CN

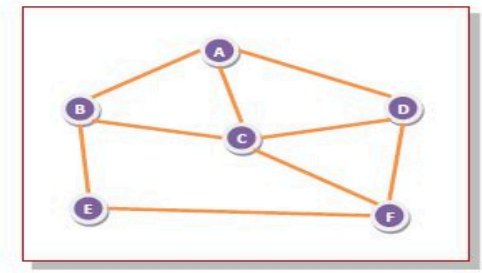
Previous Solve by Maisha

Why is flooding called a robust routing algorithm? Describe flooding with an illustration.

Why is Flooding Called a Robust Routing Algorithm?

Flooding is considered a **robust routing algorithm** because it ensures the delivery of packets under almost all circumstances, even in unreliable or damaged network conditions. Here's why:

1. **Guaranteed Delivery:**
   - Flooding sends packets through every possible path. As long as a path exists between the source and destination, the packet will eventually reach its target.
2. **Fault Tolerance:**
   - Even if multiple routers or links fail, flooding can still deliver packets via alternative paths.
3. **No Need for Routing Tables:**
   - Routers only need to know their direct neighbors, making it resilient to changes in the network topology.
4. **Useful in Emergency Scenarios:**
   - Flooding can function effectively in scenarios like military networks or disaster recovery, where the network is unreliable, and rapid message delivery is critical.



Using flooding technique –

- An incoming packet to A, will be sent to B, C and D.
- B will send the packet to C and E.
- C will send the packet to B, D and F.
- D will send the packet to C and F.
- E will send the packet to F.
- F will send the packet to C and E.

How does congestion control differ from flow control? Describe any two congestion control algorithms.

Flow Control:

Flow control manages the rate of data transmission between a sender and a receiver to ensure that the receiver can handle the incoming data without being overwhelmed. This end-to-end mechanism prevents data loss by ensuring the sender does not transmit more data than the receiver's buffer can accommodate. GEEKSFORGEEKS

Congestion Control:

Congestion control regulates the flow of data into the network to prevent or alleviate network congestion, which occurs when the aggregate data load exceeds the network's capacity. This mechanism ensures efficient utilization of network resources and maintains overall network performance. GEEKSFORGEEKS

Key Differences:

- **Scope:**
  - *Flow Control:* Operates between a single sender and receiver.
  - *Congestion Control:* Operates across the entire network.
- **Purpose:**
  - *Flow Control:* Prevents the receiver from being overwhelmed.
  - *Congestion Control:* Prevents the network from becoming congested.
- **Implementation Layer:**
  - *Flow Control:* Implemented at the data link and transport layers.
  - *Congestion Control:* Implemented primarily at the transport layer.

Congestion Control Algorithms:

1. **Leaky Bucket Algorithm:**
   - **Function:** Regulates data transmission by controlling the rate at which packets are sent into the network, smoothing out bursts of traffic.
   - **Mechanism:** Packets are added to a finite-capacity bucket and transmitted at a constant rate. If the bucket overflows due to excessive incoming packets, those packets are discarded, effectively controlling congestion.
   - **Analogy:** Similar to water dripping from a leaky bucket at a steady rate, regardless of the input rate.
   - **Use Case:** Effective in shaping traffic to ensure a consistent flow rate, preventing sudden bursts that could overwhelm the network.

2. **Token Bucket Algorithm:**
    - **Function:** Allows data transmission bursts up to a certain limit while controlling the average data rate, providing flexibility in data flow.
    - **Mechanism:** Tokens are generated at a constant rate and collected in a bucket. To transmit a packet, a corresponding number of tokens is required. This mechanism permits bursty transmissions when tokens have accumulated, while ensuring the average rate remains controlled.
    - **Analogy:** Similar to having tokens that permit sending packets; if enough tokens are available, a burst of packets can be sent.
    - **Use Case:** Suitable for applications that experience variable data rates, allowing for efficient utilization of available bandwidth.

How does the environment of the transport layer differ from the environments of the network layer and the data link layer?

In computer networking, the **Transport Layer**, **Network Layer**, and **Data Link Layer** each operate within distinct environments and serve unique functions within the OSI (Open Systems Interconnection) model.

Data Link Layer (Layer 2):

- **Scope:** Manages node-to-node communication over a single physical link.
- **Functions:**
    - **Framing:** Packages raw bits from the Physical Layer into frames.
    - **Error Detection and Correction:** Identifies and rectifies errors that occur at the Physical Layer.
    - **Flow Control:** Ensures that the rate of data transmission does not overwhelm the receiving node.
- **Environment:** Operates within a local network segment, facilitating communication between directly connected devices.

Network Layer (Layer 3):

- **Scope:** Handles communication between devices across multiple networks.
- **Functions:**
  - **Routing:** Determines optimal paths for data to travel from source to destination across interconnected networks.
  - **Logical Addressing:** Assigns IP addresses to devices, enabling identification and communication over diverse networks.
  - **Packet Forwarding:** Moves packets through routers from the source network to the destination network.
- **Environment:** Operates in an internetwork environment, managing data transmission across various network boundaries.

Transport Layer (Layer 4):

- **Scope:** Facilitates end-to-end communication between applications on different hosts.
- **Functions:**
  - **Segmentation and Reassembly:** Divides large messages into smaller segments for transmission and reassembles them at the destination.
  - **Connection Management:** Establishes, maintains, and terminates connections between applications.
  - **Error Correction and Flow Control:** Ensures complete data transfer without errors and manages data flow to prevent congestion.
- **Environment:** Operates in a host-to-host environment, providing reliable data transfer services directly to applications.

Key Differences:

- **Operational Scope:**
  - *Data Link Layer:* Confined to local network segments.
  - *Network Layer:* Spans multiple networks, managing inter-network communication.
  - *Transport Layer:* Manages end-to-end communication between applications on different hosts.
- **Addressing:**
  - *Data Link Layer:* Utilizes physical (MAC) addresses.
  - *Network Layer:* Employs logical (IP) addresses.
  - *Transport Layer:* Uses port numbers to identify specific applications.
- **Error Handling and Flow Control:**
  - *Data Link Layer:* Manages errors and flow control for node-to-node communication.
  - *Network Layer:* Primarily focuses on routing and forwarding, with limited error handling.
  - *Transport Layer:* Provides comprehensive error correction and flow control for end-to-end communication.

Transport layers uses port address of a process instead of using process ID, why it is so? What is the use of process server?

In computer networks, the **Transport Layer** utilizes **port numbers** instead of operating system-specific **process IDs** to identify communication endpoints. This design choice offers several advantages:

1. **Operating System Independence:**
    - Process IDs are unique only within a specific operating system and can vary across different systems. Relying on process IDs would make network protocols OS-dependent, hindering interoperability. Port numbers, being standardized and universally recognized, facilitate seamless communication across diverse systems. GATE OVERFLOW
2. **Support for Multiple Connections:**
    - A single process may manage multiple network connections simultaneously. Process IDs cannot distinguish between these multiple connections, whereas unique port numbers can. This distinction enables efficient multiplexing and demultiplexing of data streams, ensuring that each connection is correctly identified and managed. GATE OVERFLOW
3. **Well-Known Services Identification:**
    - Certain services are associated with standardized port numbers (e.g., HTTP uses port 80, HTTPS uses port 443). This standardization allows clients to easily locate and connect to specific services without prior knowledge of process IDs, streamlining the connection process.

Use of Process Servers:

A **process server** in networking refers to a server process that manages specific tasks or services, acting as a software engine that manages shared resources such as databases, printers, communication links, or high-powered processors. ECOMPUTERNOTES

Functions of a Process Server:

- **Resource Management:**
  - Oversees access to shared resources, ensuring efficient utilization and preventing conflicts among multiple clients.
- **Service Provisioning:**
  - Offers specific services (e.g., file storage, printing) to clients, handling requests, processing data, and returning appropriate responses.
- **Concurrency Handling:**
  - Manages multiple client connections simultaneously, ensuring that each request is processed correctly without interference.

By utilizing port numbers and process servers, the transport layer ensures reliable, efficient, and organized communication between applications across diverse networked systems.

5.3.1 Approaches to Congestion Control

The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: increase the resources or decrease the load. As shown in Fig. 5-22, these solutions are usually applied on different time scales to either prevent congestion or react to it once it has occurred.

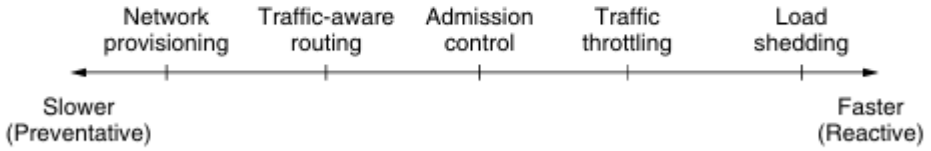


Figure 5-22. Timescales of approaches to congestion control.

Explicit Congestion Notification

Instead of generating additional packets to warn of congestion, a router can tag any packet it forwards (by setting a bit in the packet's header) to signal that it is experiencing congestion. When the network delivers the packet, the destination can note that there is congestion and inform the sender when it sends a reply packet. The sender can then throttle its transmissions as before.

This design is called **ECN** (**Explicit Congestion Notification**) and is used in the Internet (Ramakrishnan et al., 2001). It is a refinement of early congestion signaling protocols, notably the binary feedback scheme of Ramakrishnan and Jain (1988) that was used in the DECNET architecture. Two bits in the IP packet header are used to record whether the packet has experienced congestion. Packets are unmarked when they are sent, as illustrated in Fig. 5-25. If any of the routers they pass through is congested, that router will then mark the packet as having experienced congestion as it is forwarded. The destination will then echo any marks back to the sender as an explicit congestion signal in its next reply packet. This is shown with a dashed line in the figure to indicate that it happens above the IP level (e.g., in TCP). The sender must then throttle its transmissions, as in the case of choke packets.

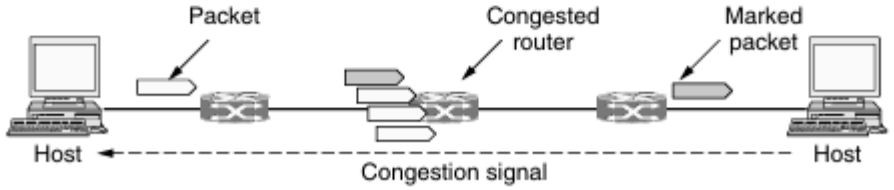


Figure 5-25. Explicit congestion notification

What is ECN?

- **Explicit Congestion Notification (ECN):**
  - A signaling mechanism for congestion control in the Internet.
  - Allows routers to mark packets (rather than dropping them) to indicate congestion.

How ECN Works:

1. **Marking Packets:**
   - Routers experiencing congestion mark packets by setting a specific bit in the packet's header.
   - Packets are unmarked when initially sent but may be marked as they pass through congested routers.
2. **Congestion Signal:**
   - The destination checks the marked packets and sends a congestion notification back to the sender in its next reply packet.
3. **Action Taken:**
   - The sender, upon receiving the congestion signal, reduces its transmission rate (similar to handling choke packets).

5.4 QUALITY OF SERVICE

The techniques we looked at in the previous sections are designed to reduce congestion and improve network performance. However, there are applications (and customers) that demand stronger performance guarantees from the network than "the best that could be done under the circumstances." Multimedia applications in particular, often need a minimum throughput and maximum latency to work. In this section, we will continue our study of network performance, but now with a sharper focus on ways to provide quality of service that is matched to application needs. This is an area in which the Internet is undergoing a long-term upgrade.

An easy solution to provide good quality of service is to build a network with enough capacity for whatever traffic will be thrown at it. The name for this solution is **overprovisioning**. The resulting network will carry application traffic without significant loss and, assuming a decent routing scheme, will deliver packets with low latency. Performance doesn't get any better than this. To some extent, the telephone system is overprovisioned because it is rare to pick up a telephone and not get a dial tone instantly. There is simply so much capacity available that demand can almost always be met.

The trouble with this solution is that it is expensive. It is basically solving a problem by throwing money at it. Quality of service mechanisms let a network with less capacity meet application requirements just as well at a lower cost. Moreover, overprovisioning is based on expected traffic. All bets are off if the traffic pattern changes too much. With quality of service mechanisms, the network can honor the performance guarantees that it makes even when traffic spikes, at the cost of turning down some requests.

Four issues must be addressed to ensure quality of service:

1. What applications need from the network.
2. How to regulate the traffic that enters the network.
3. How to reserve resources at routers to guarantee performance.
4. Whether the network can safely accept more traffic.

No single technique deals efficiently with all these issues. Instead, a variety of techniques have been developed for use at the network (and transport) layer. Practical quality-of-service solutions combine multiple techniques. To this end, we will describe two versions of quality of service for the Internet called Integrated Services and Differentiated Services.

5.4.1 Application Requirements

A stream of packets from a source to a destination is called a **flow** (Clark, 1988). A flow might be all the packets of a connection in a connection-oriented network, or all the packets sent from one process to another process in a connectionless network. The needs of each flow can be characterized by four primary parameters: bandwidth, delay, jitter, and loss. Together, these determine the **QoS** (**Quality of Service**) the flow requires.

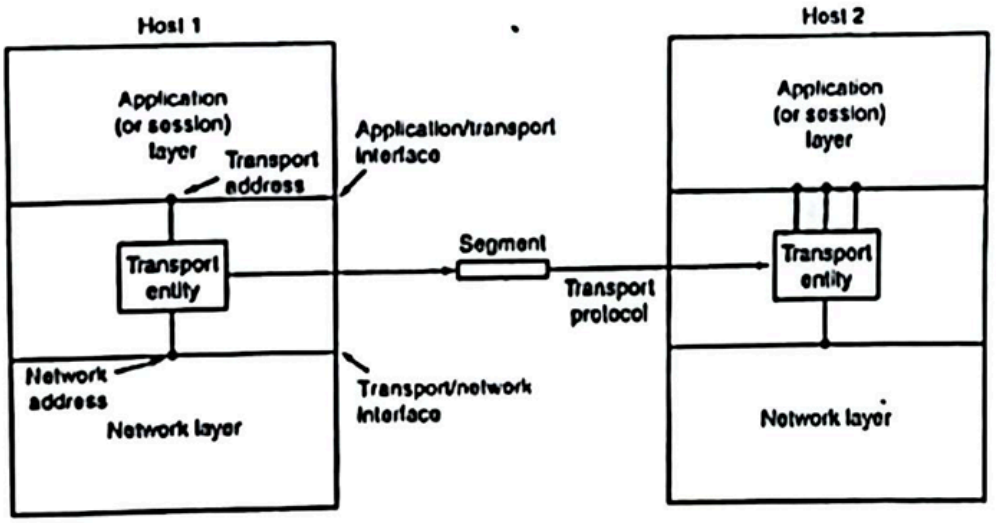
Several common applications and the stringency of their network requirements are listed in Fig. 5-27. Note that network requirements are less demanding than application requirements in those cases that the application can improve on the service provided by the network. In particular, networks do not need to be lossless for reliable file transfer, and they do not need to deliver packets with identical delays for audio and video playout. Some amount of loss can be repaired with retransmissions, and some amount of jitter can be smoothed by buffering packets at the receiver. However, there is nothing applications can do to remedy the situation if the network provides too little bandwidth or too much delay.

Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

Figure 5-27. Stringency of applications' quality-of-service requirements.

1. **What is QoS?**
    - A mechanism to guarantee network performance for specific applications beyond general network optimizations.
    - Focuses on ensuring reliability, low latency, and sufficient bandwidth for critical applications.
2. **Overprovisioning:**
    - A simple method to achieve QoS by building a network with enough capacity to handle all traffic without congestion.
    - Guarantees low latency and no packet loss.
    - **Drawback:** Expensive and inefficient as it requires excessive resources.
3. **QoS Mechanisms:**
    - Instead of overprovisioning, QoS techniques allow networks with limited capacity to meet application needs efficiently.
    - QoS adapts to expected traffic but may struggle with unexpected traffic spikes.
4. **Challenges in QoS:**
    - Ensuring that:
        1. Applications receive the resources they need.
        2. Traffic doesn't exceed the network's capacity.
        3. Network performance is maintained even during traffic spikes.
        4. All applications receive fair treatment.
5. **Solutions:**
    - Multiple techniques are implemented at the **network and transport layers**.
    - Two key approaches:
        - **Integrated Services:** Guarantees specific levels of service for individual flows.
        - **Differentiated Services:** Prioritizes traffic by categories rather than individual flows.

Given the following figure, explain roles of layers, entities and addresses:



Roles of the Layers:

1. Application (or Session) Layer:
   - Provides the interface for user applications to communicate.
   - Uses the **Transport Address** to identify the specific application or process involved in the communication.
2. Transport Layer:
   - Manages end-to-end communication between hosts.
   - Ensures reliable data delivery, error correction, and segmentation/reassembly.
   - Uses **Transport Protocols** like TCP/UDP to deliver data.
3. Network Layer:
   - Handles the delivery of data packets between devices across multiple networks.
   - Uses **Network Addresses** (e.g., IP addresses) to route packets to the destination host.

Roles of Entities:

1. Transport Entity:
   - Manages the transport-layer functionality, such as establishing, maintaining, and terminating connections.
   - Interfaces with both the **application/session layer** and the **network layer**.
   - Converts messages into **segments** for transmission.

Roles of Addresses:

1. **Transport Address (Port Number):**
   - Identifies the specific process or application on the host.
   - Ensures the correct application receives the transmitted data.
2. **Network Address (IP Address):**
   - Identifies the specific host in the network.
   - Ensures the packet reaches the correct device.

How does TCP take advantage of successful segment delivery? How does TCP tackle congestion?

TCP (Transmission Control Protocol) employs several mechanisms to ensure efficient and reliable data transmission over networks. Here's how it leverages successful segment delivery and addresses congestion:

Leveraging Successful Segment Delivery:

- **Cumulative Acknowledgments (ACKs):** TCP uses cumulative acknowledgments, where the receiver acknowledges the highest sequence number of bytes received in order. This acknowledgment implies that all preceding bytes have been successfully received. By receiving these ACKs, the sender infers successful segment delivery and can continue transmitting new data without waiting for individual acknowledgments for each segment. TUTORIALSPOINT

Congestion Control Mechanisms:

1. **Congestion Window (cwnd):** TCP maintains a congestion window that dictates the amount of data that can be sent without receiving an ACK. The size of this window adjusts based on network conditions to prevent congestion. GEEKSFORGEEKS
2. **Slow Start:** At the beginning of a connection or after a timeout, TCP starts with a small congestion window and increases it exponentially with each ACK received. This approach rapidly discovers the network's capacity. GEEKSFORGEEKS
3. **Congestion Avoidance:** Once the congestion window reaches a threshold, TCP shifts to a linear increase of the window size to avoid overwhelming the network. This phase helps in probing the network capacity more cautiously. GEEKSFORGEEKS
4. **Fast Retransmit and Fast Recovery:** If duplicate ACKs are received, indicating potential packet loss, TCP retransmits the missing segment without waiting for a timeout (Fast Retransmit). Following this, it reduces the congestion window and performs additive increase to recover from the loss efficiently (Fast Recovery). GEEKSFORGEEKS

Explain the TCP Congestion Control Mechanism of the Transport Layer.

TCP (Transmission Control Protocol) employs a comprehensive congestion control mechanism to ensure efficient and reliable data transmission across networks. The key components of this mechanism include:

1. **Congestion Window (cwnd):**
    - TCP maintains a congestion window that limits the amount of data a sender can transmit without receiving an acknowledgment. This window adjusts dynamically based on network conditions to prevent congestion. GEEKSFORGEEKS
2. **Slow Start:**
    - At the beginning of a connection or after a timeout, TCP initializes the congestion window to a small size. It then increases the window size exponentially with each acknowledgment received, effectively probing the network to discover its capacity. GEEKSFORGEEKS
3. **Congestion Avoidance:**
    - Once the congestion window reaches a certain threshold, TCP transitions to a linear increase of the window size. This cautious approach helps prevent overwhelming the network, maintaining stability and avoiding congestion. GEEKSFORGEEKS
4. **Fast Retransmit and Fast Recovery:**
    - Upon detecting packet loss, typically inferred from duplicate acknowledgments, TCP promptly retransmits the missing segment without waiting for a timeout (Fast Retransmit). Subsequently, it reduces the congestion window and employs a linear increase strategy to recover from the loss efficiently (Fast Recovery). GEEKSFORGEEKS

Explain whether reliable connection release is possible in the transport layer or not.

Yes, **reliable connection release** is achievable in the transport layer, particularly through protocols like **TCP (Transmission Control Protocol)**. TCP ensures that both parties involved in a communication session can terminate the connection gracefully, confirming that all transmitted data has been received and acknowledged before closing the connection.

TCP Connection Termination Process:

The termination of a TCP connection involves a process known as the **four-way handshake**, which includes the following steps:

1. **FIN Initiation:**
    - The party wishing to terminate the connection sends a segment with the **FIN (Finish)** flag set, indicating its intent to close the connection.
2. **ACK Response:**
    - Upon receiving the FIN segment, the opposite party responds with an **ACK (Acknowledgment)** segment, acknowledging the receipt of the FIN request.
3. **FIN from Receiver:**
    - The receiver of the initial FIN then sends its own FIN segment to indicate that it is ready to terminate the connection from its side.
4. **Final ACK:**
    - The initiator of the termination responds with a final ACK, confirming the receipt of the FIN from the other party.

This process ensures that both sides have agreed to terminate the connection and that all data has been successfully transmitted and acknowledged before the connection is closed.

) TCP uses a Transmission Timer to retransmit lost segments. Discuss the Transmission Timer with a figure.

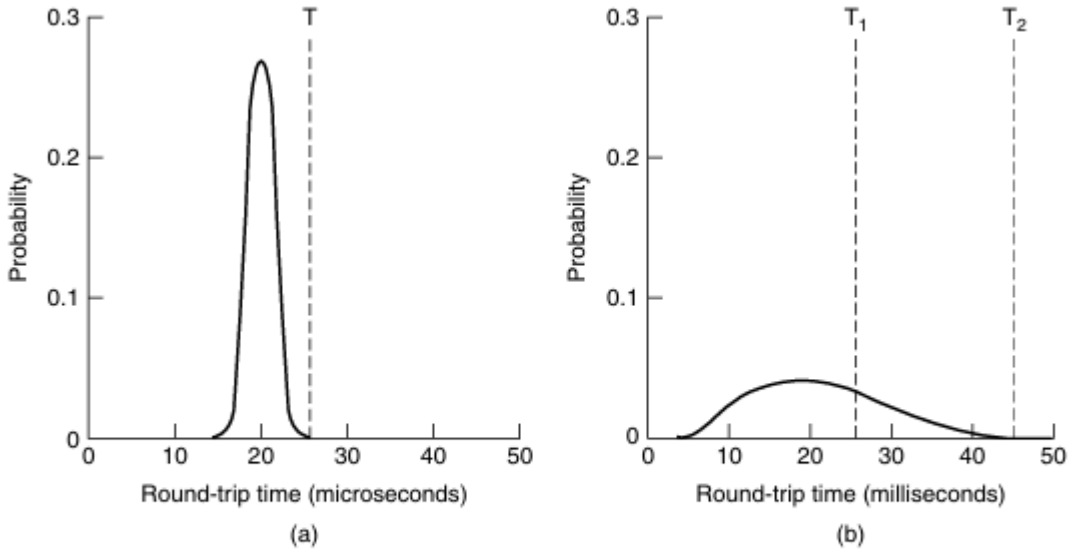


Figure 6-42. (a) Probability density of acknowledgement arrival times in the data link layer. (b) Probability density of acknowledgement arrival times for TCP.

Steps in Using the Retransmission Timer

1. **Starting the Timer:**
   - When a segment is sent, the timer starts.
   - TCP waits for the acknowledgment (ACK) from the receiver.
2. **Stopping the Timer:**
   - If the ACK is received within the timeout, the timer stops.
   - The connection proceeds with sending the next segment.
3. **Retransmission:**
   - If the timer expires before the ACK is received, TCP retransmits the segment.
   - The timer is restarted, with a doubled timeout interval to handle congestion.

Illustrative Figure

Referencing **Figure 6-42 (a)** and **Figure 6-42 (b)**:

1. **Figure 6-42 (a):** Shows predictable and low RTT variance (e.g., in a data link layer). Timer is set slightly longer than the RTT.
2. **Figure 6-42 (b):** Illustrates the unpredictable, variable RTT environment in TCP. The timeout interval dynamically adjusts based on RTT variance.

a) Suppose you have registered your newly created website named www. cse.iiuc.ac.bd with a DNS server. Describe the DNS iterative query for searching the IP address of your cse web server with proper diagram.

→ metallic pat

When a client seeks to resolve the domain name `www.cse.iiuc.ac.bd` into its corresponding IP address using an **iterative DNS query**, the process involves multiple steps, with each DNS server providing the best information it has, often referring the client to another server closer to the desired information.

Iterative DNS Query Process:

1. **Client to Local DNS Resolver:**
   - The client initiates a query to its local DNS resolver, requesting the IP address for `www.cse.iiuc.ac.bd`.
2. **Local DNS Resolver to Root Name Server:**
   - If the local resolver lacks the cached information, it sends an iterative query to a Root Name Server.
   - The Root Server doesn't have the exact IP but knows the authoritative servers for the Top-Level Domain (TLD) `.bd`.
   - It responds with a referral to the `.bd` TLD Name Servers.
3. **Local DNS Resolver to `.bd` TLD Name Server:**
   - The resolver queries a `.bd` TLD Name Server for `www.cse.iiuc.ac.bd`.
   - The TLD server doesn't have the exact IP but knows the authoritative servers for the `iiuc.ac.bd` domain.
   - It provides a referral to the `iiuc.ac.bd` Name Servers.
4. **Local DNS Resolver to `iiuc.ac.bd` Name Server:**
   - The resolver queries an `iiuc.ac.bd` Name Server for `www.cse.iiuc.ac.bd`.
   - This server doesn't have the exact IP but knows the authoritative servers for the `cse.iiuc.ac.bd` subdomain.
   - It provides a referral to the `cse.iiuc.ac.bd` Name Servers.
5. **Local DNS Resolver to `cse.iiuc.ac.bd` Name Server:**
   - The resolver queries a `cse.iiuc.ac.bd` Name Server for `www.cse.iiuc.ac.bd`.
   - This server is authoritative for the `cse.iiuc.ac.bd` subdomain and responds with the IP address for `www.cse.iiuc.ac.bd`.
6. **Local DNS Resolver to Client:**
   - The local resolver caches the obtained IP address for future queries.
   - It returns the IP address to the client, completing the resolution process.

Describe the architecture of email and compare between the POP3 and IMAP protocol.

Architecture of Email:

1. **User Agents (UAs):**
   - Applications like Outlook, Gmail, or Thunderbird that enable users to compose, send, receive, and manage emails.
2. **Mail Transfer Agents (MTAs):**
   - Servers responsible for transferring emails from the sender's server to the recipient's server using protocols such as SMTP (Simple Mail Transfer Protocol).
3. **Mail Delivery Agents (MDAs):**
   - Components that deliver emails to the recipient's mailbox, ensuring proper storage and organization.
4. **Mail Access Agents (MAAs):**
   - Servers that allow users to access and retrieve their emails from the mail server, typically using protocols like POP3 or IMAP.

Email Transmission Process:

- A user composes an email using a User Agent.
- The email is sent to an MTA, which routes it across the internet to the recipient's MTA.
- The recipient's MDA stores the email in the appropriate mailbox.
- The recipient accesses the email via an MAA using an email client.

Comparison between POP3 and IMAP:

Feature	POP3	IMAP
Storage Location	Downloads emails from the server to a single device and often deletes them from the server, limiting access to that device. MICROSOFT SUPPORT	Emails remain on the server, allowing access from multiple devices; actions like reading or deleting sync across all devices. MICROSOFT SUPPORT
Synchronization	Lacks synchronization; actions taken on one device don't reflect on others, leading to inconsistent email states across devices. MICROSOFT SUPPORT	Provides real-time synchronization; changes are mirrored across all devices, ensuring a consistent email experience. MICROSOFT SUPPORT
Folder Management	Limited to downloading emails; doesn't support server-side folder management, hindering organization. GEEKSFORGEEKS	Supports server-side folder management, allowing users to organize emails into folders accessible from any device. GEEKSFORGEEKS
Offline Access	Enables offline access to downloaded emails; suitable for environments with limited internet connectivity. MAKEUSEOF	Primarily designed for online access; some clients offer offline access, but full functionality typically requires an internet connection. MAKEUSEOF
Server Storage	Reduces server storage usage by downloading and deleting emails from the server. MAKEUSEOF	Requires more server storage, as emails are stored on the server to facilitate synchronization across devices. MAKEUSEOF
Use Case Suitability	Ideal for users who access email from a single device and prefer offline access. MAKEUSEOF	Best for users needing access to their email from multiple devices with consistent synchronization. MAKEUSEOF

OR

5. b) Compare and contrast the client-server architecture with the peer-to-peer architecture within the application layer.

Comparison:

- **Control and Management:**
  - *Client-Server:* Centralized control facilitates easier management and enforcement of policies.
  - *P2P:* Lacks centralized oversight, making management more complex.
- **Reliability:**
  - *Client-Server:* Server failures can disrupt services for all clients.
  - *P2P:* The network can remain operational even if some peers fail, due to its distributed nature.
- **Performance:**
  - *Client-Server:* Performance may degrade with high client loads, leading to potential bottlenecks.
  - *P2P:* Performance can improve as more peers contribute resources, though it may suffer if peers have limited capabilities.

Use Cases:

- *Client-Server:* Suitable for environments requiring centralized control, such as corporate networks, web services, and online banking.
- *P2P:* Ideal for applications like file sharing, collaborative platforms, and decentralized communication systems.

Or

Aspect	Client-Server Architecture	Peer-to-Peer (P2P) Architecture
Structure	Features a centralized server that provides resources or services to multiple client devices. Clients initiate requests, and servers respond accordingly.	In this decentralized model, each node (peer) functions both as a client and a server, sharing resources directly with other peers without relying on a central server.
Control and Management	Centralized control facilitates easier management and enforcement of policies.	Lacks centralized oversight, making management more complex.
Reliability	Server failures can disrupt services for all clients.	The network can remain operational even if some peers fail, due to its distributed nature.
Scalability	While servers can handle multiple clients, performance may decline if the number of clients exceeds server capacity.	As more peers join, the network's resource availability increases, potentially enhancing performance.
Security	Centralized control often leads to enhanced security measures and data integrity.	The absence of centralized control can lead to vulnerabilities, as security depends on individual peers.
Cost	Implementing and maintaining servers can be expensive due to hardware and administrative requirements.	Generally more cost-effective, as it leverages existing hardware without the need for dedicated servers.
Data Storage	Data is stored on a centralized server, ensuring consistency and ease of backup.	Each peer maintains its own data, which can lead to inconsistencies and challenges in data synchronization.
Performance	Performance can be affected by server load; high demand on the server may lead to bottlenecks.	Performance can improve with more peers, but may suffer if many peers access the same resource simultaneously.
Examples	Web services, email servers, and online banking systems.	File-sharing networks like BitTorrent, and communication platforms like Skype.

) **Write down the difference between BGP and OSPF routing protocols.**

Aspect	OSPF	BGP
Full Form	Open Shortest Path First	Border Gateway Protocol
Protocol Type	Interior Gateway Protocol (IGP)	Exterior Gateway Protocol (EGP)
Operational Domain	Used within a single autonomous system (intra-domain routing)	Used between different autonomous systems (inter-domain routing)
Routing Algorithm	Utilizes the Dijkstra algorithm (Shortest Path First) to compute the shortest path based on link-state information TECH DIFFERENCES	Employs a path vector mechanism, maintaining the path information that gets updated dynamically TECH DIFFERENCES
Convergence Speed	Generally achieves faster convergence due to immediate propagation of link-state changes PYNET LABS	Converges more slowly, as it relies on incremental updates and complex path selection processes PYNET LABS
Scalability	Suitable for smaller to medium-sized networks; may face challenges scaling in very large networks TECH DIFFERENCES	Highly scalable, designed to handle the vast number of routes on the internet TECH DIFFERENCES
Path Selection	Determines the shortest path based on cumulative link costs TECH DIFFERENCES	Selects the best path based on various attributes, including policies, path length, and network policies TECH DIFFERENCES
Resource Requirements	Can be resource-intensive due to the need to maintain a complete map of the network topology TECH DIFFERENCES	Resource usage depends on the size of the routing table; can be optimized for large-scale deployments TECH DIFFERENCES
Configuration Complexity	Generally easier to configure with automated neighbor discovery PYNET LABS	More complex to configure, requiring manual setup of peers and detailed policy definitions PYNET LABS
Use Cases	Ideal for internal routing within an organization or enterprise network TECH DIFFERENCES	Essential for routing between different organizations, such as between ISPs or large data centers TECH DIFFERENCES

What is the necessity of transport layer? How does transport layer achieve the reliability?

The **Transport Layer** is a crucial component in network architecture, serving as an intermediary between the application layer and the underlying network layers. Its primary responsibilities include:

- **Process-to-Process Communication:** It facilitates direct data exchange between applications on different hosts, ensuring that messages reach the correct process. GEEKSFORGEEKS
- **Multiplexing and Demultiplexing:** By utilizing port numbers, the transport layer allows multiple applications to operate concurrently over the same network connection, directing incoming data to the appropriate application. PREPBYTES
- **Flow Control:** It manages the rate of data transmission to prevent overwhelming the receiver, ensuring efficient and orderly data transfer. PREPBYTES
- **Error Detection and Correction:** The transport layer identifies and rectifies errors that may occur during data transmission, maintaining data integrity. PREPBYTES
- **Segmentation and Reassembly:** It divides large messages into smaller segments for transmission and reassembles them at the destination, accommodating the network's maximum transmission unit (MTU) limitations. WIKIPEDIA

Achieving Reliability:

To ensure reliable data transmission, the transport layer employs several mechanisms:

- **Connection Establishment and Termination:** Protocols like TCP establish a connection-oriented session between sender and receiver, providing a structured communication pathway. WIKIPEDIA
- **Acknowledgments (ACKs):** Upon receiving data segments, the receiver sends acknowledgments back to the sender, confirming successful delivery. WIKIPEDIA
- **Retransmission of Lost Segments:** If an acknowledgment is not received within a specified timeframe, the sender retransmits the unacknowledged segments, ensuring data is not lost. WIKIPEDIA
- **Sequence Numbering:** Each segment is assigned a sequence number, allowing the receiver to detect missing segments and reorder out-of-sequence data appropriately. WIKIPEDIA
- **Error Detection:** Checksums are used to detect errors in transmitted segments. If a segment is found to be corrupted, it is discarded, and the sender is prompted to retransmit it. WIKIPEDIA
- **Flow Control Mechanisms:** Techniques such as the sliding window protocol enable the sender to adjust its transmission rate based on the receiver's capacity, preventing buffer overflow and ensuring smooth data flow. WIKIPEDIA

Discuss on crash recovery strategies of the transport layer.

Crash recovery in the transport layer is essential for maintaining reliable communication, especially when hosts or routers experience failures during data transmission. Effective recovery strategies ensure that connections can be restored without data loss or duplication.

Key Strategies for Crash Recovery:

1. **Client Retransmission Strategies:**
   - **Always Retransmit Last Packet:** Upon detecting a server crash, the client resends the last transmitted packet. This approach can lead to duplicate data if the server had already received and processed the packet before crashing.
   - **Never Retransmit Last Packet:** The client refrains from resending the last packet, assuming it was successfully received. This can result in data loss if the server crashed before acknowledging the packet.
   - **Conditional Retransmission Based on State:** The client maintains states, such as S0 (no outstanding acknowledgment) and S1 (one outstanding acknowledgment). Depending on its state, the client decides whether to retransmit the last packet, balancing the risks of data loss and duplication. SANFOUNDRY
2. **Server Processing Strategies:**
   - **Acknowledge-Then-Write (Strategy 1):** The server first sends an acknowledgment to the client and then writes the data to the application. A crash between these operations can lead to the client believing the data was processed when it wasn't.
   - **Write-Then-Acknowledge (Strategy 2):** The server writes the data to the application before sending an acknowledgment. A crash after writing but before acknowledging can cause the client to retransmit, leading to potential data duplication. FU BERLIN

Challenges in Crash Recovery:

- **Atomicity of Operations:** Ensuring that acknowledgment and data writing are atomic (indivisible) operations is challenging. Non-atomic operations increase the risk of inconsistencies during crashes.
- **State Synchronization:** Maintaining synchronized states between client and server is difficult, especially after unexpected failures.
- **Data Integrity:** Balancing the risks of data loss and duplication requires careful strategy selection and implementation.

How does the email architecture differs from the web architecture?

Email and web architectures serve distinct purposes within the digital communication landscape, each with unique structures and operational models. Here's a comparative overview:

Email Architecture:

- **Purpose:** Facilitates asynchronous communication by enabling users to send and receive messages that can be accessed at the recipient's convenience.
- **Protocols:** Utilizes protocols such as SMTP (Simple Mail Transfer Protocol) for sending emails, and POP3 (Post Office Protocol) or IMAP (Internet Message Access Protocol) for retrieving messages from mail servers.
- **Components:**
  - **Mail User Agent (MUA):** Email clients like Outlook or Gmail that allow users to compose, send, and receive emails.
  - **Mail Transfer Agent (MTA):** Servers responsible for transferring emails from the sender's server to the recipient's server.
  - **Mail Delivery Agent (MDA):** Handles the delivery of emails to the recipient's mailbox.
- **Communication Model:** Operates on a store-and-forward mechanism, where messages are stored on intermediate servers before reaching the final recipient.
- **Data Storage:** Emails are stored on mail servers and can be downloaded to local devices or accessed remotely.
- **Interaction:** Primarily involves composing, sending, receiving, and organizing messages, often with attachments.

Web Architecture:

- **Purpose:** Enables synchronous or asynchronous access to web resources, allowing users to interact with websites and web applications in real-time.
- **Protocols:** Relies mainly on HTTP (Hypertext Transfer Protocol) and HTTPS (HTTP Secure) for transmitting web content.
- **Components:**
  - **Client-Side (Frontend):** The user interface rendered in web browsers, built using HTML, CSS, and JavaScript.
  - **Server-Side (Backend):** Servers that process requests, execute application logic, and interact with databases.
  - **Database Layer:** Stores dynamic content and user data accessed by the backend.
- **Communication Model:** Follows a request-response model, where clients send requests to servers, which then respond with the requested resources.
- **Data Storage:** Web content is hosted on servers and delivered to clients upon request; dynamic data is managed through databases.
- **Interaction:** Involves browsing, submitting forms, streaming media, and other interactive activities facilitated by web applications.

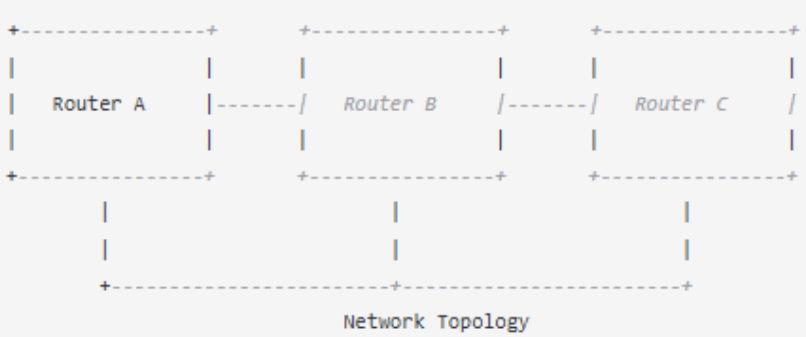
Explain with a figure how the Link State database is built.

In link-state routing protocols, such as OSPF (Open Shortest Path First), each router constructs a **Link-State Database (LSDB)** to maintain a comprehensive view of the network's topology. This database is built through a systematic process involving several key steps:

1. **Link-State Advertisements (LSAs):**
   - Each router generates LSAs containing information about its directly connected links, neighboring routers, and the state (up or down) of those links.
   - These LSAs include details like link costs, which may represent metrics such as bandwidth, delay, or administrative weight.
2. **Flooding:**
   - Routers distribute their LSAs to all other routers within the same area using a process called flooding.
   - Flooding ensures that every router receives the LSAs from all other routers, allowing them to have a consistent and complete view of the network topology.
3. **Link-State Database Construction:**
   - Upon receiving LSAs, a router updates its LSDB, which serves as a repository of the network's topology information.
   - The LSDB contains entries for all routers and their associated links, effectively representing the network as a graph.
4. **Shortest Path Calculation:**
   - With the LSDB populated, each router independently computes the shortest path to every other router using algorithms like Dijkstra's Shortest Path First (SPF).
   - This computation results in the formation of a shortest-path tree, with the calculating router as the root, determining optimal routing paths.

The following diagram illustrates the process of building the Link-State Database:

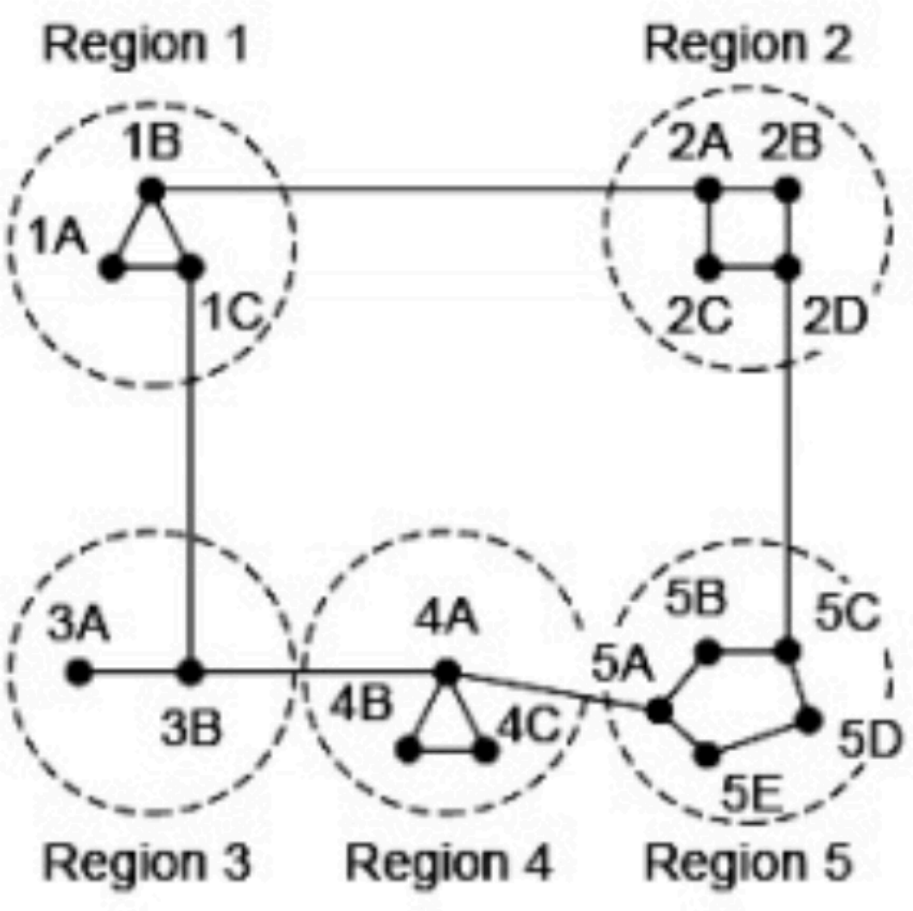
lua Copy code



2. a) Write down the advantages and disadvantages of hierarchical routing with the help of an appropriate network diagram.

Advantages of HRP

- **Scalability:** Hierarchical routing protocols exhibit excellent scalability by partitioning the network into smaller segments or areas. This division reduces the demand for routing tables and updates on each router, enhancing network efficiency and decreasing overall network traffic.
- **Better Traffic Control:** Hierarchical routing protocols demonstrate superior traffic management compared to flat routing protocols. The hierarchical framework enables more efficient traffic control, mitigating the need for unnecessary routing updates and preventing loops in the network.
- **Easy to Manage:** The organisational framework in these protocols facilitates simplified management and maintenance. Segmentation of the network into manageable sections enhances the ease of troubleshooting and diagnosing issues.



Full Table for 1A

Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6

5E	1C	5

When routing is done hierarchically then there will be only 7 entries as shown below –

Hierarchical Table for 1A

Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Unfortunately, this reduction in table space comes with the increased path length.