
CN

MOD-4 PART-A

1.Assume An end system sends 50 packets for second using UDP over a full duplex mode 100 Mbps Ethernet LAN Connection. Each packet consists of 1500 Bytes of the Ethernet frame payload data. What is the throughput when measured at UDP protocol?

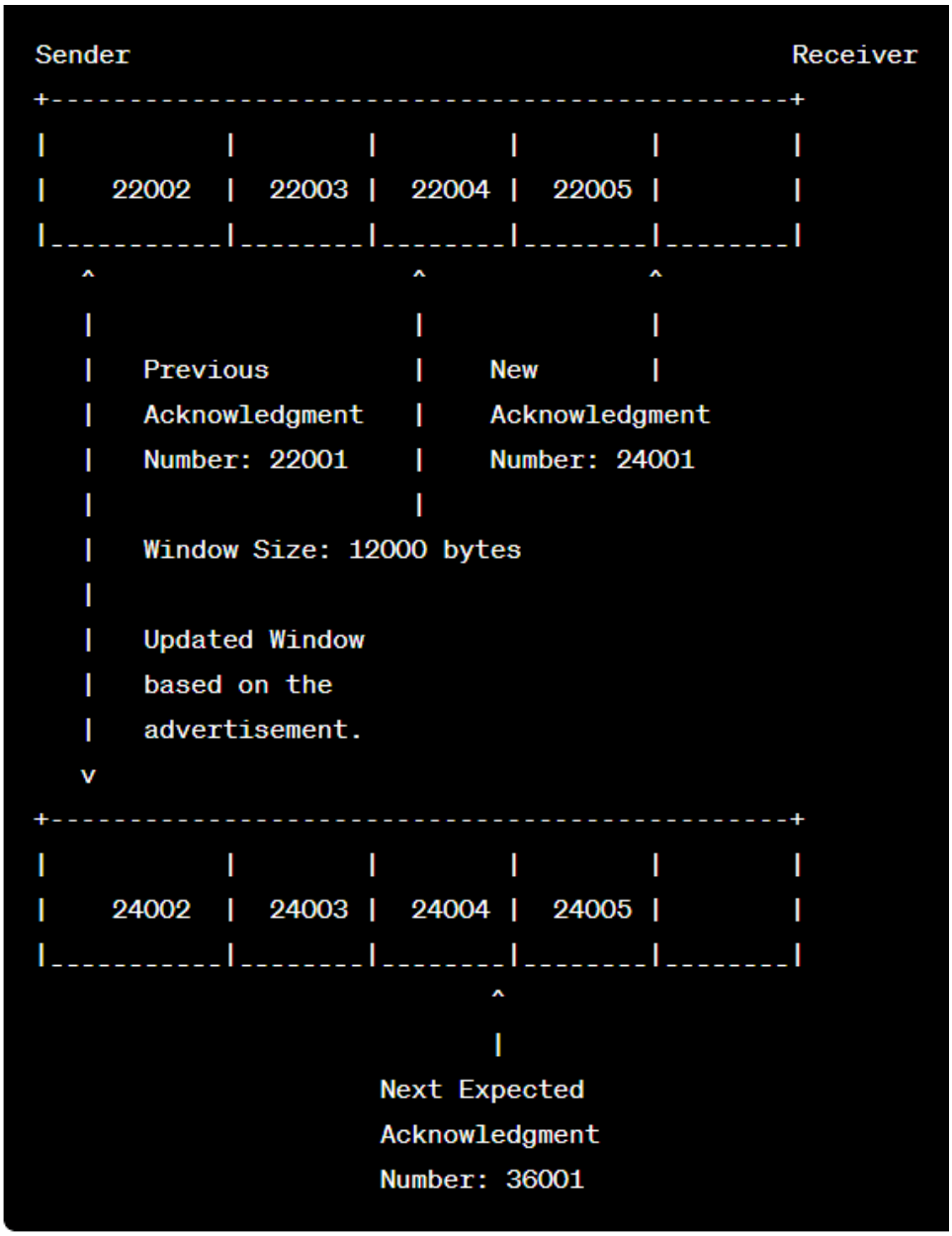
1. Determine the payload size per packet:
    - Ethernet frame payload data = 1500 bytes
    - UDP header = 8 bytes
2. Convert payload size to bits:
    - 1500 bytes * 8 bits/byte = 12000 bits
3. Multiply payload size by packets per second:
    - 12000 bits/packet * 50 packets/second = 600,000 bits/second
4. Convert to more convenient units:
    - 600000 bits/second = 600 kbps or 0.6 Mbps

Therefore, the throughput when measured at the UDP protocol is **600 kbps or 0.6 Mbps.**

2.Assume each packet has typical TCP and IP headers each 20bytes long. If we have three computers, A, B and C. The link between A and B has an MTU of 3000 bytes, while the link between B and C has an MTU of 1000 bytes. Consider the case where a packet needs to be sent from A to C that has a size of 3000 bytes (including headers). How many fragments will we have from B to C, and how much data will be in each fragment (i.e. excluding headers). (all connections are assumed to be Ethernet)

1. Packet Size and Headers:
    - Total packet size (including headers): 3000 bytes
    - TCP header size: 20 bytes
    - MTU=1000 bytes
    - IP header size: 20 bytes
    - Total header size: 40 bytes
2. Maximum Data Sizes:
    - Maximum data size on link A-B: 3000 - 40 = 2960 bytes
    - Maximum data size on link B-C: 1000 - 40 = 960 bytes
3. Number of Fragments: [max data size /MTU] ➔ 2960/1000 == 2.9 ~ 3
    - To fit within the MTU of 1000 bytes on link B-C, the original packet must be fragmented into 3 fragments.
4. Data Size per Fragment:
    - Each fragment will contain **960 bytes** of data (excluding headers).

3.Design a TCP connection is using a window size of 12000 bytes and the previous acknowledgement remembrance number was 22001.It receives a segment with acknowledgment number 24001 and window size advertisement of 12000. Design a diagram to show the situation of the window before and after.



4.Organize a client uses UDP to send data to a server. The data are 15 bytes. Calculate the efficiency of this transmission at the UDP level (ratio of useful bytes to total bytes)

1. Determine the total bytes sent:
    - Data bytes = 15 bytes
    - UDP header = 8 bytes
    - Total bytes = 15 bytes + 8 bytes = 23 bytes
2. Calculate the efficiency:
    - Efficiency = (Useful bytes / Total bytes) * 100%
    - Efficiency = (15 bytes / 23 bytes) * 100%

- Efficiency = 65.22%

Therefore, the efficiency of this UDP transmission is **65.22%.**

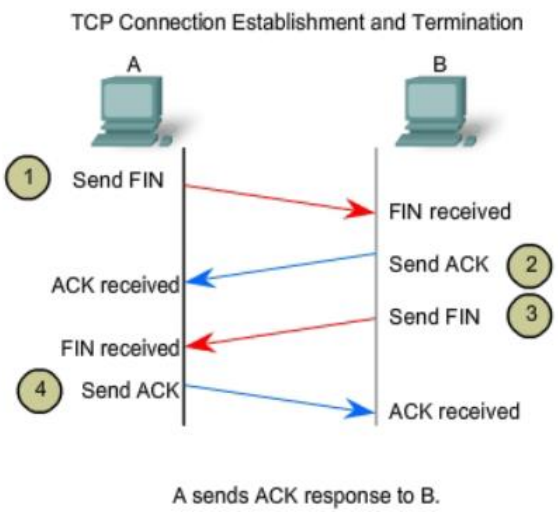
5.Discuss in detail about the connection establishment and release in TCP

Connection Establishment (Three-Way Handshake):

1. **SYN (Synchronize)**:
    - Client sends a SYN segment to the server, including a randomly generated Initial Sequence Number (ISN).
2. **SYN-ACK (Synchronize-Acknowledge)**:
    - Server responds with a SYN-ACK segment:
        - Includes its own ISN.
        - Acknowledges the client's ISN.
        - Indicates readiness to establish a connection.
3. **ACK (Acknowledge)**:
    - Client completes the handshake by sending an ACK segment:
        - Acknowledges the server's ISN, signalling synchronization for data exchange.

Connection Release (Four-Way Handshake):

1. **FIN (Finish)**:
    - Either party initiates termination by sending a FIN segment, indicating no more data to send.
2. **ACK (Acknowledge)**:
    - Recipient acknowledges the FIN, confirming the intent to close the connection.
3. **FIN (Finish)**:
    - The recipient, having no more data to send, sends its own FIN to gracefully close the connection from its side.
4. **ACK (Acknowledge)**:
    - The original sender of the FIN acknowledges the received FIN, completing the connection termination.



6.Describe in detail about TCP segment header and connection Establishment.

TCP Segment Header:
The TCP header contains crucial information for data transmission. It includes source and destination ports for identifying the sender and recipient, sequence and acknowledgment numbers to manage ordered delivery and acknowledgments, control flags (URG, ACK, PSH, RST, SYN, FIN) for various control purposes like acknowledging data receipt or initiating connection termination, window size for flow control, checksum for error detection, urgent pointer for urgent data, and optional fields like TCP options for additional functionalities.

Connection Establishment (Three-Way Handshake):

1. **Client Sends SYN:** The client initiates a connection by sending a SYN (Synchronize) packet to the server. This packet includes a randomly generated initial sequence number (ISN).
2. **Server Responds with SYN-ACK:** The server acknowledges the client's SYN by responding with a SYN-ACK packet. It includes its own ISN, acknowledges the client's ISN, and indicates readiness to establish a connection.
3. **Client Completes Handshake with ACK:** The client acknowledges the server's SYN-ACK packet by sending an ACK packet. This packet acknowledges the server's ISN, finalizing the connection setup, and both sides are synchronized and ready to exchange data.

7. a) Explain the Services of Transport layer. b) Explain leaky bucket and token bucket algorithms

a) Transport Layer Services:

Imagine the transport layer as a helpful courier:

- Connects Apps Across Devices: It sets up conversations between apps on different devices, like a phone call.
- Chop and Deliver Data: It divides messages into smaller packets for transport, like fitting boxes into a delivery truck.
- Reliable Delivery (TCP): It ensures all packets arrive safely and in order, like a careful mail carrier.
- Speedy Delivery (UDP): It focuses on fast delivery without guarantees, like a quick courier for urgent items.
- Handle Multiple Conversations: It lets multiple apps use the network at once, like sorting mail for different people.
- Manage Traffic Flow: It prevents network overload by regulating data flow, like a traffic controller.

b) Traffic Shaping Algorithms (Leaky Bucket and Token Bucket):

Imagine these algorithms as water regulators:

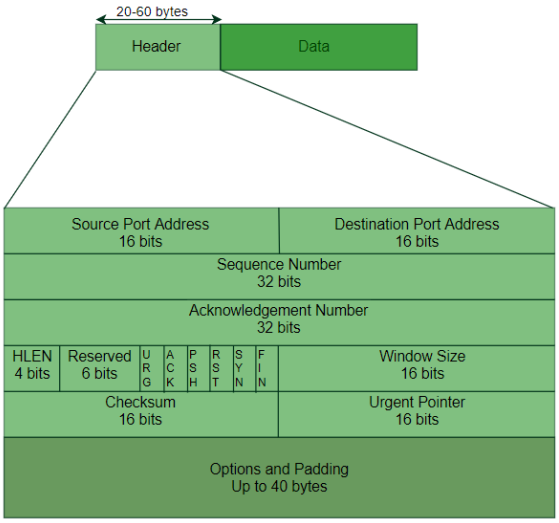
- Leaky Bucket:
    - A bucket with a hole at the bottom.
    - Water (data) flows in but trickles out steadily.
    - Prevents flooding by discarding excess water.

- Token Bucket:
    - A bucket filled with tokens (like coins).
    - Water can only flow out when you have a token.
    - Tokens are added regularly, allowing bursts if you save tokens.
    - More flexible for varying water needs.

8.Draw and explain each field in the TCP Segment header.

Key Fields:

1. Source Port and Destination Port:
    - Like addresses for the sender and receiver apps (e.g., web browser and website).
    - Similar to "From" and "To" on a package.



2. Sequence Number:
    - Assigns a unique number to each segment, ensuring they arrive in the correct order.
    - Like numbering pages in a book to reassemble the story.
3. Acknowledgment Number:
    - Confirms receipt of previous segments, letting the sender know what's been received successfully.
    - Like sending a "Got it!" message back.
4. Data Offset:
    - Specifies the length of the header, indicating where the actual data begins.
    - Like marking where the wrapping paper ends and the gift starts.
5. Flags:
    - Control various actions, such as setting up a connection, ending a conversation, or requesting urgent delivery.
    - Like special instructions or delivery options.
6. Window Size:
    - Determines how much data the receiver can accept at once, preventing a backlog.
    - Like saying "I can only fit a certain amount of boxes in my garage."
7. Checksum:
    - Detects errors during transmission, ensuring data integrity.

- Like checking for damage or missing pieces during delivery.

8. Urgent Pointer:
    - Flags specific data as urgent, prioritizing its delivery.
    - Like marking a package "Rush!" for immediate attention.
9. Options:
    - Allow for additional settings or customization.
    - Like choosing extra features or special handling.
10. Padding:
    - Fills any empty space to make the header a standard size for efficiency.
    - Like adding packing peanuts to keep things stable.

Together, these fields ensure reliable and organized data delivery over TCP connections.

PART-B

1.Explain the real transport protocol of UDP and how will you calculate checksum in UDP

UDP (User Datagram Protocol) is a connectionless, fast, and efficient transport protocol that prioritizes speed over reliability. It's like sending a postcard without tracking:

Key Features:

- Connectionless: No handshake or setup process, data is sent directly.
- Unordered Delivery: Packets (datagrams) may arrive out of order or not at all.
- No Retransmission: Lost or corrupted packets aren't retransmitted.
- Simple Header: Minimal overhead with only 8 bytes of header information.

Checksum Calculation:

1. Pseudo Header Creation: Combine source IP address, destination IP address, protocol number (17 for UDP), UDP length, and UDP data.
2. Segmentation: Divide data into 16-bit words.
3. One's Complement Sum: Add all words, including any carry bits.
4. Invert the Sum: Flip all bits to get the checksum.
5. Append Checksum: Attach the calculated checksum to the UDP header.

Receiver Verification:

- Calculates the checksum using the same method.
- If the calculated checksum matches the received checksum, data is likely error-free.
- Otherwise, data is discarded.

UDP is ideal for:

- Real-time applications (e.g., live streaming, gaming)

- Time-sensitive tasks (e.g., DNS lookups)
- Applications tolerating some loss (e.g., VoIP)
- When speed is more important than guaranteed delivery.

2.Show neatly the TCP segment format and describe each of it. (PART A -8)

3.List out the network performance characteristics

Network performance characteristics are essential metrics in understanding and optimizing how data flows across computer networks. These metrics help network administrators troubleshoot potential issues, predict resource demands, and ensure smooth user experience. Let's explore some key network performance characteristics:

1. Bandwidth:

- Definition: The maximum rate at which data can be transmitted across a network connection, measured in bits per second (bps).
- Analogy: Think of bandwidth as the width of a road. A wider road allows more cars to pass through per second, similar to how greater bandwidth enables faster data transfer.
- Measurement: Mbps (megabits per second), Gbps (gigabits per second), etc.

---

2. Latency:

- Definition: The time it takes for data to travel from one point to another on a network, measured in milliseconds (ms).
- Analogy: Think of latency as the delay between pressing a button and seeing the result. Lower latency means a faster response, like a quicker reaction time in a video game.
- Measurement: ms (milliseconds), µs (microseconds), etc.

stopwatch representing latency

---

3. Packet Loss:

- Definition: The percentage of data packets that are lost during transmission and never reach their destination.
- Analogy: Think of packet loss as dropped letters in a sentence. Missing packets can disrupt or even render data unusable, similar to an incomplete message.
- Measurement: Percentage (%)

---

4. Throughput:

- Definition: The actual rate at which data is successfully delivered across a network, measured in bps.
- Analogy: Think of throughput as the average number of cars that successfully reach their destination per second, considering traffic lights and potential

obstacles. It takes into account bandwidth and other factors affecting data transfer.

- Measurement: Mbps (megabits per second), Gbps (gigabits per second), etc.

---

5. Jitter:

- Definition: The variation in the arrival time of data packets, measured in milliseconds (ms).
- Analogy: Think of jitter as the uneven pacing of a runner. Inconsistent packet arrival times can disrupt real-time applications like video conferencing or online gaming.
- Measurement: ms (milliseconds)

---

6. Network Availability:

- Definition: The percentage of time a network is operational and accessible to users.
- Analogy: Think of network availability as the uptime of a restaurant. A higher availability ensures consistent access to network resources, while downtime is like the restaurant being closed.
- Measurement: Percentage (%)

4.Illustrate the adaptive retransmission policy in detail.

Key Principles:

- Continuous Monitoring: The courier (sender) actively tracks delivery success and adjusts strategies based on conditions.
- Dynamic Retransmission Timeouts: The courier adjusts waiting times for acknowledgments (ACKs) before resending, adapting to network congestion or delays.
- Congestion Handling: The courier employs techniques like slow start and congestion avoidance to prevent overwhelming the recipient (receiver).

Technical Steps:

1. Initial Transmission: The sender transmits a packet and starts a timer.
2. Acknowledgment Check: If the recipient confirms receipt (ACK) before the timeout, the courier moves on to the next package.
3. Timeout: If no ACK arrives within the timeout, the courier assumes a problem and resends the package.
4. Adaptive Timeout Adjustment: The courier considers factors like recent delays and congestion to adjust future timeouts, preventing unnecessary retransmissions or premature timeouts.
5. Congestion Control: The courier implements strategies like slow start (gradual data transmission increase) and congestion avoidance (transmission rate reduction when congestion detected) to maintain network stability.

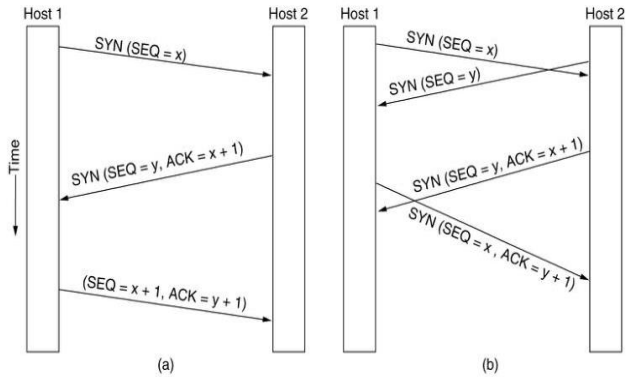
6. Continuous Adaptation: The courier refines timeouts and congestion control measures throughout the delivery process, ensuring efficient and reliable delivery even in varying network conditions.

Benefits:

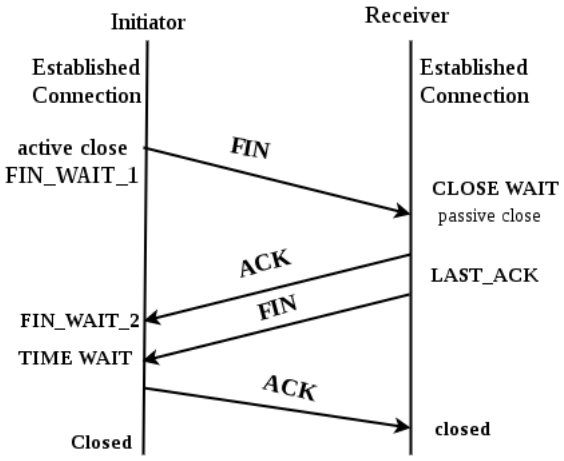
- Improved Efficiency: Minimizes unnecessary retransmissions, reducing network traffic and resource usage.
- Enhanced Reliability: Ensures successful packet delivery even in unpredictable network conditions.
- Responsiveness to Network Changes: Adapts to varying congestion levels and latency to maintain performance.

Adaptive retransmission is crucial for reliable and efficient data transfer in dynamic network environments.

5.Show the TCP connection establishment and termination using timeline diagram.



(a) TCP connection establishment in the normal case.
(b) Call collision.



TERMINATION

TCP Connection Establishment Timeline:

1. **H1 sends SYN to H2:** H1 initiates the connection by sending a SYN packet to H2.
2. **H2 responds with SYN-ACK to H1:** H2 acknowledges H1's SYN by sending a SYN-ACK packet.
3. **H1 acknowledges SYN-ACK from H2:** H1 acknowledges H2's SYN-ACK, completing the three-way handshake.
4. **Data Transfer:** Bidirectional data transfer is established between H1 and H2.

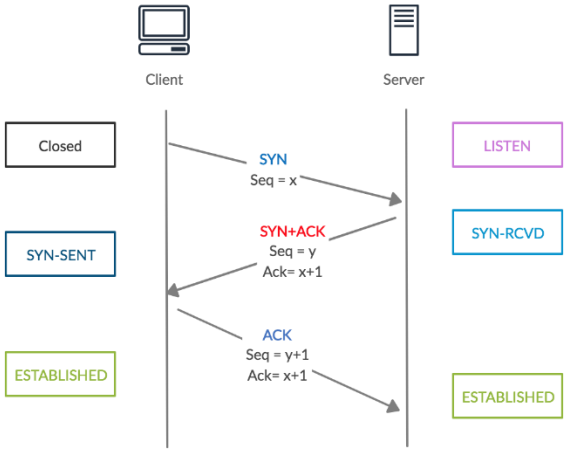
TCP Connection Termination Timeline:

1. **H1 initiates termination by sending FIN to H2:** H1 sends a FIN packet to H2, indicating the desire to close the connection.
2. **H2 acknowledges FIN from H1:** H2 acknowledges H1's FIN, agreeing to terminate the connection.
3. **H2 sends its own FIN to H1:** H2 sends a FIN packet to H1, expressing its intent to close the connection.
4. **H1 acknowledges FIN from H2:** H1 acknowledges H2's FIN, completing the four-way handshake, and the connection is terminated.

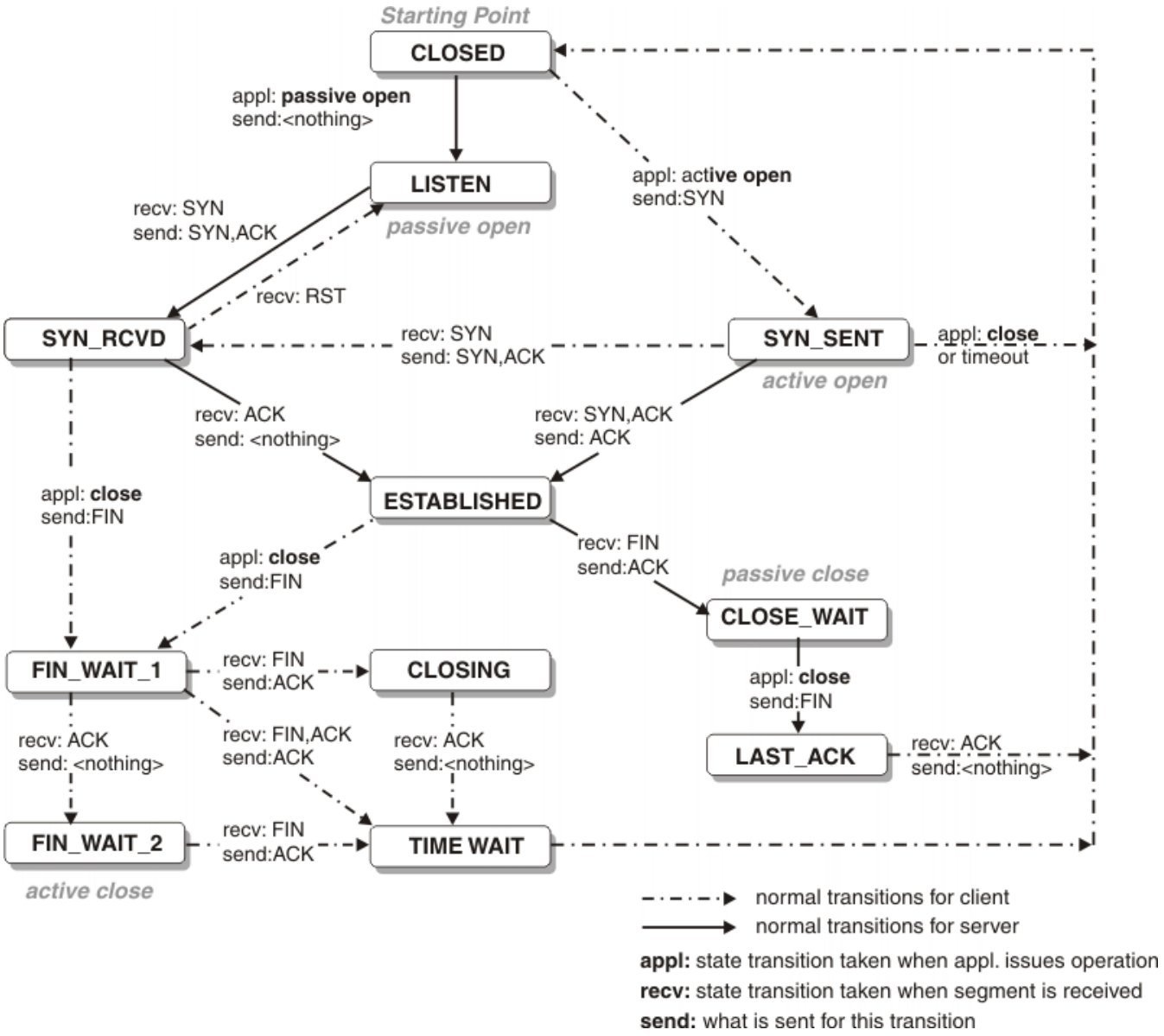
6.Explain the three way handshake protocol to establish the transport level connection.

1. Client Initiates (SYN):
    - Client sends a SYN segment to the server, indicating its desire to connect.
    - Includes a randomly generated Initial Sequence Number (ISN) for tracking data.
    - Client enters SYN-SENT state, awaiting server's response.
2. Server Acknowledges and Synchronizes (SYN-ACK):
    - Server responds with a SYN-ACK segment:
        - Acknowledges client's SYN by setting ACK number to client's ISN + 1.
        - Includes its own ISN for data tracking.
        - Requests synchronization with client.
    - Server enters SYN-RCVD state, awaiting client's final acknowledgment.
3. Client Confirms (ACK):
    - Client sends an ACK segment:
        - Acknowledges server's ISN by setting ACK number to server's ISN + 1.
        - Confirms synchronization.
    - This completes the three-way handshake, establishing the connection.
    - Both client and server transition to the ESTABLISHED state.

- Reliable connection establishment requires three segments (SYN, SYN-ACK, ACK).
- Each segment contains sequence and acknowledgment numbers for tracking data and detecting lost packets.
- Sequence numbers ensure ordered delivery, while acknowledgments confirm successful reception.



7.Design TCP state transition diagram and describe each of it.



1. **CLOSED:**

- Initial state before any connection is established or attempted.

2. **LISTEN:**

- The server is ready to accept incoming connection requests (passive open).

3. **SYN-SENT:**

- The client sends a connection request (active open) and awaits an acknowledgment.

4. **SYN-RCVD:**

- The server has received a connection request and is waiting for the client to acknowledge.

5. **ESTABLISHED:**

- Both client and server have acknowledged the connection request, and data can be exchanged.

6. **FIN-WAIT-1:**

- The side initiating the connection termination sends a FIN to indicate it has finished sending data.

7. **FIN-WAIT-2:**

- The side that received a FIN acknowledges the termination request, indicating it is also finished sending data.

8. **CLOSE-WAIT:**

- The server side has received a FIN from the client and has closed its connection but is waiting for the application to acknowledge termination.

9. **CLOSING:**

- Both sides have initiated termination, and the side that received the first FIN is waiting for an acknowledgment from the other side.

10. **LAST-ACK:**

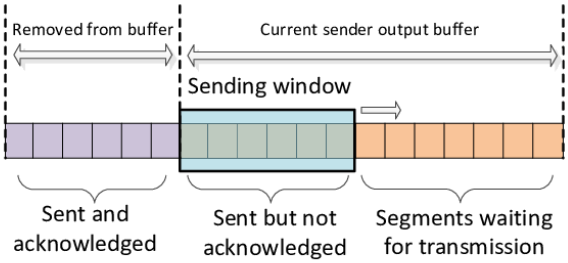
- The side that received the second FIN acknowledges termination and waits for the acknowledgment from the other side.

11. **TIME-WAIT:**

- A connection has been terminated, and the socket is waiting for any remaining packets to arrive before completely closing the connection.

8.Explain a detailed note on connection establishment.(PART-B 6Q)

9.Discuss about the TCP sliding window algorithm for flow control



Purpose:

- Regulates the rate of data transmission between sender and receiver to prevent overwhelming the receiving end and ensure reliable delivery.
- Adapts to dynamic network conditions, such as congestion or varying receiver buffer sizes.

Algorithm Steps:

1. Sender Initiates:
    - Sender sets its initial window size, often based on the receiver's advertised window.
    - Transmits segments up to the window size limit.
2. Receiver Acknowledges:
    - Receiver sends ACKs for each successfully received segment, indicating the next expected byte.
    - ACKs slide the sender's window forward, allowing transmission of new segments.

3. Congestion Control:
    - If the sender detects congestion (e.g., lost packets, timeouts), it reduces the window size to slow down transmission.
4. Adaptive Window Adjustment:
    - The sender dynamically adjusts the window size based on ACKs, congestion signals, and network conditions.

Benefits:

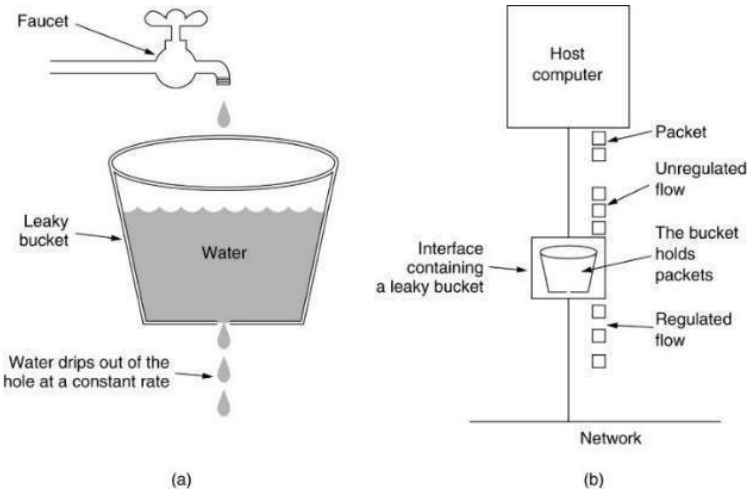
- Prevents Buffer Overflow: Ensures the sender doesn't overwhelm the receiver's buffer, preventing data loss.
- Congestion Avoidance: Helps mitigate network congestion, improving overall performance.
- Efficient Utilization: Optimizes network resources by adjusting transmission rates based on available capacity.
- Ordered Delivery: Guarantees that data is received in the correct order, essential for many applications.

10.Summarize all congestion control algorithms and describe how it works

Congestion Control Algorithm	Working Principle
Slow Start	Exponentially increases congestion window until a threshold or congestion is detected.
Congestion Avoidance	Linearly increases congestion window to alleviate congestion, slowing growth to prevent overload.
Fast Recovery	Reduces congestion window by half upon triple duplicate ACK, enters fast recovery, retransmits missing segment, and exits upon acknowledgment.
TCP Reno	Utilizes fast recovery and fast retransmit mechanisms triggered by triple duplicate ACKs, reducing congestion window to mitigate congestion.
TCP Vegas	Measures congestion based on packet delay rather than packet loss, utilizing RTT deviation to identify and alleviate congestion proactively.
TCP New Reno	Allows multiple packet transmissions in fast recovery mode, addressing halting of transmission upon the first packet loss in a window.
TCP Cubic	Uses a cubic function to adjust congestion window, improving bandwidth estimation and fairness over high-speed networks.

11.Illustrate leaky bucket and token bucket algorithm.

LEAKY BUCKET



- The leaky bucket algorithm discovers its use in the context of network traffic shaping or rate-limiting.
- A leaky bucket execution and a token bucket execution are predominantly used for traffic shaping algorithms.
- The disadvantages compared with the leaky-bucket algorithm are the inefficient use of available network resources.
- The large area of network resources such as bandwidth is not being used effectively.

Key Points:

- Smooths out bursty traffic, enforcing a consistent transmission rate.
- Prevents network congestion by limiting peak traffic. Simple to Implement

TOKEN BUCKET:

- In the Token Bucket, tokens (representing data transmission permissions) are added to the bucket at a predefined rate.
- Packets arriving at the bucket can only be transmitted if enough tokens are available to cover their size.
- If tokens are insufficient, the packet may be buffered, delayed, or dropped.
- Unused tokens are retained, allowing bursts up to the bucket's capacity if enough tokens accumulate.

Key Points:

- More flexible than Leaky Bucket, allowing bursts of traffic up to the bucket's capacity.
- Accommodates varying traffic patterns while still enforcing an average rate limit.
- Often used in network switches and routers for traffic shaping.

Main Differences:

- Leaky Bucket: Enforces a strict, constant rate.
- Token Bucket: Allows bursts up to a certain capacity, providing more flexibility.

Choosing the Right Algorithm:

- Leaky Bucket: Ideal for applications requiring strict rate control and predictability.
- Token Bucket: Better suited for handling bursty traffic and accommodating varying rates.

12.Compare & Contrast UDP & TCP with suitable example.

Aspect	UDP (User Datagram Protocol)	TCP (Transmission Control Protocol)
Reliability	Low: No guarantee of delivery or order.	High: Ensures reliable, ordered delivery of data.
Connection State	Connectionless: No ongoing connection setup.	Connection-oriented: Establishes a connection before data transfer.
Acknowledgments	No acknowledgment of receipt.	Acknowledges each segment, ensuring data integrity.
Order of Delivery	Unordered delivery: No guarantee of sequence.	Ordered delivery: Ensures data arrives in the correct sequence.
Speed	Fast: Lightweight and quick transmission.	Slower: Additional overhead for reliability and ordering.
Use Cases	Real-time applications (e.g., VoIP, streaming).	Applications requiring data integrity (e.g., file transfer).
Example Applications	Video streaming, online gaming.	File transfer, web browsing.

13.Explain congestion avoidance techniques in detail.

Open-Loop Congestion Avoidance:

- Concept: Prevents congestion proactively without relying on direct feedback from the network.
- Mechanisms:
    - Traffic Shaping: Enforces predetermined transmission rates using Leaky Bucket or Token Bucket algorithms, smoothing out traffic and preventing sudden bursts.
    - Admission Control: Regulates the number of active connections or flows allowed in a network, ensuring available resources aren't overwhelmed.
    - Resource Reservation: Pre-allocates bandwidth or buffers for specific traffic types or applications, guaranteeing priority for critical flows.

Closed-Loop Congestion Avoidance:

- Concept: Reacts to congestion based on feedback from the network, adjusting transmission rates dynamically.
- Mechanisms:

- Window-Based Flow Control (e.g., TCP Slow Start, Congestion Avoidance): Uses sliding windows and acknowledgments to regulate data flow between sender and receiver.
- Explicit Congestion Notification (ECN): Receivers explicitly signal congestion to senders, prompting them to reduce transmission rates.
- Congestion-Based Routing: Protocols select paths that avoid congested links, distributing traffic more efficiently.

▪ Hybrid Approaches:
▪ Some congestion control methods use a combination of open-loop and closed-loop techniques.
▪ For instance, TCP's AIMD algorithm is open-loop in its general behavior but adapts to network feedback (like packet loss) to adjust its congestion window.

Key Differences:

- Open-Loop: Preventative measures, acting before congestion occurs.
- Closed-Loop: Reactive measures, responding to congestion signals.

14.List major types of networks and give brief note on each of it.

All types of networks have significance based on their specific functionalities and applications. However, the importance of a network type often depends on the context and the requirements of a particular scenario. Here are a few that are particularly crucial in modern computing and connectivity:

1. Local Area Network (LAN):

- Importance: Vital for in-house communication, resource sharing, and local connectivity within organizations, homes, or campuses.
- Range: Typically spans a small geographic area, like a single building, office, or campus, covering up to a few kilometers.

2. Wide Area Network (WAN):

- Importance: Enables global connectivity, facilitating internet access and communication between geographically dispersed locations, crucial for interconnecting offices, cloud services, and the internet.
- Range: Spans large geographical distances, connecting networks across cities, countries, or continents, encompassing global or transcontinental distances.

3. Wireless LAN (WLAN):

- Importance: Ubiquitous for wireless connectivity in various settings like homes, offices, public spaces, providing flexibility and convenience in device connectivity.
- Range: Covers a limited area, functioning wirelessly within buildings, homes, or outdoor spaces, spanning from a few hundred feet to a few hundred meters.

4. Virtual Private Network (VPN):

- Importance: Offers secure and encrypted connections over public networks, crucial for remote work, accessing private networks securely, and maintaining privacy.

- Range: Operates over existing public networks (like the internet), connecting networks or devices globally, extending across vast geographical distances.

5. Client-Server Network:

- Importance: Fundamental for numerous services, such as web hosting, email, and database management, as it supports centralized resource management and user access.
- Range: Connection between client devices and centralized servers, often within LANs or WANs, scaling from local environments to global networks.

6. Storage Area Network (SAN):

- Importance: Critical in data centers and enterprises for high-speed, centralized storage access, facilitating efficient data management and storage.
- Range: Operates within or across data centers, facilitating high-speed data access and storage, spanning from localized setups within a single building to distributed configurations across multiple locations.

15.Illustrate data units at different layers of the TCP / IP protocol suite.

TCP/IP Protocol Suite Layers:

1. Application Layer:

- **Data Unit:** Message, Segment or Datagram.
- **Description:** This layer deals with user interactions, providing services directly to applications. The data unit at this layer is referred to as a "message" in terms of the information exchanged between applications. When passed to the transport layer, it's segmented into "segments" (TCP) or "datagrams" (UDP).

2. Transport Layer:

- **Data Unit:** Segment or Datagram.
- **Description:** Responsible for end-to-end communication, ensuring data delivery between hosts. For TCP, the data unit is a "segment," which includes header information and actual data. For UDP, it's a "datagram," a basic unit of data that gets passed to the network layer.

3. Internet Layer:

- **Data Unit:** Packet or Datagram.
- **Description:** Focuses on routing and forwarding data packets across networks. It takes the "segment" (from TCP) or "datagram" (from UDP) and encapsulates it into an IP packet or datagram, attaching routing information (source and destination IP addresses) for transmission.

4. Link Layer:

- **Data Unit:** Frame.
- **Description:** Handles data transfer within a local network. It encapsulates the IP packet or datagram into a "frame" and includes physical addresses (MAC addresses) for devices within the same network segment.

16.Discuss in detail about the connection establishment and release in TCP.

TCP Connection Establishment (Three-Way Handshake):

1. **SYN (Synchronize):**
    - **Client → Server:** Initiates the connection by sending a SYN segment to the server.
    - **Function:** Includes a randomly generated Initial Sequence Number (ISN) to start communication.
2. **SYN-ACK (Synchronize-Acknowledge):**
    - **Server → Client:** Responds to the client's SYN with a SYN-ACK segment.
    - **Function:** Acknowledges the SYN, includes its own ISN, and confirms the client's readiness to establish a connection.
3. **ACK (Acknowledge):**
    - **Client → Server:** Acknowledges the server's SYN-ACK by sending an ACK segment.
    - **Function:** Confirms the receipt of the server's ISN, indicating that both sides are synchronized and ready for data exchange.

TCP Connection Termination (Four-Way Handshake):

1. **FIN (Finish):**
    - **Initiator (Client or Server) → Other Party:** Initiates termination by sending a FIN segment.
    - **Function:** Indicates the sender has no more data to transmit and wants to terminate the connection.
2. **ACK (Acknowledge):**
    - **Recipient → Initiator:** Responds to the FIN with an ACK segment.
    - **Function:** Confirms the FIN, acknowledging the termination request.
3. **FIN (Finish):**
    - **Recipient → Initiator:** When the recipient has no more data to send, it sends its FIN segment.
    - **Function:** Signals its completion of data transmission and the desire to terminate the connection.
4. **ACK (Acknowledge):**
    - **Initiator → Recipient:** Responds to the second FIN with an ACK segment.
    - **Function:** Confirms the second FIN, acknowledging the termination of the connection.

Detailed Process:

- **Connection Establishment:** Initiates with SYN, acknowledges with SYN-ACK, and confirms with ACK, establishing a synchronized state for data transfer.

- **Connection Termination:** Starts with FIN, acknowledged by an ACK; followed by a reciprocal FIN, acknowledged by another ACK, completing the termination.

17.Draw and explain each fieldin the TCP Segment header. Part-A last ques

18.Explain leaky bucket and token bucket algorithms. Part-b 11q

19.Explain in detail about transport layer protocols

There are two main protocols at the transport layer:

1. Transmission Control Protocol (TCP):

- Connection-oriented: Establishes a virtual circuit between sender and receiver before data exchange begins.
- Reliable delivery: Uses sequence numbers, acknowledgments, and retransmissions to ensure all data packets arrive in the correct order and without errors.
- Flow control: Regulates the rate of data transmission to prevent the receiver from being overwhelmed.
- Congestion control: Adapts to network conditions and avoids congestion by dynamically adjusting the window size.
- Common applications: Web browsing, email, file transfer, online gaming.

2. User Datagram Protocol (UDP):

- Connectionless: Sends data packets independently without establishing a connection beforehand.
- Unreliable delivery: Does not guarantee ordering or error-checking of data packets.
- Faster and simpler: Requires less overhead than TCP, making it suitable for real-time applications.
- Flow control: Optional and application-specific.
- Congestion control: Minimal or application-specific.
- Common applications: Streaming media, video conferencing, online gaming, DNS queries.

Choosing the right transport protocol depends on the specific application requirements:

- Use TCP for applications that require reliable and ordered data delivery, such as file transfer or email.
- Use UDP for applications that prioritize speed and efficiency over reliability, such as streaming media or real-time gaming.

20.Explain the services provided by the transport layer

Transport Layer Services: Keeping the Data Flowing Smoothly

The transport layer sits between the application layer and the network layer, acting as a critical mediator in network communication. It offers several key services that ensure reliable and efficient data transfer between applications running on different devices:

1. End-to-End Delivery:

- Guarantees data reaches the intended recipient application, not just the destination device.
- Achieved through:
    - Connection establishment: TCP sets up a virtual circuit, while UDP utilizes port addressing.
    - Addressing: Transport headers include source and destination port numbers for application identification.
    - Routing: Transport layer information guides packets through the network based on the destination address.

2. Reliable Delivery:

- Ensures data arrives complete and in the correct order, especially relevant for sensitive applications.
- TCP prioritizes reliable delivery through:
    - Sequence numbers: Track the order of data segments.
    - Acknowledgments (ACKs): Confirm successful data reception.
    - Retransmissions: Re-send lost or corrupted packets.

3. Flow Control:

- Prevents overwhelming the receiver by regulating the rate of data transmission.
- Achieved through:
    - Window size: Defines the maximum amount of data the sender can transmit without an ACK.
    - Congestion control algorithms: Adapt window size based on network conditions to avoid congestion.

4. Multiplexing:

- Enables multiple applications on the same device to share a single network connection.
- Achieved through:
    - Ports: Unique identifiers assigned to applications for data differentiation.
    - Demultiplexing: Transport layer identifies and delivers data to the correct application based on its port number.

5. Error Detection and Correction:

- Identifies and corrects errors that may occur during data transmission.
- Achieved through:
    - Checksums: Calculated values appended to data to detect corruption.
    - Negative acknowledgments (NAKs): Signals receiver identified corrupted data, prompting retransmission.

6. Connection Management:

- Established in TCP for reliable communication, involves:
    - Three-way handshake: Initiates and confirms the connection before data exchange.
    - Tear-down process: Gracefully closes the connection upon completion.