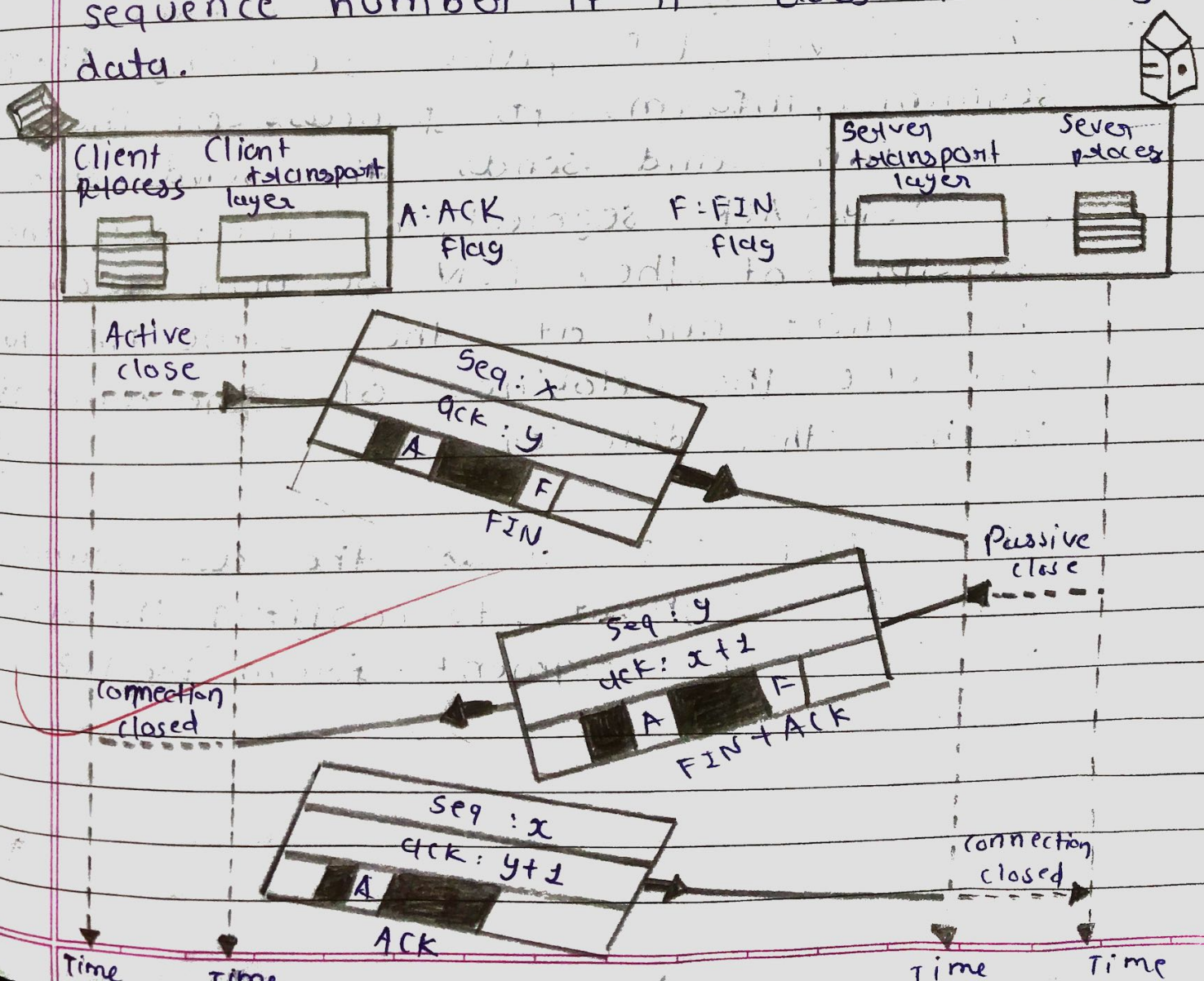
→ The data segments sent by the client have the PSH (push) flag set so that the server TCP tries to deliver data to the server process as soon as they are received.



(3) Connection Termination :-

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

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



• Three - Way Handshaking :-

(1) In a common situation, the client TCP, after receiving a close command from the client process, sends the first segement, a FIN segement in which the FIN flag is set

(2) The server TCP, after receiving the FIN segment, informs its process of the situation and sends the sends segment, a FIN + ACK segment, to confirm the receipt of the FIN Segment from the client and at the same time to announce the closing of the connection in the other direction.

(3) The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server.

* Silly Window Syndrome

→ A serious problem can arise in the sliding window operation when either the sending application program creates data slowly or the receiving application program consumes data slowly, or both.

→ Any of these situations results in the sending of data in very small segments, which reduces the efficiency of the operation.

→ for example, if TCP sends segments containing only 1 byte of data, it means that a 41-byte datagram transfer only 1 byte user data. Hear the overhead is 41/1, which indicates

that we are using the capacity of the network very inefficiently. The inefficiency is even worse after accounting for the data link layer and physical layer overhead. This problem is called is called the silly window syndrome.

- **Syndrome created by the Sender:**

→ The sending TCP may create a silly window syndrome if it is serving an application program that creates data slowly, for example, 1 byte at a time.

* **Nagle's Algorithm:**

(1) The sending TCP sends the first piece of data it receives from the sending application.

(2) After sending the first segment, the sending TCP accumulates data in the output buffer and waits until either the receiving TCP sends an acknowledgement or until

enough data has accumulated to fill a maximum size segment. At this time, the sending TCP can send the segment.

(3) Step 2 is repeated for the rest of the transmission.

- **Syndrome Created by the Receiver.**

→ The receiving TCP may create a silly window syndrome if it is serving an application program that consumes data slowly, for example, 1 byte at a time.

- **Clark's solution** is to send an acknowledgement as soon as the data arrive, but to announce a window a window size of zero until either there is enough space to accommodate a segment of maximum size or until at least half of the receive buffer is empty.

- **Delayed Acknowledgment** The second solution is to delay sending the

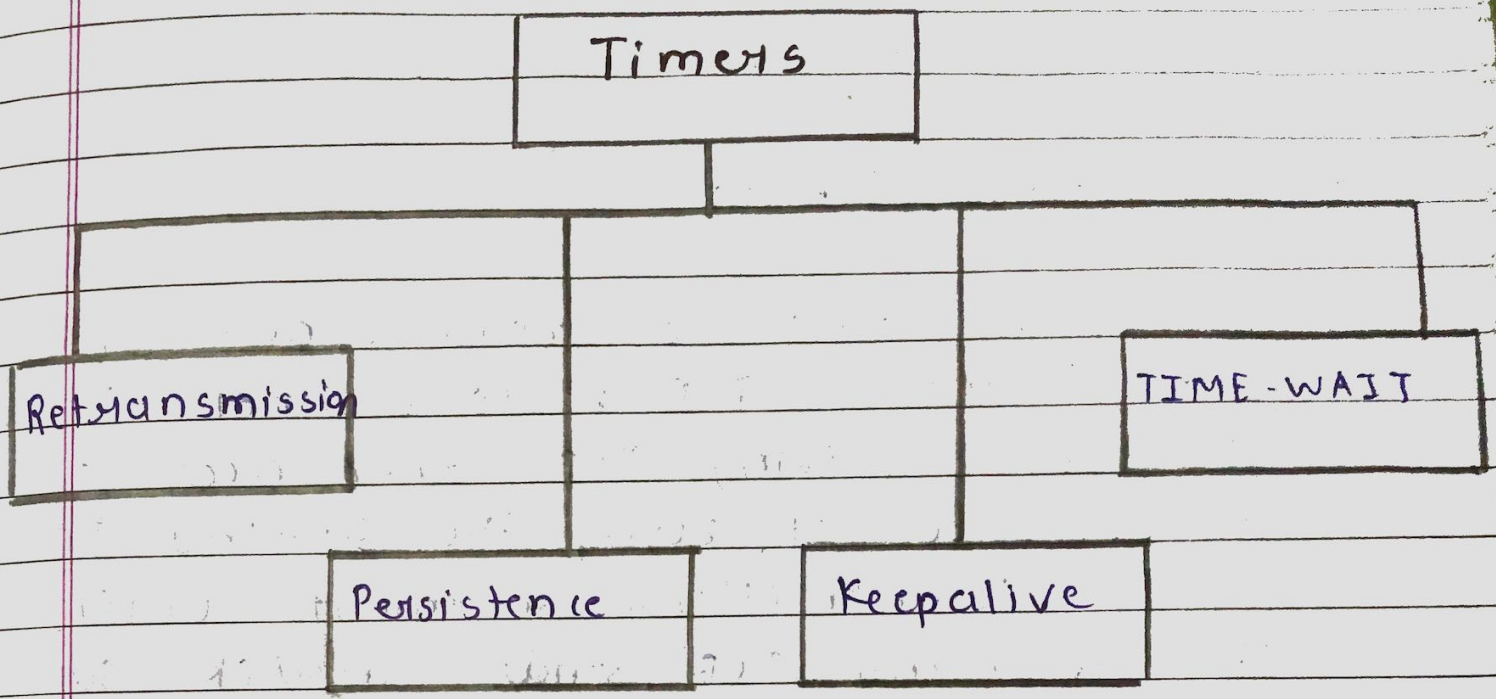
acknowledgement. This means that when a segement arrives, it is not acknowledgement immediately The receiver waits until there is a decent amount of space in its incoming buffer before acknowledging the arrived segements.

→ The delayed acknowledgement prevents the sending TCP from sliding window. After the sending TCP has sent data in the window, it stops. This kills Syndrome.

→ Delayed acknowledgement also has another advantage : it reduces traffic. The receiver does not have to acknowledge each segment.

* TCP Timers :-

→ To perform its operation smoothly, most TCP implementations use at least four timers as shown in this figure.



- **Retransmission Timer :**

→ To retransmit lost segments, TCP employs one retransmission timer (for the whole connection period) that handles the retransmission time-out (RTO), the waiting time for an acknowledgement of a segment.

- **Round-Trip Time (RTT)**

→ To caculate the retransmission time-out (RTO), we first need to calculate the

round trip time (RTT).

- **Persistence Timer:**

→ To deal with a zero-window-size advertisement, TCP needs another timer. If the receiving TCP announces a window size of zero, the sending TCP spots transmitting segements until the receiving TCP sends an ACK segment announcing a nonzero window size.

→ This ACK segment can be lost. Remember that ACK segments are not acknowledgemead nor retransmitted in TCP. If this acknowledgement is lost, the receiving TCP thinks that it has done its job and waits for the sending TCP to send more segements. There is no retransmission time for a segement containing only an acknowledgement. The sending TCP has not received an acknowledgement and waits for the other TCP to send an acknowledgement advertising the size of the window.

Both TCPs might continue to wait for each other forever (a deadlock).

→ To correct this deadlock, TCP uses a persistence timer for each connection. when the sending TCP receives an acknowledgement with a window size of zero. it starts a persistence timer. when the ~~p~~ persistence timer goes off, the sending TCP sends TCP sends a special segment called ~~a~~ a probe. This segment contains only 1 byte of new data. It has a sequence number, but its sequence number is never acknowledged; it is even ignored in calculating the sequence number for the rest of the data. The probe causes the receiving TCP to resend the acknowledgement.

→ The value of the persistence timer is set to the value of the retransmission time. However, if a response is not received from the receiver, another probe segement is sent and the value

of the persistence timer is doubled and reset. The sender continues sending the probe segments and doubling and resetting the value of the persistence timer until the value reaches a threshold. After that the sender sends one probe segment every 60s until the window is reopened.

- **Keepalive Timer:**

→ A keepalive timer is used in some implementations to prevent a long idle connection between two TCPs. Suppose that a client opens a TCP connection to a server, transfer some data, and becomes silent. Perhaps the client has crashed. In this case, the connection remains open forever.

→ To remedy this situation, most implementation equip a server with a keepalive timer. Each time the server

hears from a client, it resets this timer. The time-out is usually 2 hours. If the server ~~the~~ does not hear from the client after 2 hours, it sends a probe segement. If there is no response after 10 probes, each of which is 75s apart it assumes that the client is down and terminates the connection.

- TIME-WAIT Timer :-

→ The TIME-WAIT (2MSL) timer is used during connection termination.