Computer Network Quiz 2

1. List at least four usable (consecutive) IP addresses under the 174.53.4.32/27 subnet.

174.53.4.33

174.53.4.34

174.53.4.35

174.53.4.36

2.(a) Class A: 0.0.0.0 to 127.255.255.255

Class B: 128.0.0.0 to 191.255.255.255

Class C: 192.0.0.0 to 223.255.255.255

Class D: 224.0.0.0 to 239.255.255.255 (reserved for multicast)

Class E: 240.0.0.0 to 255.255.255.255 (reserved for future use)

The IPv4 address 192.168.1.0 belongs to Class C.

(b) Valid six subnets are –

192.168.1.0/27

192.168.1.32/27

192.168.1.64/27

192.168.1.96/27

192.168.1.128/27

192.168.1.160/27

(C) The first three IP addresses of the first subnet -

192.168.1.0

192.168.1.1

192.168.1.2

(d) So, the subnet mask for all these subnets is 255.255.255.224 or expressed in CIDR notation as /27.

3. List all the /23 subnets in the 141.67.128.0/21 address block.

2 ^ 23-21 = 2^2 = 4

/23 is in 3rd octet.

The maximum is 24

2 ^ 24-23 = 2

141.67.128.0

141.67.130.0

141.67.132.0

141.67.134.0

141.67.136.0

141.67.138.0

141.67.140.0

141.67.142.0

4. Assume that a peer-to-peer network link as the following details: link Bandwidth is 8 million bits per second, round-trip time (RTT) is 90 milliseconds, and frame size is 2 Kilobytes. Assume the link is fully opened and the Go-Back-N ARQ sliding window protocol runs on both the receiver and the sender hosts. Based on the given data, calculate (a) the window size (N) of the ARQ algorithm and (b) The maximum data size allowed in the window.

Bandwidth = 8 \* 10^6 bits

RTT = 90 \* 10^-3 seconds

Frame size = 2 \* 1024 \* 8 bit (1 byte = 8 bits)

N = bandwidth \* RTT / Frame size

1. N = 43.75
2. Maximum data = N \* frame size

43 \* 2 \* 1024 \* 8

=722 944 bits

Byte = 722 944 / 8 = 90368

5. (a) Advertised Window =

max rc buffer – (LastByteRecvd – LastByteRead)  
Rcv Window = ?

= 8 – ( 18 – 15 )

= 8 – ( 3)

= 5

1. Effective Window size = 18 – (19- 15 )

18 – 4

14

6.

In TCP's sliding window protocol, an advertised window size of 0 from the receiver indicates that the receiver's buffer for incoming data is currently full. It means that the receiver cannot accept any more data at the moment because it doesn't have enough available buffer space to store additional incoming segments or bytes.

7. In TCP's sliding window protocol, the size of the Advertised Window (often referred to as the receiver's window or receive window) becomes equal to the size of the MaxRcvBuffer (maximum receive buffer) when the receiver's buffer is fully utilized or reaches its capacity.

Here's a breakdown of the conditions and what this means:

1. Advertised Window Size: The Advertised Window is the receiver's way of informing the sender about the amount of available buffer space for incoming data. It represents the maximum amount of unacknowledged data that the receiver can accept without overflowing its buffer.
2. MaxRcvBuffer (Maximum Receive Buffer): MaxRcvBuffer represents the total capacity of the receiver's buffer, i.e., the maximum amount of data that the receiver can store.

When the Advertised Window size equals the MaxRcvBuffer size, it means that the receiver's buffer is completely full, and it cannot accommodate any more incoming data. In this state:

* The receiver has received data up to its buffer's capacity.
* The receiver cannot accept any additional data from the sender until it has processed and freed up space in its buffer by acknowledging previously received data.
* The sender is effectively blocked from sending more data until the receiver acknowledges some of the previously received data, making space in the buffer.

This equilibrium between the Advertised Window and the MaxRcvBuffer is a flow control mechanism in TCP that helps ensure that the sender does not overwhelm the receiver with data that it cannot handle. It allows for efficient and controlled data transfer, ensuring that the sender's rate of sending is matched with the receiver's ability to receive and process data. Once the receiver acknowledges some data and frees up space in its buffer, the Advertised Window size will increase, allowing the sender to resume sending more data.

8. TCP's three-way handshake is a fundamental part of how the TCP protocol establishes a network connection between two devices (e.g., computers or servers) over an IP network. The three-way handshake process is used to ensure reliable and orderly communication between the two parties. Here's a step-by-step description of how it works:

1. **Step 1 - SYN (Synchronize)**:
   * The process begins with one device (the client) initiating the connection by sending a TCP packet with the SYN (synchronize) flag set to the other device (the server).
   * The client also selects an initial sequence number (ISN) for the connection. This ISN is typically a random number or a value based on a clock.
2. **Step 2 - SYN-ACK (Synchronize-Acknowledge)**:
   * Upon receiving the SYN packet, the server acknowledges the request by sending its own TCP packet.
   * The server sets the SYN flag and the ACK (acknowledge) flag in this packet.
   * Like the client, the server selects its own initial sequence number (ISN) for the connection.
3. **Step 3 - ACK (Acknowledge)**:
   * Finally, the client responds to the server's SYN-ACK packet by sending an acknowledgment.
   * This acknowledgment packet has the ACK flag set, indicating that the client acknowledges the server's response.
   * The client also acknowledges the server's ISN by incrementing it by 1.

At this point, both devices have exchanged initial sequence numbers, and they have agreed to establish a connection. The connection is considered open and established, allowing data to be exchanged in both directions reliably.

Key points to note about the three-way handshake:

* The three-way handshake ensures that both the client and server agree on initial sequence numbers, reducing the risk of old or duplicate packets interfering with the connection.
* The SYN flag is used to initiate and respond to connection requests, while the ACK flag is used to acknowledge receipt of packets and to confirm the establishment of the connection.
* If any of the three steps fails or if there is a timeout without receiving an expected response, the connection is not established, and the devices may attempt the handshake again or take appropriate error-handling actions.
* The three-way handshake is an essential part of TCP's reliability and connection-oriented nature, helping to establish a solid foundation for data transmission between devices in a network.

9 TCP (Transmission Control Protocol) uses an adaptive timeout mechanism to estimate the Round-Trip Time (RTT) and dynamically adjust the retransmission timeout (RTO) value. The RTO is the time interval that TCP waits before retransmitting a packet if it doesn't receive an acknowledgment (ACK) from the receiver. This adaptive mechanism helps TCP adapt to varying network conditions.

In your scenario, you have the following data:

* SampleRTT of the current segment transmission (SRTT\_new) = 8ms
* SampleRTT of the previous segment transmission (SRTT\_old) = 12ms
* RTT smoothing parameter (alpha) = 0.85

To update the estimated RTT (SRTT) and the RTO, you can use the following formulas:

1. Update the Estimated RTT (SRTT):
   * SRTT\_new = (1 - alpha) \* SRTT\_old + alpha \* SampleRTT\_new
2. Update the RTO:
   * RTO = SRTT + 4 \* RTT\_variance

However, we need to calculate the RTT variance (RTTVAR) first. The RTT variance represents the variability or fluctuations in the RTT values. Here's the formula to calculate RTTVAR:

* RTTVAR\_new = (1 - beta) \* RTTVAR\_old + beta \* |SampleRTT\_new - SRTT\_new|

Where:

* beta is another parameter (typically set to 0.25 in practice) to control the influence of new RTT samples on the RTTVAR.

Now, let's calculate RTTVAR:

* beta = 0.25 (commonly used value)
* |SampleRTT\_new - SRTT\_new| = |8ms - SRTT\_new| (where SRTT\_new is the updated value from the first formula)

Assuming that SRTT\_old was 12ms:

* RTTVAR\_new = (1 - 0.25) \* RTTVAR\_old + 0.25 \* |8ms - SRTT\_new|

Next, you can use the updated RTTVAR and the SRTT to calculate the RTO:

* RTO = SRTT\_new + 4 \* RTTVAR\_new

This adaptive timeout mechanism ensures that the RTO value reflects the current network conditions and adapts to changes over time. Smoothing with alpha and beta parameters helps prevent rapid adjustments due to occasional variations in RTT while still allowing the RTO to react to significant changes in network conditions.

10. The Stop-and-Wait Automatic Repeat reQuest (ARQ) sliding window algorithm prevents the issue of duplicate copies of frame reception at the receiver through a simple and effective mechanism. Here's how it works:

1. **Sender Behavior**:
   * The sender sends a single frame to the receiver.
   * After sending the frame, the sender starts a timer to wait for an acknowledgment (ACK) from the receiver.
2. **Receiver Behavior**:
   * The receiver receives the frame.
   * It checks the frame for errors and verifies its integrity.
   * If the frame is error-free and in sequence, the receiver sends an ACK back to the sender.
3. **Sender's Reaction to ACK**:
   * Upon receiving the ACK, the sender knows that the frame was successfully received and acknowledged by the receiver.
   * The sender can then proceed to send the next frame.
4. **Sender's Reaction to Timeout**:
   * If the sender's timer expires before receiving an ACK, it assumes that the frame was lost or damaged in transit.
   * In this case, the sender retransmits the same frame.
   * The receiver, upon receiving a duplicate frame, detects it as such and discards it.

This mechanism ensures that only one frame is in transit at a time, and the sender waits for an acknowledgment before sending the next frame. If the receiver receives a duplicate frame (due to a retransmission), it recognizes it as such based on the sequence number and discards it. This prevents the issue of duplicate copies of frames being processed by the receiver.

In summary, the Stop-and-Wait ARQ algorithm relies on sequence numbers and acknowledgments to ensure that frames are delivered reliably without duplication. It's a simple and reliable protocol suitable for scenarios where low data rates or low error rates are acceptable, such as in serial communication or scenarios with minimal network congestion.

11 The Selective Repeat Automatic Repeat reQuest (ARQ) sliding window algorithm is a more sophisticated variation of ARQ compared to Stop-and-Wait. It allows multiple frames to be in transit simultaneously and provides better network efficiency. Here's how it works in your scenario:

1. **Sender and Receiver Initialization**:
   * The sender, receiver, and the network are set up.
   * Eight frames with sequence numbers 0 to 7 are to be sent.
2. **Window Size**:
   * The size of the sender's window is 4, which means it can have up to 4 unacknowledged frames in transit at any given time.
3. **Sending Frames**:
   * Initially, the sender sends the first 4 frames (0, 1, 2, 3) because its window size is 4.
   * These frames are marked as sent but not acknowledged.
4. **Receiver's Behavior**:
   * The receiver receives the frames and checks them for errors and for being in sequence.
   * If a frame is received correctly and in sequence (e.g., frames 0, 1, 2), the receiver sends individual acknowledgments (ACKs) for each of these frames.
5. **Negative Acknowledgment (NACK)**:
   * In your scenario, the receiver encounters an issue with frame 3, and it sends a Negative Acknowledgment (NACK) to the sender specifically for frame 3.
   * The NACK indicates that frame 3 was received but has an issue (e.g., it might be damaged or out of sequence).
6. **Sender's Reaction to NACK**:
   * Upon receiving the NACK for frame 3, the sender knows that frame 3 needs to be retransmitted.
   * The sender retransmits frame 3, but it doesn't stop sending new frames.
7. **Continuing Transmission**:
   * While frame 3 is being retransmitted, the sender continues to send frames within its window (e.g., frames 4, 5, 6, and 7).
8. **Receiver's Behavior after Retransmission**:
   * When the receiver receives the retransmitted frame 3 correctly, it sends an ACK for frame 3.
9. **Completion of Transmission**:
   * The sender continues to send the remaining frames, and the receiver acknowledges them as they arrive.
   * Once all frames are received and acknowledged, the transmission is complete.

The key advantage of Selective Repeat over Stop-and-Wait is that it allows the sender to continue sending new frames even if there are missing or damaged frames. It's more efficient because it doesn't require the sender to wait for a specific missing frame to be retransmitted before sending more data. Instead, it selectively retransmits only the frames that are missing or damaged, leading to better network utilization and throughput.

12.