Module 4 previous year questions

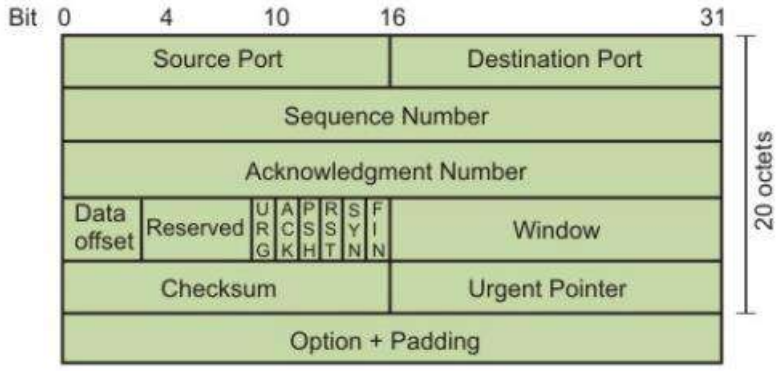
1. Explain the features of TCP – 5 marks

Ans.

- **Connection**: TCP provides connections between clients and servers. A TCP client establishes a connection with a server, exchanges data across the connection, and then terminates the connection.
- **Reliability**: TCP requires acknowledgment when sending data. If an acknowledgment is not received, TCP automatically retransmits the data and waits a longer amount of time.
- **Round-trip time** (RTT): TCP estimates RTT between a client and server dynamically so that it knows how long to wait for an acknowledgment.
- **Sequencing**: TCP associates a sequence number with every byte (**segment**, unit of data that TCP passes to IP.) it sends. TCP reorders out-of-order segments and discards duplicate segments.
- **Flow control**
- **Full-duplex**: an application can send and receive data in both directions on a given connection at any time.

2. Draw and explain TCP Header format – 10 marks

Ans.



• Source port (16 bits): It defines the port number of the application program in the host of the sender

• Destination port (16 bits): It defines the port number of the application program in the host of the receiver

• Sequence number (32 bits): It conveys the receiving host which octet in this sequence comprises the first byte in the segment

• Acknowledgement number (32 bits): This specifies the sequence number of the next octet that receiver expects to receive

• HLEN (4 bits): This field specifies the number of 32-bit words present in the TCP header

• Control flag bits (6 bits):

- URG: Urgent pointer
- ACK: Indicates whether acknowledge field is valid
- PSH: Push the data without buffering
- RST: Resent the connection
- SYN: Synchronize sequence numbers during connection establishment
- FIN: Terminate the connection

• Window (16 bits): Specifies the size of window

• Checksum (16 bits): Checksum used for error detection.

• User pointer (16 bits): Used only when URG flag is valid

• Options: Optional 40 bytes of information

3. The following is a dump of a TCP header in hexadecimal format:
   05320017 00000001 00000000 500207FF 00000000 – 5 Marks
   i. What is the source port number?
   ii. What is the destination port number?
   iii. What is the length of the header?
   iv. What is the type of segment?
   v. What is the window size?

Ans.

The dump of a TCP header in hexadecimal format:

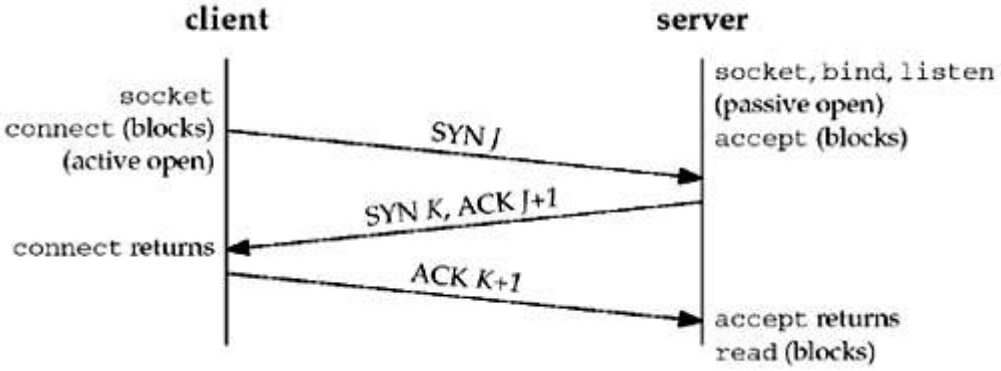
05320017 00000001 00000000 500207FF 00000000

i. Source port number: - (2 byte) -> 0532(in hex)
   or
   $(0532)_{16} = (0 \times 16^3) + (5 \times 16^2) + (3 \times 16^1) + (2 \times 16^0) = 1330$ (in decimal)
ii. Destination port number: - (2 byte) -> 0017(in hex) or
   $(0017)_{16} = (0 \times 16^3) + (0 \times 16^2) + (1 \times 16^1) + (7 \times 16^0) = 23$(in decimal)
iii. Length of the header (4 bits) -> 5(in hex and decimal) Means 5*4= 20 bytes header
iv. Type of the segment -> 0X02: - is control field and this indicates a SYN packet.
   (5002)->0101 0000 0000 0010->0101 is header, 000000 is reserved,
   Urg=0,ack=0,psh=0,rst=0,syn=1,fin=0
v. Window size -> 07FF (in hex) or
   $(07FF)_{16} = (0 \times 16^3) + (7 \times 16^2) + (15 \times 16^1) + (15 \times 16^0) = 2047$ (in decimal)

4. Explain connection establishment in TCP using three-way handshaking
5. Explain three-way handshaking for connection establishment in TCP
6. Draw and explain connection establishment using 3 way handshaking in TCP
7. Explain with diagram the connection establishment and connection termination in TCP using Three way handshaking

10 Marks

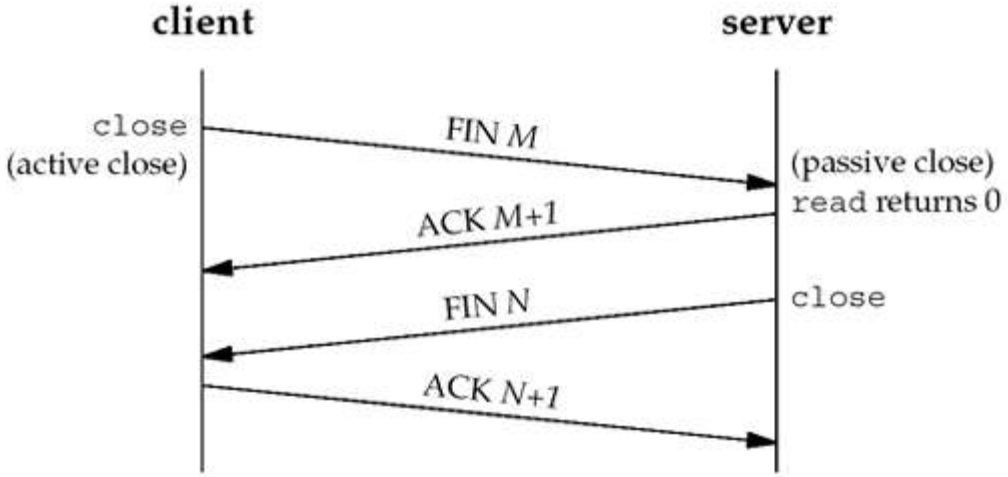
Ans. Three-Way Handshake - TCP Connection Establishment



1. Server: **passive open**, by calling socket, bind, and listen
2. Client: **active open**, by calling connect. The client TCP to send a "synchronize" (SYN) segment with no data but it contains client's initial sequence number for the data to be sent on the connection.
3. Server: acknowledges (ACK) client's SYN. The server sends its SYN and the ACK of the client's SYN in a single segment which also contains its own SYN containing the initial sequence number for the data to be sent on the connection.
4. Client: acknowledges the server's SYN.

The client's initial sequence number as *J* and the server's initial sequence number as *K*. The acknowledgment number in an ACK is the next expected sequence number for the end sending the ACK. Since a SYN occupies one byte of the sequence number space, the acknowledgment number in the ACK of each SYN is the initial sequence number plus one.

TCP Connection Termination

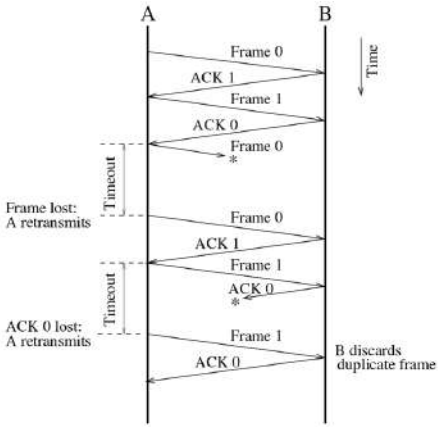


It takes four segments to terminate a connection:

1. One end calls close first by sending a FIN segment to mean it is finished sending data. This is called **active close**.
2. The other end that receives the FIN performs the **passive close**. The received FIN is acknowledged by TCP (sending an ACK segment). The receipt of the FIN is also passed to the application as an end-of-file.
3. Sometime later, the application that received the end-of-file will close its socket. This causes its TCP to send a FIN.
4. The TCP on the system that receives this final FIN (the end that did the active close) acknowledges the FIN

8. Write short note on flow control using TCP

Ans. **Stop and Wait Flow Control**



i. Use sequence numbers to individually identify each frame and the corresponding acknowledgement.
ii. Acknowledgement contains sequence no. of *next expected frame* in TCP actually.

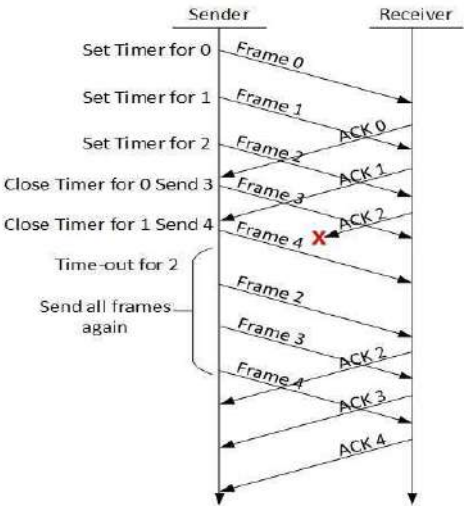
Problem with Stop and Wait

- Every packet needs to wait for the acknowledgement of the previous packet.

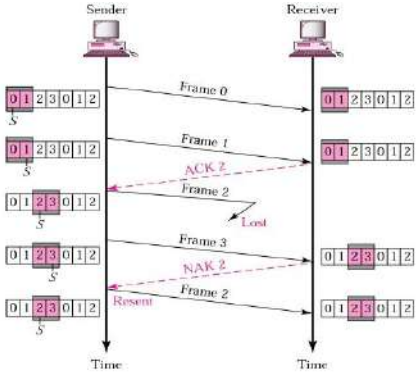
- For bidirectional connections – use two instances of the stop and wait protocol at both directions – further waste of resources
- A possible solution: Piggyback (Piggybacking is the technique of delaying outgoing acknowledgment and attaching it to the next data packet)data and acknowledgement from both the directions.
- Reduce resource waste based on **sliding window protocols (a pipelined protocol)**

Go Back N ARQ

- If segment N is lost, all the segments from segment 0 (start of the sliding window) to segment N are retransmitted.



Selective Repeat ARQ

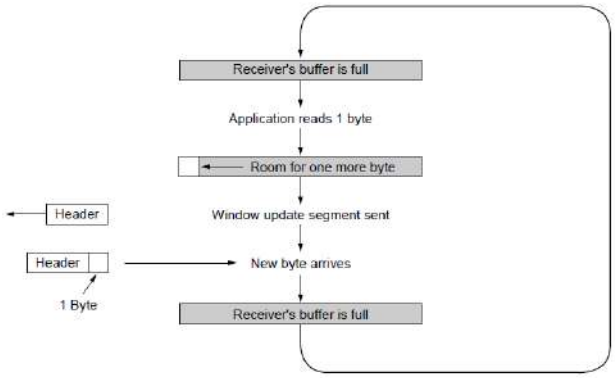


- **Selective Repeat (SR) ARQ:** Only the lost packets are selectively retransmitted
  - **Negative Acknowledgement (NAK) or Selective Acknowledgements (SACK):** Informs the sender about which packets need to be retransmitted (not received by the receiver)

Silly Window Syndrome is a problem that arises due to poor implementation of TCP. It degrades the TCP performance and makes the data transmission

extremely inefficient. The problem is called so because:

- It causes the sender window size to shrink to a silly value.
- The window size shrinks to such an extent that the data being transmitted is smaller than TCP Header.



Nagles's algorithm: This algorithm improves the efficiency of TCP /IP networks by reducing the number of packets that need to be sent over the network.

Nagle's algorithm works by combining a number of small outgoing messages and sending them all at once. Specifically, as long as there is a sent packet for which the sender has received no acknowledgment, the sender should keep buffering its output until it has a full packet's worth of output, thus allowing output to be sent all at once.

Clark's solution suggests:

- Receiver should not send a window update for 1 byte.
- Receiver should wait until it has a decent amount of space available.
- Receiver should then advertise that window size to the sender

9. Short note on congestion control in TCP
10. Explain TCP Congestion control policy

10 Marks

Ans.

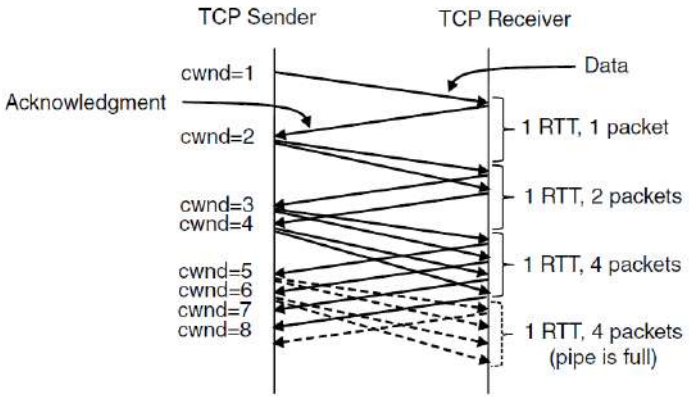
- TCP congestion control is a method used by the TCP protocol to manage data flow over a network and prevent congestion.
- TCP uses a congestion window and congestion policy that avoids congestion

Congestion Policy in TCP

- Slow Start Phase: Starts slow increment is exponential to the threshold.
- Congestion Avoidance Phase: After reaching the threshold increment is by 1.
- Congestion Detection Phase: The sender goes back to the Slow start phase or the Congestion avoidance phase

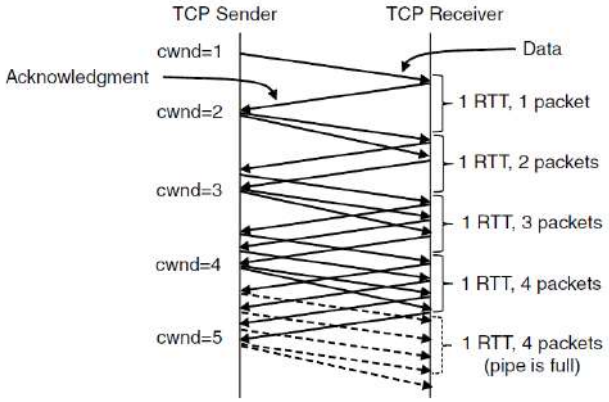
Slow Start Phase

- **Exponential increment**: In this phase after every RTT the congestion window size increments exponentially.
- **Example:-** If the initial congestion window size is 1 segment, and the first segment is successfully acknowledged, the congestion window size becomes 2 segments. If the next transmission is also acknowledged, the congestion window size doubles to 4 segments. This exponential growth continues as long as all segments are successfully acknowledged.



Congestion Avoidance Phase

- **Additive increment:** This phase starts after the threshold value also denoted as ssthresh. The size of cwnd(congestion window) increases additive. After each RTT cwnd = cwnd + 1.
- **Example:-** if the congestion window size is 20 segments and all 20 segments are successfully acknowledged within an RTT, the congestion window size would be increased to 21 segments in the next RTT. If all 21 segments are again successfully acknowledged, the congestion window size would be increased to 22 segments, and so on.



Additive increase from an initial congestion window of 1 segment

Congestion Detection Phase

- **Multiplicative decrement:** If congestion occurs, the congestion window size is decreased. The only way a sender can guess that congestion has happened is the need to retransmit a segment. Retransmission is needed to recover a missing packet that is assumed to have been dropped by a router due to congestion. Retransmission can occur in one of two cases: when the RTO timer times out or when three duplicate ACKs are received.

11. List and explain various timers in TCP
12. What are the types of timers used in TCP

Ans. TCP Timers

- Retransmission
- Persistence
- Keepalive
- Time-waited

Retransmission timer

- For lost or discarded segment, TCP employs a retransmission timer
- Measures waiting time for an ack of a segment
- When TCP sends a segment, time is created for that segment
- If an ack is received for the segment before timer expires, timer is destroyed
- If timer expires before ack arrives, the segment is retransmitted and timer is reset

Persistence timer

- Addresses zero (0) window size advertisement
- Sender will stop sending until ack received from destination TCP
- If ack gets lost, destination TCP will wait indefinitely for more data from the sender
- This deadlock situation must be avoided
- After persistence timer elapses, sender sends a probe segment (only 1 byte)
- Probe alerts destination TCP that ack was lost and must be resent

Keep Alive timer

- Implemented in some TCP servers
- Prevents a long idle connection between two connected TCP implementations
- Timer is typically set at 2 hours
- After timer elapses, 10 “probe” segments are rapidly sent
- If no response after 10 probes, it is assumed that the client is down so connection is terminated

Time-waited timer

- Used during connection termination
- Keeps connection alive long enough for any remaining FIN segments to arrive (which are then discarded)
- The MSL is the maximum time a segment can exist in the Internet before it is dropped.The common value for MSL is between 30 seconds and 1 minute

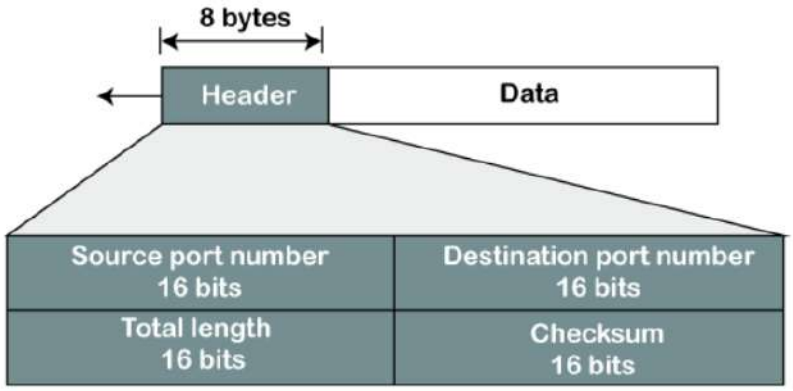
13. How UDP is different from TCP for data transmission

Ans.

Basis	Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Type of Service	TCP is a connection-oriented protocol. Connection orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.	UDP is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, or terminating a connection. UDP is efficient for broadcast and multicast types of network transmission.
Reliability	TCP is reliable as it guarantees the delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
Error checking mechanism	TCP provides extensive error-checking mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error-checking mechanism using checksums.
Acknowledgment	An acknowledgment segment is present.	No acknowledgment segment.
Sequence	Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in order at the receiver.	There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer.

Speed	TCP is comparatively slower than UDP.	UDP is faster, simpler, and more efficient than TCP.
Retransmission	Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in the User Datagram Protocol (UDP).
Header Length	TCP has a (20-60) bytes variable length header.	UDP has an 8 bytes fixed-length header.
Weight	TCP is heavy-weight.	UDP is lightweight.
Handshaking Techniques	Uses handshakes such as SYN, ACK, SYN-ACK	It's a connectionless protocol i.e. No handshake
Broadcasting	TCP doesn't support Broadcasting.	UDP supports Broadcasting
Protocols	TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet.	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.
Stream Type	The TCP connection is a byte stream.	UDP connection is a message stream.
Overhead	Low but higher than UDP.	Very low.
Applications	This protocol is primarily utilized in situations when a safe and trustworthy communication procedure is necessary, such as in email, on the web surfing, and in military services.	This protocol is used in situations where quick communication is necessary but where dependability is not a concern, such as VoIP, game streaming, video, and music streaming, etc

Q. The following is a dump of a udp header in hexadecimal format: CB84000D001C001C



i. What is the source port number? CB84 -> 52100

ii. What is the destination port number?000D

$(000D)_{16} = (0 \times 16^3) + (0 \times 16^2) + (0 \times 16^1) + (13 \times 16^o) = (13)$

iii. What is the total length of the user datagram? 001C
$(001C)_{16} = (0 \times 16^3) + (0 \times 16^2) + (1 \times 16^1) + (12 \times 16^o) = (28)_{10}$

iv. What is the length of the data? Total length- Header = 28-8=20

v. Is the packet directed from a client to server or vice versa?

At server side,

0-1023 port numbers are called as well known port numbers. These are the port numbers assigned to diff services that are running at the application layer.

Here the destination port number 13 relies in this range. That means the packet is directed to server.

For example, a remote job entry application has the port number of 5; the Hypertext Transfer Protocol (HTTP) application has the port number of 80; and the Post Office Protocol Version 3 (POP3) application, commonly used for e-mail delivery, has the port number of 110