Tahoe Vs Reno :

If you're asking about a comparison between congestion control algorithms used in Tahoe and Reno, both of these algorithms are used for TCP congestion control, but there are some key differences between them.

Tahoe is an older congestion control algorithm that uses a conservative approach to avoid network congestion. It uses a slow start algorithm to initially establish a connection and gradually increase the amount of data transmitted. If congestion is detected, it will quickly reduce the transmission rate and enter a congestion avoidance state. Tahoe will then slowly increase the transmission rate until congestion is detected again.

Reno is a newer congestion control algorithm that builds on the Tahoe algorithm, but with some modifications to improve performance. One of the key differences is that Reno uses a fast recovery algorithm to quickly recover from congestion. When congestion is detected, Reno reduces the transmission rate, but it also enters a fast recovery state, which allows it to quickly resume transmitting data when the congestion subsides.

In summary, Tahoe and Reno are both congestion control algorithms used in TCP, but Reno is generally considered to be a more advanced algorithm that performs better in congested network conditions. However, there may be specific use cases where Tahoe is still preferable, such as in scenarios where network congestion is consistently high or in situations where a more conservative approach is desired.

Congestion control policy problems and solutions

Congestion control policies are used to manage network traffic to prevent network congestion, which can cause slow data transmission rates, packet loss, and other network performance problems. However, even with congestion control policies in place, there can still be problems that arise. Here are some common congestion control problems and solutions:

1. Slow start: A slow start algorithm is commonly used to establish a TCP connection and gradually increase the transmission rate. However, if the congestion control policy is too conservative, slow start can take a long time to ramp up, resulting in slow data transfer rates. To solve this problem, some congestion control policies use more aggressive slow start algorithms or adjust the initial window size to allow for faster data transfer rates.
2. Over-zealous congestion control: Some congestion control policies can be too aggressive in detecting and responding to congestion, leading to unnecessary data transmission rate reductions. This can result in reduced network performance and slower data transfer rates. To solve this problem, congestion control policies can be adjusted to be more conservative or use more intelligent algorithms to detect and respond to congestion.
3. Network heterogeneity: Network heterogeneity refers to the situation where there are multiple types of devices and networks in use, each with their own bandwidth and congestion control policies. This can lead to network congestion and performance issues, especially if there are differences in the congestion control policies used by different devices. To solve this problem, congestion control policies can be designed to be more adaptable to different network conditions, or network devices can be configured to use similar congestion control policies.
4. Fairness: Congestion control policies should be fair and distribute network bandwidth evenly among all users. However, some congestion control policies can favor certain users or types of traffic over others, leading to unfairness and network performance issues. To solve this problem, congestion control policies can be designed to be more equitable, or additional network resources can be added to ensure fairness.

In summary, congestion control policies are critical for managing network traffic and preventing congestion. However, to ensure optimal network performance, congestion control policies should be carefully designed and adjusted to account for network heterogeneity and fairness, while avoiding over-zealousness and slow start problems.

TCP slow start

TCP slow start is a congestion control algorithm used by TCP (Transmission Control Protocol) to establish a connection and gradually increase the transmission rate. The goal of slow start is to avoid overwhelming the network with too much traffic, which can cause congestion, packet loss, and other performance issues.

During slow start, the sender initially transmits a small number of packets and then gradually increases the number of packets sent per round-trip time. The number of packets sent doubles with each round-trip time until the congestion window size reaches a threshold value. This threshold value is usually based on the available network bandwidth and the amount of congestion present in the network.

Once the congestion window size reaches the threshold value, the sender switches from slow start to congestion avoidance mode. In congestion avoidance mode, the sender continues to gradually increase the transmission rate, but at a slower rate than during slow start. The sender also monitors the network for signs of congestion and reduces the transmission rate if congestion is detected.

Slow start is an effective algorithm for avoiding network congestion and ensuring reliable data transmission, but it can also cause performance issues in certain situations. For example, slow start can be slow to ramp up and may result in slow data transfer rates, especially in high-bandwidth networks. Additionally, slow start may cause unnecessary packet loss or network congestion if the congestion window size is set too low or if there are other factors that cause congestion.

To address these issues, there are alternative congestion control algorithms that can be used, such as the Reno and New Reno algorithms, which build on the slow start algorithm but provide additional features for faster recovery from congestion and improved network performance.

What Is Network congestion ?

Network congestion occurs when there is more traffic on a network than it is capable of handling. Congestion can occur in any type of network, including local area networks (LANs), wide area networks (WANs), and the internet.

When a network experiences congestion, data packets can be delayed, lost, or dropped, which can lead to degraded network performance and slower data transfer rates. This can cause issues such as increased response times, interrupted data transmission, and reduced network throughput. In severe cases, network congestion can even result in complete network failure.

There are several causes of network congestion, including network design issues, equipment failure or malfunction, software issues, and unexpected spikes in network traffic. Congestion can also occur when network resources are oversubscribed, meaning that there is not enough bandwidth or processing power available to handle the amount of traffic on the network.

To prevent network congestion, network administrators can implement congestion control policies, such as traffic shaping, to manage network traffic and prioritize critical data traffic. Congestion control policies can also help to reduce the impact of congestion on network performance by slowing down traffic rates and regulating traffic flows. Additionally, network administrators can upgrade network equipment, add more bandwidth, and optimize network traffic routing to help prevent congestion and improve network performance.

What is session? in Netsim

In NetSim, a session refers to a communication session between two or more network devices, using a specific protocol or application. A session can be thought of as a logical connection between devices that allows them to exchange data and communicate with each other.

Why TCP uses 4 way finishing for connection termination instead of 3way like

connection establishment?

Answer :

TCP uses a 4-way handshake for connection termination to ensure that both the client

and server have completely closed the connection and that all data has been

successfully transferred.

In a 3-way handshake for connection establishment, the client sends a SYN packet to

the server, the server responds with a SYN-ACK packet, and the client sends an ACK

packet to confirm the connection.

During a connection termination, the client sends a FIN packet to the server to indicate

that it has no more data to send. The server then responds with an ACK packet to

confirm that it has received the FIN packet. However, the server may still have data to

send to the client, so it sends a FIN packet to the client to indicate that it has no more

data to send. Finally, the client responds with an ACK packet to confirm that it has

received the FIN packet.

By using a 4-way handshake, TCP ensures that all data has been successfully

transmitted and acknowledged by both parties before the connection is fully terminated.

This helps to avoid potential data loss or corruption that could occur if the connection

was terminated abruptly with a 3-way handshake.

**Topology:**

What is a network topology? A network topology is the physical and logical arrangement of nodes and connections in a network. Nodes usually include devices such as switches, routers and software with switch and router features. Network topologies are often represented as a graph.

**Three-way handshaking:**

Three-way handshaking is a communication process used in TCP to establish a connection between two devices. The client sends a SYN packet to the server, which responds with a SYN-ACK packet containing an acknowledgment number. Finally, the client sends an ACK packet to complete the process. This ensures both devices agree on connection parameters and can communicate effectively. Once the handshake is complete, data can be transmitted between the devices.

LAB 7

What Is Application Method:

Unicast In networking, the unicast application method refers to a method of communication where a single message is sent from one sender to one specific recipient. This means that data is transmitted from one source to one destination, and only that destination receives the data.

In unicast communication, the sender typically sends data to the IP address of the intended recipient. The recipient's network interface card (NIC) then checks the destination IP address of the incoming data packet and receives the data if the destination address matches its own IP address.

—------\

In TCP (Transmission Control Protocol), there are several types of control packets, including TCP ACK (acknowledgment) and TCP FIN (finish) packets.

TCP ACK Packet: A TCP ACK packet is sent by the receiver to acknowledge the receipt of data packets sent by the sender. When a device receives a TCP packet from another device, it responds with an ACK packet to confirm that it received the data packet successfully. The ACK packet contains an acknowledgment number, which indicates the next expected sequence number of data that the receiver is expecting to receive. The ACK packet is an essential part of TCP's reliability mechanism, as it ensures that data is successfully transmitted from sender to receiver.

TCP FIN Packet: A TCP FIN packet is sent by a device to indicate that it has finished transmitting data to another device and wants to close the connection. The FIN packet contains a sequence number that indicates the last byte of data that the device has sent, and the ACK number of the next expected sequence number of data that it is expecting to receive. Once both devices have sent and received FIN packets, the TCP connection is closed. The FIN packet is a part of TCP's reliable data transmission mechanism, as it ensures that data is transmitted and received correctly before closing the connection.

Both TCP ACK and TCP FIN packets are essential for the reliable transmission of data in TCP connections. Without ACK packets, there would be no way for devices to confirm that data has been received successfully, and without FIN packets, TCP connections would not be able to close reliably.

—-------------------------------------------------------------------------------------------------------------------------------------

LAB 8

**Congestion Policy Of TCP**

TCP (Transmission Control Protocol) uses a congestion control mechanism to prevent network congestion, which can cause packet loss and slow down the data transmission. When TCP detects congestion, it reduces the sending rate to prevent further congestion.

TCP uses a variety of congestion control algorithms, such as Slow Start, Congestion Avoidance, and Fast Recovery. Slow Start is used when the connection is established or when congestion is detected. It starts by sending a few packets and gradually increases the sending rate until congestion is detected.

Congestion Avoidance is used to maintain the sending rate once it is established. It increases the sending rate gradually until congestion is detected and then reduces it to prevent further congestion.

Fast Recovery is used when a packet is lost and the receiver sends a duplicate acknowledgment. It allows the sender to recover quickly without going through the entire Slow Start process again.

Overall, TCP's congestion control mechanism helps ensure the stability and efficiency of network communication by dynamically adjusting the sending rate based on network conditions.

**1. Answer:**(a) TCP slow start is the initial phase of congestion control, in which the sender gradually increases the number of packets in flight. In the graph, we can identify TCP slow start as a steep increase in the congestion window size at the beginning of the transmission. It is characterized by a linear increase in the congestion window size until a threshold is reached.

(b) Packet loss and timeout occur when a packet is lost or not acknowledged by the receiver. In the graph, we can identify packet loss as a sudden drop in the congestion window size. Timeouts are characterized by a sudden drop in the congestion window size to one, followed by a slow restart of the congestion control mechanism.

(c) TCP congestion avoidance is the phase of congestion control that maintains the sending rate once it is established. In the graph, we can identify the intervals of time when TCP congestion avoidance is operating as a gradual increase in the congestion window size with occasional drops due to packet loss. It is characterized by a sawtooth pattern with a gradual increase and sharp drops.

**Tahoe & Reno:**

Tahoe and Reno are two variations of TCP congestion control algorithms that were developed to improve network performance in different ways.

Tahoe is the original TCP congestion control algorithm that was implemented in the early versions of TCP. It uses a simple slow start algorithm and congestion avoidance mechanism to regulate the sending rate. In Tahoe, the congestion window is cut in half whenever a packet loss is detected. Tahoe has a conservative approach to congestion control, and it takes a longer time to recover from congestion than other algorithms.

Reno is a modified version of Tahoe that was developed to improve its performance. Reno uses a more aggressive fast recovery algorithm that allows the sender to recover quickly from packet loss. In Reno, the congestion window is reduced by only a small amount when a packet loss is detected, allowing it to maintain a higher sending rate. Reno also includes a mechanism called fast retransmit, which allows the sender to retransmit a lost packet without waiting for a timeout.

**CBR**

CBR (Constant Bit Rate) applications are a type of network application that requires a fixed amount of bandwidth and consistent transmission rate. They are used for real-time applications such as voice and video streaming, where delay and jitter can have a significant impact on the quality of the transmission. CBR applications transmit data in fixed-sized packets at a constant rate, regardless of the network conditions.

**EstimateRTT:**

EstimatedRTT (Estimated Round Trip Time) is a network parameter used by the TCP protocol to estimate the time it takes for a packet to travel from the sender to the receiver and back again. It is calculated using a weighted moving average of the measured Round Trip Time (RTT) values, which is the time it takes for a packet to travel from the sender to the receiver and back again.

**UDP Header**:

UDP Header contains 4 fields. The six fields are Source Port, Destination Port, Length &

Checksum. Length Of UDP Header: 8 bytes. The largest possible Source Port Number is (2^16 – 1) = 65535. We use DNS commands to capture UDP packets because DNS uses UDP as the transport protocol for the majority of its queries and responses.