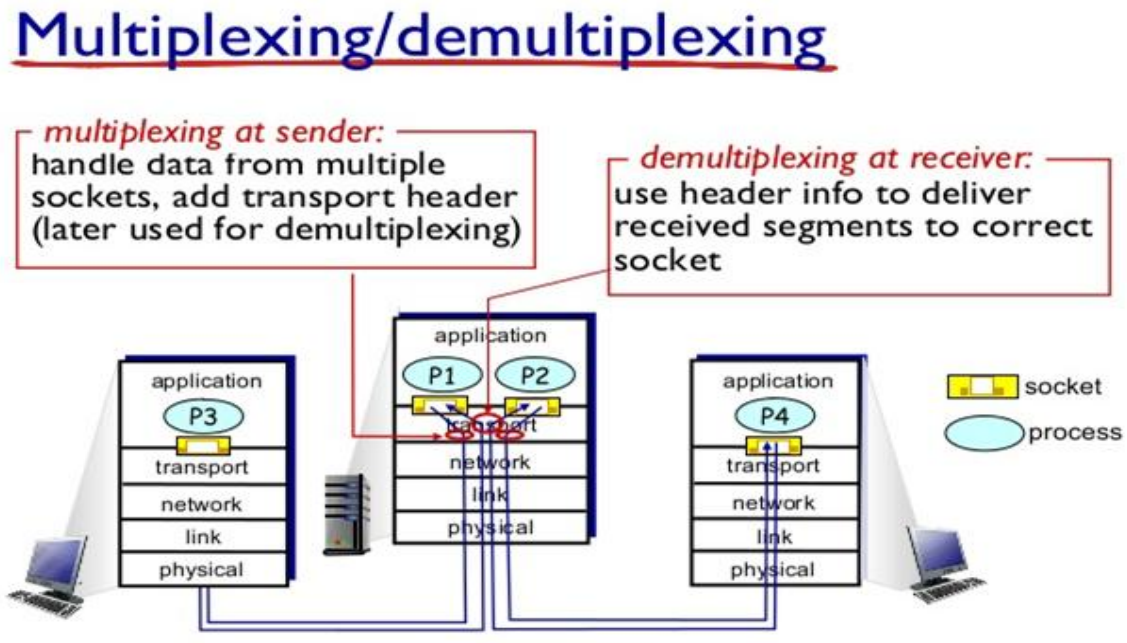
Chapter-3 Transport Layer MIMP Questions

Q.1 What is multiplexing in computer networks? (Nov-2016, June-2017, Dec-2015, Nov-2017)

Ans:

- The job of gathering data chunks at the source host from different sockets, encapsulating each data chunk with header information (that will later be used in demultiplexing) to create segments, and passing the segments to the network layer is called multiplexing.
- At the receiving end, the transport layer examines these fields to identify the receiving socket and then directs the segment to that socket. This job of delivering the data in a transport-layer segment to the correct socket is called demultiplexing.
- Transport layer in the middle host in Figure 1 must demultiplex segments arriving from the network layer below to either process P1 or P2 above; this is done by directing the arriving segment's data to the corresponding process's socket.
- The transport layer in the middle host must also gather outgoing data from these sockets, form transport-layer segments, and pass these segments down to the network layer.



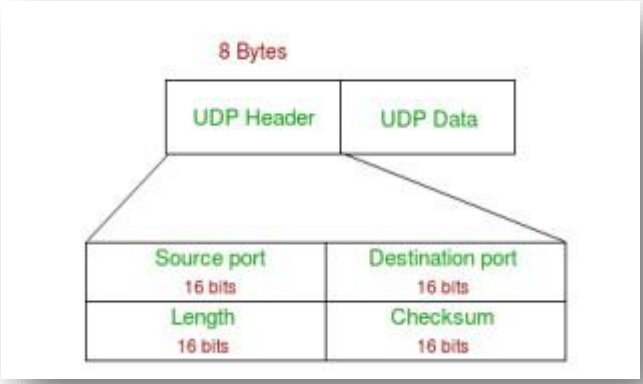
Q.2 The following is a dump of a UDP header in hexadecimal format.

CB84000D001C001C

i. **What is the source port number?**
ii. **What is the destination port number?**
iii. **What is the total length of the user datagram?**

What is the length of the data? (May-2016)

Ans:



i) The source port number is the first four hexadecimal digits.

$(CB84)_{16}$ or $(52100)_{decimal}$

ii) The destination port number is the second four hexadecimal digits.

$(000D)_{16}$ or $(13)_{decimal}$

iii) The third four hexadecimal digits define the length of the whole UDP packet.

$(001C)_{16}$ or $(28\ bytes)_{decimal}$

iv) The length of the data is the length of the whole packet minus the length of the header.

28 - 8 = 20 bytes.

Q.3 List and explain the services provided by the transport layer. (June-2017)

Ans:

1. Process to process delivery .
2. End-to-end Connection between hosts.
3. Multiplexing and Demultiplexing.
4. Congestion Control.
5. Data integrity and Error correction.
6. Flow control

a. Process to process delivery:

i. While Data Link Layer requires the MAC address of source- destination hosts

to correctly deliver a frame and Network layer requires the IP address for appropriate routing of packets , in a similar way Transport Layer requires a Port number to correctly deliver the segments of data to the correct process amongst the multiple processes running on a particular host.

ii. A port number is a 16 bit address used to identify any client-server program uniquely.

b. End-to-end Connection between hosts:

i. Transport layer is also responsible for creating the end-to-end Connection between hosts for which it mainly uses TCP and UDP.

ii. TCP is a secure, connection- orientated protocol which uses a handshake protocol to establish a robust connection between two end- hosts.

iii. TCP ensures reliable delivery of messages and is used in various applications.

iv. UDP on the other hand is a stateless and unreliable protocol which ensures best-effort delivery.

C. Multiplexing and Demultiplexing:

i. Multiplexing allows simultaneous use of different applications over a network which are running on a host. Transport layer provides this mechanism which enables us to send packet streams from various applications simultaneously over a network.

ii. Transport receives the segments of data from network layer and delivers it to the appropriate process running on the receiver's machine.

d. Congestion Control:

i. Congestion is a situation in which too many sources over a network attempt to send data and the router buffers start overflowing due to which loss of packets occur.

ii. In this situation Transport layer provides Congestion Control in different ways.

iii. It uses open loop congestion control to prevent the congestion and closed loop congestion control to remove the congestion in a network once it occurred.

e. Data integrity and Error correction:

i. Transport layer checks for errors in the messages coming from application layer by using error detection codes, computing checksums, it checks whether the received data is not corrupted and uses the ACK and NACK.

ii. Services to inform the sender if the data is arrived or not and checks for the integrity of data.

f. Flow control:

i. Transport layer provides a flow control mechanism between the adjacent layers of the TCP/IP model.

ii. TCP also prevents the data loss due to a fast sender and slow receiver by imposing some flow control techniques.

iii. TCP also prevents the data loss due to a fast sender and slow receiver by imposing some flow control techniques.

Q.4 How UDP checksum value is calculated? Explain with suitable example. (May-2016)

Ans:

- The checksum is used to determine whether bits within the UDP segment have been altered (for example, by noise in the links or while stored in a router) as it moved from source to destination.
- UDP at the sender side performs the **1's complement** of the sum of all the 16-bit words in the segment, with any overflow encountered during the sum being wrapped around.
- This result is put in the checksum field of the UDP segment.

suppose that we have the following three 16-bit words:

0110011001100000

0101010101010101

1000111100001100

The sum of first two of these 16-bit words is

0110011001100000

0101010101010101

1011101110110101

Adding the third word to the above sum gives

1011101110110101

1000111100001100

0100101011000010

Note:-

- This last addition had overflow, which was wrapped around. The 1s complement is obtained by converting all the 0s to 1s and converting all the 1s to 0s. Thus the 1s complement of the sum 0100101011000010 is **1011010100111101**.
- At the receiver, all four 16-bit words are added, including the checksum.

- If no errors are introduced into the packet, then clearly the sum at the receiver will be **1111111111111111.**
    - If one of the bits is a **0**, then we know that errors have been introduced into the packet.

Q.5 What do you mean by congestion and overflow? Explain the slow-start component of the TCP congestion-control algorithm. (May-2015)

Ans:

- **Congestion:** An abnormal or excessive accumulation of a body fluid. The term is used broadly in medicine. Examples include nasal congestion (excess mucus and secretions in the air passages of the nose) seen with a common cold and congestion of blood in the lower extremities seen with some types of heart failure.
- An error that occurs when the computer attempts to handle a number that is too large for it. ... If during execution of a program it arrives at a number outside this range, it will experience an overflow error. Overflow errors are sometimes referred to as overflow conditions.
- Slow start is part of the congestion control strategy used by TCP in conjunction with other algorithms to avoid sending more data than the network is capable of forwarding, that is, to avoid causing network congestion. The algorithm is specified by RFC 5681.
- **Definition:** TCP slow start is an algorithm which balances the speed of a network connection. Slow start gradually increases the amount of data transmitted until it finds the network's maximum carrying capacity.
- TCP slow start is one of the first steps in the congestion control process. It balances the amount of data a sender can transmit (known as the congestion window) with the amount of data the receiver can accept (known as the receiver window). The lower of the two values becomes the maximum amount of data that the sender is allowed to transmit before receiving an acknowledgment from the receiver.

STEPS OF SLOW START:

1. A sender attempts to communicate to a receiver. The sender's initial packet contains a small congestion window, which is determined based on the sender's maximum window.
2. The receiver acknowledges the packet and responds with its own window size. If the receiver fails to respond, the sender knows not to continue sending data.
3. After receiving the acknowledgement, the sender increases the next packet's window size. The window size gradually increases until the receiver can no longer acknowledge each packet, or until either the sender or the receiver's window limit is reached.
4. Once a limit has been determined, slow start's job is done. Other congestion control algorithms take over to maintain the speed of the connection.

EXAMPLE OF TCP SLOW START:

I. Content providers often adjust their slow start window to maximize performance.

II. The initial congestion window parameter (initcwnd) can have a significant impact on the speed of a network. When the receiver has to send fewer acknowledgments to the sender, more data can be transmitted faster.

III. Most large CDN providers default to an initcwnd of 10, meaning the CDN will transmit 10 packets before requesting an acknowledgment. Cachefly has an initcwnd of 70, compared to Microsoft's initcwnd of 2. For comparison, MaxCDN has an initcwnd of about 30. A good balance will be determined by the type of data transmitted and the general speed of the network.

BENEFITS OF TCP SLOW START:

I. Slow start ensures the speed and integrity of a network while protecting content providers and consumers.

II. Users experience uninterrupted connections since packets are no longer dropped due to congestion.

III. Users also experience faster downloads since slow start finds and uses the maximum connection speed.

IV. Enterprises see less network congestion since slow start regulates bandwidth and prevents the sender from having to continuously retransmit data.

Q.6 Explain the TCP Segment structure and justify the importance of its field values.

OR

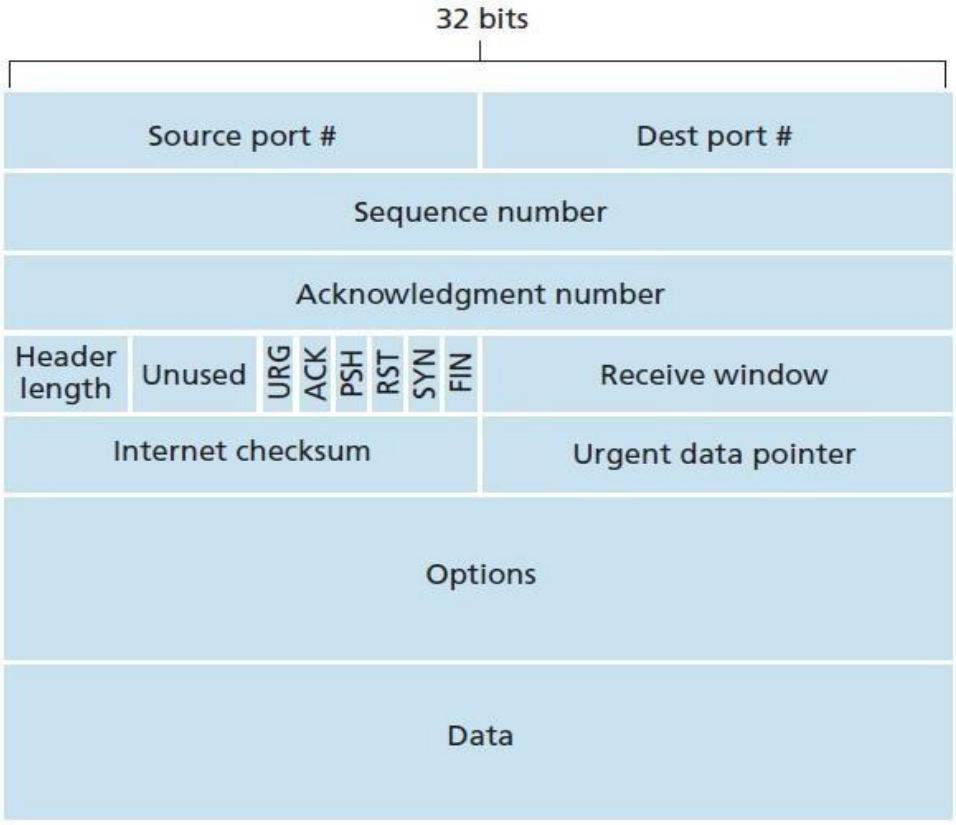
Explain Transmission Control Protocol with TCP header fields. (May-2015, May-2016, Nov-2017)

Ans:

- The segment consists of a header of 20 to 60 bytes, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.
- **A TCP segment header contains the following fields:**
- **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. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence is the first byte in the segment.
- **Acknowledgment 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 acknowledgment number. Acknowledgment and data can be piggybacked together.
- **Header length:** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the

value of this field is always between 5 (5 *4 =20) and 15 (15 *4 =60).

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



- **Control Flags:** This field defines 6 different control bits or flags as shown in Figure. One or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.
- **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 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver.
- **Checksum:** This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP.
- **Urgent pointer:** This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.
- **Options:** There can be up to 40 bytes of optional information in the TCP header for example end of operation (EOP), Timestamp etc.

Q.7 Compare UDP and TCP (June-2017, Nov-2017)

Ans:

TRANSMISSION CONTROL PROTOCOL (TCP)	USER DATAGRAM PROTOCOL (UDP)
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, and terminating a connection. UDP is efficient for broadcast and multicast type of network transmission.
TCP is reliable as it guarantees delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
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.
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 ordering is required, it has to be managed by the application layer.
TCP is comparatively slower than UDP.	UDP is faster, simpler and more efficient than TCP.
Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in User Datagram Protocol (UDP).
TCP header size is 20 bytes.	UDP Header size is 8 bytes.
TCP is heavy-weight.	UDP is lightweight.
TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.

Q.8 What is congestion? List the approaches congestion control. (Nov-2016)

Ans:

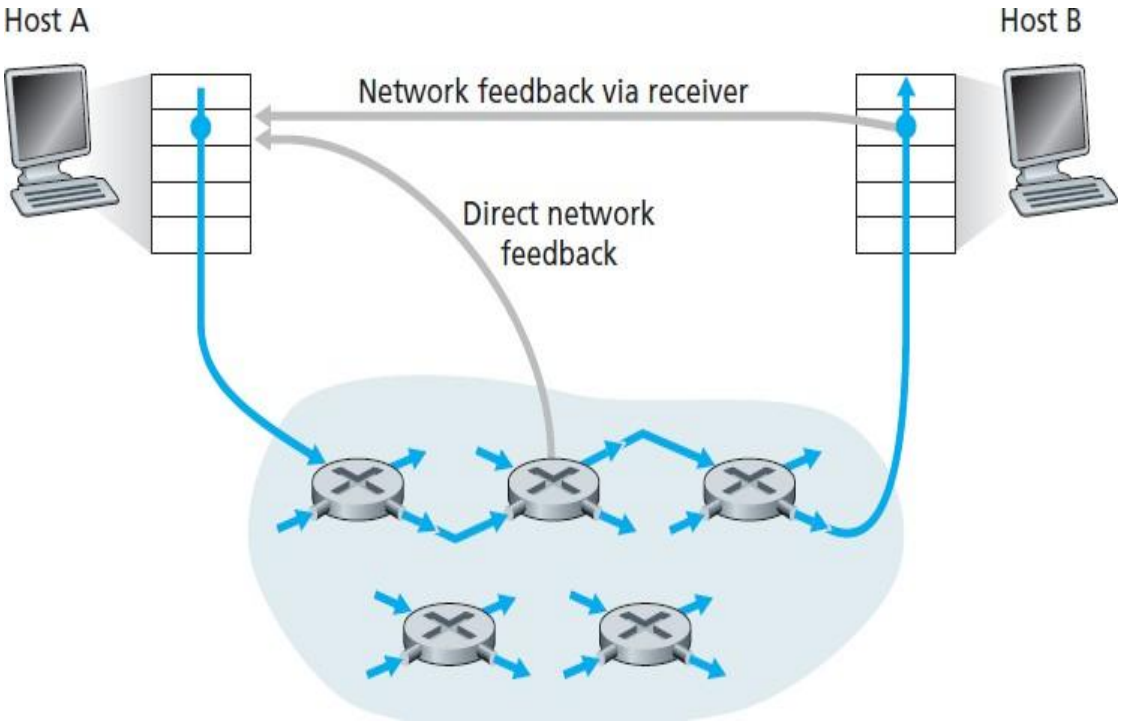
congestion:

- **informally:** "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- **Cost of congestion:**
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
  - Retransmissions

- **Approaches towards congestion control :-**
- two broad approaches towards congestion control:

  (i) end-end congestion control:
  - no explicit feedback from network
  - congestion inferred from end-system observed loss, delay
  - approach taken by TCP

  (ii) network-assisted congestion control:
  - routers provide feedback to end systems
  - single bit indicating congestion.
  - Choke packets



Q.9 Draw the reliable data transfer service model. OR Discuss the principles of reliable data transfer. (Nov-2016, June-2017)

Ans:

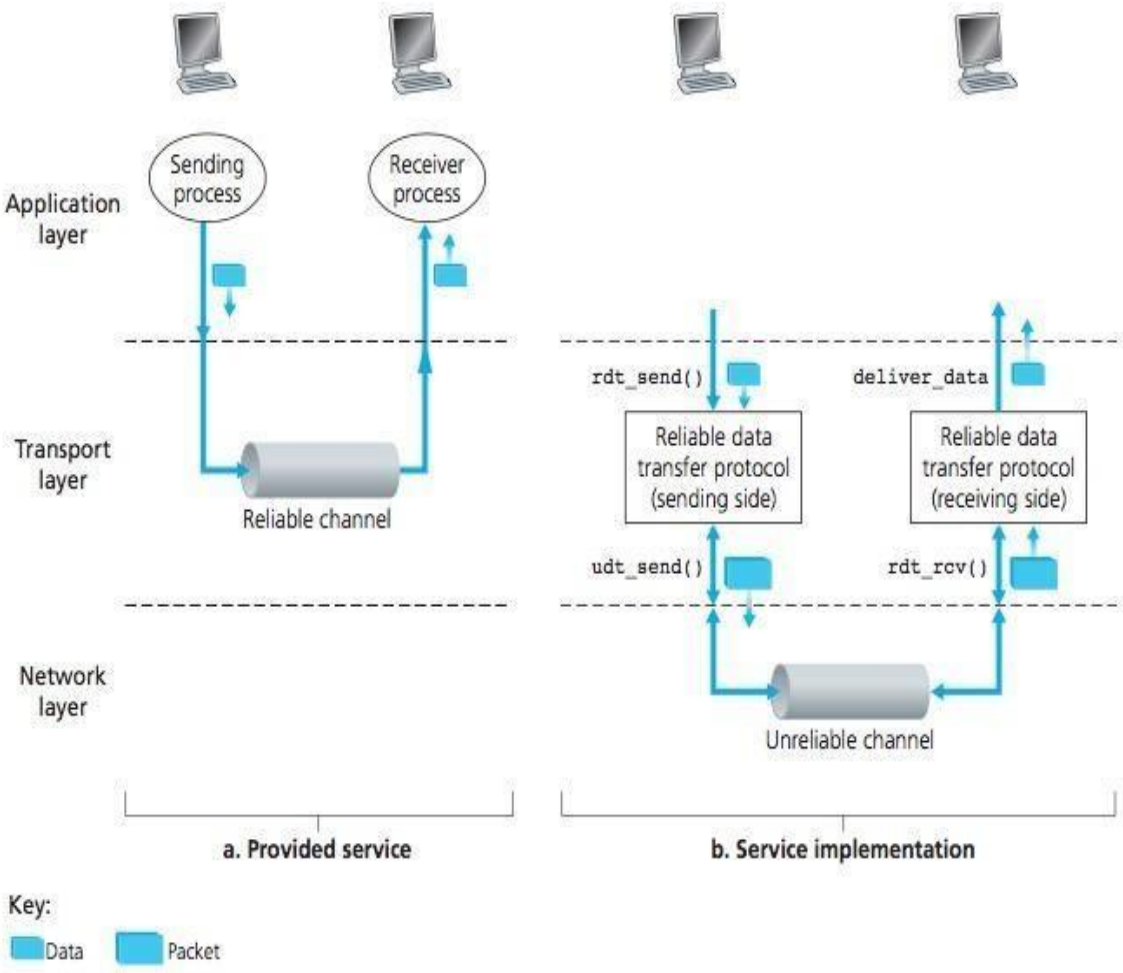


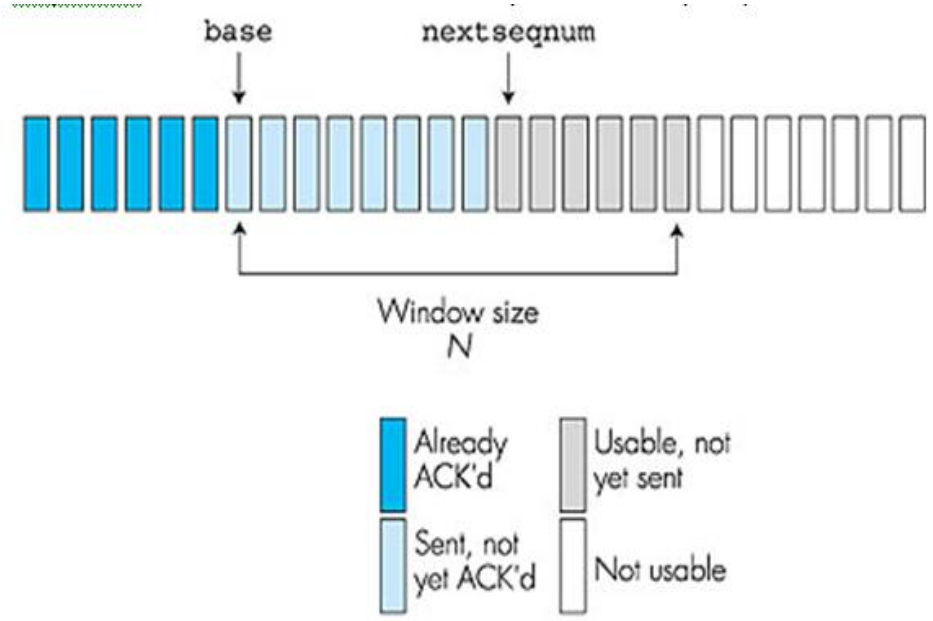
Figure 3.8 ♦ Reliable data transfer: Service model and service implementation

- The sending side of the data transfer protocol will be invoked from above by a call to **rdt_send().** It will pass the data to be delivered to the upper layer at the receiving side. (Here rdt stands for *reliable data transfer)*
- On the receiving side, **rdt_rcv()** will be called when a packet arrives from the receiving side of the channel.
- When the rdt protocol wants to deliver data to the upper layer, it will do so by calling **deliver_data().**
- **udt_send():** called by rdt, to transfer packet over unreliable channel to receiver.

Q.10 How pipeline approach improves the overall sender utilization time? Explain Go-Back-N pipeline approach in transport layer? (May-2015)

Ans:

- In a Go-Back-N (GBN) protocol, the sender is allowed to transmit multiple packets (when available) without waiting for an acknowledgment, but is constrained to have no more than some maximum allowable number, N, of unacknowledged packets in the pipeline. Figure shows the sender's view of the range of sequence numbers in a GBN protocol.



- If we define base to be the sequence number of the oldest unacknowledged packet and next segment number to be the smallest unused sequence number (i.e., the sequence number of the next packet to be sent), then four intervals in the range of sequence numbers can be identified. Sequence numbers in the interval [0, base-1] correspond to packets that have already been transmitted and acknowledged.
- The interval [base, nextseqnum-1] corresponds to packets that have been sent but not yet acknowledged. Sequence numbers in the interval [nextseqnum, base+N-1] can be used for packets that can be sent immediately, should data arrive from the upper layer.
- Finally, sequence numbers greater than or equal to base+N cannot be used until an unacknowledged packet currently in the pipeline has been acknowledged.
- As suggested by Figure, the range of permissible sequence numbers for transmitted but not-yet-acknowledged packets can be viewed as a ``window'' of size N over the range of sequence numbers. As the protocol operates, this window slides forward over the sequence number space. For this reason, N is often referred to as the window size and the GBN protocol itself as a sliding window protocol.

Q. 11 How many packets overhead while doing the data communication using TCP? Draw the TCP connection establishment and termination process with diagram. OR
Explain connection establishment and connection release in Transport protocols. (June-2017)

Ans:

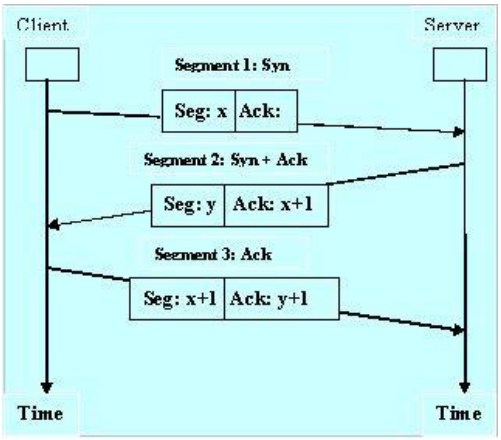
Connection Establishment: Connection establishment is performed by the concept called Three-way Handshake.To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the three-way handshake occurs:

1. The active open is performed by the client sending a SYN to the server.
2. In response, the server replies with a SYN-ACK.
3. Finally the client sends an ACK back to the server.

At this point, both the client and server have received an acknowledgement of the connection.

Example:

1. The initiating host (client) sends a synchronization packet (SYN flag set to 1) to initiate a connection. It sets the packet's sequence number to a random value x.
2. The other host receives the packet, records the sequence number x from the client, and replies with an acknowledgement and synchronization (SYN-ACK). The Acknowledgement is a 32-bit field in TCP segment header. It contains the next sequence number that this host is expecting to receive (x+1). The host also initiates a return session. This includes a TCP segment with its own initial Sequence Number of value y.
3. The initiating host responds with the next Sequence Number (x+1) and a simple Acknowledgement Number value of y+1, which is the Sequence Number value of the other host +1.



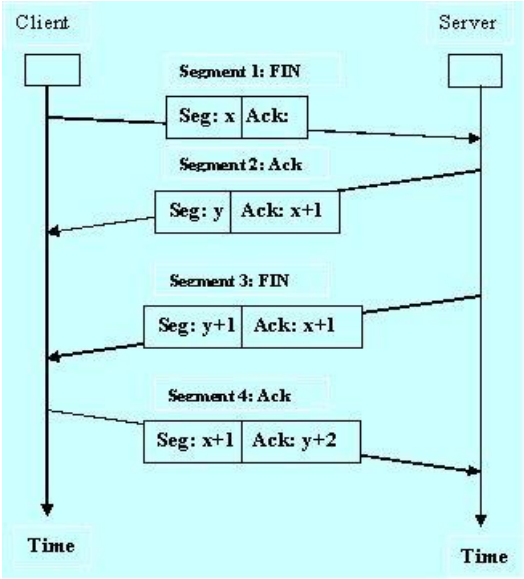
Connection Release: Connection release is performed by a concept called Four-way hand shake.]

The server as well as client both should participate in the connection release. When connection in one direction is terminated, the other party can continue sending data in the

other direction. Four steps need to perform the connection release from both server and client.

The four steps are as follows,

1. The client TCP sends the FIN segment first.
2. The server TCP sends the ACK segment to confirm the receipt of the FIN from the client. It increments the sequence number of FIN by 1 and no other user data will add with the ACK segment.
3. Server does not have any data for transmission, then it sends the FIN segment to Client side.
4. Then client sends the ACK segment again to the server side. The connection termination fulfilled.



Q.12 Explain rdt2.0 with FSM diagram. (Dec-2015)

Ans:

➢ Transfer Over a channel with Bit Error:

- A more realistic model of the underlying channel is one in which bits in a packet may be corrupted. Such bit errors typically occur in the physical components of a network as a packet is transmitted, propagates, or is buffered.
- To recover from bit errors new mechanism is required which will inform sender regarding its message is correctly received or not: ARQ (Automatic Repeat Request) protocol.

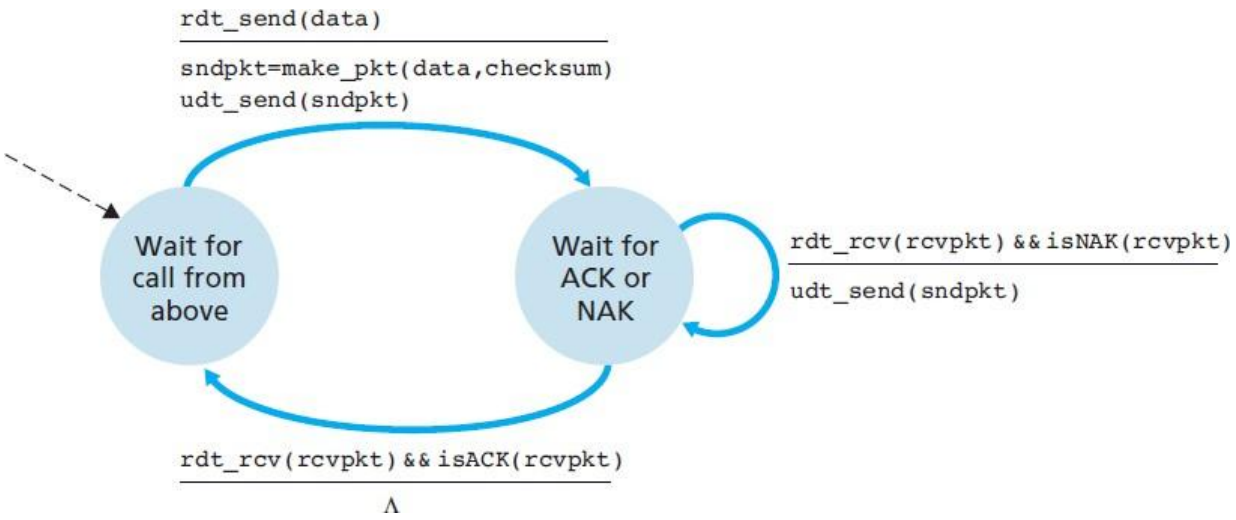
It includes two types of messages:-

- **Acknowledgment:-** for correctly received packet
- **Negative Acknowledgement:-** for corrupted packet

- Fundamentally, three additional protocol capabilities are required in ARQ protocols to handle the presence of bit errors:
  - **Error detection: -** First, a mechanism is needed to allow the receiver to detect when bit errors have occurred. The Internet checksum field is for

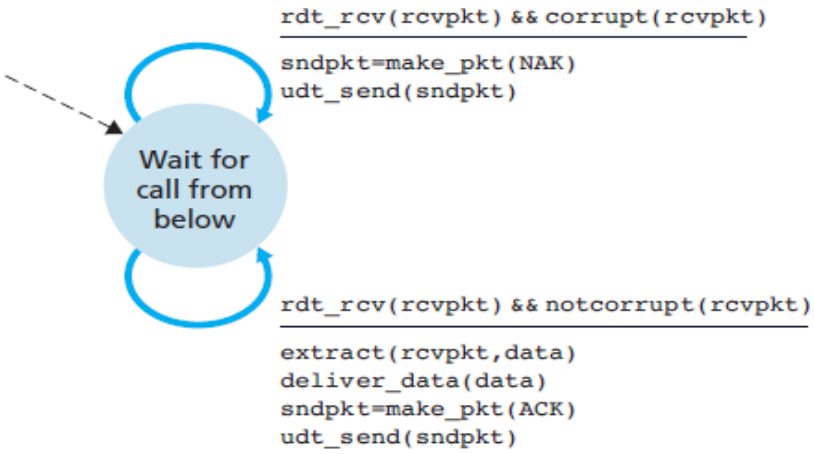
exactly this purpose.

- **Receiver feedback: -** The only way for the sender to learn whether or not a packet was received correctly is to provide explicit feedback to the sender. The positive (ACK) and negative (NAK) acknowledgment replies in the message-dictation scenario are examples of such feedback. Our rdt2.0 protocol. Feedback is one bit long; for example, a 0 value could indicate a NAK and a value of 1 could indicate an ACK.
- **Retransmission: -** A packet that is received in error at the receiver will be retransmitted by the sender.



rdt2.0:- Sending Side

- When the **rdt_send(data)** event occurs, the sender will create a packet containing the data to be sent, along with a packet checksum and then send the packet via the **udt_send(sndpkt)** operation.
- In the rightmost state, the sender protocol is waiting for an ACK or a NAK packet from the receiver. If an ACK packet is received **(the notation rdt_rcv(rcvpkt) && isACK (rcvpkt))**, then packet received correctly by receiver and thus the protocol returns to the state of waiting for data from the upper layer.
- If a NAK is received, the protocol retransmits the last packet and waits for an ACK or NAK to be returned by the receiver.
- When the sender is in the wait-for-ACK-or-NAK state, it cannot get more data from the upper layer; that is, the rdt_send() event cannot occur; that will happen only after the sender receives an ACK and leaves this state.
- Because of this behavior, protocols such as rdt2.0 are known as **stop-and-wait Protocol.**



rdt2.0:- Receiving Side

- The receiver-side FSM for rdt2.0 still has a single state.
- On packet arrival, the receiver replies with either an ACK or a NAK, depending on whether or not the received packet is corrupted. The **notation rdt_rcv(rcvpkt) && corrupt(rcvpkt)** corresponds to the event in which a packet is received and is found to be in error.

➢ Problem in rdt2.0:

It has a fatal error. No possibility of corruption of the ACK or NAK has been accounted. Minimally, we will need to add checksum bits to ACK/NAK packets in order to detect such errors.

Consider three possibilities for handling corrupted ACKs or NAKs:

1. Add third type of message indicating ACK/NAK is corrupted.
2. A second alternative is to add enough checksum bits to allow the sender not only to detect, but also to recover from, bit errors.
3. A third approach is for the sender simply to resend the current data packet when it receives a garbled ACK or NAK packet.
    - This approach, however, introduces duplicate packets into the sender-to-receiver channel. The fundamental difficulty with duplicate packets is that the receiver doesn't know whether the ACK or NAK it last sent was received correctly at the sender.

➢ Solution to duplicate packets:

- A simple solution to this new problem is to add a new field to the data packet and have the sender number its data packets by putting a sequence number into this field.
- The receiver then need only check this sequence number to determine whether or not they received packet is a retransmission.
- For this simple case of a stop-and wait protocol, a 1-bit sequence number will be sufficient.

Q.13 Describe the sliding window protocol. (Nov-2017)

Ans:

- In sliding window method, multiple frames are sent by sender at a time before need.

- Multiple frames sent by source are acknowledged by receiver using a single ACK frame.

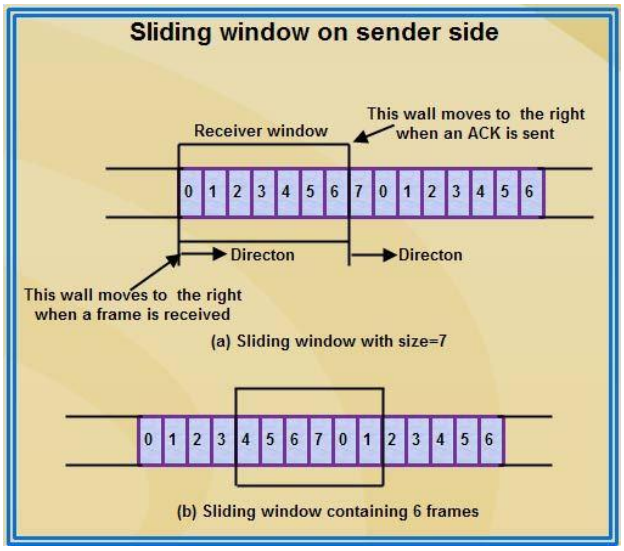
Sliding Window:

- Sliding window refers to an imaginary boxes that hold the frames on both sender and receiver side.
- It provides the upper limit on the number of frames that can be transmitted before requiring an acknowledgment.
- Frames may be acknowledged by receiver at any point even when window is not full on receiver side.
- Frames may be transmitted by source even when window is not yet full on sender side.
- The windows have a specific size in which the frames are numbered modulo- n, which means they are numbered from 0 to n-l. For e.g. if n = 8, the frames are numbered 0, 1,2,3,4,5,6, 7, 0, 1,2,3,4,5,6, 7, 0, 1, ...
- The size of window is n-1. For e.g. In this case it is 7. Therefore, a maximum of n-l frames may be sent before an acknowledgment.
- When the receiver sends an ACK, it includes the number of next frame it expects to receive. For example in order to acknowledge the group of frames ending in frame 4, the receiver sends an ACK containing the number 5. When sender sees an ACK with number 5, it comes to know that all the frames up to number 4 have been received.

Sliding window

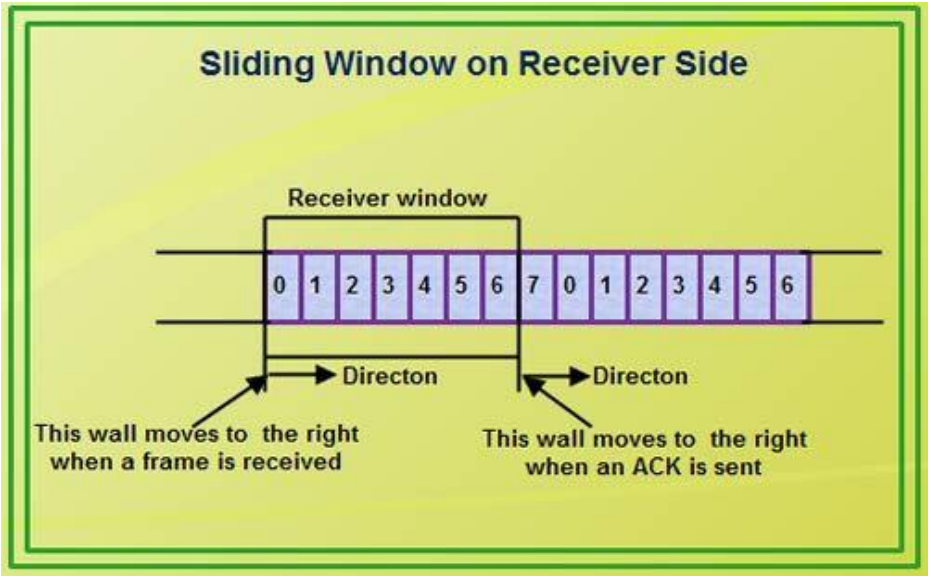
Sliding window on sender side:

- At the beginning of a transmission, the sender's window contains n-l frames
- As the frames are sent by source, the left boundary of the window moves inward, shrinking the size of window. This means if window size is w, if four frames are sent by source after the last acknowledgment, then the number of frames left in window is w-4.
- When the receiver sends an ACK, the source's window expand i.e. (right boundary moves outward) to allow in a number of new frames equal to the number of frames acknowledged by that ACK
- For example, Let the window size is 7 (see diagram (a)), if frames 0 through 3 have been sent and no acknowledgment has been received, then the sender's window contains three frames - 4,5,6
- Now, if an ACK numbered 3 is received by source, it means three frames (0, 1, 2) have been received by receiver and are undamaged



Sliding window on receiver side:

- At the beginning of transmission, the receiver's window contains n-1 spaces for frame but not the frames.
- As the new frames come in, the size of window shrinks.
- Therefore the receiver window represents not the number of frames received but the number of frames that may still be received without an acknowledgment ACK must be sent.
- Given a window of size w, if three frames are received without an ACK being returned, the number of spaces in a window is w-3.
- As soon as acknowledgment is sent, window expands to include the number of frames equal to the number of frames acknowledged.
- For example, let the size of receiver's window is 7 as shown in diagram. It means window contains spaces for 7 frames.
- With the arrival of the first frame, the receiving window shrinks, moving the boundary from space 0 to 1. Now, window has shrunk by one, so the receiver may accept six more frame before it is required to send an ACK.
- If frames 0 through 3 have arrived but have DOC been acknowledged, the window will contain three frame spaces.



- Therefore, the sliding window of sender shrinks from left when frames of data are sending. The sliding window of the sender expands to right when acknowledgments are received.
- The sliding window of the receiver shrinks from left when frames of data are received. The sliding window of the receiver expands to the right when acknowledgement is sent.