**Q1)What do you mean by Transport layer multiplexing and demultiplexing. Explain in brief.**

Transport layer multiplexing and demultiplexing are fundamental concepts in network communication, primarily associated with how data is handled as it moves between devices over a network.

**Multiplexing**

Multiplexing at the transport layer involves combining data from multiple applications or processes into a single stream of data that can be transmitted over a network. This process ensures efficient use of the network by sharing the same transport connection (e.g., a TCP or UDP connection) among multiple applications.

**Key aspects of multiplexing include:**

**Source Port Number:** Each application or process on a sender's machine is identified by a unique port number. When an application sends data, the transport layer adds a header to the data segment containing the source port number.

**Encapsulation:** The data from different applications is encapsulated with the appropriate headers (such as TCP or UDP headers) which include the source port numbers.

**Demultiplexing**

Demultiplexing is the process of separating this combined stream of data at the receiving end and delivering the segments to the correct application or process. It involves using the header information added during multiplexing to determine where each segment of data should be directed.

**Key aspects of demultiplexing include:**

**Destination Port Number:** The transport layer at the receiving end examines the header of each incoming data segment to identify the destination port number.

**Data Delivery:** Based on the destination port number, the transport layer directs the data segment to the appropriate application or process.

**Example**

Consider a computer running a web browser and an email client simultaneously:

Multiplexing: When the user sends a request from the web browser (using HTTP over TCP) and simultaneously checks for new emails (using SMTP over TCP), the transport layer assigns different source port numbers to each application. It combines the data streams and sends them over the network.

Demultiplexing: At the server, the transport layer reads the destination port numbers from the incoming data segments to determine whether the data should be delivered to the web server (HTTP) or the email server (SMTP).

**Q2) What are the two Transport layer protocols? Compare them.**

The two main Transport layer protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Both are used to send data over the internet, but they have different characteristics and use cases. Below is a comparison of these two protocols:

**Transmission Control Protocol (TCP)**

1. Characteristics:

Connection-oriented: TCP establishes a connection between the sender and receiver before data transmission begins. This is done through a process called the TCP three-way handshake.

Reliable: TCP ensures that data is delivered accurately and in the correct order. It uses acknowledgments, retransmissions, and sequence numbers to guarantee delivery.

Flow control: TCP uses flow control mechanisms to prevent the sender from overwhelming the receiver with too much data at once.

Congestion control: TCP includes algorithms to avoid network congestion by adjusting the rate of data transmission.

Byte-stream oriented: TCP treats data as a continuous stream of bytes, rather than discrete packets.

2. Use Cases:

Applications requiring reliable data delivery, such as web browsing (HTTP/HTTPS), email (SMTP, IMAP), file transfers (FTP), and remote administration (SSH, Telnet).

**User Datagram Protocol (UDP)**

1. Characteristics:

Connectionless: UDP does not establish a connection before sending data. Each packet (datagram) is sent independently, without a handshake process.

Unreliable: UDP does not guarantee delivery, order, or error-checking of data. If packets are lost or arrive out of order, UDP does not handle retransmission or reordering.

No flow control: UDP does not have built-in mechanisms to control the flow of data between sender and receiver.

No congestion control: UDP does not implement congestion control algorithms, making it less suitable for networks with fluctuating bandwidth.

Message-oriented: UDP treats data as independent messages (datagrams), preserving message boundaries.

2. Use Cases:

Applications that can tolerate some loss of data and require low latency, such as live video or audio streaming, online gaming, Voice over IP (VoIP), and simple query-response protocols like DNS.

**Comparison**

| **Feature** | **TCP** | **UDP** |
| --- | --- | --- |
| **Connection** | Connection-oriented | Connectionless |
| **Reliability** | Reliable (with error-checking, retransmission, and acknowledgment) | Unreliable (no guarantees of delivery, order, or duplication) |
| **Flow Control** | Yes | No |
| **Congestion Control** | Yes | No |
| **Data Transfer Method** | Byte-stream oriented | Message-oriented |
| **Overhead** | Higher (due to connection setup, error-checking, etc.) | Lower (minimal protocol overhead) |
| **Latency** | Higher (due to reliability mechanisms) | Lower (faster data transmission) |
| **Use Cases** | Web browsing, email, file transfer, remote administration | Streaming media, online gaming, VoIP, DNS queries |

**Q3)What is port number? What is its size? How many port numbers are available? Which are well known port numbers?**

What is a Port Number?

A port number is a numerical identifier in networking used to specify a specific process or service on a computer within a network. When data is transmitted over the internet or any other network, it is directed to a specific application or process by using a port number.

**Size of a Port Number**

A port number is a 16-bit integer, which means it can range from 0 to 65535.

How Many Port Numbers are Available?

Since a port number is a 16-bit number, there are

216=65536

216=65536 possible port numbers, ranging from 0 to 65535.

**Well-Known Port Numbers**

Well-known port numbers are those from 0 to 1023. These ports are assigned to specific services and protocols by the Internet Assigned Numbers Authority (IANA). Here are some common well-known port numbers and their associated services:

Port 20: FTP (File Transfer Protocol) - Data Transfer

Port 21: FTP (File Transfer Protocol) - Command Control

Port 22: SSH (Secure Shell)

Port 23: Telnet

Port 25: SMTP (Simple Mail Transfer Protocol)

Port 53: DNS (Domain Name System)

Port 80: HTTP (HyperText Transfer Protocol)

Port 110: POP3 (Post Office Protocol 3)

Port 143: IMAP (Internet Message Access Protocol)

Port 443: HTTPS (HyperText Transfer Protocol Secure)

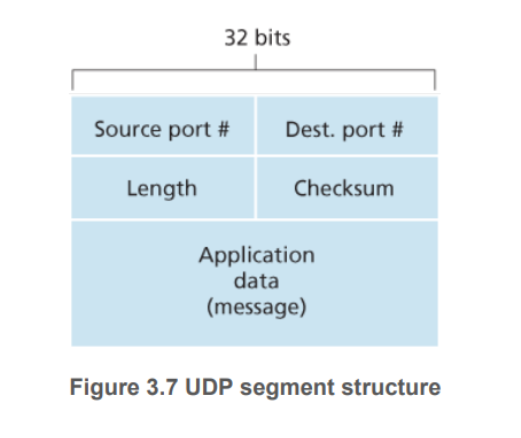
Port Ranges

Well-Known Ports: 0 to 1023

Registered Ports: 1024 to 49151 (These are assigned by IANA for specific services upon request.)

Dynamic/Private Ports: 49152 to 65535 (These can be used dynamically by applications and are also known as ephemeral ports.)

**Q4) With a neat diagram explain UDP segment structure.**

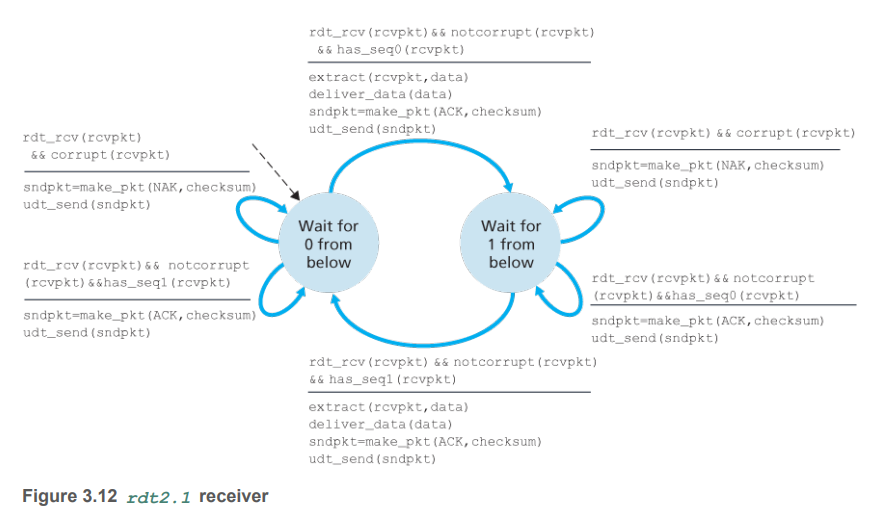
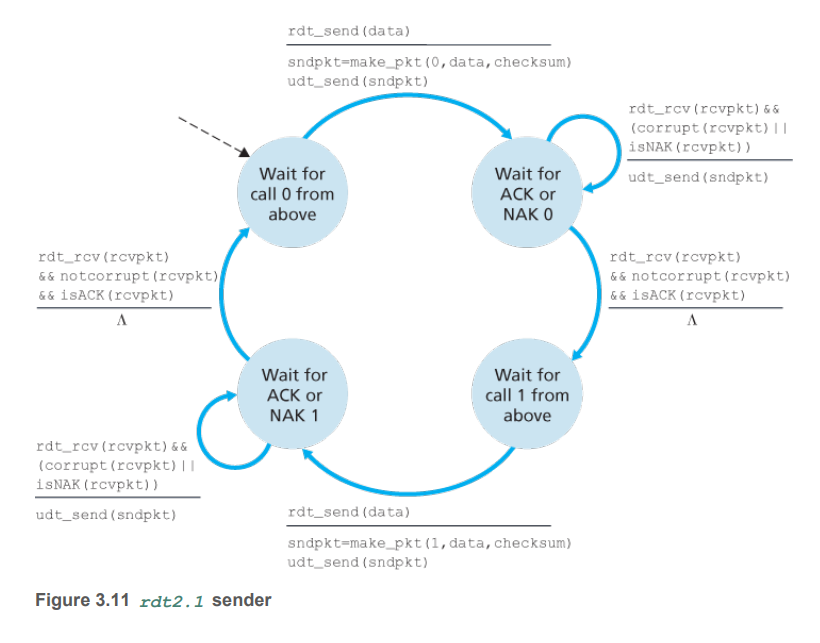
****

* The application data occupies the data field of the UDP segment.

For example,

* For DNS, the data field contains either a query message or a response message.
* For a streaming audio application, audio samples fill the data field.
* The UDP header has only four fields, each consisting of two bytes.
* The port numbers allow the destination host to pass the application data to the correct process running on the destination end system (that is, to perform the demultiplexing function).
* The length field specifies the number of bytes in the UDP segment (header plus data). An explicit length value is needed since the size of the data field may differ from one UDP segment to the next.
* The checksum is used by the receiving host to check whether errors have been introduced into the segment. In truth, the checksum is also calculated over a few of the fields in the IP header in addition to the UDP segment

**Q5)Explain rdt2.1 protocol with a neat finite state machine for sender and receiver.**

* The rdt2.1 sender and receiver FSMs each now have twice as many states as before.
* This is because the protocol state must now reflect whether the packet currently being sent (by the sender) or expected (at the receiver) should have a sequence number of 0 or 1.
* Note that the actions in those states where a 0- numbered packet is being sent or expected are mirror images of those where a 1-numbered packet is being sent or expected; the only differences have to do with the handling of the sequence number.
* Protocol rdt2.1 uses both positive and negative acknowledgments from the receiver to the sender. When an out-of-order packet is received, the receiver sends a positive acknowledgment for the packet it has received. When a corrupted packet is received, the receiver sends a negative acknowledgment.
* We can accomplish the same effect as a NAK if, instead of sending a NAK, we send an ACK for the last correctly received packet.
* A sender that receives two ACKs for the same packet (that is, receives duplicate ACKs) knows that the receiver did not correctly receive the packet following the packet that is being ACKed twice

**Q6)Which are the different types of ARQ protocols? Explain with examples.**

Automatic Repeat reQuest (ARQ) protocols are mechanisms used in data communication to ensure reliable transmission by detecting errors and retransmitting data. Here are the main types of ARQ protocols:

**1. Stop-and-Wait ARQ**

In Stop-and-Wait ARQ, the sender transmits one frame and then waits for an acknowledgment (ACK) from the receiver before sending the next frame. If the sender does not receive an ACK within a certain timeframe, it retransmits the same frame.

Example

* Transmission: The sender sends Frame 1.
* ACK: The receiver sends an acknowledgment (ACK) if Frame 1 is received correctly.
* Timeout: If the sender does not receive an ACK within a specified timeout period, it retransmits Frame 1.
* Advantage: Simple and easy to implement.
* Disadvantage: Inefficient use of bandwidth due to waiting for ACKs.

**2. Go-Back-N ARQ**

In Go-Back-N ARQ, the sender can send multiple frames before needing an acknowledgment, but the receiver can only receive frames in order. If an error is detected in a frame, all subsequent frames are discarded, and the sender must retransmit from the erroneous frame onwards.

Example

* Transmission: The sender sends Frame 1, Frame 2, Frame 3, ..., Frame N.
* ACK: The receiver acknowledges frames received correctly (e.g., ACK for Frame 2, Frame 3, etc.).
* Error: If Frame 4 is erroneous, the receiver discards Frame 4 and all subsequent frames.
* Retransmission: The sender retransmits Frame 4 and all subsequent frames.
* Advantage: Better bandwidth utilization than Stop-and-Wait.
* Disadvantage: Can lead to inefficiencies if errors occur frequently, as many frames might need to be retransmitted.

**3. Selective Repeat ARQ**

Selective Repeat ARQ allows the sender to send multiple frames before needing an acknowledgment, and the receiver can accept and buffer frames even if they are out of order. Only erroneous frames are retransmitted.

Example

* Transmission: The sender sends Frame 1, Frame 2, Frame 3, ..., Frame N.
* ACK: The receiver acknowledges each frame individually (e.g., ACK for Frame 2, NACK for Frame 4).
* Error: If Frame 4 is erroneous, the receiver buffers subsequent frames (Frame 5, Frame 6, etc.) but does not deliver them to the upper layer until Frame 4 is correctly received.
* Retransmission: The sender retransmits only the erroneous Frame 4.
* Advantage: More efficient use of bandwidth, as only erroneous frames are retransmitted.
* Disadvantage: More complex to implement due to the need for buffering and managing out-of-order frames.

**Q7) Explain Stop and wait , Go back N and selective repeat protocols with examples.**

**Stop-and-Wait ARQ**

Concept: The sender sends one frame and waits for an acknowledgment (ACK) from the receiver before sending the next frame. If an ACK is not received within a specified timeout period, the sender retransmits the same frame.

Example:

* Sender sends Frame 1.
* Receiver receives Frame 1 and sends an ACK.
* Sender receives the ACK and sends Frame 2.
* If Sender does not receive an ACK for Frame 2 within the timeout period, it retransmits Frame 2.

Scenario:

* Sender transmits Frame 1.
* Receiver successfully receives Frame 1 and sends an ACK for Frame 1.
* Sender receives the ACK for Frame 1 and transmits Frame 2.
* ACK for Frame 2 is lost.
* Sender does not receive the ACK within the timeout period, so it retransmits Frame 2.
* Receiver receives Frame 2 (again) and sends an ACK for Frame 2.

Advantages: Simple to implement.

Disadvantages: Inefficient for high-latency or high-bandwidth networks because the sender must wait for an ACK before sending the next frame.

**Go-Back-N ARQ**

Concept: The sender can send several frames before needing an acknowledgment, but the receiver can only receive frames in order. If a frame is received with an error, all subsequent frames are discarded. The sender must retransmit the erroneous frame and all subsequent frames.

Example:

* Sender sends Frames 1, 2, 3, 4, and 5.
* Receiver successfully receives Frames 1 and 2, and sends ACKs for them.
* Receiver detects an error in Frame 3 and discards Frames 3, 4, and 5.
* Sender retransmits Frame 3, 4, and 5 upon receiving a NACK or upon timeout.

Scenario:

* Sender transmits Frames 1, 2, 3, 4, and 5.
* Receiver successfully receives Frames 1 and 2 and sends ACKs for them.
* Receiver detects an error in Frame 3 and discards Frames 3, 4, and 5.
* Sender receives ACKs for Frames 1 and 2 but does not receive an ACK for Frame 3 (either receives a NACK or a timeout occurs).
* Sender retransmits Frames 3, 4, and 5.

Advantages: More efficient than Stop-and-Wait since multiple frames are sent before waiting for an acknowledgment.

Disadvantages: Inefficient if errors are frequent because all subsequent frames are retransmitted.

**Selective Repeat ARQ**

Concept: The sender can send several frames before needing an acknowledgment, and the receiver can accept and buffer frames even if they are out of order. Only erroneous frames are retransmitted.

Example:

* Sender sends Frames 1, 2, 3, 4, and 5.
* Receiver successfully receives Frames 1, 2, and 4, and sends ACKs for them.
* Receiver detects an error in Frame 3 and does not acknowledge it.
* Receiver buffers Frame 4.
* Sender retransmits only Frame 3 upon receiving a NACK or upon timeout.
* Receiver receives the retransmitted Frame 3 and sends an ACK.

Scenario:

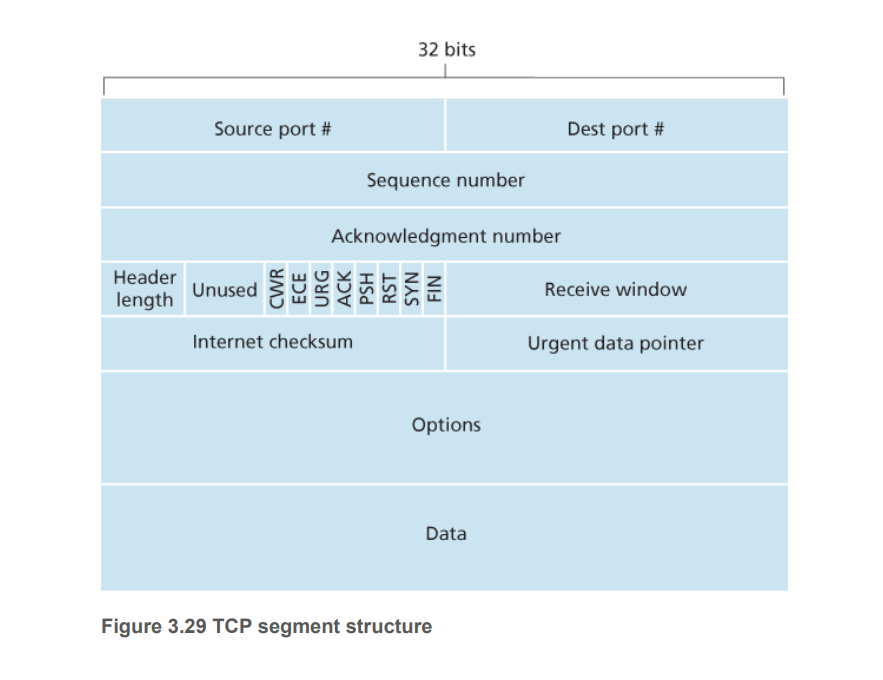
* Sender transmits Frames 1, 2, 3, 4, and 5.
* Receiver successfully receives Frames 1, 2, and 4, and sends ACKs for Frames 1, 2, and 4.
* Receiver detects an error in Frame 3 and does not send an ACK for it.
* Receiver buffers Frame 4.
* Sender receives ACKs for Frames 1, 2, and 4 but not for Frame 3.
* Sender retransmits only Frame 3.
* Receiver receives Frame 3, sends an ACK for it, and delivers the buffered Frame 4 to the application layer.

Advantages: Most efficient use of bandwidth as only erroneous frames are retransmitted.

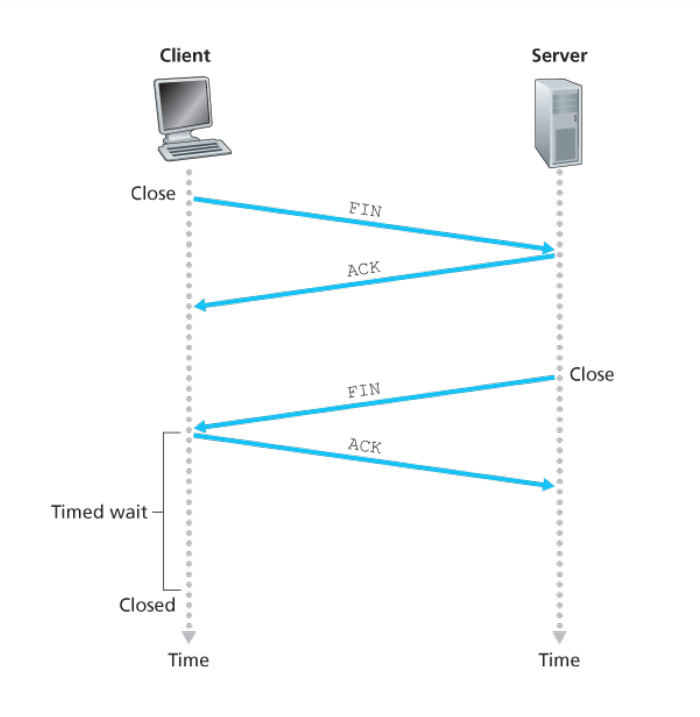
Disadvantages: More complex to implement due to the need for buffering and managing out-of-order frames.

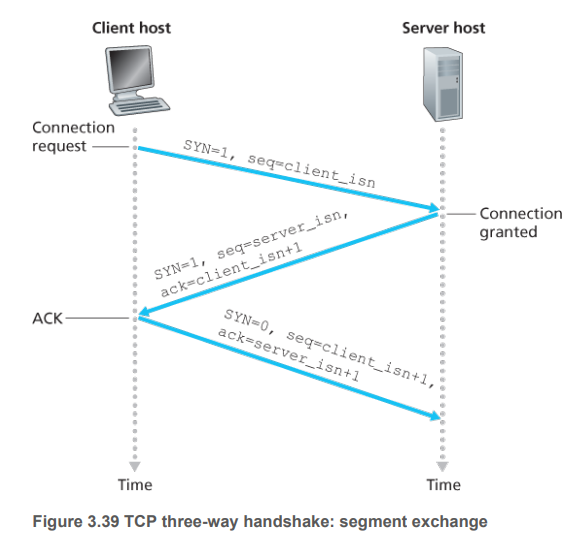
**Q8) With a neat diagram explain TCP three-way handshake and closing a TCP connection.**

* TCP is said to be connection-oriented because before one application process can begin to send data to another, the two processes must first “handshake” with each other—that is, they must send some preliminary segments to each other to establish the parameters of the ensuing data transfer.
* As part of TCP connection establishment, both sides of the connection will initialize many TCP state variables associated with the TCP connection.
* TCP in the client then proceeds to establish a TCP connection with TCP in the server. At the end of this section we discuss in some detail the connection-establishment procedure.
* For now it suffices to know that the client first sends a special TCP segment; the server responds with a second special TCP segment; and finally the client responds again with a third special segment.
* The first two segments carry no payload, that is, no application-layer data; the third of these segments may carry a payload.
* Because three segments are sent between the two hosts, this connection-establishment procedure is often referred to as a three-way handshake.

****

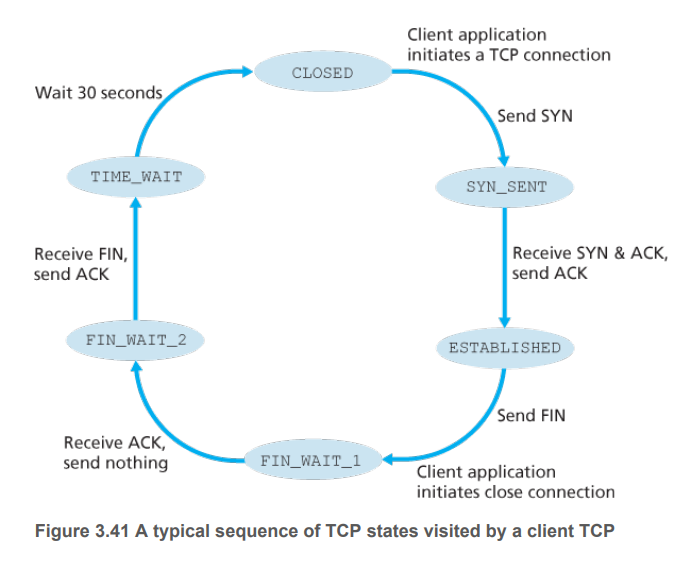
* When a connection ends, the in the hosts are deallocated.
* As an example, suppose the client decides to close the connection. The client application process issues a close command.
* This causes the client TCP to send a special TCP segment to the server process. This special segment has a flag bit in the segment’s header, the FIN bit set to 1.
* When the server receives this segment, it sends the client an acknowledgment segment in return.
* The server then sends its own shutdown segment, which has the FIN bit set to 1. Finally, the client acknowledges the server’s shutdown segment.
* At this point, all the resources in the two hosts are now deallocated. During the life of a TCP connection, the TCP protocol running in each host makes transitions through various TCP states. Figure 3.41 illustrates a typical sequence of TCP states that are visited by the client TCP. The client TCP begins in the CLOSED state

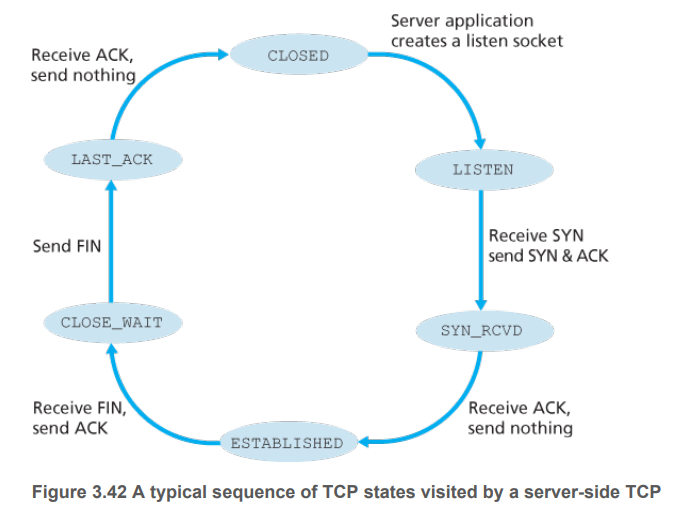
****

****

**Q9)What are the different states visited by the TCP client and server? Explain with a neat diagram.**

* During the life of a TCP connection, the TCP protocol running in each host makes transitions through various TCP states.
* The sequence of TCP states that are visited by the client TCP. The client TCP begins in the CLOSED state. The application on the client side initiates a new TCP connection by creating a Socket object This causes TCP in the client to send a SYN segment to TCP in the server.
* After having sent the SYN segment, the client TCP enters the SYN\_SENT state. While in the SYN\_SENT state, the client TCP waits for a segment from the server TCP that includes an acknowledgment for the client’s previous segment and has the SYN bit set to 1.
* Having received such a segment, the client TCP enters the ESTABLISHED state. While in the ESTABLISHED state, the TCP client can send and receive TCP segments containing payload (that is, application-generated) data.
* Suppose that the client application decides it wants to close the connection. (Note that the server could also choose to close the connection.)
* This causes the client TCP to send a TCP segment with the FIN bit set to 1 and to enter the FIN\_WAIT\_1 state.
* While in the FIN\_WAIT\_1 state, the client TCP waits for a TCP segment from the server with an acknowledgment. When it receives this segment, the client TCP enters the FIN\_WAIT\_2 state.
* While in the FIN\_WAIT\_2 state, the client waits for another segment from the server with the FIN bit set to 1; after receiving this segment, the client TCP acknowledges the server’s segment and enters the TIME\_WAIT state.
* The TIME\_WAIT state lets the TCP client resend the final acknowledgment in case the ACK is lost. The time spent in the TIME\_WAIT state is implementation-dependent, but typical values are 30 seconds, 1 minute, and 2 minutes.
* After the wait, the connection formally closes and all resources on the client side (including port numbers) are released.

****

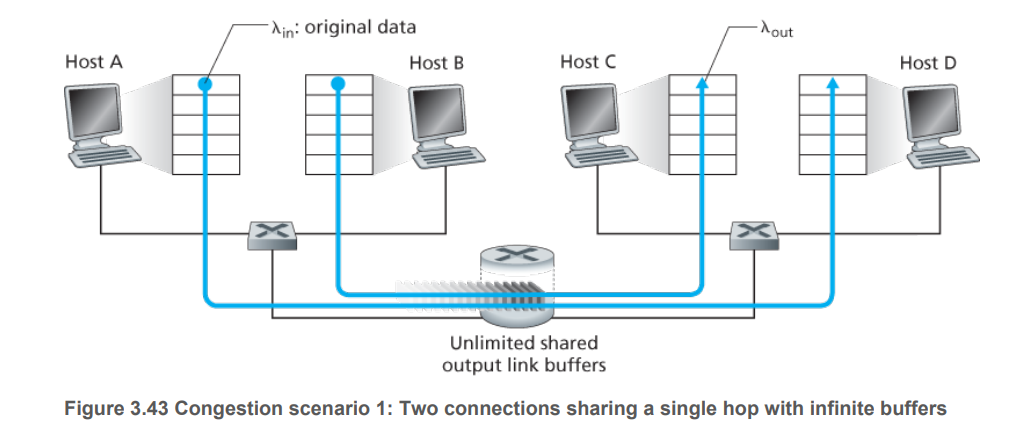
****

**Q10)What is the cause for congestion? Explain congestion scenario Where there are two connections sharing a single hop with infinite buffer.**

congestion occurs in the first place and at the cost of congestion (in terms of resources not fully utilized and poor performance received by the end systems). Rather focus on the simpler issue of understanding what happens as hosts increase their transmission rate and the network becomes congested.

**Two Senders, a Router with Infinite Buffers**

* We begin by considering perhaps the simplest congestion scenario possible: Two hosts (A and B) each have a connection that shares a single hop between source and destination.
* Let’s assume that the application in Host A is sending data into the connection (for example, passing data to the transport-level protocol via a socket) at an average rate of λ bytes/sec.
* These data are original in the sense that each unit of data is sent into the socket only once. The underlying transport level protocol is a simple one. Data is encapsulated and sent; no error recovery (for example, retransmission), flow control, or congestion control is performed.
* Ignoring the additional overhead due to adding transport- and lower-layer header information, the rate at which Host A offers traffic to the router in this first scenario is thus λ bytes/sec.
* Host B operates in a similar manner, and we assume for simplicity that it too is sending at a rate of λ bytes/sec. Packets from Hosts A and B pass through a router and over a shared outgoing link of capacity R.
* The router has buffers that allow it to store incoming packets when the packet-arrival rate exceeds the outgoing link’s capacity. In this first scenario, we assume that the router has an infinite amount of buffer space.
* For a sending rate between 0 and R/2, the throughput at the receiver equals the sender’s sending rate—everything sent by the sender is received at the receiver with a finite delay.
* When the sending rate is above R/2, however, the throughput is only R/2. This upper limit on throughput is a consequence of the sharing of link capacity between two connections.
* The link simply cannot deliver packets to a receiver at a steady-state rate that exceeds R/2. No matter how high Hosts A and B set their sending rates, they will each never see a throughput higher than R/2.
* Achieving a per-connection throughput of R/2 might actually appear to be a good thing, because the link is fully utilized in delivering packets to their destinations.
* However, shows the consequence of operating near link capacity. As the sending rate approaches R/2 (from the left), the average delay becomes larger and larger. When the sending rate exceeds R/2, the average number of queued packets in the router is unbounded, and the average delay between source and destination becomes infinite Thus, while operating at an aggregate throughput of near R may be ideal from a throughput standpoint, it is far from ideal from a delay standpoint.
* Even in this (extremely) idealized scenario, we’ve already found one cost of a congested network—large queuing delays are experienced as the packet-arrival rate nears the link capacity.



**Q11)Discuss about the two congestion control approaches of TCP.**

Here, we identify the two broad approaches to congestion control that are taken in practice and discuss specific network architectures and congestion-control protocols embodying these approaches.

At the highest level, we can distinguish among congestion-control approaches by whether the network layer provides explicit assistance to the transport layer for congestion-control purposes:

**1.End-to-end congestion control:** In an end-to-end approach to congestion control, the network layer provides no explicit support to the transport layer for congestion-control purposes.

* Even the presence of network congestion must be inferred by the end systems based only on observed network behavior (for example, packet loss and delay). TCP takes this end-to-end approach toward congestion control, since the IP layer is not required to provide feedback to hosts regarding network congestion.
* TCP segment loss (as indicated by a timeout or the receipt of three duplicate acknowledgments) is taken as an indication of network congestion, and TCP decreases its window size accordingly.
* We’ll also see a more recent proposal for TCP congestion control that uses increasing round-trip segment delay as an indicator of increased network congestion

**2.Network-assisted congestion control:** With network-assisted congestion control, routers provide explicit feedback to the sender and/or receiver regarding the congestion state of the network.

This feedback may be as simple as a single bit indicating congestion at a link – an approach taken in the early IBM SNA [Schwartz 1982], DEC DECnet [Jain 1989; Ramakrishnan 1990] architectures, and ATM [Black 1995] network architectures.

More sophisticated feedback is also possible. For example, in ATM Available Bite Rate (ABR) congestion control, a router informs the sender of the maximum host sending rate it (the router) can support on an outgoing link.

As noted above, the Internet-default versions of IP and TCP adopt an end-to-end approach towards congestion control.

**Q12)Explain the services provided by the Network layer.**

The network service model defines the characteristics of end-to-end delivery of packets between sending and receiving hosts. Let’s now consider some possible services that the network layer could provide.

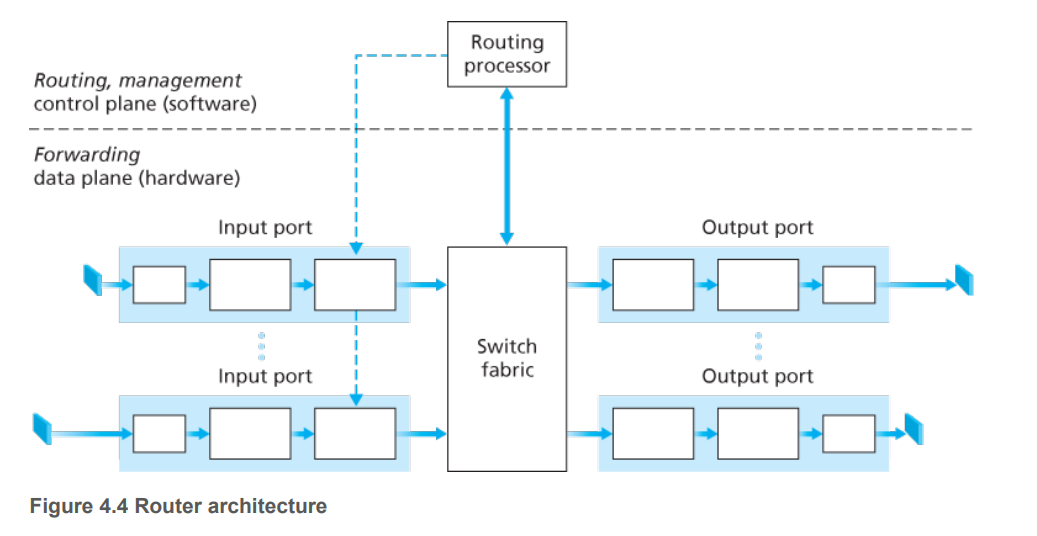
These services could include:

* **Guaranteed delivery:** This service guarantees that a packet sent by a source host will eventually arrive at the destination host.
* **Guaranteed delivery with bounded delay:** This service not only guarantees delivery of the packet, but delivery within a specified host-to-host delay bound (for example, within 100 msec).
* **In-order packet delivery:** This service guarantees that packets arrive at the destination in the order that they were sent
* **Guaranteed minimal bandwidth:** This network-layer service emulates the behavior of a transmission link of a specified bit rate (for example, 1 Mbps) between sending and receiving hosts. As long as the sending host transmits bits (as part of packets) at a rate below the specified bit rate, then all packets are eventually delivered to the destination host.
* **Security:** The network layer could encrypt all datagrams at the source and decrypt them at the destination, thereby providing confidentiality to all transport-layer segments.

**Q13) Differentiate between forwarding and routing .**

| **Aspect** | **Forwarding** | **Routing** |
| --- | --- | --- |
| **Definition** | Moving packets from an incoming interface to an outgoing interface within a single network device | Determining the path that data packets should take through the network from the source to the destination |
| **Scope** | Local (within a single device) | Global (across the network) |
| **Functionality** | Uses a pre-built forwarding table to determine the appropriate outgoing interface for each packet | Uses routing algorithms and protocols (such as OSPF, BGP, and RIP) to build and maintain routing tables |
| **Speed** | Fast (often hardware-based for efficiency) | Slower (involves complex calculations) |
| **Decision Basis** | Destination IP address matched against entries in the forwarding table | Routing protocols and algorithms considering factors like hop count, link cost, and bandwidth |
| **Example** | A router receiving a packet and determining the best outgoing interface to send it to reach its next hop | A router using OSPF to calculate the most efficient route to a remote network and updating its routing table |

**Q14)With the neat diagram explain Router architecture**

****

**Input ports:** An input port performs several key functions. It performs the physical layer function of terminating an incoming physical link at a router; this is shown in the leftmost box of an input port and the rightmost box of an output port .

* An input port also performs link-layer functions needed to interoperate with the link layer at the other side of the incoming link; this is represented by the middle boxes in the input and output ports.
* Perhaps most crucially, a lookup function is also performed at the input port; this will occur in the rightmost box of the input port.
* It is here that the forwarding table is consulted to determine the router output port to which an arriving packet will be forwarded via the switching fabric. Control packets (for example, packets carrying routing protocol information) are forwarded from an input port to the routing processor.
* Note that the term “port” here —referring to the physical input and output router interfaces—is distinctly different from the software ports associated with network applications and sockets.

**Switching fabric:** The switching fabric connects the router’s input ports to its output ports. This switching fabric is completely contained within the router—a network inside of a network router!

**Output ports:** An output port stores packets received from the switching fabric and transmits these packets on the outgoing link by performing the necessary link-layer and physical-layer functions.

* When a link is bidirectional (that is, carries traffic in both directions), an output port will typically be paired with the input port for that link on the same line card.

**Routing processor**: The routing processor performs control-plane functions. In traditional routers, it executes the routing protocols maintains routing tables and attached link state information, and computes the forwarding table for the router.

* In SDN routers, the routing processor is responsible for communicating with the remote controller in order to (among other activities) receive forwarding table entries computed by the remote controller, and install these entries in the router’s input ports.

**Q15)What is Switching ? Explain 3 Switching Techniques.**

**1.Switching via memory:** The simplest, earliest routers were traditional computers, with switching between input and output ports being done under direct control of the CPU (routing processor). Input and output ports functioned as traditional I/O devices in a traditional operating system.

* An input port with an arriving packet first signaled the routing processor via an interrupt. The packet was then copied from the input port into processor memory.
* The routing processor then extracted the destination address from the header, looked up the appropriate output port in the forwarding table, and copied the packet to the output port’s buffers.
* In this scenario, if the memory bandwidth is such that a maximum of B packets per second can be written into, or read from, memory, then the overall forwarding throughput (the total rate at which packets are transferred from input ports to output ports) must be less than B/2. Note also that two packets cannot be forwarded at the same time, even if they have different destination ports, since only one memory read/write can be done at a time over the shared system bus.

**2.Switching via a bus:** In this approach, an input port transfers a packet directly to the output port over a shared bus, without intervention by the routing processor.

* This is typically done by having the input port pre-pend a switch-internal label (header) to the packet indicating the local output port to which this packet is being transferred and transmitting the packet onto the bus.
* All output ports receive the packet, but only the port that matches the label will keep the packet. The label is then removed at the output port, as this label is only used within the switch to cross the bus.
* If multiple packets arrive to the router at the same time, each at a different input port, all but one must wait since only one packet can cross the bus at a time. Because every packet must cross the single bus, the switching speed of the router is limited to the bus speed; in our roundabout analogy, this is as if the roundabout could only contain one car at a time.
* The Cisco 6500 router [Cisco 6500 2016] internally switches packets over a 32-Gbps-backplane bus.

**3.Switching via an interconnection network:** One way to overcome the bandwidth limitation of a single, shared bus is to use a more sophisticated interconnection network, such as those that have been used in the past to interconnect processors in a multiprocessor computer architecture.

* A crossbar switch is an interconnection network consisting of 2N buses that connect N input ports to N output ports.
* Each vertical bus intersects each horizontal bus at a crosspoint, which can be opened or closed at any time by the switch fabric controller (whose logic is part of the switching fabric itself).
* When a packet arrives from port A and needs to be forwarded to port Y, the switch controller closes the crosspoint at the intersection of busses A and Y, and port A then sends the packet onto its bus, which is picked up (only) by bus Y.
* Note that a packet from port B can be forwarded to port X at the same time, since the A-to-Y and B-to-X packets use different input and output busses. Thus, unlike the previous two switching approaches, crossbar switches are capable of forwarding multiple packets in parallel.

**Q16)Explain the different packet scheduling algorithms.**

**First-in-First-Out (FIFO)**

* The queuing model abstraction for the FIFO link-scheduling discipline. Packets arriving at the link output queue wait for transmission if the link is currently busy transmitting another packet.
* If there is not sufficient buffering space to hold the arriving packet, the queue’s packet discarding policy then determines whether the packet will be dropped (lost) or whether other packets will be removed from the queue to make space for the arriving packet.
* When a packet is completely transmitted over the outgoing link (that is, receives service) it is removed from the queue.
* The FIFO (also known as first-come-first-served, or FCFS) scheduling discipline selects packets for link transmission in the same order in which they arrived at the output link queue.

**Priority Queuing**

Under priority queuing, packets arriving at the output link are classified into priority classes upon arrival at the queue.

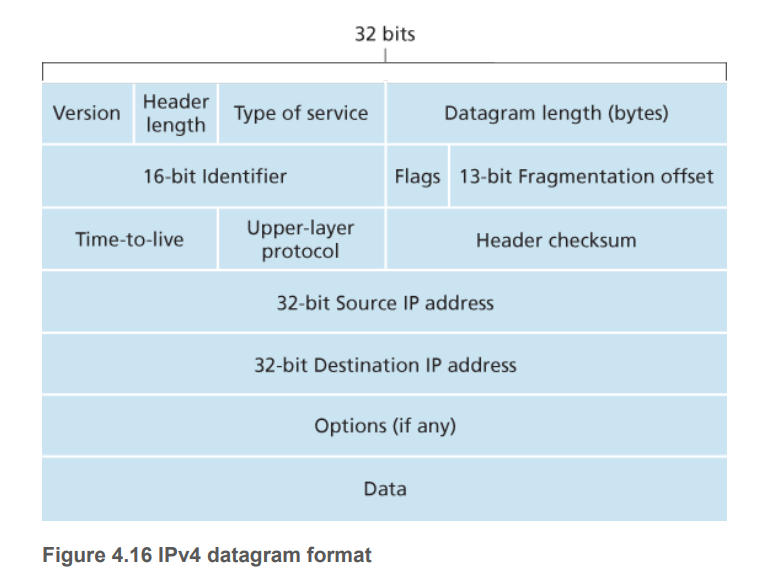
* In practice, a network operator may configure a queue so that packets carrying network management information (e.g., as indicated by the source or destination TCP/UDP port number) receive priority over user traffic; additionally, real-time voice-over-IP packets might receive priority over non-real traffic such as SMTP or IMAP e-mail packets.
* Each priority class typically has its own queue. When choosing a packet to transmit, the priority queuing discipline will transmit a packet from the highest priority class that has a nonempty queue (that is, has packets waiting for transmission).
* The choice among packets in the same priority class is typically done in a FIFO manner.

**Round Robin and Weighted Fair Queuing (WFQ)**

Under the round robin queuing discipline, packets are sorted into classes as with priority queuing. However, rather than there being a strict service priority among classes, a round robin scheduler alternates service among the classes.

* In the simplest form of round robin scheduling, a class 1 packet is transmitted, followed by a class 2 packet, followed by a class 1 packet, followed by a class 2 packet, and so on.
* A so-called work-conserving queuing discipline will never allow the link to remain idle whenever there are packets (of any class) queued for transmission.
* A work-conserving round robin discipline that looks for a packet of a given class but finds none will immediately check the next class in the round robin sequence.

**Q17)With the neat diagram explain IPv4 datagram format.**

****

The key fields in the IPv4 datagram are the following:

**1.Version number:** These 4 bits specify the IP protocol version of the datagram. By looking at the version number, the router can determine how to interpret the remainder of the IP datagram. Different versions of IP use different datagram formats. The datagram format for IPv4.

**2.Header length**: Because an IPv4 datagram can contain a variable number of options (which are included in the IPv4 datagram header), these 4 bits are needed to determine where in the IP datagram the payload (e.g., the transport-layer segment being encapsulated in this datagram) actually begins. Most IP datagrams do not contain options, so the typical IP datagram has a 20-byte header.

**3.Type of service:** The type of service (TOS) bits were included in the IPv4 header to allow different types of IP datagrams to be distinguished from each other. For example, it might be useful to distinguish real-time datagrams (such as those used by an IP telephony application) from non-realtime traffic (for example, FTP). The specific level of service to be provided is a policy issue determined and configured by the network administrator for that router.

**4.Datagram length**: This is the total length of the IP datagram (header plus data), measured in bytes. Since this field is 16 bits long, the theoretical maximum size of the IP datagram is 65,535 bytes. However, datagrams are rarely larger than 1,500 bytes, which allows an IP datagram to fit in the payload field of a maximally sized Ethernet frame.

**5.Identifier, flags, fragmentation offset**: These three fields have to do with so-called IP fragmentation, a topic we will consider shortly. Interestingly, the new version of IP, IPv6, does not allow for fragmentation.

**6.Time-to-live:** The time-to-live (TTL) field is included to ensure that datagrams do not circulate forever (due to, for example, a long-lived routing loop) in the network. This field is decremented by one each time the datagram is processed by a router. If the TTL field reaches 0, a router must drop that datagram.

**7.Protocol:** This field is typically used only when an IP datagram reaches its final destination. The value of this field indicates the specific transport-layer protocol to which the data portion of this IP datagram should be passed.

For example, a value of 6 indicates that the data portion is passed to TCP, while a value of 17 indicates that the data is passed to UDP. For a list of all possible values, see. Note that the protocol number in the IP datagram has a role that is analogous to the role of the port number field in the transport-layer segment.

**8.Header checksum:** The header checksum aids a router in detecting bit errors in a received IP datagram. The header checksum is computed by treating each 2 bytes in the header as a number and summing these numbers using 1s complement arithmetic.

**9.Source and destination IP addresses:** When a source creates a datagram, it inserts its IP address into the source IP address field and inserts the address of the ultimate destination into the destination IP address field. Often the source host determines the destination address via a DNS lookup.

**10.Options:** The options fields allow an IP header to be extended. Header options were meant to be used rarely—hence the decision to save overhead by not including the information in options fields in every datagram header.

**11.Data (payload):** Finally, we come to the last and most important field—the raison d’etre for the datagram in the first place! In most circumstances, the data field of the IP datagram contains the transport-layer segment (TCP or UDP) to be delivered to the destination.

**Q18)What is IP address?What are the 2 versions of IP addressing, what is the size of IP address in each version ?**

IP addressing is the method used to assign unique identifiers to devices on a network, enabling them to communicate with each other over the Internet or other networks. There are two primary types of IP addresses: IPv4 and IPv6.

**IPv4 (Internet Protocol version 4)**

* Address Format: IPv4 addresses are 32-bit numbers, usually represented in decimal format as four octets separated by periods (e.g., 192.168.0.1).
* Address Space: IPv4 provides approximately 4.3 billion unique addresses.
* Header Size: The IPv4 header is 20 bytes long without options.
* Fragmentation: Routers along the path can fragment packets to accommodate smaller Maximum Transmission Units (MTUs).
* Security: Security features are not inherent but can be added using protocols like IPsec.
* Example: An IPv4 address looks like 192.0.2.1.

**IPv6 (Internet Protocol version 6)**

* Address Format: IPv6 addresses are 128-bit numbers, typically represented in hexadecimal format, separated by colons (e.g.,2001:0db8:85a3:0000:0000:8a2e:0370:7334). Leading zeros in each block can be omitted and consecutive blocks of zeros can be abbreviated with "::" once per address (e.g., 2001:db8::8a2e:370:7334).
* Address Space: IPv6 provides a vastly larger address space, with 3.4 x 10^38 unique addresses.
* Header Size: The IPv6 header is 40 bytes long and designed to be simpler and more efficient than the IPv4 header.
* Fragmentation: Fragmentation is handled only at the source, not by routers along the path.
* Security: IPv6 was designed with IPsec support as a fundamental component.
* Example: An IPv6 address looks like 2001:0db8:85a3::8a2e:0370:7334.

**Q19)Explain how a host obtains IP address using DHCP**

1. The operating system creates a DHCP request message and puts this message within a UDP segment with destination port 67 (DHCP server) and source port 68 (DHCP client). The UDP segment is then placed within an IP datagram with a broadcast IP destination address (255.255.255.255) and a source IP address of 0.0.0.0 .

2. The IP datagram containing the DHCP request message is then placed within an Ethernet frame . The Ethernet frame has a destination MAC addresses of FF:FF:FF:FF:FF:FF so that the frame will be broadcast to all devices connected to the switch (hopefully including a DHCP server).

3. The broadcast Ethernet frame containing the DHCP request is the first frame sent by operating system to the Ethernet switch. The switch broadcasts the incoming frame on all outgoing ports, including the port connected to the router.

4. The router receives the broadcast Ethernet frame containing the DHCP request on its interface with MAC address 00:22:6B:45:1F:1B and the IP datagram is extracted from the Ethernet frame. The datagram’s broadcast IP destination address indicates that this IP datagram should be processed by upper layer protocols at this node, so the datagram’s payload (a UDP segment) is thus demultiplexed up to UDP, and the DHCP request message is extracted from the UDP segment. The DHCP server now has the DHCP request message.

5. Let’s suppose that the DHCP server running within the router can allocate IP addresses in the CIDR block 68.85.2.0/24. In this example, all IP addresses used within the school are thus within Comcast’s address block. Let’s suppose the DHCP server allocates address 68.85.2.101 to OS. The DHCP server creates a DHCP ACK message containing this IP address, as well as the IP address of the DNS server (68.87.71.226), the IP address for the default gateway router (68.85.2.1), and the subnet block (68.85.2.0/24) (equivalently, the “network mask”). The DHCP message is put inside a UDP segment, which is put inside an IP datagram, which is put inside an Ethernet frame. The Ethernet frame has a source MAC address of the router’s interface to the home network (00:22:6B:45:1F:1B) and a destination MAC address of OS (00:16:D3:23:68:8A).

6. The Ethernet frame containing the DHCP ACK is sent (unicast) by the router to the switch. Because the switch is self-learning and previously received an Ethernet frame (containing the DHCP request) from OS, the switch knows to forward a frame addressed to 00:16:D3:23:68:8A only to the output port leading to OS.

7. OS receives the Ethernet frame containing the DHCP ACK, extracts the IP datagram from the Ethernet frame, extracts the UDP segment from the IP datagram, and extracts the DHCP ACK message from the UDP segment. OS’s DHCP client then records its IP address and the IP address of its DNS server. It also installs the address of the default gateway into its IP forwarding table. OS will send all datagrams with destination address outside of its subnet 68.85.2.0/24 to the default gateway. At this point, OS has initialized its networking components and is ready to begin processing the Web page fetch.

**Q20) What is the advantage of NAT ? Explain with example.**

One of the primary advantages of Network Address Translation (NAT) is its ability to conserve public IP addresses. NAT allows multiple devices on a local network to share a single public IP address (or a small pool of public IP addresses) when accessing the internet. This is particularly important given the limited availability of IPv4 addresses.

How NAT Conserves IP Addresses

1.Using Private IP Addresses Internally:

Example: In a typical home network, devices like computers, smartphones, tablets, and smart home devices are assigned private IP addresses (e.g., 192.168.1.x, 10.x.x.x).

Private IP Ranges:

* 10.0.0.0 to 10.255.255.255
* 172.16.0.0 to 172.31.255.255
* 192.168.0.0 to 192.168.255.255
* These private IP addresses are used within the local network and are not routable on the public internet.

2.Single Public IP Address for Multiple Devices:

An ISP assigns a single public IP address (e.g., 203.0.113.1) to a household.

* Without NAT: Each device would need a unique public IP address to communicate with the internet. This would quickly exhaust the limited pool of IPv4 addresses available.
* With NAT: The router uses the single public IP address for all devices in the local network. It translates the private IP addresses to the public IP address for outbound traffic and manages the return traffic accordingly.

**Q21) Compare IPv4 and IPv6**

Comparing the IPv6 datagram format with the IPv4 datagram format that we saw we notice that several fields appearing in the IPv4 datagram are no longer present in the IPv6 datagram:

**Fragmentation/reassembly:** IPv6 does not allow for fragmentation and reassembly at intermediate routers; these operations can be performed only by the source and destination.

* If an IPv6 datagram received by a router is too large to be forwarded over the outgoing link, the router simply drops the datagram and sends a “Packet Too Big” ICMP error message (see Section 5.6) back to the sender.
* The sender can then resend the data, using a smaller IP datagram size. Fragmentation and reassembly is a time-consuming operation; removing this functionality from the routers and placing it squarely in the end systems considerably speeds up IP forwarding within the network.

**Header checksum:** Because the transport-layer (for example, TCP and UDP) and link-layer (for example, Ethernet) protocols in the Internet layers perform checksumming, the designers of IP probably felt that this functionality was sufficiently redundant in the network layer that it could be removed.

* Once again, fast processing of IP packets was a central concern. Recall from our discussion of IPv4 that since the IPv4 header contains a TTL field (similar to the hop limit field in IPv6), the IPv4 header checksum needed to be recomputed at every router.
* As with fragmentation and reassembly, this too was a costly operation in IPv4.

**Options:** An options field is no longer a part of the standard IP header. However, it has not gone away.

* Instead, the options field is one of the possible next headers pointed to from within the IPv6 header.
* That is, just as TCP or UDP protocol headers can be the next header within an IP packet, so too can an options field. The removal of the options field results in a fixed-length, 40-byte IP header.

**Q22) What are the different ways to overcome the scarcity of IP addresses, explain in brief.**

The scarcity of IPv4 addresses has prompted the development of several strategies to efficiently manage and extend the existing pool of IP addresses. Here are the primary methods used to address IP address scarcity:

**1. Network Address Translation (NAT)**

NAT allows multiple devices on a local network to share a single public IP address. It translates private IP addresses used within a local network to a public IP address for communication over the internet.

**Advantages:**

* Conserves public IP addresses by enabling multiple devices to share one.
* Adds a layer of security by hiding internal network structures from external networks.

**Example:** A home network with devices using private IP addresses (e.g., 192.168.1.x) shares a single public IP address provided by the ISP through a NAT-enabled router.

**2. Private IP Addressing**

Private IP addressing involves using specific IP address ranges designated for private networks. These addresses are not routable on the public internet and are used within local networks.

Private IP Ranges:

10.0.0.0 to 10.255.255.255

172.16.0.0 to 172.31.255.255

192.168.0.0 to 192.168.255.255

**Advantages:**

* Frees up public IP addresses for use on the internet.
* Provides flexibility in internal network design.

**Example:** A corporate network uses private IP addresses (e.g., 10.0.0.x) for all internal devices, with a NAT gateway handling external communication.

**3. Dynamic Host Configuration Protocol (DHCP)**

DHCP dynamically assigns IP addresses to devices on a network. Addresses are leased for a specific period, after which they can be reassigned to other devices.

**Advantages:**

* Efficient use of a limited pool of IP addresses by reusing them as devices connect and disconnect from the network.
* Simplifies IP address management by automating the assignment process.

**Example:** A Wi-Fi network in a coffee shop uses DHCP to assign IP addresses to customers' devices as they connect.

**4. IPv6 Adoption**

IPv6 is the successor to IPv4, designed to address the limitations of IPv4, including address scarcity. IPv6 uses 128-bit addresses, providing an almost inexhaustible supply of unique IP addresses.

**Advantages:**

* Vast address space (approximately 3.4 x 10^38 addresses).
* Improved routing and network autoconfiguration capabilities.
* Enhanced security features like IPsec are built into the protocol.

**Example:** An ISP begins to roll out IPv6 addresses to its customers, allowing for a significant increase in the number of devices that can be connected to the internet.

**5. Classless Inter-Domain Routing (CIDR)**

CIDR allows for more efficient allocation of IP addresses by using variable-length subnet masking (VLSM), which enables finer granularity in address assignment compared to the fixed classes (A, B, C) used in the traditional classful addressing scheme.

**Advantages:**

* Reduces wasted IP address space by allowing subnets to be created in sizes that fit the actual number of hosts needed.
* Helps in aggregating routes to reduce the size of routing tables.

**Example:** Instead of allocating an entire Class C network (256 addresses) to a small network that needs only 50 addresses, a CIDR block like 192.168.1.0/26 (64 addresses) can be used.

**Q23)Problems on Forwarding the packets to the proper interfaces(problems given in the class)**

**Q24) What are the different categories of routing algorithms ?**

1. The Link-State (LS) Routing Algorithm
2. The Distance-Vector (DV) Routing Algorithm

**Q25) Explain link state and distance vector routing algorithms with examples.**

**Link-State Routing Algorithm**

Link-state routing algorithms work by having each router know the complete layout of the network. This means every router knows about every other router and the cost to reach them.

**Steps:**

* **Discovery:** Each router discovers its neighbors and the cost to reach them.
* **Advertisement:** Each router creates a packet containing this information and sends it to all other routers.
* **Database:** Every router builds a database of the network using these packets.
* **Calculation:** Each router uses this database to calculate the shortest path to every other router using Dijkstra's algorithm.
* **Routing Table:** Based on these calculations, each router creates its routing table.

**Example:**

Imagine a network with routers A, B, C, and D:

* A connects to B (cost 2)
* A connects to C (cost 5)
* B connects to C (cost 1)
* B connects to D (cost 3)
* C connects to D (cost 2)

**Discovery and Advertisement:** Each router learns about its neighbors and sends this information to all other routers.

**Database:** Every router now knows the entire network layout.

**Calculation:** Each router calculates the shortest path to all other routers.

From A: A → B (2), A → B → C (3), A → B → D (5)

Routing Table: Router A’s table will show the best routes:

* To B: Directly (cost 2)
* To C: Through B (total cost 3)
* To D: Through B (total cost 5)

**Distance-Vector Routing Algorithm**

Distance-vector routing algorithms have each router know only about its immediate neighbors. They periodically share their routing table with their neighbors and update their own table based on the received information.

**Steps:**

**Initialization:** Each router knows the cost to reach its directly connected neighbors.

**Sharing:** Routers periodically share their routing tables with their neighbors.

**Update:** Each router updates its table if it finds a cheaper way to reach a destination through a neighbor.

**Convergence:** This process continues until all routers have the best routes.

**Example:**

**Using the same network:**

Initially, routers know only their direct connections:

* A: B (2), C (5)
* B: A (2), C (1), D (3)
* C: A (5), B (1), D (2)
* D: B (3), C (2)

**Initialization:** Each router sets the cost to reach neighbors.

**Sharing:** Routers share their tables with neighbors.

**Update:** Router A learns from B that it can reach C more cheaply via B:

A updates its table: C (cost 3 via B).

**Convergence:** After several exchanges, all routers have the optimal paths.

Router A’s final table might look like this:

* To B: Directly (cost 2)
* To C: Via B (cost 3)
* To D: Via B (cost 5)

**Q26) Solve problems on Link state and distance vector algorithms which are given in the class.**

**Q27) In CIDR approach how the network address and host addresses are identified from a given IP address.**

**Q28) For the following masks, identify the number of host IP addresses.**

**/8, /10 ,/20 ,/26 , /28**