CSCE 5580 – COMPUTER NETWORK

Assignment 2

Set 1 (Q2, Q4, Q6, Q8, Q10, Q12, Q14, Q16, Q22, Q25, Q27, Q29, Q31)

Comparison Questions: (5 Points each)

2. What are the differences between Stateless and Stateless.

Stateful	Stateless
In stateful protocol, the server must retain state information for each client. It retains information from past communications	In stateless protocol, the server to retain any state information
It requires constant connection between the client and server	It does can handle each request on their own, doesn't require constant connection.
It is hard because it provides additional functionality.	It is easier to implement and can scale more effectively
Retaining state information can increase server's overhead	Doesn't retain information, resulting in lower overhead
Ex: TCP and FTP	Ex: HTTP and DNS

4. Briefly compare the Go-Back-N and Selective Repeat transfer protocols?

Go-Back-N	Selective Repeat
Go-Back-N protocol retransmits all the packets in the window if an error is detected in any of the packets.	Selective Repeat protocol only retransmits the packet with an error, and the receiver buffers the rest of the packets until the missing one is received.
Go-Back-N protocol is simpler to implement.	Selective Repeat protocol requires more complex buffer management at the receiver's end.
Go-Back-N protocol, the receiver can accept packets out of order and buffer them until the missing packet arrives.	Selective Repeat protocol requires packets to be delivered in order.
Go-Back-N protocol is more efficient case of burst errors.	Selective Repeat protocol is more efficient in case of random error.

6. What is the difference between flow control and congestion control, give any suitable real-life example?

Flow Control	Congestion Control
Flow control is a mechanism used to regulate the rate of data transmission between the sender and receiver.	Congestion control is a mechanism used to control the rate of data transmission in the network
It is used to avoid overwhelming the receiver with data	It is used to avoid network congestion,
It is used in point-to-point communication between a sender and a receiver.	It is used in a network with multiple devices and shared resources.
Flow control is implemented by the receiver, which sends feedback to the sender to control the rate of data transmission.	Congestion control is implemented by network devices, which detect congestion and signal the sender to reduce the rate of data transmission.
Ex: use of buffer in streaming videos	Ex: TCP's congestion control algorithm

Short answers: (2 Points each)

8. In a reliable transfer protocol, can a sender tell the difference between a lost data packet and a lost ACK?

The sender does not know whether a data packet was lost, an ACK was lost, or if the packet or ACK was simply overly delayed.

10. What is the size of the TCP header without options?

The size of the TCP header without options is 20 bytes. The TCP header is composed of several fields, including source and destination port numbers, sequence and acknowledgment numbers, TCP flags, window size, and checksum.

12. Why shouldn't we set the TCP timeout value to be extremely large to avoid early timeouts?

We should not set the TCP timeout value to be excessively large to avoid early timeouts because doing so can result in several problems.

1. A large timeout can cause delay in detecting segment loss.
2. A large timeout value can delay the retransmission of lost packets, causing a significant increase in transmission time, decreasing the network performance.
3. A large timeout value can also increase the vulnerability of a network to attacks, such as denial of service attacks, as attackers can exploit the long timeout to tie up network resources.

14. Why does TCP need to exchange three packets during connection setup (3- way handshake)?

TCP (Transmission Control Protocol) requires the exchange of three packets during the connection setup, also known as the three-way handshake, for several reasons.

1. To ensure that both endpoints are ready to communicate.
2. Confirms that both the sender and receiver are available and ready to transmit and receive data.

3. To establish and synchronize sequence numbers between the sender and receiver.
4. The sequence numbers are used by TCP to ensure that packets are received in the correct order and to detect the loss of packets. During the handshake, the two endpoints exchange their initial sequence numbers and confirm that they have received each other's sequence numbers.

16. What are the two symptoms of congestion?

Congestion in computer networks is due to overloading of network resources due to excess traffic.

1. **Packet delay:**
   If the network is congested, the packets compete for the available resources like buffer space and bandwidth, which increases the transmission delay and decreases the network performance.
2. **Packet loss:**
   Another symptom of congestion is packet loss, which occurs when a network device is unable to forward a packet because its buffers are full, and the packet is dropped due to errors.

Problems: (10 Points each)

22. Suppose that we want to distribute a file with a size of F = 25 Gbits to N = 100 clients/peers. The server supports an upload rate of u! = 30 Mbps while each client/peer has a download rate of d" = 3 Mbps and an upload rate of u, where u = 400 Kbps. Show your calculations.

Given,

File size F = 25 Gbits

N = 100 clients

a. Calculate the minimum distribution time for a client-server distribution using the two values of u given above.

   d" = 3 Mbps.

   u! = 30Mbps

   minimum distribution time for a client-server = max{NF/u! , F/d"}

   NF/u! = 100*25Gbit/30Mbps = 100*25600Megabit/30 Megabits per second

   = 85333.33 sec

   = 23.70 hours

   F/d" = 25Gbit/3Mbps = 25Gbit/3Mbps = 25600Megabit/3 Megabits per second

   = 8533.33 sec

= 2.37 hours

Hence the minimum distribution time for a client-server is 23.70 hours / 85333.33 sec

b. Calculate the minimum distribution time for a peer-to-peer distribution using the two values of u given above.

In a peer-to-peer distribution, each peer can upload to other peers, so the bottleneck link is the upload rate of each peer, which is u = 400 Kbps.

d" = 3 Mbps.

u! = 30Mbps

u = 400Kbps = 0.4 Mbps

minimum distribution time for a peer-peer = max{F/u! , F/d", NF/ u! +∑ u}

F/u! = 25Gbit/30Mbps = 25600Megabit/30 Megabits per second

= 853.33 sec

= 0.237 hours

F/d" = 25Gbit/3Mbps = 25Gbit/3Mbps = 25600Megabit/3 Megabits per second

= 8533.33 sec

= 2.37 hours

NF/ u! +∑ u = 100*25Gbit/30Mbps+ 400Kbps = 2560000Megabit/30.39 Megabits per second

= 84238.23 sec

= 23.39 hours

Hence the minimum distribution time for a peer-peer is 23.39 hours / 84238.23 sec

25. UDP -Consider UDP Segment, the source port is 3306, destination port is 80, Draw UDP segment header. Data is 9324 and 8545. Calculate the length and checksum from the information given. (Hint: Convert the data into 16-bit binary to evaluate the checksum included in the header and show the process).

UDP Header:

0 16 32

Source Port	Destination Port
Length	Checksum
Data	

UDP header with binary data:

0000110011101010	0000000001010000
0100010111001101	1011101000110010
0010010001101100	0010000101100001

UDP header with decimal data:

3306	80
17869	-17870
9324	8545

Sender:

Length = 8 bytes(header) + 9324 +8545 = 17869

Checksum:

Data 1 = 9324 = 0010010001101100

Data 2 =8545 = 0010000101100001

Sum = **0100010111001101**

1s complement = 1011101000110010

Checksum = **1011101000110010**

Now this checksum is sent as a part for header to the receiver, Now are the receiver end, receiver will calculate the Sum of data and the check sum.

Receiver:

Data 1 = 9324 = 0010010001101100

Data 2 =8545 = 0010000101100001

0100010111001101

Checksum = 1011101000110010

Sum = 1111111111111111

1s complement = 0000000000000000

Now we can see that sum is all 0s, indicating that these is no error

27. Hosts A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 162. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 88 and 44 bytes of data respectively. In the first segment, the sequence number is 127, the source port number is 1002, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

a. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?

   The sequence number of the second segment = 215 (127 + 88 =215)

   source port number: 1002

   destination port number: 80

b. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segments, what is the ACK number, the source port number. and the destination port number?
   If the first segment arrives before the second segment, in the acknowledgment of the first arriving segments = 215
   source port number: 80 (The Host B's port number)
   destination port number: 1002 (The Host A's port number)

c. If the second segment arrives before the first segment, in the ACK of the first arriving segment, what is the ACK number?
   If the second segment arrives before the first segment, ACK of the first arriving segment = 127
   That is it is still waiting for the 127 bytes onwards.

Finite state machine: (10 Points each)

29. Draw the finite state machine for following scenarios (explain briefly),

• reliable data transfer for operation with no error

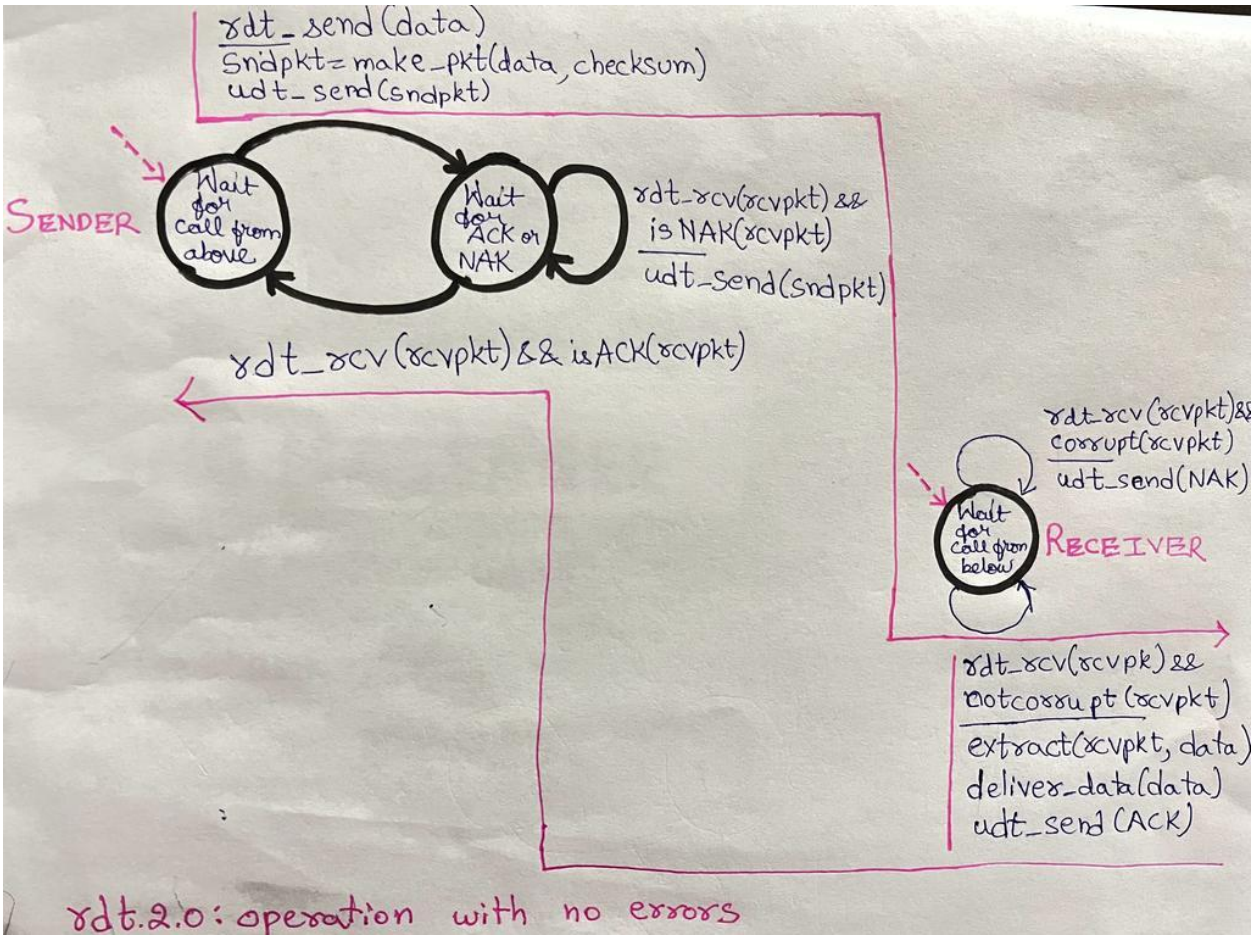
• reliable data transfer for operation with corrupted packet

1. **Reliable Data Transfer for Operation with No Error:**
   This finite state machine is used in a reliable data transfer protocol where the data packets are not corrupted during transmission.
   The sender sends data packets to the receiver and waits for an acknowledgment (ACK) from the receiver.
   The receiver checks if the received packet is corrupt or not.

i. If it is a corrupt packet the receiver sends a NAK to the sender. If the sender gets a NAK, it retransmits the data packet. As the data has no errors, NAK is not sent.
ii. If not a corrupt packet, the receiver extracts the data, delivers data and sends an ACK to the sender, that data is received without error. The receiver sends a ACK as there are no errors.

If the sender receives the ACK within a certain time frame, it sends the next data packet. If the ACK is not received within the time frame, the sender resends the same data packet. The sender receives the ACK on time and sends the next packet as this case operates with No errors.

The finite state machine for this scenario is as follows:



rdt_send: Send data through reliable data channel.
make_pkt: makes packet with data and checksum.
udt_send: Send data or ACK or NAK through unreliable data channel.
rdt_rcv: Received data through reliable data channel.
make_pkt: makes packet with data and checksum.
notcorrupt: Set if packet received is not corrupt.
corrupt: Set if packet received is corrupt.
extract: extracts the received data packet.

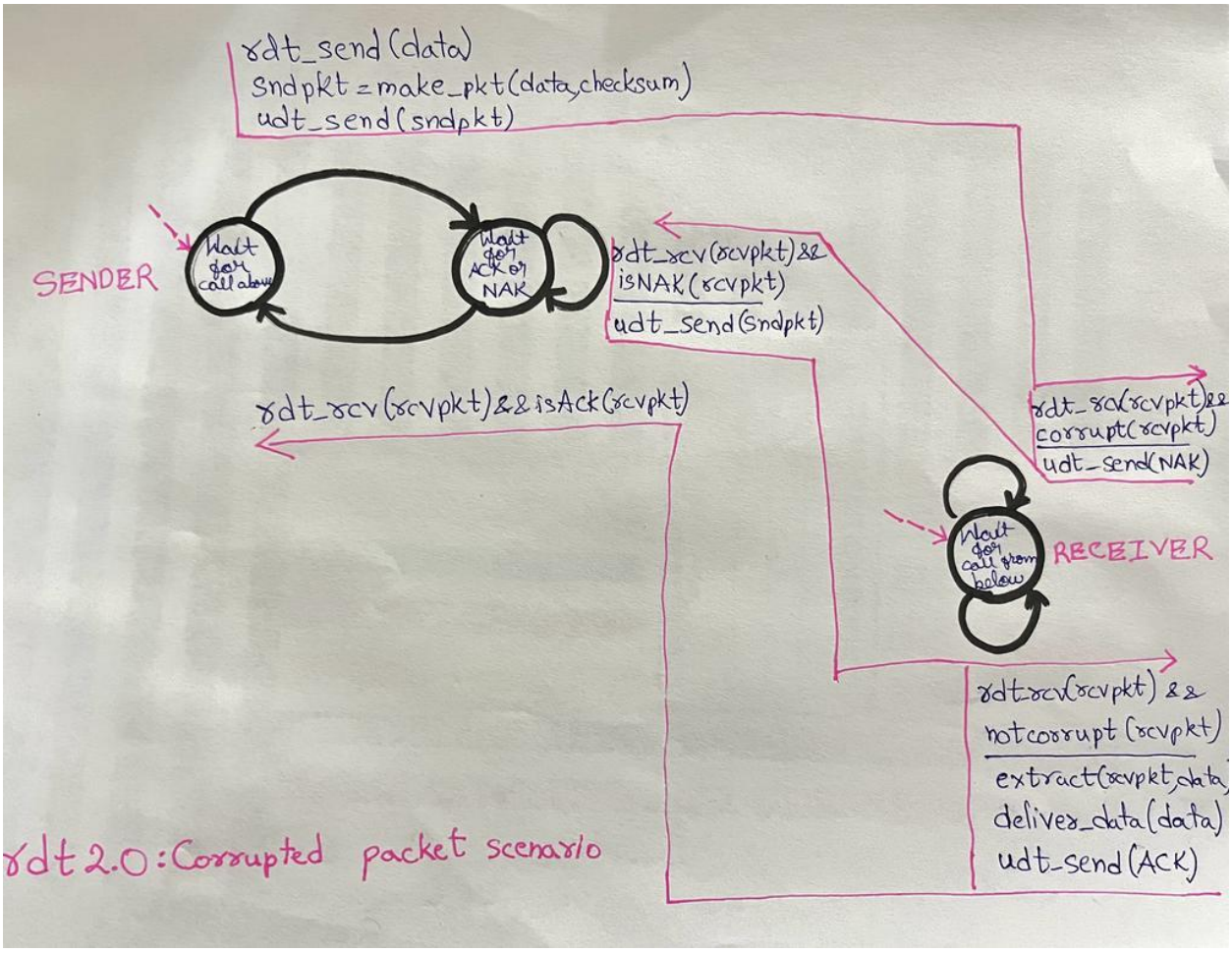
deliver_data: delivers received data.
isACK: receiver explicitly tells sender that pkt received is ok
isNAK: receiver explicitly tells sender that pkt had errors

2. Reliable Data Transfer for Operation with Corrupted Packet:
   This finite state machine is used in a reliable data transfer protocol where the data packets may be corrupted during transmission.
   The sender sends data packets to the receiver and waits for an acknowledgment (ACK) from the receiver.
   The receiver checks if the received packet is corrupt or not.

   If it is a corrupt packet the receiver sends a NAK to the sender.

   In addition to waiting for ACKs, the sender also waits for a negative acknowledgment (NAK) from the receiver. If the receiver detects a corrupted packet, it sends a NAK to the sender. The sender then resends the corrupted packet and waits for the acknowledgement from receiver.

   The finite state machine for this scenario is as follows:

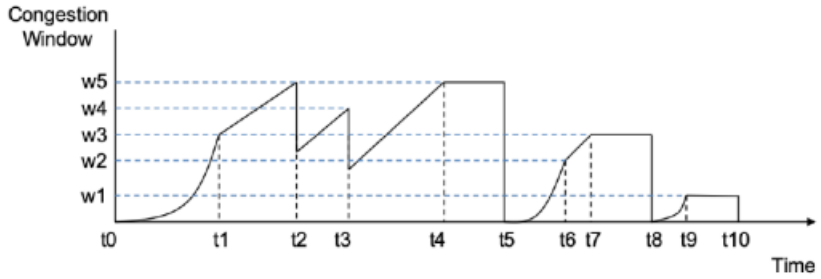


rdt_send: Send data through reliable data channel.
make_pkt: makes packet with data and checksum.
udt_send: Send data or ACK or NAK through unreliable data channel.
rdt_rcv: Received data through reliable data channel.
make_pkt: makes packet with data and checksum.
notcorrupt: Set if packet received is not corrupt.
corrupt: Set if packet received is corrupt.
extract: extracts the received data packet.
deliver_data: delivers received data.
isACK: receiver explicitly tells sender that pkt received is ok
isNAK: receiver explicitly tells sender that pkt had errors

In both scenarios, the sender and receiver can transition back and forth between the waiting and receiving states as data packets are sent and received. The main difference in the second scenario is the addition of the NAK message, which allows the receiver to request retransmission of corrupted packets.

Graphs Questions: (10 Points each)

31. Suppose the change of the congestion window size of a TCP connection is as depicted below.



a) State the time periods that the TCP connection is in the slow start phase. State the time periods in the congestion avoidance phase.
Slow start:
t0-t1
t5-t6
t8-t10
Congestion avoidance: t1-t5, t6-t8

b) State the ssthresh during time periods t0-t1, t2-t3, t3-t4, t7-t8, and t8- t9.
t0-t1: w3
t2-t3: w5/2
t3-t4: w4/2

t7-t8: w2
t8- t9: w3/2

c) State the TCP version, Tahoe or Reno, this connection is using and how you tell.

We are using TCP Reno in the above diagram provided. The window size does drop at t2, t3, t5, t8. At t2 and t3, the window size does not drop all the way down to 1. This indicates TCP Reno behavior, whereas in TCP Tahoe, the window size always drops to 1 and slow-starts again.