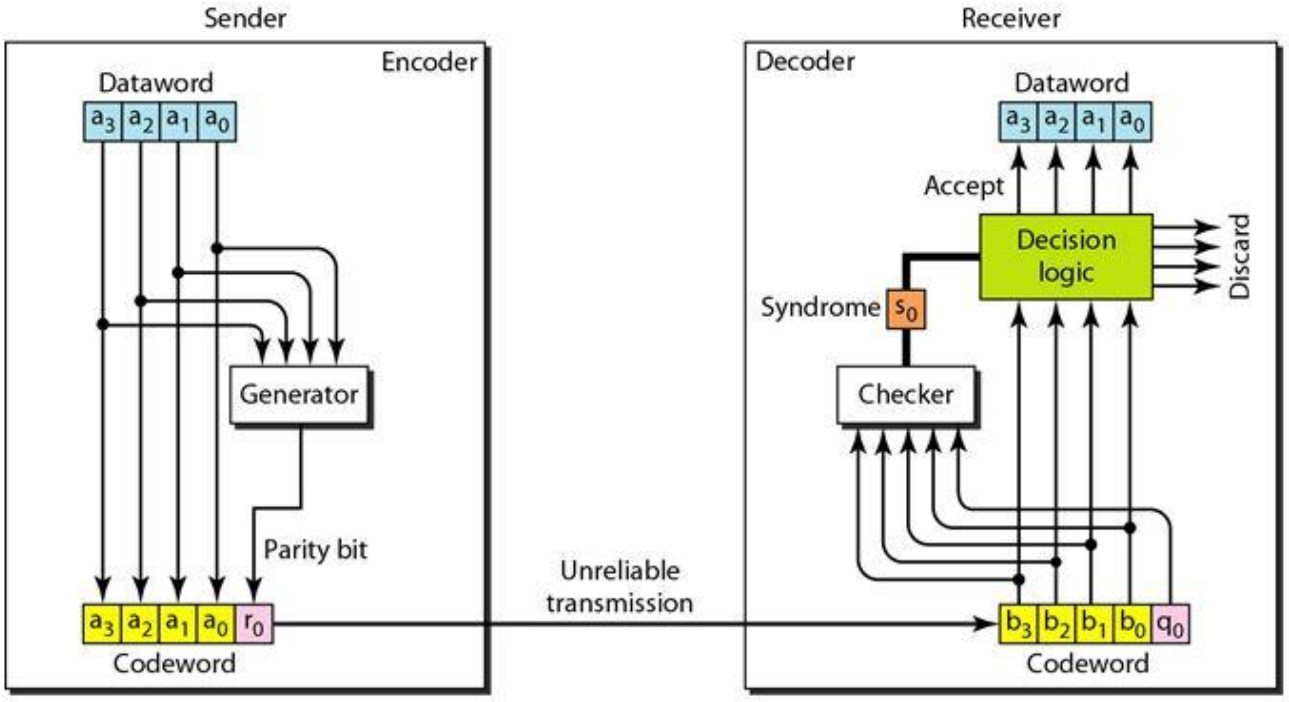
FCN QB PT2

1) Explain simple parity check code process with neat diagram and example.

Simple Parity-Check Code

Perhaps the most familiar error-detecting code is the simple parity-check code. In this code, a *k-bit* dataword is changed to an n-bit codeword where $n = k + 1$. The extra bit, called the parity bit, is selected to make the total number of Is in the codeword even. Although some implementations specify an odd number of Is, we discuss the even case. The minimum Hamming distance for this category is $d_{min} = 2$, which means that the code is a single-bit error-detecting code; it cannot correct any error.



- In a simple parity-check code, a 4-bit data word (a3, a2, a1, a0) is sent along with a parity bit r0 to make the total number of 1s even. The parity bit is calculated as:

r0 = a3 + a2 + a1 + a0 (mod 2)

- The 5-bit code word (a3, a2, a1, a0, r0) is transmitted. At the receiver, all 5 bits (b3, b2, b1, b0, q0) are added using modulo-2 to calculate the syndrome:

s0 = b3 + b2 + b1 + b0 + q0 (mod 2)

- If s0 = 0, no error is detected and the data word is accepted. If s0 = 1, an error is detected and the data is discarded.

For example, if the sender sends 1011, r0 = 1 → code word = 10111.

- *If received correctly → syndrome = 0 → data accepted.*
- *If one bit is changed (e.g., a1 → 0) → 10011 → syndrome = 1 → data discarded.*
- *If r0 is changed → 10110 → syndrome = 1 → data discarded.*
- *If two bits are changed (e.g., a3 and r0) → 00110 → syndrome = 0 → incorrect data accepted.*
- *If three bits are changed → syndrome = 1 → error detected.*

In conclusion, this method detects single-bit and all odd-numbered errors but fails to detect even-numbered errors and cannot locate the error position.

2) Explain different types of errors. Give example.

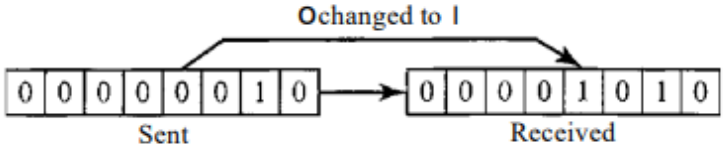
Types of Errors

Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference. This interference can change the shape of the signal. In a single-bit error, a 0 is changed to a 1 or a 1 to a 0. In a burst error, multiple bits are changed. For example, a 11100 s burst of impulse noise on a transmission with a data rate of 1200 bps might change all or some of the 12 bits of information.

Single-Bit Error

The term *single-bit error* means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

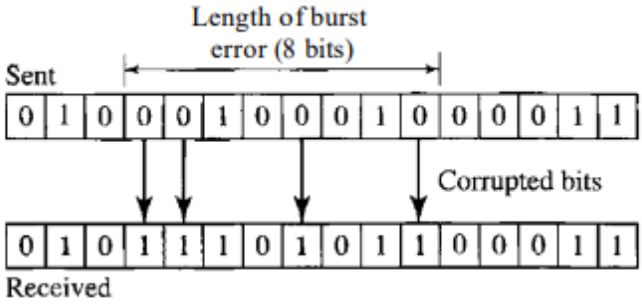
In a single-bit error, only 1 bit in the data unit has changed.



Single-bit errors are the least likely type of error in serial data transmission. To understand why, imagine data sent at 1 Mbps. This means that each bit lasts only 1/1,000,000 s, or 1 µs. For a single-bit error to occur, the noise must have a duration of only 1 µs, which is very rare; noise normally lasts much longer than this.

Burst Error

The term *burst error* means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

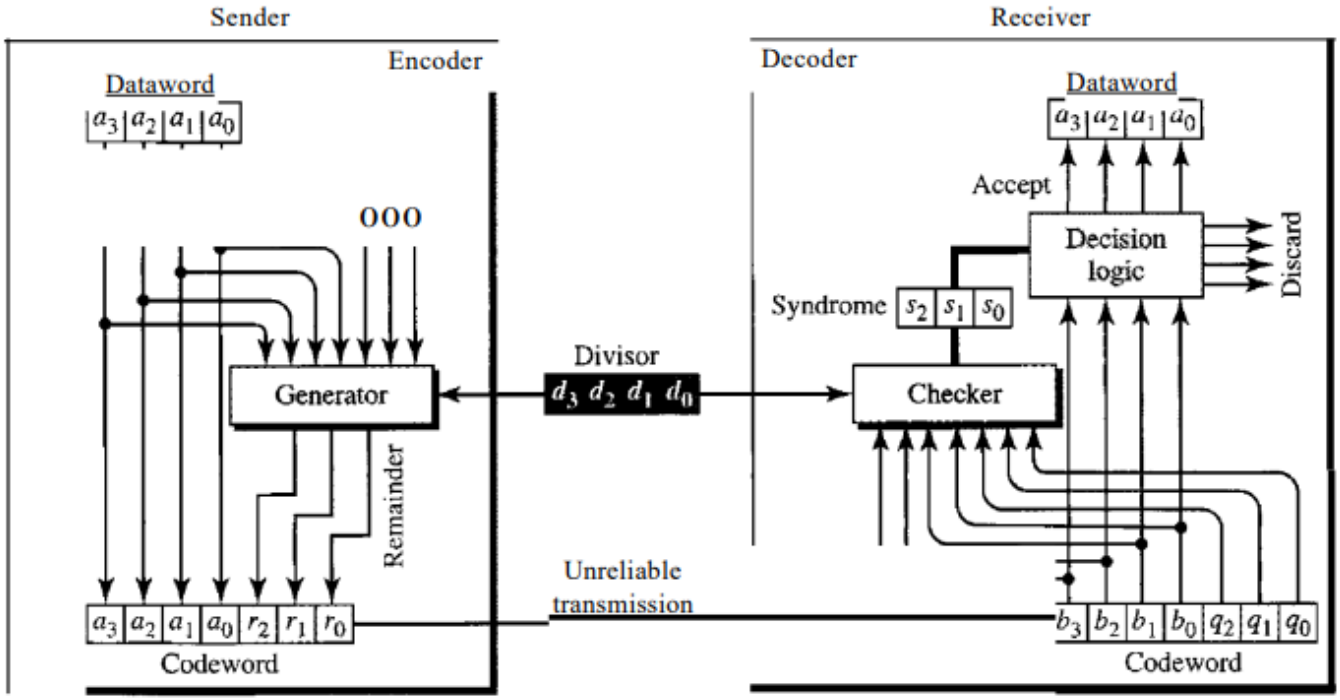


A burst error is more likely to occur than a single-bit error. The duration of noise is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at I kbps, a noise of 11100 s can affect 10 bits; if we are sending data at I Mbps, the same noise can affect 10,000 bits.

3) Describe Cyclic Redundancy check process with neat diagram and example.

Cyclic Redundancy Check

We can create cyclic codes to correct errors. However, the theoretical background required is beyond the scope of this book. In this section, we simply discuss a category of cyclic codes called the cyclic redundancy check (CRC) that is used in networks such as LANs and WANs.



Encoder

Let us take a closer look at the encoder. The encoder takes the dataword and augments it with n - k number of as. It then divides the augmented dataword by the divisor, as shown in Figure 10.15.

Decoder

The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all Os, there is no error; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded. Figure 10.16 shows two cases: The left-hand figure shows the value of syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is one single error. The syndrome is not all Os (it is OIl).

4) Justify “Error correction is more complex than error detection.”

Detection Versus Correction

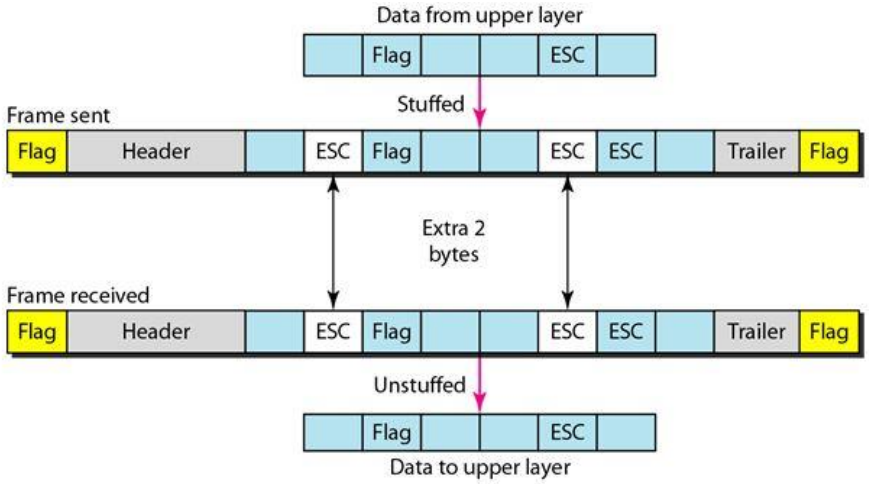
The correction of errors is more difficult than the detection. In error detection, we are looking only to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of errors. A single-bit error is the same for us as a burst error.

In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message. The number of the errors and the size of the message are important factors. If we need to correct one single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 possibilities. You can imagine the receiver's difficulty in finding 10 errors in a data unit of 1000 bits.

5) Describe character oriented and bit oriented protocols.

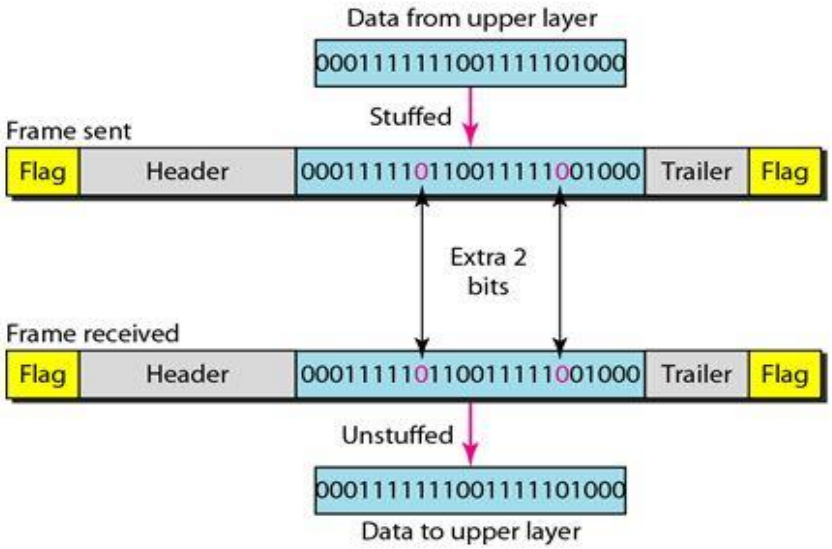
Character-Oriented Protocols

Character-oriented framing was popular when only text was exchanged by the data link layers. The flag could be selected to be any character not used for text communication. Now, however, we send other types of information such as graphs, audio, and video. Any pattern used for the flag could also be part of the information. If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame. To fix this problem, a byte-stuffing strategy was added to character-oriented framing.



Bit-Oriented Protocols

In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame, as shown in Figure 11.3.



6) Explain flow control and error control.

Flow Control

Flow control coordinates the amount of data that can be sent before receiving an acknowledgment and is one of the most important duties of the data link layer. In most protocols, flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgment from the receiver. The flow of data must not be allowed to overwhelm the receiver. Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data. The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily. Incoming data must be checked and processed before they can be used. The rate of such processing is often slower than the rate of transmission. For this reason, each receiving device has a block of memory, called a *buffer,* reserved for storing incoming data until they are processed. If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

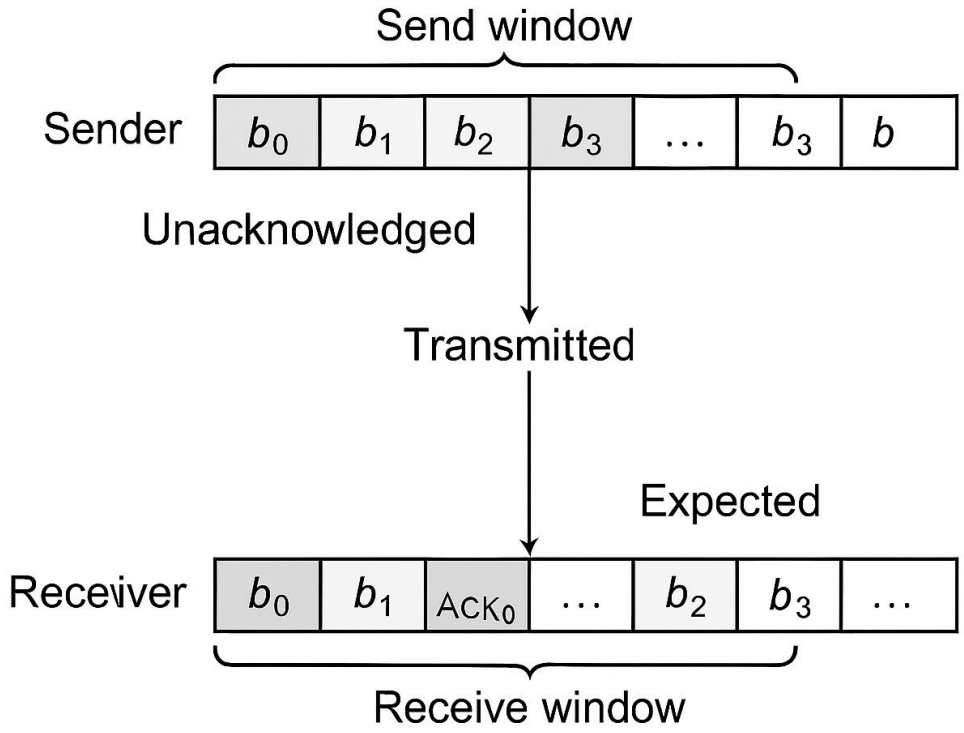
Error Control

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term *error control* refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

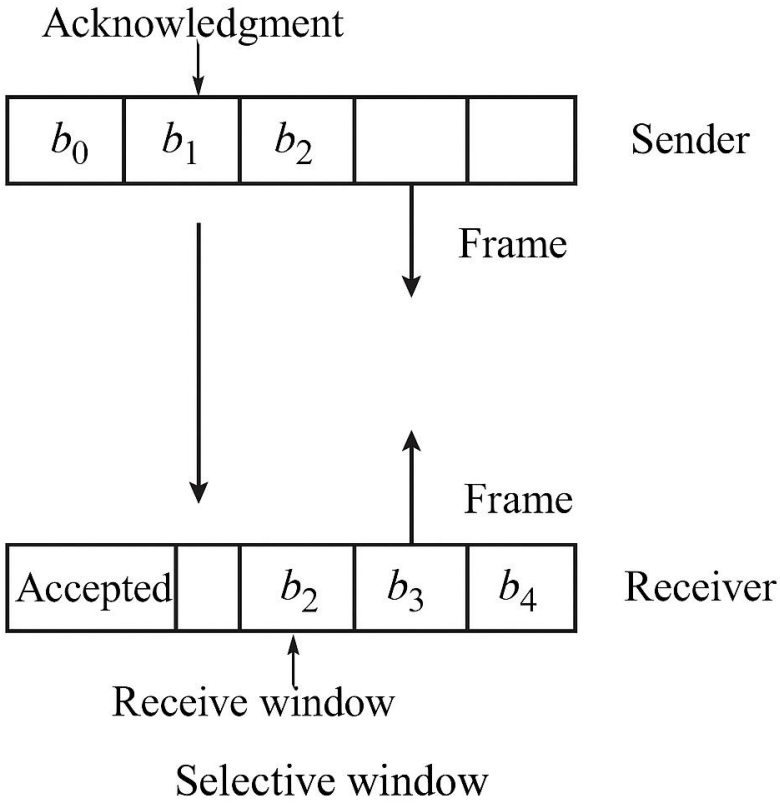
---

7) Explain with neat diagram Go Back N ARQ protocol detail



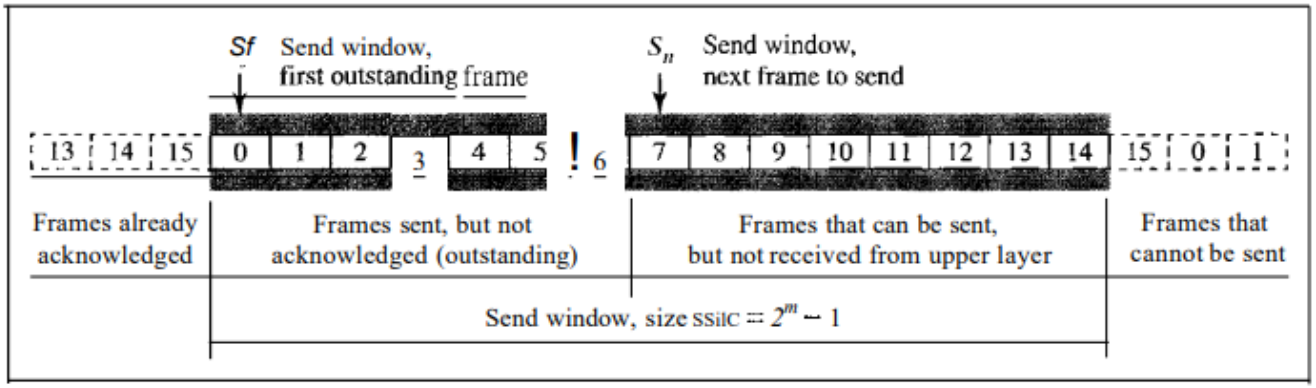
The **Go-Back-N (GBN) ARQ protocol** is a reliable data transmission method that uses a sliding window mechanism to send multiple frames before waiting for acknowledgments. The sender can transmit up to **N frames** without receiving an ACK, but if an error occurs or a frame is lost, it **retransmits that frame and all subsequent frames**, leading to some inefficiency. The receiver only accepts frames in **strict order** and discards any out-of-order frames, sending cumulative acknowledgments for the last correctly received frame. This makes GBN simple to implement but can result in redundant retransmissions, especially in noisy environments. While it is more efficient than the **Stop-and-Wait ARQ**, it is less efficient than **Selective Repeat ARQ**, which only retransmits corrupted frames. GBN is commonly used in **TCP, wireless networks, and satellite communications**, where simplicity is prioritized over bandwidth efficiency.

8) Describe with neat diagram Selective repeat ARQ protocol in detail.

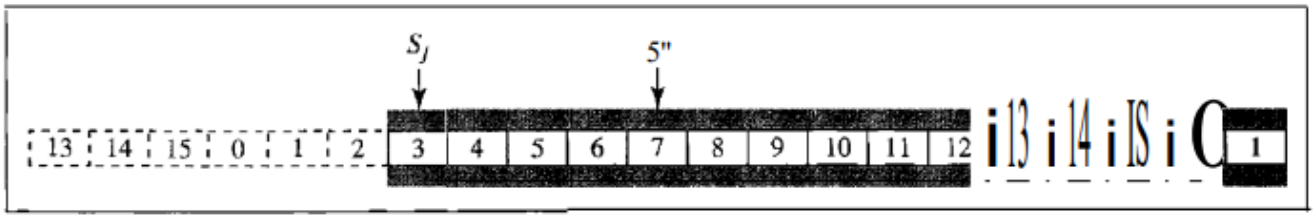


The **Selective Repeat (SR) ARQ protocol** is a reliable data transmission method that improves efficiency by retransmitting only the erroneous or lost frames, rather than all subsequent frames like in **Go-Back-N ARQ**. It uses a **sliding window** approach, where both the sender and receiver maintain a window of size **N**, allowing multiple frames to be sent and received out of order. The receiver individually acknowledges each correctly received frame and buffers out-of-order frames until missing frames arrive, ensuring proper sequencing. This reduces redundant retransmissions and improves bandwidth efficiency, especially in noisy networks. However, SR requires **more memory and processing power** at the receiver since it must store and reorder frames before passing them to the application. It is widely used in **high-speed networks, wireless communication, and TCP variations**, where minimizing retransmissions is crucial for performance.

9) Explain concept of sliding window with neat diagram.



a. Send window before sliding



b. Send window after sliding

The **sliding window** is an abstract concept in networking that defines the range of sequence numbers managed by the sender and receiver. It consists of two key parts: the **send window**, which tracks frames sent but not yet acknowledged, and the **receive window**, which handles incoming frames.

The **send window** is a moving range of sequence numbers representing frames in transit. It is divided into four regions: (1) acknowledged frames that are no longer of concern, (2) **outstanding frames** that have been sent but not yet acknowledged, (3) frames that **can be sent** but are waiting for data from the network layer, and (4) sequence numbers that cannot be used until the window moves.

Three key variables define the window's operation: **Sf (Send First)**, which is the oldest unacknowledged frame; **Sn (Send Next)**, which is the next frame to be sent; and **Ssize (Send Window Size)**, which determines the maximum number of frames in transit. The window slides forward as acknowledgments arrive, ensuring efficient and continuous data transmission. The maximum window size is $2^m - 1$, where **m** is the number of bits used for sequence numbers. This concept is essential in protocols like **Go-Back-N ARQ and Selective Repeat ARQ** to maintain reliable data transfer.

10) Bit-stuff the following data and highlight stuffed bit. (each 2)

Original Data:

`01111110111111011111110111110`

Bit-Stuffed Data (Stuffed bits in brackets [ ]):

`011111[0]1011111[0]1011111[0]0111110`

Explanation:

- A **0** is inserted after every five consecutive **1s** to prevent confusion with frame delimiters.
- The **stuffed bits** are shown inside **[brackets]**.
- The receiver removes these bits to retrieve the original data.

11) Numericals on CRC and Hamming coding.

1. CRC (Cyclic Redundancy Check) Numerical

Problem Statement:

A sender wants to transmit the **dataword 110101** using the generator polynomial $x^3 + x + 1$, which corresponds to the **binary divisor 1011**. Find the **CRC code** that will be transmitted.

Step 1: Append Zeros

Since the generator polynomial has a **degree of 3**, we append **three 0s** to the dataword:

$$\text{Dataword} = 110101$$

$$\text{Appended Data} = 110101000$$

Step 2: Perform Binary Division

We now divide **110101000** by **1011** using **XOR division** (since CRC works with modulo-2 arithmetic).

Binary Division Process:

```lua
  110101000 ÷ 1011
------------------
  1011        (1011 × 1)
  ------
  011111000
   1011      (1011 × 1)
   ------
    01011000
     1011    (1011 × 1)
     ------
      0111000
       1011   (1011 × 1)
       ------
        011
```

The remainder is **011**.

Step 3: Append Remainder to the Dataword

$$\text{Transmitted CRC Code} = 110101011$$

Thus, the **final transmitted codeword is 110101011**, which includes the original data **110101** plus the remainder **011**.

2. Hamming Code (Error Detection & Correction)

Problem Statement:

Encode the **4-bit data 1011** using **7-bit Hamming code with even parity.**

Step 1: Identify Parity Bit Positions

For a **4-bit** dataword, we need **3 parity bits (P1, P2, and P4)** in a **7-bit** Hamming code.

Bit Position	1	2	3	4	5	6	7
Bit Type	P1	P2	D1	P4	D2	D3	D4
Data	?	?	1	?	0	1	1

So, our Hamming code structure is:

$$P1, P2, 1, P4, 0, 1, 1$$

Step 2: Calculate Parity Bits (Even Parity)

- P1 covers bits 1, 3, 5, 7 → (P1 ⊕ 1 ⊕ 0 ⊕ 1) = 0
  - P1 = 0
- P2 covers bits 2, 3, 6, 7 → (P2 ⊕ 1 ⊕ 1 ⊕ 1) = 1
  - P2 = 1
- P4 covers bits 4, 5, 6, 7 → (P4 ⊕ 0 ⊕ 1 ⊕ 1) = 0
  - P4 = 0

Step 3: Final Hamming Code

$$0110011$$

Thus, the **final 7-bit Hamming code is 0110011.**

(for more info or numericals about crc and hamming code check page 109 of second book for networking)

12) Define piggybacking and its usefulness.

Definition of Piggybacking and Its Usefulness

Piggybacking is a technique used in **bidirectional data transmission** to improve efficiency by combining **control information (ACK/NAK)** with data frames. Instead of sending separate acknowledgment frames, a node **attaches the ACK/NAK** to the outgoing data frame being transmitted in the opposite direction.

Usefulness of Piggybacking:

1. **Increases Efficiency:** Reduces the number of separate acknowledgment frames, saving bandwidth.
2. **Reduces Overhead:** Merges control and data transmission into one frame, minimizing extra communication costs.
3. **Optimizes Bidirectional Communication:** Ensures data and control information flow in both directions without unnecessary delays.
4. **Improves Network Performance:** Fewer individual control frames reduce congestion and processing time.

Piggybacking is widely used in protocols like **Go-Back-N ARQ and Selective Repeat ARQ**, where acknowledgments are essential for reliable transmission.

13) Bit-stuff the following data and highlight stuffed bit. (each 2)

100011111110011111010001111111111000011111

Bit-Stuffing Process

Given Data:

`100011111110011111010001111111111000011111`

Bit-Stuffing Rule:

- Insert a **0** after every sequence of **five consecutive 1s** to prevent confusion with control signals.

Bit-Stuffed Data (Stuffed Bits Highlighted in [Brackets])

`1000111111[0]10011111[0]01000111111[0]1111[0]000011111[0]`

Explanation:

- A **0** is inserted after each sequence of **five consecutive 1s** (highlighted in brackets).
- The receiver will remove these extra **0s** to recover the original data.

14) A senders ends a series of packets to the same destination using 5-bit sequence numbers. If the sequence number starts with 0, what is the sequence number after sending 100 packets?

A sender uses **5-bit sequence numbers**, meaning the numbers range from **0 to 31** (since $2^5 = 32$). After reaching 31, the numbering **wraps around to 0** and continues cyclically.

To find the sequence number after sending **100 packets**, we perform the modulo operation:

$$\text{New Sequence Number} = 100 \mod 32$$

Step-by-step Calculation:

1. **Find how many full cycles of 32 fit into 100:**

$$100 \div 32 = 3 \text{ (quotient)}, \quad 3 \times 32 = 96$$

2. **Find the remainder:**

$$100 - 96 = 4$$

Thus, after sending **100 packets**, the sequence number will be **4.** ✅

15) Using 5-bit sequence numbers, what is the maximum size of the send and receive windows for each of the following protocols?

a. Stop-and-Wait ARQ

b. Go-Back-NARQ

c. Selective-Repeat ARQ

In protocols using **5-bit sequence numbers**, the sequence numbers range from **0 to 31** $(2^5 = 32)$. The **maximum window size** depends on the ARQ (Automatic Repeat reQuest) protocol.

(a) Stop-and-Wait ARQ

- Stop-and-Wait allows only one frame to be sent at a time before receiving an acknowledgment.
- **Maximum sender and receiver window size = 1**

(b) Go-Back-N ARQ

- The **sender window size** is $2^m - 1$, where **m = 5**:

$$\text{Sender Window Size} = 2^5 - 1 = 31$$

- The **receiver window size** is always **1**.

(c) Selective-Repeat ARQ

- The **maximum sender and receiver window size** is 2^{m-1} to prevent sequence number ambiguity:

$$\text{Window Size} = 2^{5-1} = 2^4 = 16$$

Final Answer Summary:

Protocol	Sender Window Size	Receiver Window Size
Stop-and-Wait ARQ	1	1
Go-Back-N ARQ	31	1
Selective-Repeat ARQ	16	16

16) Describe ICMP protocols (any 4 methods in each type)

ICMP stands for **Internet Control Message Protocol**. It is used with IP to report errors and send diagnostic messages. Since IP is unreliable and does not give feedback, ICMP helps by sending messages when packets face issues like delivery failure, time expiry, or wrong routing.

ICMP works at the **network layer** and its messages are carried inside IP packets. These messages help the sender know about the problem so that corrective action can be taken at higher layers.

There are two main types of ICMP messages – **error messages** and **query messages**.

Error messages are sent when something goes wrong during packet delivery. These are:

1. **Destination Unreachable** – if the packet can't reach the destination.
2. **Time Exceeded** – if the packet's TTL (Time To Live) becomes zero.
3. **Parameter Problem** – if there's an issue in the IP header.
4. **Redirection** – to inform the sender of a better route.
5. **Source Quench** – tells the sender to slow down (now outdated).

Error messages are always sent to the original sender. ICMP does not send error messages in response to another ICMP error message. Also, no error is sent for special addresses or for later fragments of a packet.

In error reporting, ICMP includes the IP header and first 8 bytes of the original data. This helps the sender identify which packet caused the issue.

Query messages are used to request or exchange information. These are used in request–reply form. Important types are:

- **Echo Request and Echo Reply** – used in ping to check host availability.
- **Timestamp Request and Reply** – used to calculate delay or time.
- **Address Mask Request and Reply** – to know the subnet mask.
- **Router Solicitation and Advertisement** – to discover nearby routers.

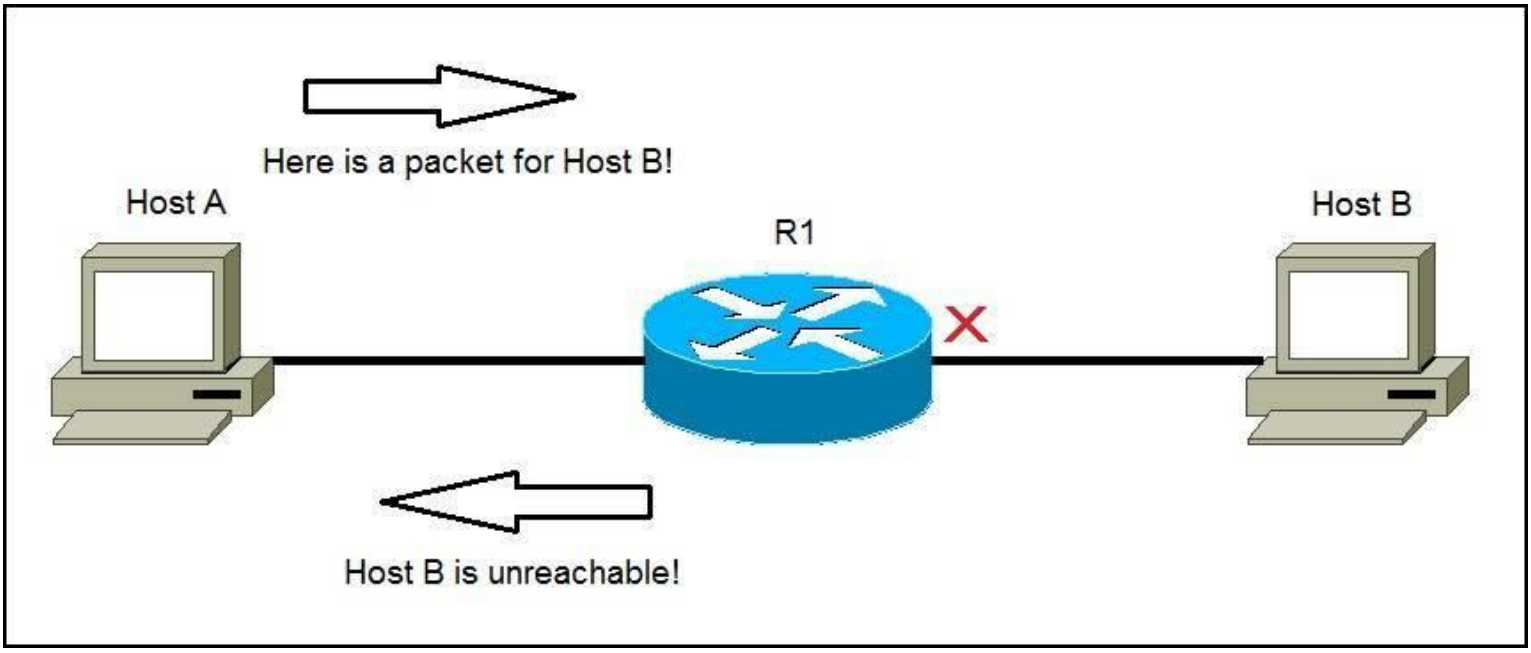
ICMP message format has the following fields:

- **Type** – shows the message category.
- **Code** – gives the specific reason.
- **Checksum** – for error detection.
- **Data** – includes extra information or part of the original packet.

Applications of ICMP:

- **Ping** sends Echo Request and waits for Echo Reply to check if a device is active and measure round-trip time.
- **Traceroute** increases TTL value in each packet to receive Time Exceeded messages from routers, which helps trace the path of the packet.

In conclusion, ICMP is an important protocol that helps monitor and report problems in an IP network. It doesn't correct errors, but it helps in identifying issues, making it very useful for network management and testing.



17) Compare TCP and UDP

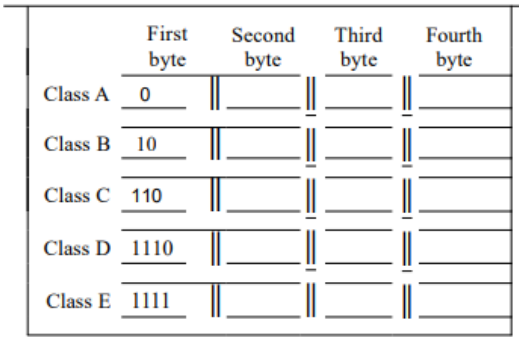
Feature	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Connection Type	Connection-oriented (establishes a connection before data transfer)	Connectionless (sends data without establishing a connection)
Reliability	Reliable, ensures data is delivered correctly and in order	Unreliable, does not guarantee delivery or order
Error Checking	Uses checksums, acknowledgments (ACK), and retransmissions for error detection and correction	Uses checksums for error detection but no retransmission
Flow Control	Uses mechanisms like sliding window and congestion control	No flow control mechanisms
Speed	Slower due to error checking and retransmission	Faster as it does not wait for acknowledgments
Overhead	High due to additional control mechanisms	Low since there are minimal control mechanisms
Data Order	Ensures data packets arrive in the correct sequence	No guarantee of packet order
Usage	Used for applications needing reliable communication, e.g., web browsing (HTTP, HTTPS), email (SMTP, IMAP)	Used for real-time applications where speed is important, e.g., video streaming, VoIP, online gaming

18) Discuss concept of Classful addressing

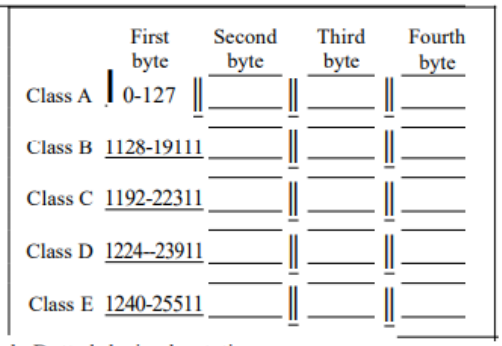
Classful Addressing

IPv4 addressing, at its inception, used the concept of classes. This architecture is called classful addressing. Although this scheme is becoming obsolete, we briefly discuss it here to show the rationale behind classless addressing.

In classful addressing, the address space is divided into five classes: A, B, C, D, and E. Each class occupies some part of the address space.



a. Binary notation



b. Dotted-decimal notation

Class	*Number of Blocks*	*Block Size*	*Application*
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved

Classful addressing was a system used in IPv4 to divide IP addresses into five classes: **A, B, C, D, and E.** When an organization needed a block of addresses, it was assigned one from Class A, B, or C.

- **Class A** addresses were designed for large organizations with a vast number of attached hosts or routers. However, these blocks were too large for most organizations, leading to wasted addresses.
- **Class B** addresses were meant for midsize organizations with thousands of hosts, but even these blocks were often larger than needed, causing inefficiency.
- **Class C** addresses, meant for small organizations, often proved too small for practical use.

Class D was specifically designed for multicasting, where each address defined a group of hosts on the Internet. However, the predicted need for these addresses was overestimated, resulting in wasted addresses. Similarly, Class E was reserved for future use, but only a few were ever utilized, leading to further address wastage.

This rigid structure of classful addressing caused inefficient utilization of IP addresses, leading to the eventual adoption of **Classless Inter-Domain Routing (CIDR)** to optimize address allocation.

19) Find the error, if any, in the given IPv4 address. 4 (Problems)

20) Compare IPv4 and IPv6. (8 points)

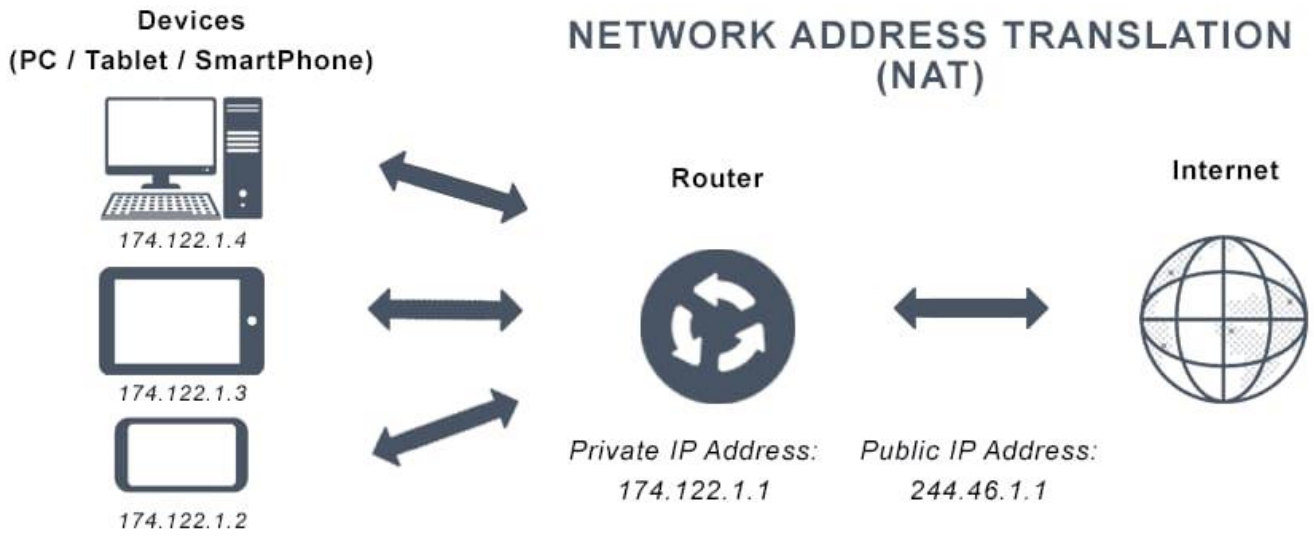
Comparison of IPv4 and IPv6

Feature	IPv4	IPv6
Address Length	32-bit (4.3 billion addresses)	128-bit (340 undecillion addresses)
Address Format	Decimal, written in **dotted decimal notation** (e.g., 192.168.1.1)	Hexadecimal, written in **colon-separated notation** (e.g., 2001:db8::1)
Header Complexity	Complex, contains **checksum**	Simplified, no checksum for efficiency
Address Allocation	Manually or DHCP	Auto-configuration using **Stateless Address Autoconfiguration (SLAAC)**
Security	Security is optional (IPSec is not built-in)	IPSec is **mandatory**, providing better security
Broadcasting	Supports broadcasting	No broadcasting, uses **multicasting** instead
Network Address Translation (NAT)	Widely used due to limited address space	Not required due to vast address availability
Routing Efficiency	Large routing tables, less efficient	More efficient routing with hierarchical addressing

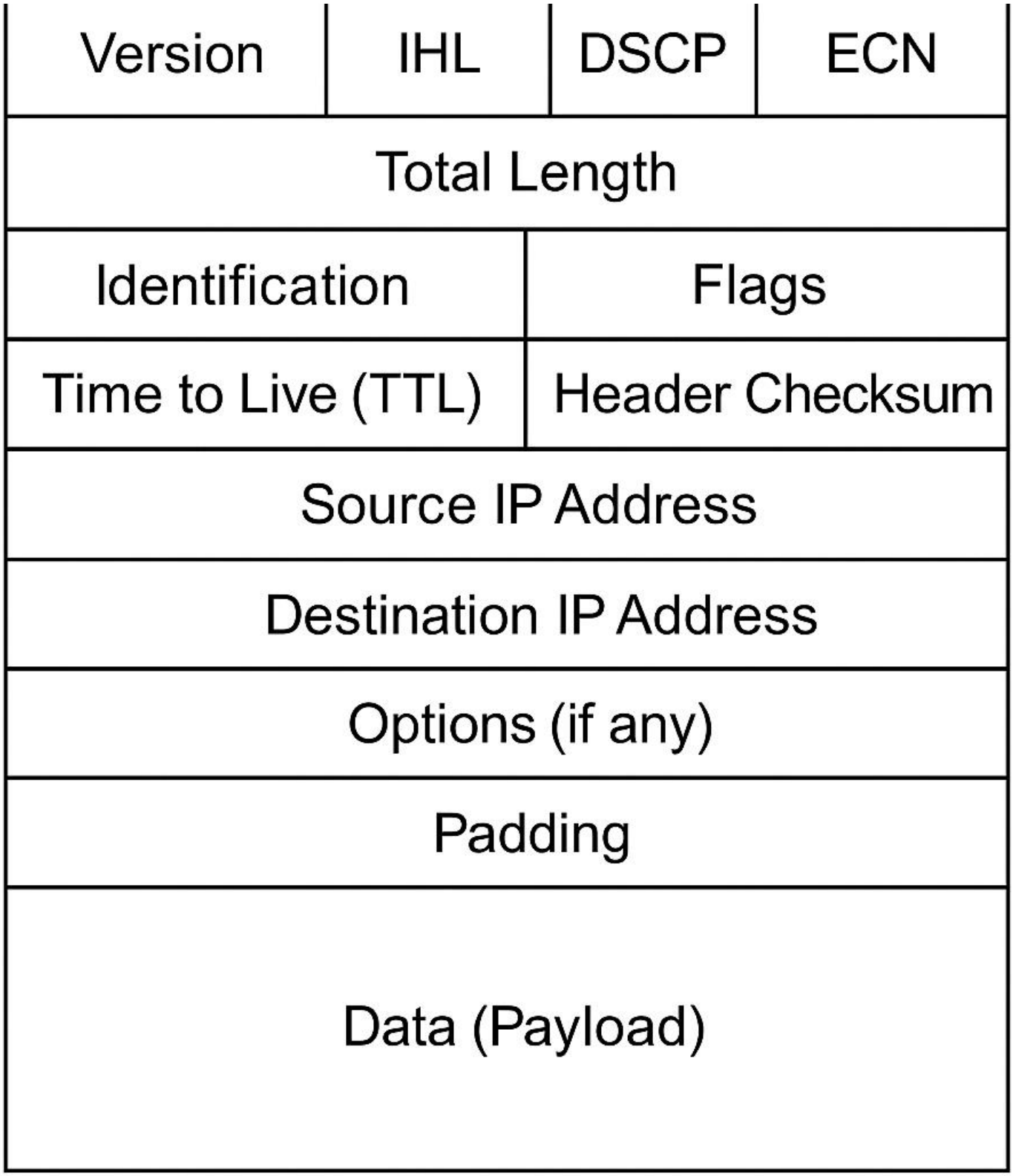
21) Explain the need of the Network address translation.

Network Address Translation (NAT) is used to modify IP addresses in network packets, allowing multiple devices on a private network to share a single public IP address. NAT is essential for several reasons:

1. **IPv4 Address Shortage** – Since IPv4 has a limited number of addresses (32-bit), NAT allows multiple devices to use a single public IP, conserving address space.
2. **Improved Security** – NAT hides internal private IP addresses from external networks, making it harder for attackers to directly target devices.
3. **Simplifies IP Address Management** – Organizations can use private IP addresses internally without needing to acquire large numbers of public IP addresses.
4. **Allows Multiple Devices to Access the Internet** – Home and business networks can connect several devices (PCs, smartphones, etc.) to the internet using one public IP address.
5. **Supports Legacy IPv4 Networks** – NAT helps in transitioning to IPv6 by enabling IPv4-based devices to continue functioning without requiring immediate upgrades.



22) Draw frame format of datagram.



23) Write advantages of IPv6 over IPv4.

Advantages of IPv6 over IPv4

IPv6 was developed to overcome the limitations of IPv4. Here are the key advantages:

1. **Larger Address Space** – IPv6 uses **128-bit addresses** compared to IPv4's **32-bit**, providing **virtually unlimited** IP addresses.
2. **Better Security** – IPv6 has **built-in IPsec (Internet Protocol Security)** for encryption and authentication, while IPv4 security is optional.
3. **Simplified Header** – IPv6 has a **more efficient and fixed header** format, reducing processing time and improving performance.
4. **No Need for NAT** – Since IPv6 provides **enough addresses**, there is no need for **Network Address Translation (NAT)**, simplifying communication.
5. **Improved Multicasting and Anycast** – IPv6 supports **better multicasting and anycast** for efficient data distribution and network optimization.
6. **Auto-configuration** – IPv6 allows **Stateless Address Autoconfiguration (SLAAC)**, enabling devices to configure themselves without a DHCP server.
7. **Better Quality of Service (QoS)** – IPv6 includes a **Flow Label field** in its header, ensuring **faster** and **more efficient** data handling.
8. **More Efficient Routing** – IPv6 reduces the size of routing tables and allows for **faster and more scalable** routing.

24) Explain frame fields related to the Fragmentation in datagram.

Fields Related to Fragmentation

The fields that are related to fragmentation and reassembly of an IPv4 datagram are the identification, flags, and fragmentation offset fields.

- Identification. This 16-bit field identifies a datagram originating from the source host. The combination of the identification and source IPv4 address must uniquely define a datagram as it leaves the source host. To guarantee uniqueness, the IPv4 protocol uses a counter to label the datagrams. The counter is initialized to a positive number. When the IPv4 protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by 1. As long as the counter is kept in the main memory, uniqueness is guaranteed. When a datagram is fragmented, the value in the identification field is copied to all fragments. In other words, all fragments have the same identification number, the same as the original datagram. The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value must be assembled into one datagram.
- Flags. This is a 3-bit field. The first bit is reserved. The second bit is called the *do notfragment* bit. If its value is 1, the machine must not fragment the datagram. If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host (see Chapter 21). If its value is 0, the datagram can be fragmented if necessary. The third bit is called the *more fragment* bit. If its value is 1, it means the datagram is not the last fragment; there are more fragments after this one. If its value is 0, it means this is the last or only fragment (see Figure 20.10).

or

In IPv4, fragmentation occurs when a datagram is too large to pass through a network with a lower **Maximum Transmission Unit (MTU)**. To handle fragmentation and reassembly, three key fields are used:

1. **Identification (16 bits)**
   - Each datagram from a source is assigned a **unique identification number** to help in reassembly.
   - If a datagram is **fragmented**, all fragments carry the **same identification number** as the original datagram.
   - This ensures that the destination can correctly group fragments belonging to the same datagram.
2. **Flags (3 bits)**
   - **Bit 1 (DF - Don't Fragment)**: If set to **1**, fragmentation is **not allowed**. If fragmentation is required but DF is set, the packet is discarded, and an **ICMP error message** is sent to the source.
   - **Bit 2 (MF - More Fragments)**: If set to **1**, it indicates **more fragments** are coming. If **0**, it is the **last fragment** or the only fragment.
3. **Fragment Offset (13 bits)**
   - Specifies the **position of a fragment** within the original datagram.
   - Helps the receiver **reassemble** fragments in the correct order.

25) Design a network and assign IP address to host system in a network. Find Last address, First address and subnet mask of sub networks. 8 (Problem)

26) Numerical on IP address.

Given: IP Address **192.168.10.0/26**

1. **Subnet Mask:** `255.255.255.192`
2. **Total Subnets:** `4` (using `2^2 = 4`)
3. **Valid Hosts per Subnet:** `62` (using `(2^6) - 2 = 62`)
4. **First and Last Valid IP in First Subnet:**
   - First Host: **192.168.10.1**
   - Last Host: **192.168.10.62**

27) Explain different functions of Network Layer.

Functions of the Network Layer

The **Network Layer** is responsible for **delivering packets** from the **source host** to the **destination host** across multiple networks. It ensures efficient data transmission using the following key functions:

1. **Logical Addressing:**
   - The **Data Link Layer** provides only local addressing within a network. However, for communication between different networks, a **unique logical address** (such as an **IP address**) is required.
   - The **Network Layer** adds a header containing **source and destination IP addresses** to help identify devices globally.
2. **Routing:**
   - In large networks or **internetworks (network of networks)**, packets must be forwarded across multiple devices to reach their destination.
   - The **Network Layer** determines the **best path** using **routing algorithms** and directs packets via **routers** to their correct destination.
3. **Packet Forwarding:**
   - Once the **best route** is determined, packets must be forwarded correctly to the **next hop** in the network.
   - Devices like **routers** handle packet forwarding based on **routing tables**.
4. **Fragmentation and Reassembly:**
   - If a packet is too large for the network's **Maximum Transmission Unit (MTU)**, it is **fragmented** into smaller packets.
   - The **Network Layer** ensures these fragments are **reassembled correctly** at the destination.

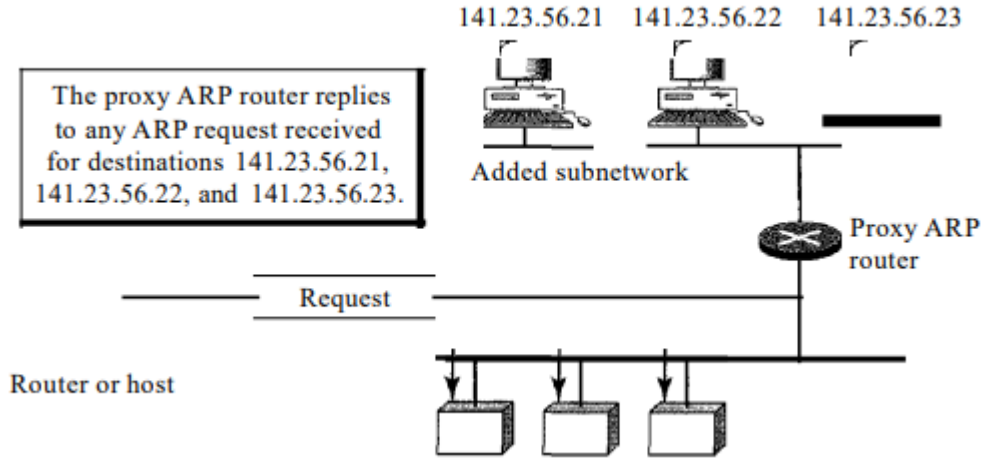
28) Explain different functions of Data Link Layer.

Functions of the Data Link Layer

The **Data Link Layer** is the second layer of the **OSI model** and is responsible for the **reliable transfer of data** between two directly connected devices. It ensures error-free communication over the physical medium by performing the following key functions:

1. **Framing:**
   - The Data Link Layer divides the network layer's data into **frames** for easier transmission.
   - Each frame contains **source and destination MAC addresses**, error detection bits, and control information.
2. **Addressing (MAC Addressing):**
   - Unlike the **Network Layer**, which uses logical (IP) addresses, the Data Link Layer assigns a **unique MAC (Media Access Control) address** to each device in a network.
   - This ensures proper **device-to-device communication** within the same network.
3. **Error Detection and Correction:**
   - During data transmission, errors may occur due to noise or interference.
   - The Data Link Layer uses techniques like **Cyclic Redundancy Check (CRC)** and **Parity Check** to detect and correct errors.
4. **Flow Control:**
   - Prevents a **fast sender** from overwhelming a **slow receiver** by using flow control mechanisms like **Stop-and-Wait** and **Sliding Window Protocols**.
5. **Access Control (Media Access Control - MAC):**
   - Determines how multiple devices share the same communication channel using methods like **CSMA/CD (for wired networks)** and **CSMA/CA (for wireless networks)**.
   - Prevents **collisions** and ensures efficient data transfer.

29) Explain ARP and RARP with diagram.



Address Resolution Protocol (ARP)

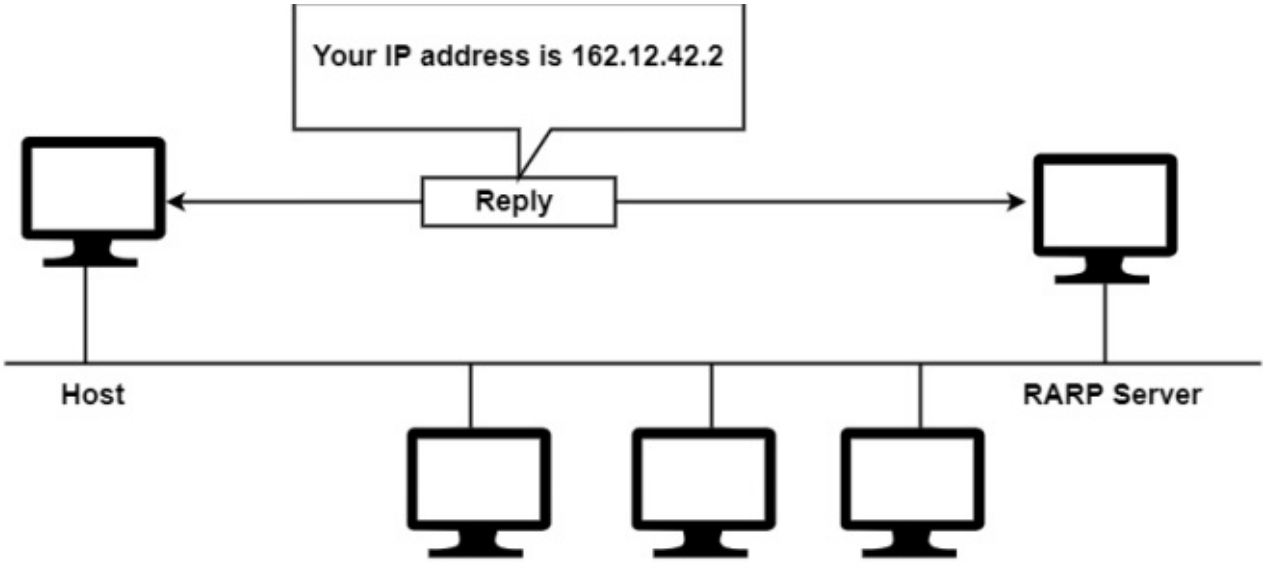
ARP is a network protocol used to map an **IP address to a MAC address**. When a device wants to communicate within a network, it needs to know the **MAC address** of the destination device.

Working of ARP:

1. The sender broadcasts an **ARP request** asking, "Who has this IP address?"
2. The device with the matching IP responds with an **ARP reply**, providing its **MAC address**.
3. The sender stores this MAC address in its **ARP cache** for future communication.

Example:

- If a computer wants to send data to another device with IP **192.168.1.5**, it first uses **ARP** to get its MAC address before sending data.



(b) RARP reply is unicast

Reverse Address Resolution Protocol (RARP)

RARP is used to obtain an **IP address when a device only knows its MAC address**. It is mainly used by diskless computers that do not store their own IP addresses.

Working of RARP:

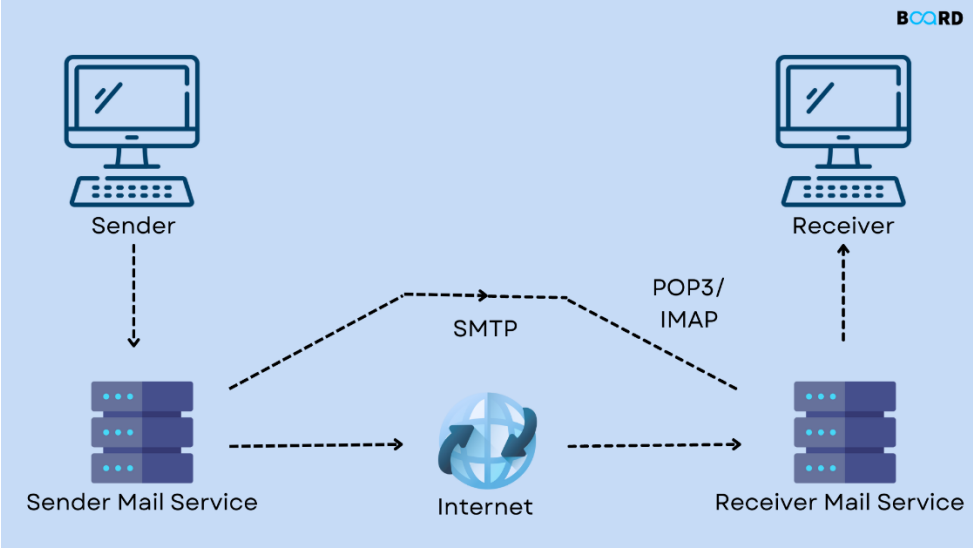
1. The device sends a **RARP request** asking, "What is my IP address?" and includes its **MAC address**.
2. A **RARP server** responds with the **corresponding IP address**.

Example:

- A diskless workstation, after booting, sends a **RARP request** to a server to receive its IP address.

RARP has been replaced by modern protocols like **DHCP** for dynamic IP assignment.

30) Describe SMTP protocols with diagram.



Simple Mail Transfer Protocol (SMTP)

SMTP (Simple Mail Transfer Protocol) is an **application layer protocol** used for **sending emails** between mail servers over the Internet. It follows a **client-server model** and works on **port 25 (default), 587 (secure), or 465 (SSL/TLS).**

Working of SMTP:

1. **The sender's email client** connects to the SMTP server and sends the email.
2. **The SMTP server** forwards the email to the recipient's mail server.
3. **The recipient's mail server** stores the email until the recipient retrieves it using **POP3 or IMAP.**

Key SMTP Commands:

- **HELO/EHLO:** Identifies the sender to the server.
- **MAIL FROM:** Specifies the sender's email address.
- **RCPT TO:** Specifies the recipient's email address.
- **DATA:** Sends the actual email content.
- **QUIT:** Terminates the connection.

Limitations of SMTP:

- Only supports **sending** emails, not retrieving them.
- Does not support **email encryption** by default.
- Can be vulnerable to **spam and email spoofing** if not secured properly.

SMTP ensures **efficient and reliable email transmission** across networks, making it a fundamental protocol for email communication.
