COMPUTER NETWORKS–V SEM CSE VTU

MODULE-3: NETWORK LAYER

1. NETWORK-LAYER

1.1 Network Layer Services

1.2 Packet Switching

1.3 IPv4 Addresses

1.4 IPv4 Datagram

1.5 IPv6 Datagram

2. INTRODUCTION TO ROUTING ALGORITHMS

2.1 Introduction

2.2 Routing Algorithms

2.3 Unicast Routing Protocols:DVR, LSR, PVR

2.4 Unicast Routing Protocols:RIP, OSPF,BGP

2.5 Multicast Routing:MOSPF

1. NETWORK-LAYER

The network layer in the **TCP/IP protocol suite** plays a crucial role in ensuring **host-to-host delivery** of data. It operates between the **data-link layer** and the **transport layer**, providing services to the latter while receiving services from the former. It is responsible for routing and delivering packets across various networks, encapsulating data into packets, and decapsulating them at the destination.

1.1 Network Layer Services

The Internet, a combination of **LANs** and **WANs**, consists of many networks connected through routers and switches. The network layer is involved in:

- **Source Host (e.g., Alice):** It encapsulates the transport layer packet into a datagram and hands it over to the data-link layer for transmission.
- **Destination Host (e.g., Bob):** It decapsulates the received datagram to retrieve the packet and deliver it to the transport layer.
- **Routers (R2, R4, R5, R7):** They forward the datagrams across multiple networks until they reach the destination.

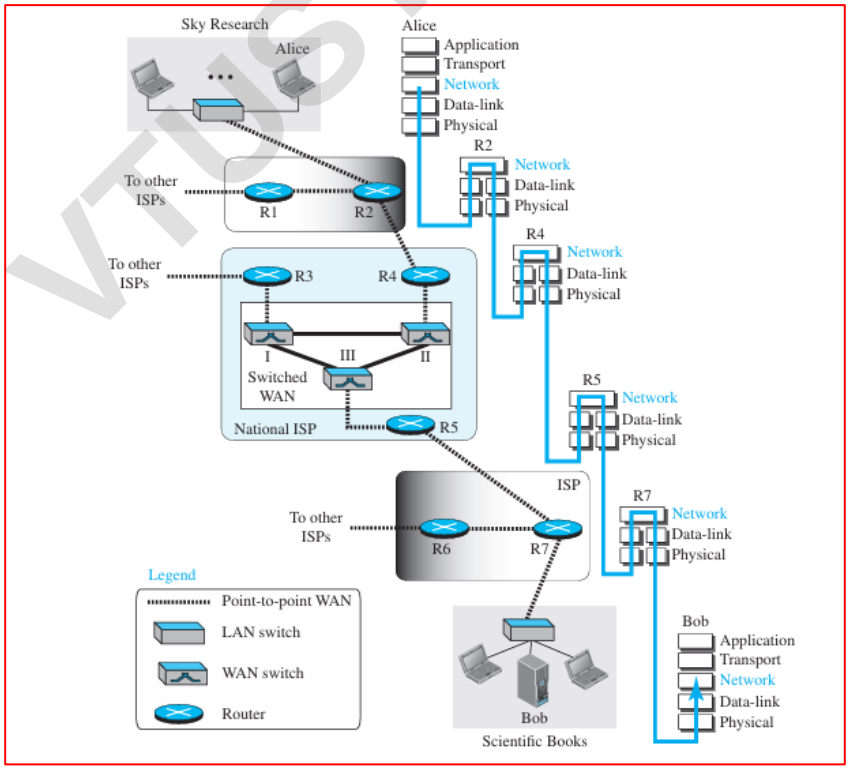


Figure 3.1:Communication at the network layer

Functions of the Network Layer

1. **Packetizing:**
    - The network layer encapsulates data from the transport layer into packets (datagrams) at the source and decapsulates them at the destination.
    - The source adds necessary headers, including source and destination addresses, to the packet.
2. **Routing:**
    - The network layer **selects the best route** for packets to travel across multiple networks (LANs, WANs).
    - Routers use algorithms to determine the most efficient path from source to destination.
3. **Forwarding:**
    - Forwarding refers to **how routers handle packets** upon arrival. Routers look up the forwarding table or routing table to determine which network interface to send the packet through.
    - The decision is based on packet headers (e.g., destination address, labels).

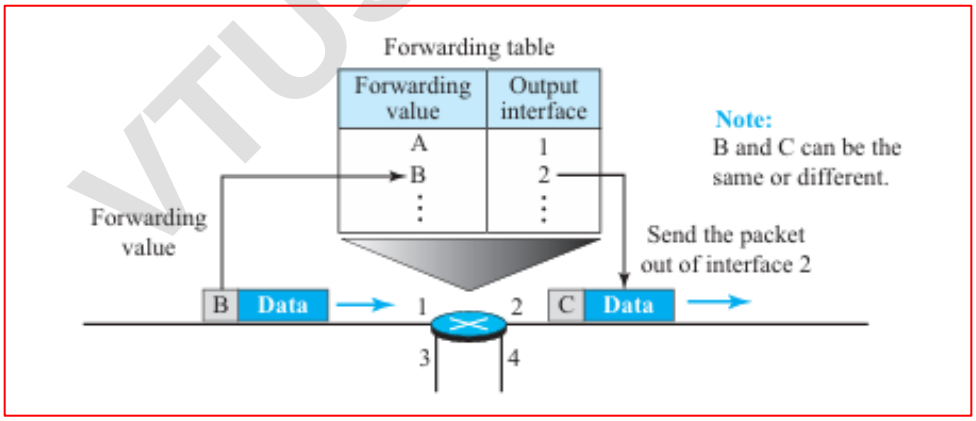


Figure 3.2: Forwarding process

Additional Services Provided by the Network Layer

- **Error Control:**
    - Although the network layer itself doesn't handle errors directly, the **ICMP protocol** helps detect errors, such as when a packet cannot be delivered.

- **Flow Control:**
    - The network layer helps prevent the receiver from becoming overwhelmed with data by ensuring the sender doesn't send more data than the receiver can process.
- **Congestion Control:**
    - The network layer helps in managing and avoiding congestion in the network by controlling the flow of packets, especially in high-traffic conditions.
- **Quality of Service (QoS):**
    - QoS ensures the network layer delivers data with a certain performance level, managing delays, throughput, and packet loss.

1.2 Packet Switching

- Packet switching is a method of data transmission in which information is broken into small, manageable pieces called packets. Each packet is sent independently through a network, and they may take different routes to reach the destination. Once all packets arrive, they are reassembled in the correct order to recreate the original message
- .Although in data communication switching techniques are divided into two broad categories, circuit switching and packet switching, only packet switching is used at the network layer because the unit of data at this layer is a packet.
- Circuit switching is mostly used at the physical layer; the electrical switch mentioned earlier is a kind of circuit switch.
- At the network layer, a message from the upper layer is divided into manageable packets and each packet is sent through the network.
- The source of the message sends the packets one by one; the destination of the message receives the packets one by one.
- The destination waits for all packets belonging to the same message to arrive before delivering the message to the upper layer

Today, a packet-switched network can use two different approaches to route the packets: the **datagram approach**and **the virtual circuit approach.**

Types of Packet Switching:

- **Datagram Packet Switching:**
  - Each packet is treated independently with no pre-established path.
  - Routing decisions are made for each packet at every node.
  - Examples: User Datagram Protocol (UDP).

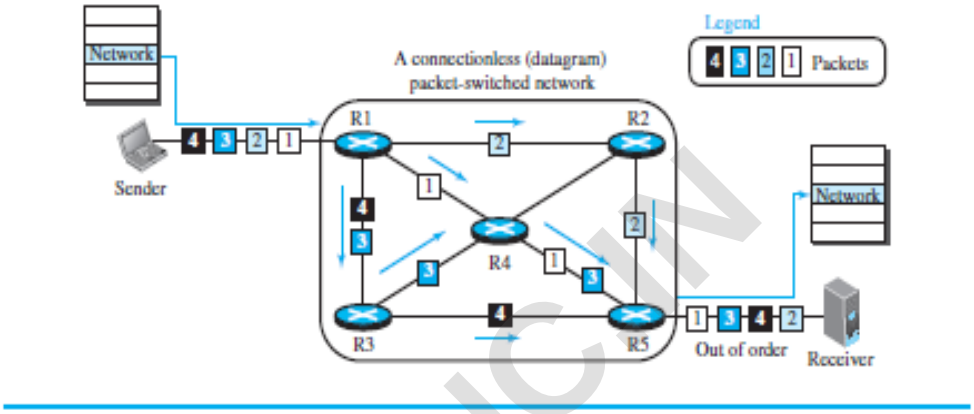


Fig 3.3: A connectionless packet-switched network

- **Virtual Circuit Packet Switching:**
  - A logical path (virtual circuit) is established between sender and receiver before any packets are sent.
  - Packets follow the same route, which helps maintain the order of transmission.
  - Examples: Transmission Control Protocol (TCP), Frame Relay.

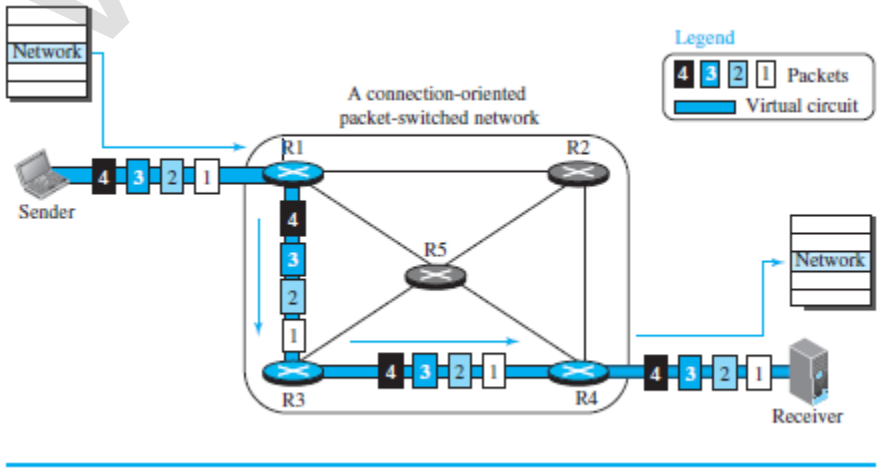


Fig 3.4 A virtual-circuit packet-switched network

The virtual-circuit approach is a type of **connection-oriented service** used in data communication networks. Before transmitting data, a logical connection, known as a virtual circuit, is established between the sender and receiver. All data packets then follow this predefined path throughout the communication session.

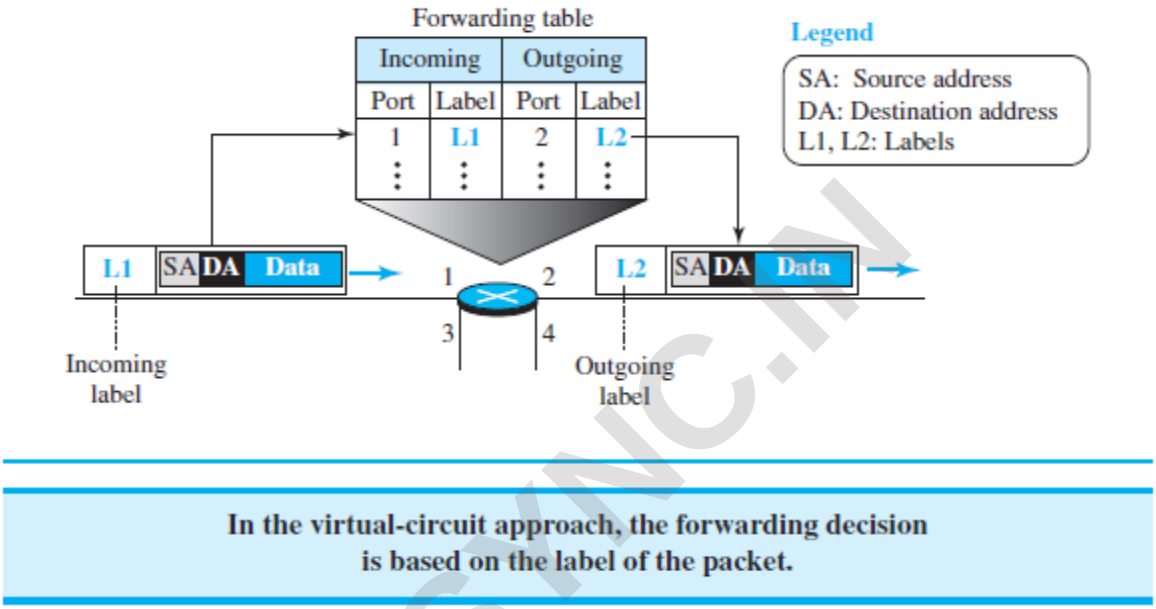


Fig 3.5: Forwarding process in a router when used in a virtual-circuit network

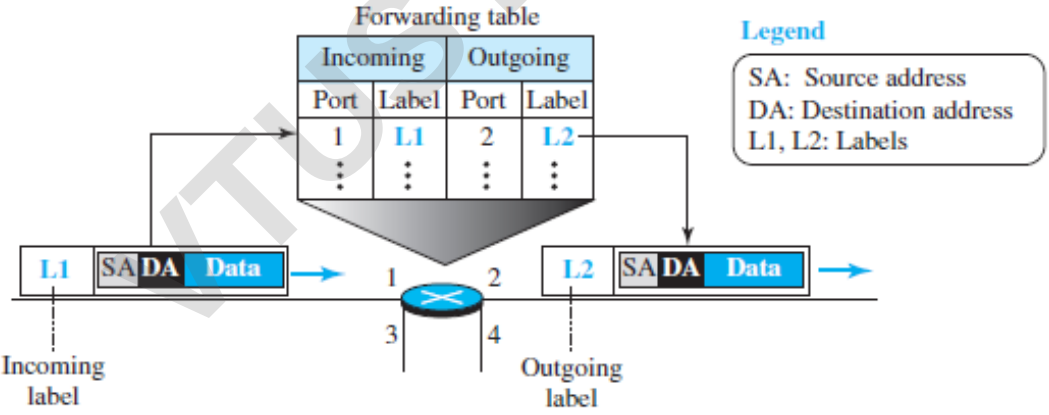
- To create a connection-oriented service, a three-phase process is used: setup, data transfer, and teardown.
- In the setup phase, the source and destination addresses of the sender and receiver are used to make table entries for the connection-oriented service.
- In the teardown phase, the source and destination inform the router to delete the corresponding entries. Data transfer occurs between these two phases.

1. Setup Phase

- **Goal:** Establish a virtual circuit between **Source A** and **Destination B** using request and acknowledgment packets.
- **Steps in Setup:**
  - **Request Packet:**
    1. **Source A** sends a **request packet** to **Router R1**.
    2. **Router R1**:

- Identifies the outgoing port for packets to B (Port 3).
- Assigns:
  - **Incoming Port:** 1
  - **Incoming Label:** 14
  - **Outgoing Port:** 3
- Forwards the packet to **Router R3**.

3. **Router R3**:
   - Fills its routing table with:
     - **Incoming Port:** 1
     - **Incoming Label:** 66
     - **Outgoing Port:** 3
   - Forwards the packet to **Router R4**.
4. **Router R4**:
   - Assigns:
     - **Incoming Port:** 1
     - **Incoming Label:** 22
     - **Outgoing Port:** 4
   - Sends the request packet to **Destination B**.
5. **Destination B** assigns **Label 77** to incoming packets from A.



In the virtual-circuit approach, the forwarding decision is based on the label of the packet.

Fig: 3.6: Sending request packet in a virtual-circuit network

- **Acknowledgment Packet:**
  1. **Destination B** sends an **acknowledgment packet** back to **Router R4**:
     - Router R4 fills the **outgoing label** as **77**.

2. **Router R4** sends the acknowledgment to **Router R3**, which fills its **outgoing label** as **22**.
3. **Router R3** sends an acknowledgment to **Router R1**, which fills its **outgoing label** as **66**.
4. **Router R1** sends the acknowledgment to **Source A**, which fills its **outgoing label** as **14** for future packets.

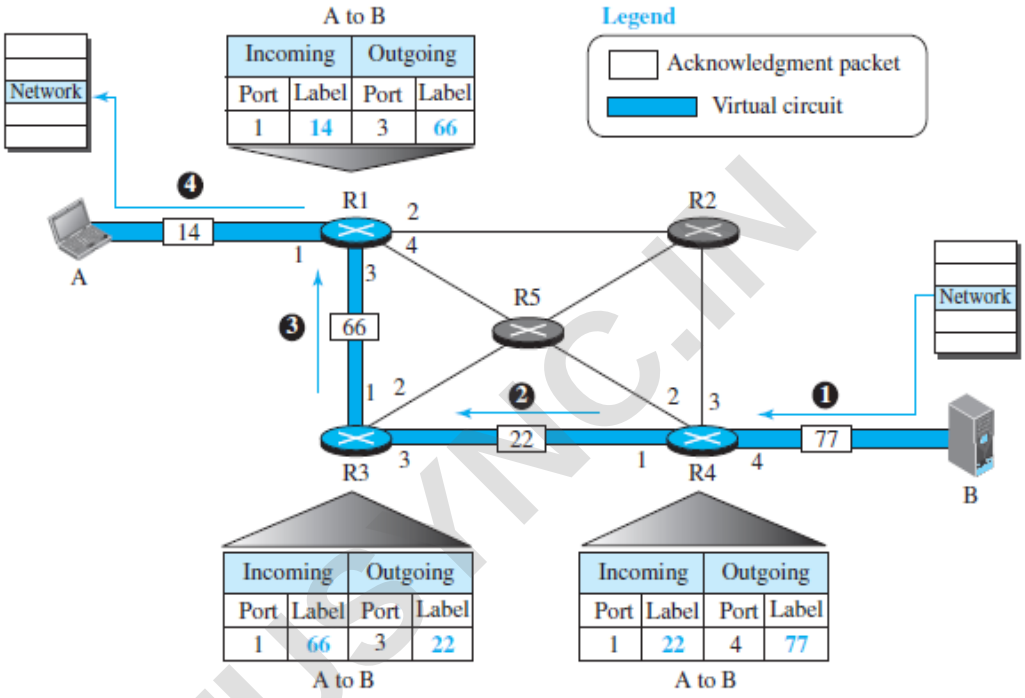


Fig: 3.7: Sending acknowledgments in a virtual-circuit network

2. Data Transfer Phase

- **Goal:** Transfer data between Source A and Destination B once the virtual circuit is established.
- **Steps:**
    1. **Source A** labels the packets with **Label 14** (from R1).
    2. **Router R1**:
        - Receives packets with Label 14.
        - Changes the label to **66** and forwards to **Router R3**.
    3. **Router R3**:
        - Receives packets with Label 66.
        - Changes the label to **22** and forwards to **Router R4**.
    4. **Router R4**:
        - Receives packets with Label 22.

- Changes the label to **77** and forwards to **Destination B**.

5. **Destination B** receives the packets with **Label 77** (which it recognizes from setup).

3. Teardown Phase

- **Goal:** End the virtual circuit once data transmission is complete.
- **Steps:**
  1. **Source A** sends a **teardown packet** to **Destination B**.
  2. **Destination B** responds with a **confirmation packet**.
  3. Each router along the path (R4, R3, R1) deletes the virtual circuit entries from its routing table.

1.3 IPv4 Datagram

• Packetizing is the primary service provided by IPv4. It defines the format of a packet (called a **datagram**) to encapsulate data from the upper layers.

• An **IPv4 datagram** is a variable-length packet, with a **header** (20 to 60 bytes) and **payload** (data).

- A **datagram** consists of two main parts:
  - **Header** (20–60 bytes): Contains control and routing information.
  - **Payload** (data): The actual data being transmitted from the upper layers (e.g., TCP/UDP).

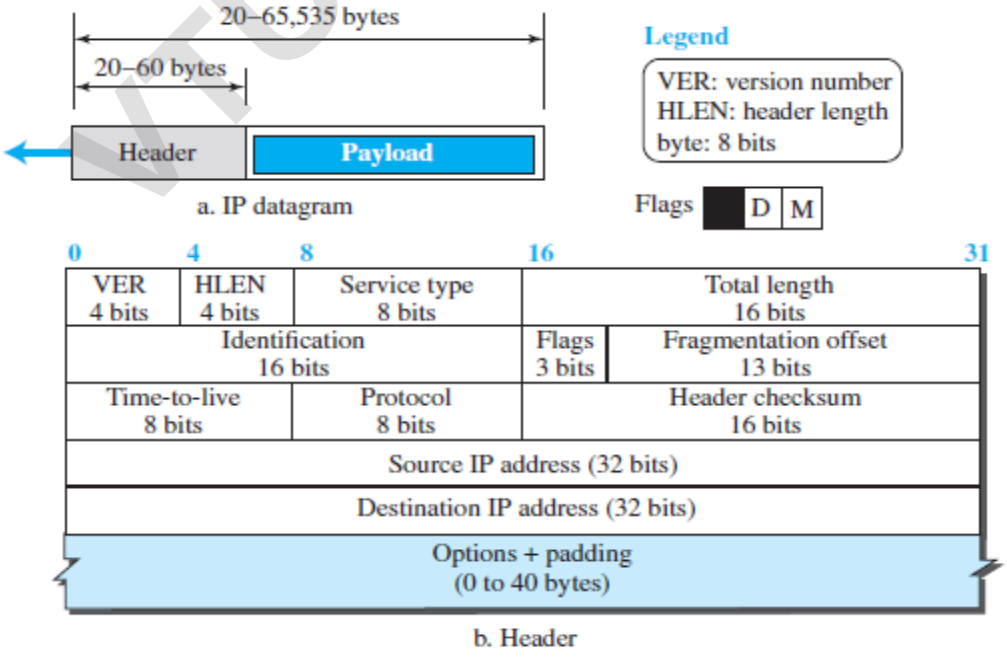


Fig 3.8: IPv4 datagram format.

IPv4 Datagram Structure

- **Header:**
  - 20 to 60 bytes in length.
  - Contains crucial fields to ensure the proper routing and handling of the packet.
  - Shown in **4-byte (32-bit) sections** for easy reference.
- **Payload:**
  - Contains the data from the upper-layer protocols, such as TCP, UDP, or other transport or network protocols.

2. Key Header Fields

1. **Version (4 bits):**
   - Specifies the IP version. For IPv4, the value is always **4**.
2. **Header Length (HLEN) (4 bits):**
   - Indicates the length of the header in **4-byte words**.
   - Helps identify where the header ends and the data (payload) begins.
3. **Service Type (8 bits):**
   - Originally called **Type of Service (TOS)**, now redefined as **Differentiated Services (DiffServ)**.
   - Used for prioritizing different types of network traffic (e.g., voice over IP).
4. **Total Length (16 bits):**
   - Specifies the total length of the datagram, including both header and data, in bytes.
   - Helps receivers distinguish between actual data and any padding that might be added (e.g., in Ethernet frames).
5. **Identification, Flags, and Fragmentation Offset:**
   - Used for **fragmenting** datagrams that exceeds the maximum transmission unit (MTU) of the network.
   - Allows reassembly of fragmented packets at the destination.
6. **Time-to-Live (TTL) (8 bits):**
   - Limits the number of hops a datagram can take.
   - Each router decrements this value, and if it reaches **zero**, the datagram is discarded.
   - Prevents datagrams from endlessly circulating due to routing errors.
7. **Protocol (8 bits):**
   - Identifies the protocol of the encapsulated payload (e.g., **TCP = 6**, **UDP = 17**).
   - Ensures the correct upper-layer protocol receives the data at the destination (demultiplexing).

8. **Header Checksum (16 bits):**
    - Used to verify the integrity of the **header** (not the payload).
    - Each router recalculates the checksum since fields like TTL change with every hop.
9. **Source and Destination IP Addresses (32 bits each):**
    - **Source IP Address:** The IP address of the sender.
    - **Destination IP Address:** The IP address of the receiver.
    - These fields remain **unchanged** as the datagram travels across the network.
10. **Options (0–40 bytes):**
    - Optional field used for network testing, debugging, or special features.
    - Adds flexibility, but may require routers to recalculate the checksum if options are altered.

3. Payload (Data)

- The **payload** is the data encapsulated in the datagram, which comes from upper-layer protocols (e.g., **TCP, UDP**).
- The payload is the primary reason for sending the datagram, with the header providing the necessary instructions for delivery.

4. Important Concepts

- **Multiplexing and Demultiplexing:**
    - The **Protocol field** in the IP header functions like **port numbers** in the transport layer.
    - It ensures data is delivered to the correct upper-layer protocol.
- **Fragmentation:**
    - Large datagrams are fragmented when they exceed the size limit of the network's maximum transmission unit (MTU).
    - The **Identification, Flags, and Fragmentation Offset** fields help in reassembling the fragments at the destination.
- **TTL and Loop Prevention:**
    - The **TTL field** prevents infinite looping by limiting the number of hops the packet can traverse.
    - If the TTL reaches zero, the datagram is discarded.

- An **IPv4 address** is a **32-bit identifier** used to uniquely define the connection of a device (host or router) to the Internet.

1.4 IPv4 Addresses

- The address is assigned to the **connection** (not the device), meaning if a device changes its network, its IP address may also change.
- IPv4 addresses are both **unique** and **universal**, ensuring global consistency and uniqueness across all devices connected to the Internet

1. 4.1 Address Space

- The **address space** is the total number of addresses available for use in the IPv4 protocol.
- IPv4 uses **32 bits**, so the address space is 2^{32} (or 4,294,967,296 addresses).
- In theory, this would allow more than 4 billion devices to connect to the Internet, but practical limitations such as network design reduce the usable address space.

1.4.2. IPv4 Address Notations

IPv4 addresses can be represented in three main formats:

- **Binary Notation (Base 2):**
  - The address is shown as 32 bits, separated into 8-bit **octets** for readability (e.g., 11000000 10101000 00000001 00000001).
- **Dotted-Decimal Notation (Base 256):**
  - The address is written in decimal format, with each octet separated by dots (e.g., 192.168.1.1).
  - Each octet is a value between **0 and 255**.
- **Hexadecimal Notation (Base 16):**
  - The address is represented as 8 hexadecimal digits (e.g., C0A80101).
  - This format is often used in network programming.

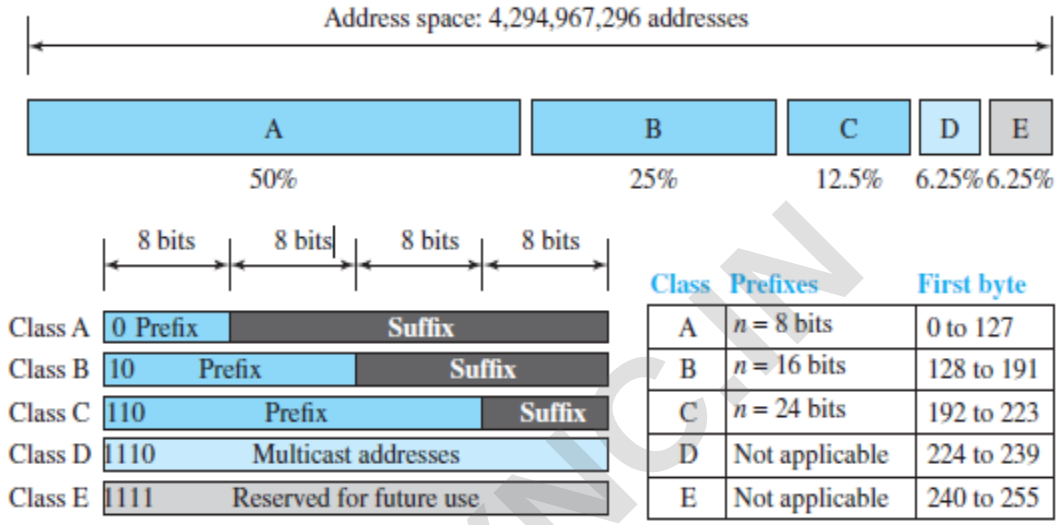
1.4.3. Hierarchy in IPv4 Addressing

- **IPv4 addresses are** hierarchical**, divided** into two parts:
  - **Prefix:** Defines the network (similar to a postal country or city code).
  - **Suffix:** Defines the specific node (device or connection) on that network (similar to a house number in postal addressing).
- There are two main types of address hierarchy:
  - **Classful Addressing:** Original fixed-length prefix system (now obsolete).
  - **Classless Addressing:** Modern variable-length prefix system.

4Classful Addressing (Obsolete)

- **Classful Addressing** divides the IPv4 address space into five classes (A, B, C, D, E), designed to accommodate different network sizes.

IPv4 Address Classes:



Class	Prefixes	First byte
A	$n = 8$ bits	0 to 127
B	$n = 16$ bits	128 to 191
C	$n = 24$ bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255

Fig 3.9: classful addressing

1. **Class A:**
   - Prefix length: **8 bits** (first bit is always 0, so only 7 bits are available for network identification).
   - Supports up to **128 networks**.
   - Example range: 1.0.0.0 to 126.0.0.0.
2. **Class B:**
   - Prefix length: **16 bits** (first two bits are 10, so 14 bits for network identification).
   - Supports up to **16,384 networks**.
   - Example range: 128.0.0.0 to 191.255.0.0.
3. **Class C:**
   - Prefix length: **24 bits** (first three bits are 110, so 21 bits for network identification).
   - Supports up to **2,097,152 networks**.
   - Example range: 192.0.0.0 to 223.255.255.0.
4. **Class D (Multicast):**
   - First four bits are 1110.
   - Reserved for **multicast** addresses (group communication).
   - Range: 224.0.0.0 to 239.255.255.255.

5. **Class E (Experimental):**
    - First four bits are 1111.
    - Reserved for **experimental** purposes.
    - Range: 240.0.0.0 to 255.255.255.255.

Address Depletion

Problem: Classful addressing led to rapid depletion of IPv4 addresses due to inefficient distribution.

- **Class A**:
    - Designed for very large organizations.
    - Only 128 networks available, each with 16,777,216 addresses.
    - Most organizations didn't need so many addresses, leading to significant waste.
- **Class B**:
    - Designed for midsize organizations.
    - Despite more networks available, many addresses remained unused.
- **Class C**:
    - Intended for smaller networks.
    - Each network had only 256 addresses, which was often too few for organizations.
- **Class E**: Reserved and almost never used, leading to wastage of the entire class.

Solutions to Address Depletion

- **Subnetting**:
    - **Concept**: Divides a large class A or class B network into smaller subnets.
    - **Method**: Increases the prefix length, creating multiple smaller networks from a single large network.
    - **Issue**: Not widely adopted because large organizations were reluctant to share unused addresses with smaller ones.
- **Supernetting**:
    - **Concept**: Combines several smaller class C networks into a larger block.
    - **Purpose**: Designed to create larger address blocks for organizations needing more than 256 addresses.
    - **Issue**: Made packet routing more complex, limiting its effectiveness.

Advantage of Classful Addressing

- **Simplicity**: The class of an address is easily identifiable from the address itself.

- **Fixed Prefix Length**:
    - Each address class (A, B, C) has a predefined prefix length.
    - No additional information is required to determine the prefix and suffix, making it easier to understand and manage

1.Introduction to Classless Addressing:

- Classless addressing was introduced to solve the IPv4 address depletion problem.
- It eliminates the rigid class structure (A, B, C) of classful addressing and allows variable-length blocks of IP addresses.
- Provides more efficient and flexible distribution of IP addresses to organizations and ISPs.
- Classful addressing wasted many IP addresses due to fixed-size blocks; classless addressing resolves this by allowing different block sizes.

2. Why Classless Addressing Was Introduced:

- **Address Depletion**: The Internet's growth led to a shortage of IPv4 addresses.
- **ISPs (Internet Service Providers)**: ISPs required a flexible address allocation system to serve individuals, small businesses, and organizations efficiently.
- **Efficient Allocation**: Classless addressing allowed ISPs to assign blocks of addresses based on actual needs, rather than forcing organizations to take larger, unused ranges.

3. Variable-Length Blocks:

- In classless addressing, IP addresses are grouped into blocks of varying sizes, which can be 2, 4, 8, 16, 32, 64 addresses, and so on.
- The block sizes must be a power of 2, such as 2^1, 2^2, 2^3 and so on.

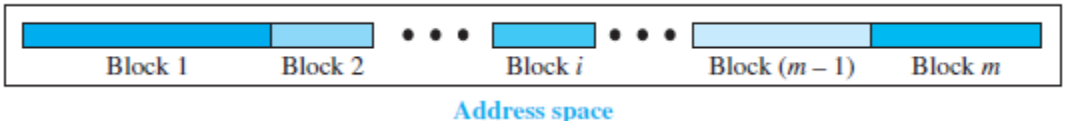


Fig: 3.10: ***Variable-length blocks in classless addressing***

4. Prefix Length and Slash Notation (CIDR):

- The length of the network portion of the IP address is indicated by the **prefix length**.
- **Slash Notation** or **CIDR (Classless Interdomain Routing)** is used to specify the prefix length, written as /n, where n is the number of bits used for the network part.

- Example: 192.168.1.0/24 means the first 24 bits are for the network and the remaining 8 bits are for hosts.

- **Larger prefix length = smaller network** (fewer hosts).
- **Smaller prefix length = larger network** (more hosts).

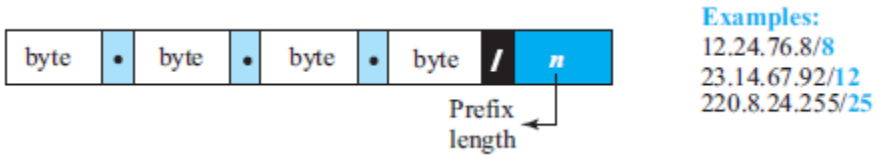


Fig: 3.11: *Slash notation (CIDR)*

5. Block of Addresses:

- A block of addresses is determined by the **prefix** (network part) and the **suffix** (host part).
- The size of the block is based on the prefix length.
- The number of addresses in a block is calculated using the formula: $N=2^{32-n}$ where n is the prefix length.

6. How to Extract Information from a Block:

- To extract key information from any given IP address block:
  1. **Number of addresses**: Use $N=2^{32-n}$ where n is the prefix length.
  2. **First address**: Set the last (32−n) bits to 0.
  3. **Last address**: Set the last (32 – n) bits to 1.

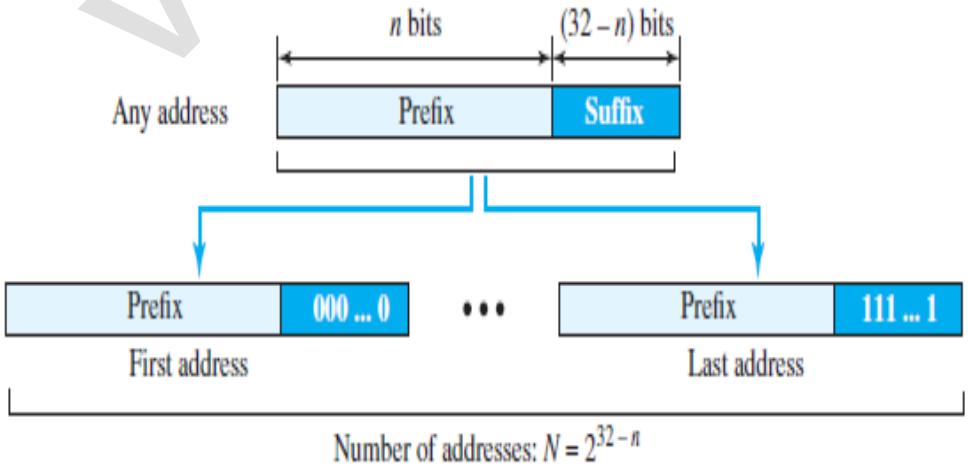


Fig 3.11: Information extraction in classless addressing

Example : A classless address is given as 167.199.170.82/27. We can find the above three pieces of information as follows. The number of addresses in the network is $2^{32-n} = 25 = 32$ addresses.

The first address can be found by keeping the first 27 bits and changing the rest of the bits to 0s.

Address: 167.199.170.82/27	10100111	11000111	10101010	01010010
First address: 167.199.170.64/27	10100111	11000111	10101010	01000000

The last address can be found by keeping the first 27 bits and changing the rest of the bits to 1s.

Address: 167.199.170.82/27	10100111	11000111	10101010	01011111
Last address: 167.199.170.95/27	10100111	11000111	10101010	01011111

Example 18.3

In classless addressing, an address cannot per se define the block the address belongs to. For example, the address 230.8.24.56 can belong to many blocks. Some of them are shown below with the value of the prefix associated with that block.

Prefix length:16	→	Block:	230.8.0.0	to	230.8.255.255
Prefix length:20	→	Block:	230.8.16.0	to	230.8.31.255
Prefix length:26	→	Block:	230.8.24.0	to	230.8.24.63
Prefix length:27	→	Block:	230.8.24.32	to	230.8.24.63
Prefix length:29	→	Block:	230.8.24.56	to	230.8.24.63
Prefix length:31	→	Block:	230.8.24.56	to	230.8.24.57

Network Address

The above examples show that, given any address, we can find all information about the block. The first address, the **network address,** is particularly important because it is used in routing a packet to its destination network.

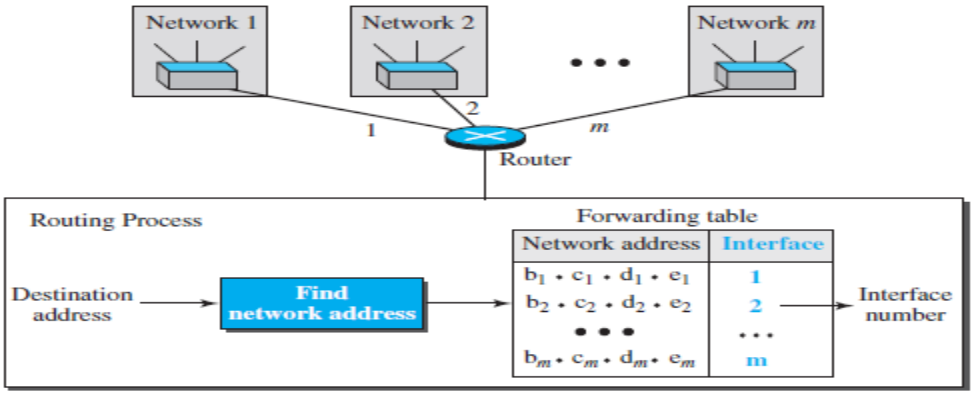


Fig 3.12: Network address

For the moment, let us assume that an internet is made of *m* networks and a router with *m* interfaces. When a packet arrives at the router from any source host, the router needs to know to which network the packet should be sent: from which interface the packet should be sent out.

7. Advantages of Classless Addressing:

- **Efficient use of IP addresses**: Avoids the waste of addresses that occurs with classful addressing.
- **Scalability**: Allows for more precise address allocation based on actual need, which is crucial as the number of devices connected to the Internet grows.
- **Better management by ISPs**: ISPs can assign blocks of different sizes to customers, ensuring everyone gets only the number of addresses they need.

Classless addressing (CIDR) was a crucial development in managing the limited IPv4 address space. By allowing variable-length prefixes and more flexible block sizes, it ensures a more efficient and scalable distribution of IP addresses across the Internet.

8. Block Allocation in Classless Addressing:

- **Global Authority (ICANN)**: The Internet Corporation for Assigned Names and Numbers (ICANN) is responsible for assigning IP address blocks.
    - ICANN does not assign addresses to individual users but allocates large blocks to ISPs or large organizations.
- **Responsibilities of ISPs**: ISPs receive large blocks of addresses from ICANN, which they further subdivide and assign to customers.

Restrictions for Block Allocation:

Two main restrictions ensure the proper functioning of CIDR:

1. **Number of Addresses (N)**:
    - The number of requested addresses must be a power of 2.
    - This is because N=232−nN = 2^{32 - n}N=232−n, and nnn must be an integer. If NNN is not a power of 2, nnn will not be a valid integer.
    - Example: If 1000 addresses are requested, since 1000 is not a power of 2, the next power of 2 (1024) addresses will be allocated.
2. **First Address Requirement**:
    - The first address of the block must be divisible by the number of addresses in the block.

- The first address should be represented as the **prefix** followed by (32−n) mzeros in binary.
- The decimal value of the first address must be divisible by the number of addresses.

Example (Block Allocation):

- An ISP requests 1000 addresses.
- Since 1000 is not a power of 2, 1024 addresses are allocated instead.
- The prefix length n is calculated as $n=32-\log_2 1024=22$ The ISP is assigned the block 18.14.12.0/22. The first address in this block (in decimal) is 302,910,464, which is divisible by 1024.

1. Overview of Subnetting:

- **Subnetting**: The process of dividing a larger block of IP addresses (assigned to an organization or ISP) into smaller blocks called **subnetworks** (or **subnets**).
- **Multi-level Hierarchy**: Subnetting can be performed at multiple levels. For example, a subnet can be further divided into **sub-subnets**, **sub-sub-subnets**, etc.
- Subnetting provides more granular control over IP address allocation and network design, allowing for efficient use of address space and easier management of smaller networks.

2. Designing Subnets:

To design subnetworks efficiently, the following steps must be followed:

1. **Number of Addresses in Subnet**:
   - The number of addresses in each subnetwork (**Nsub**) must be a **power of 2**.
   - This ensures that subnetting adheres to the same principles as block allocation, where the address block size must be a power of 2.
2. **Prefix Length for Each Subnetwork**:
   - The prefix length for each subnetwork (**nsub**) can be calculated using the formula: nsub=32−log⁡2(Nsub)nsub = 32 - \log_2 (Nsub)nsub=32−log2(Nsub)
   - This formula helps determine how many bits in the IP address are used for the network part, and how many are left for the hosts.
3. **Starting Address of Each Subnet**:
   - The **starting address** of each subnet must be divisible by the number of addresses in the subnet.

- This can be achieved by assigning addresses to **larger subnetworks first** and then working down to smaller subnets.

Example:

- Suppose an organization is granted 1024 addresses (/22).
- The organization can divide the range into subnetworks based on the number of required addresses per subnet.
  - For a subnet that needs 256 addresses, the prefix length would be $n_{sub}=32-\log_2 256=24$
  - The starting address of each subnet must be divisible by the size of the subnet (256 in this case).

3. Finding Information About Each Subnetwork:

After designing the subnetworks, key information about each subnet (such as the **first address** and **last address**) can be calculated using the same method applied to larger networks. The process is as follows:

1. **First Address**:
   - The first address of the subnet is found by setting the **host bits** (the last 32−n sub bits) to **0**.
2. **Last Address**:
   - The last address is found by setting the **host bits** to **1**.

Steps to Find First and Last Address:

- If a subnet has a prefix length of /24, the first address will have the last 8 bits set to 0, and the last address will have the last 8 bits set to 1.

Key Points to Remember:

- **Subnetting** allows an organization to efficiently use its allocated IP address block by dividing it into smaller subnetworks.
- Each **subnetwork** must contain a power-of-2 number of addresses.
- The **prefix length** for each subnetwork is calculated based on the number of addresses it requires.
- Proper **design of subnets** is crucial for efficient routing and network management.
- Once designed, the **first and last address** of each subnet can be determined using the prefix length.

This approach ensures optimal usage of IP addresses, simplifies routing, and improves the scalability of network infrastructure.

Address Aggregation

One of the advantages of the CIDR strategy is **address aggregation** (sometimes called *address summarization* or *route summarization*). When blocks of addresses are combined to create a larger block, routing can be done based on the prefix of the larger block. ICANN assigns a large block of addresses to an ISP. Each ISP in turn divides its assigned block into smaller sub blocks and grants the sub locks to its customers.

Example: An organization has been granted a block of 256 addresses, starting at **14.24.74.0/24**. The organization needs to divide this block into three subnets with the following requirements:

1. Subnet 1: 120 addresses
2. Subnet 2: 60 addresses
3. Subnet 3: 10 addresses

The solution involves allocating a number of addresses that are powers of 2, which are larger than or equal to the required number. The subnets are designed by starting with the largest requirement and working down to the smallest.

Steps and Subnet Design:

1. **Total Addresses**:
   - The block **14.24.74.0/24** has a total of: $2^{32-24}=256$ addresses
   - First address: **14.24.74.0**
   - Last address: **14.24.74.255**

Subnet A: Largest Subnet (120 addresses)

- **Required**: 120 addresses, but this is not a power of 2.
- **Allocated**: 128 addresses (since 128=is the next power of 2).
- **Prefix length**: $n_1=32-\log_2(128)=25$
- **First address**: **14.24.74.0/25**
- **Last address**:
  - The block starts at **14.24.74.0**, and since there are 128 addresses, the last address is **14.24.74.127/25**.

Subnet B: Second Largest Subnet (60 addresses)

- **Required**: 60 addresses, but this is not a power of 2.

- **Allocated**: 64 addresses (since $64=2^6$ is the next power of 2).
- **Prefix length**: $n_2=32-\log_2(64)=26$
- **First address**: The next available address after the first block is **14.24.74.128/26**.
- **Last address**:
    - The block starts at **14.24.74.128**, and since there are 64 addresses, the last address is **14.24.74.191/26**.

Subnet C: Smallest Subnet (10 addresses)

- **Required**: 10 addresses, but this is not a power of 2.
- **Allocated**: 16 addresses (since $16=2^4$ is the next power of 2).
- **Prefix length**: $n_3=32-\log_2(16)=28$
- **First address**: The next available address after the second block is **14.24.74.192/28**.
- **Last address**:
    - The block starts at **14.24.74.192**, and since there are 16 addresses, the last address is **14.24.74.207/28**.

Remaining Addresses:

- After assigning the three subnets, the total number of allocated addresses is: 128+64+16=208 addresses
- This leaves **48 addresses** unused.
- **First address of unused range**: **14.24.74.208**
- **Last address of unused range**: **14.24.74.255**
- **Prefix length**: To be determined, but the range can be used for future subnets or other purposes.

Summary of Subblocks:

- **Subnet A**:
    - **120 required, 128 allocated**
    - **First address**: 14.24.74.0/25
    - **Last address**: 14.24.74.127/25
- **Subnet B**:
    - **60 required, 64 allocated**
    - **First address**: 14.24.74.128/26
    - **Last address**: 14.24.74.191/26
- **Subnet C**:
    - **10 required, 16 allocated**
    - **First address**: 14.24.74.192/28
    - **Last address**: 14.24.74.207/28

- **Unused Block**:
  - **First address**: 14.24.74.208
  - **Last address**: 14.24.74.255
  - **48 addresses left for future use**.

This method ensures efficient utilization of IP addresses while allowing for potential future expansions.

Special IPv4 Addresses

IPv4 includes several special-purpose addresses used for specific functions in networking. Here are five important types:

1. This-host Address (0.0.0.0/32)

- **Address**: **0.0.0.0/32**
- **Purpose**: Used when a host doesn't know its own IP address and needs to communicate.
- **Use Case**: A host might use this address as a source address before it knows its IP (e.g., during DHCP requests).

2. Limited-broadcast Address (255.255.255.255/32)

- **Address**: **255.255.255.255/32**
- **Purpose**: Sends a datagram to all devices on the **local network**.
- **Restrictions**: Routers block these packets from being forwarded to other networks (they stay within the local network).
- **Use Case**: A host can broadcast to all devices within the same local network segment.

3. Loopback Address (127.0.0.0/8)

- **Address**: **127.0.0.0/8** (commonly, 127.0.0.1)
- **Purpose**: Used for testing and local communications within the same host. Packets sent to this address never leave the device.
- **Use Case**: Software testing (e.g., running a client-server application on the same host using 127.0.0.1 as the server address).

4. Private Addresses

- **Blocks**:
  - **10.0.0.0/8**
  - **172.16.0.0/12**
  - **192.168.0.0/16**
  - **169.254.0.0/16** (link-local addresses)
- **Purpose**: Used for internal/private networks. These addresses are **not routable** on the public Internet.
- **Use Case**: Networks within homes, businesses, or organizations. They require **Network Address Translation (NAT)** for accessing the Internet.

5. Multicast Addresses (224.0.0.0/4)

- **Address Block**: **224.0.0.0/4**
- **Purpose**: Used for **multicasting**, where a datagram is sent to multiple destinations simultaneously.
- **Use Case**: Applications like video conferencing, live streaming, or network services that need to send data to a group of devices.

Summary of Special IPv4 Addresses:

Type	Address/Block	Purpose
This-host	0.0.0.0/32	Host doesn't know its IP address
Limited-broadcast	255.255.255.255/32	Broadcast to all devices in the local network
Loopback	127.0.0.0/8	Testing local applications (remains on host)
Private Addresses	10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 169.254.0.0/16	Internal, non-routable IP addresses
Multicast	224.0.0.0/4	Send data to multiple devices simultaneously

These addresses serve distinct roles in networking, ensuring proper communication, testing, and private address use within internal networks.

Dynamic Host Configuration Protocol (DHCP)

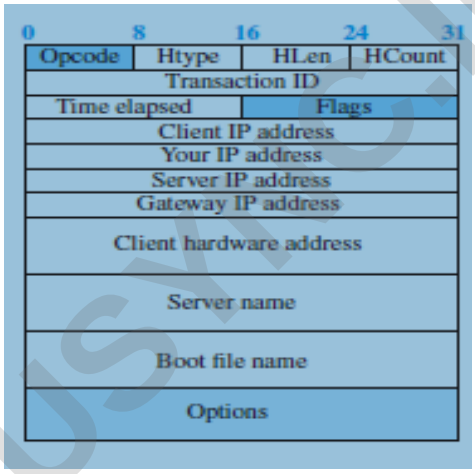
1. **What is DHCP?**
    - **Definition**: DHCP (Dynamic Host Configuration Protocol) automates the process of assigning IP addresses and other essential network configurations to devices within a network.
    - **Purpose**: Reduces the need for manual IP address setup by network administrators.
2. **IP Address Allocation Sources**
    - **Large Organizations and ISPs**: Receive IP address blocks directly from **ICANN** (Internet Corporation for Assigned Names and Numbers).
    - **Small Organizations**: Obtain IP address blocks from their **ISP**.
3. **Manual vs. Automatic Assignment**
    - **Manual Assignment**: Admins manually assign each device an IP address, which can be time-consuming.
    - **Automatic Assignment with DHCP**: DHCP automatically assigns IPs, improving efficiency and reducing errors.
4. **How DHCP Works**
    - **Layer and Model**: Operates at the **application layer** and follows a **client-server model**.
    - **Process**: Devices (clients) request IP configuration from a DHCP server, which assigns an IP and other necessary details.
5. **Plug-and-Play Protocol**
    - DHCP is often called "plug-and-play" because it allows devices to connect to the network and obtain configuration settings automatically, without user intervention.
6. **Permanent vs. Temporary IP Addressing**
    - **Permanent IP Assignment**: For devices like servers and routers needing a stable IP.
    - **Temporary IP Assignment**: For devices requiring short-term connectivity (e.g., laptops in hotels).
    - **Example of Efficiency**: ISPs with limited IPs can use DHCP to support more devices by reallocating addresses dynamically, such as serving 4,000 households with 1,000 IPs if only 25% are online simultaneously.
7. **Essential Information Provided by DHCP**
    - **IP Address**: Unique identifier for the device.
    - **Network Prefix (Subnet Mask)**: Defines the device's network range.
    - **Default Router (Gateway)**: The IP address of the router to connect to external networks.
    - **DNS Server Address**: Resolves domain names into IP addresses, allowing easy access to websites.

8. **Uses and Benefits of DHCP**
    - **Applications**: Commonly used in all types of networks (home, office, ISP) to manage IPs and provide necessary network details.
    - **Simplified Network Management**: DHCP reduces setup time, decreases the chance of configuration errors, and optimizes IP allocation.

DHCP is crucial for modern networks, making IP management simple and efficient. It provides both permanent and temporary IP assignments along with other essential network information, ensuring that devices can connect to the network with minimal setup.

DHCP Message Format

DHCP is a client-server protocol in which the client sends a request message and the server returns a response message



Opcode: Operation code, request (1) or reply (2)
Htype: Hardware type (Ethernet, ...)
HLen: Length of hardware address
HCount: Maximum number of hops the packet can travel
Transaction ID: An integer set by the client and repeated by the server
Time elapsed: The number of seconds since the client started to boot
Flags: First bit defines unicast (0) or multicast (1); other 15 bits not used
Your IP address: The client IP address sent by the server
Client IP address: Set to 0 if the client does not know it
Server IP address: A broadcast IP address if client does not know it
Gateway IP address: The address of default router
Server name: A 64-byte domain name of the server
Boot file name: A 128-byte file name holding extra information
Options: A 64-byte field with dual purpose described in text

The 64-byte option field has a dual purpose. It can carry either additional information or some specific vendor information.

The server uses a number, called a **magic cookie,** in the format of an IP address with the value of 99.130.83.99. When the client finishes reading the message, it looks for this magic cookie. If present, the next 60 bytes are options. An option is composed of three fields: a 1-byte tag field, a 1-byte length field, and a variable-length value field. There are several tag fields that are mostly used by vendors. If the tag field is 53, the value field defines one of the 8 message types shown in Figure 18.26. We show how these message types are used by DHCP.

Figure 18.26 *Option format*

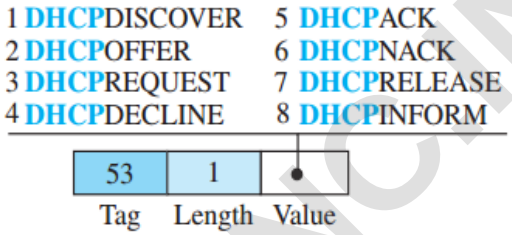
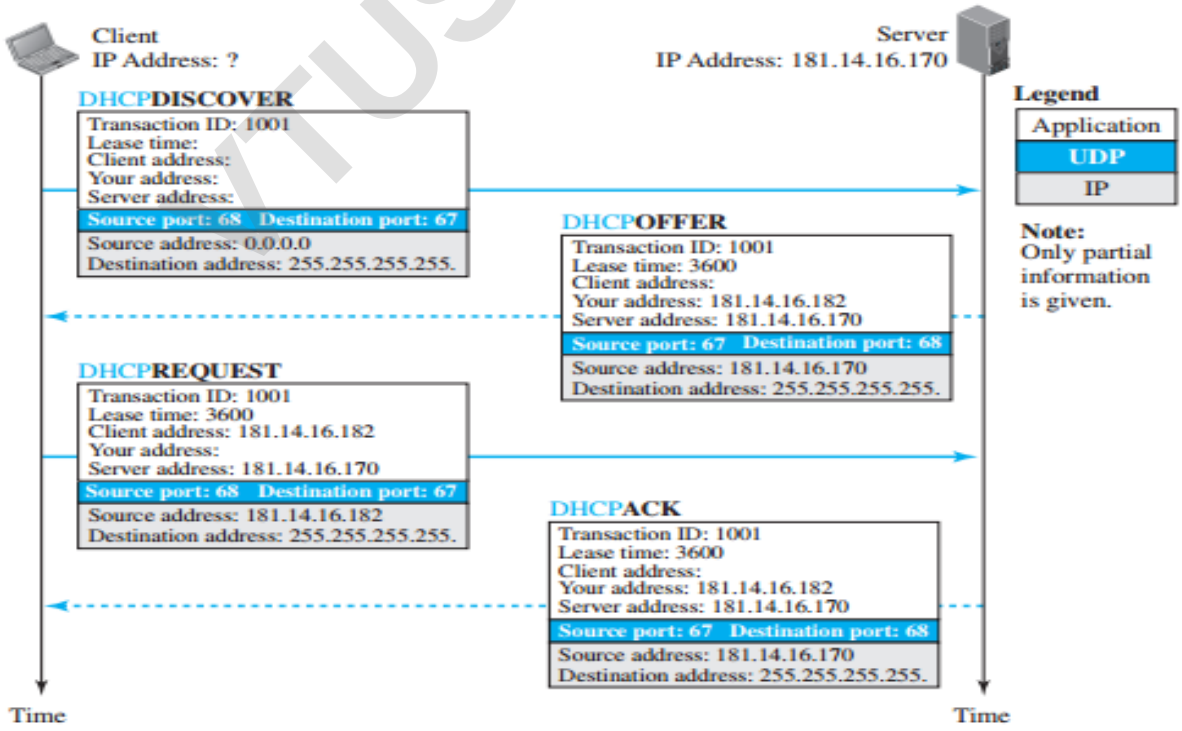


Figure 18.27 *Operation of DHCP*



DHCP Operation Steps

DHCP (Dynamic Host Configuration Protocol) enables devices to automatically acquire IP addresses and network configuration details for seamless network connectivity. Here's how the DHCP process unfolds:

1. **DHCPDISCOVER**:
    - The new host (client) sends a **DHCPDISCOVER** message to locate a DHCP server. This message includes only a **transaction ID**, a unique random identifier to track the session, as the host has no IP address or server information.
    - The message is encapsulated in a UDP datagram with **source port 68** and **destination port 67** (well-known DHCP ports).
    - The IP layer uses **source IP address 0.0.0.0** (since the host has no IP address) and **destination IP address 255.255.255.255** (broadcast address) to ensure it reaches any DHCP server on the network.
2. **DHCPOFFER**:
    - One or more DHCP servers respond with a **DHCPOFFER** message. This message includes:
        - **Your IP Address**: the offered IP address for the client.
        - **Server IP Address**: identifies the server making the offer.
        - **Lease Time**: duration for which the IP address is valid.
    - The DHCPOFFER message is encapsulated in a UDP datagram with **source port 67** and **destination port 68** (reverse of the DISCOVER message).
    - The IP layer sets the **server's IP address as the source** and **broadcasts the destination address** so other DHCP servers can see the offer and, if necessary, make better offers.
3. **DHCPREQUEST**:
    - The client selects the best offer and responds with a **DHCPREQUEST** message to the chosen server.
    - This message includes the chosen IP address and other relevant details and is sent with **source port 68** and **destination port 67**.
    - The IP source address is the **client's new IP address**, while the destination remains the **broadcast address**, informing other servers that their offers were not accepted.

4. **DHCPACK or DHCPNACK**:

    - The selected server confirms the IP assignment with a **DHCPACK** message, finalizing the process.
    - If the IP is no longer available, the server sends a **DHCPNACK** message, and the client restarts the process.

- The server broadcasts this message so other DHCP servers can see if the request was accepted or rejected.

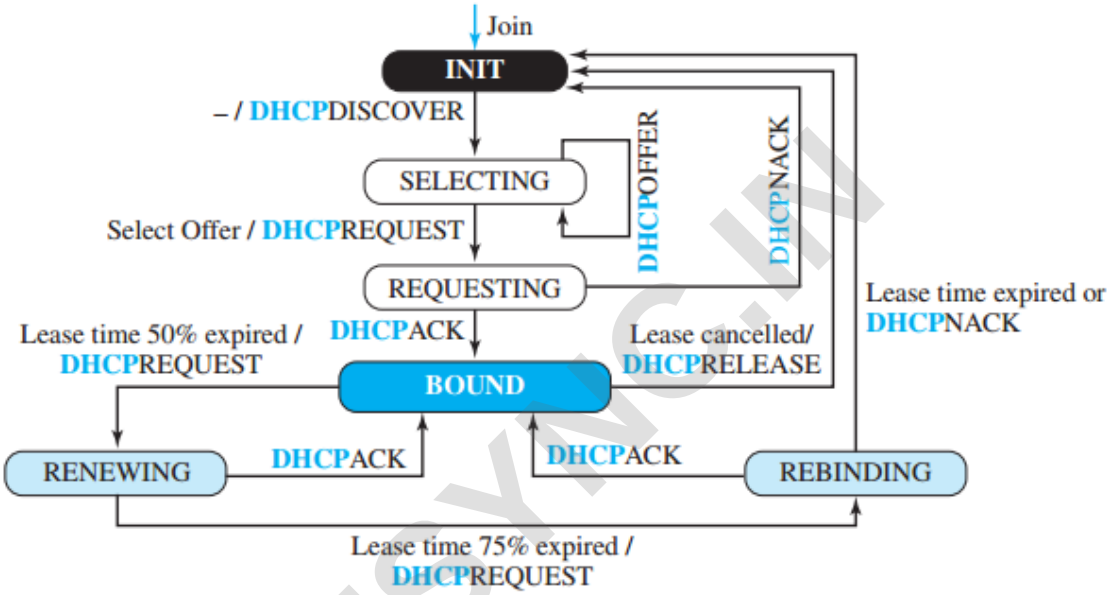
The DHCP protocol relies on two **well-known ports (68 and 67)** and includes features for file retrieval, error control, and address allocation management. Here's a breakdown of these aspects:

1. **Two Well-Known Ports (68 and 67)**:
    - **Port 68** is used by the DHCP client, while **Port 67** is used by the DHCP server.
    - The choice of a well-known client port (68) over a temporary (ephemeral) port helps avoid confusion if multiple applications are running. For example, if both a DHCP and a different service client (like DAYTIME) were waiting for responses on the same port, both could mistakenly receive each other's packets if the server broadcast a response. Using Port 68 isolates DHCP responses to the DHCP client.
    - **If multiple DHCP clients** are on the network after events like a power restoration, they can be identified by their unique **transaction ID**, avoiding interference.
2. **Using FTP for Additional Configuration**:
    - After a client receives a **DHCPACK** message with basic information, it may need more configuration details, such as the DNS server address.
    - The DHCP server includes a **pathname** in the DHCPACK message, guiding the client to a file where additional setup details are stored.
    - The client can then retrieve this file using **FTP (File Transfer Protocol)** or similar methods.
3. **Error Control in DHCP**:
    - DHCP operates over **UDP**, which lacks built-in reliability. To manage errors, DHCP implements:
        - **UDP checksum**: Ensures data integrity as UDP's checksum is mandatory for DHCP.
        - **Timers and Retransmission**: If the client doesn't receive a response, it retransmits its request. To avoid congestion, especially if multiple devices need to retransmit (e.g., after a power outage), DHCP uses a **random timer** for retransmissions.
4. **Transition States in DHCP**:
    - DHCP functions as a **state machine**, where the client transitions through several states depending on the messages exchanged. The main states include:
        - **INIT**: Client starts here when joining a network.
        - **SELECTING**: Client waits for DHCPOFFER messages.
        - **REQUESTING**: Client selects an offer and sends a DHCPREQUEST.
        - **BOUND**: Client is assigned an IP and can use the network.
        - **RENEWING/REBINDING**: Client attempts to extend its lease as it approaches expiration.

- **REBINDING**: If lease renewal fails, the client tries to contact any server to rebind its lease.

Figure 18.28 shows the transition diagram with the main states.

Figure 18.28 *FSM for the DHCP client*



DHCP Client State Transitions and Timers

The DHCP client progresses through several states as it acquires and manages its IP address lease. Key stages and timers include:

1. **INIT State (Initializing)**:
   - The client begins in the **INIT** state, where it needs an IP address.
   - It broadcasts a **DHCPDISCOVER** message to find a DHCP server.
2. **SELECTING State**:
   - Upon receiving **DHCPOFFER** messages, the client enters the **SELECTING** state.
   - It may receive multiple offers from different servers. The client picks the best offer.

3. **REQUESTING State**:
    - After selecting an offer, the client sends a **DHCPREQUEST** message to confirm its choice.
    - It then enters the **REQUESTING** state, waiting for the server's acknowledgment (ACK).
4. **BOUND State**:
    - When the client receives a **DHCPACK** message, it transitions to the **BOUND** state and begins using the assigned IP address.
5. **RENEWING State**:
    - When **50% of the lease time** has passed, the **renewal timer** triggers, moving the client to the **RENEWING** state.
    - In this state, the client attempts to renew its lease directly with the server.
    - If the server renews the lease, the client moves back to the **BOUND** state.
6. **REBINDING State**:
    - If the lease isn't renewed and **75% of the lease time** expires, the **rebinding timer** triggers, moving the client to the **REBINDING** state.
    - The client tries to renew the lease by broadcasting a request to any available DHCP server.
    - If successful, the client moves back to the **BOUND** state.
7. **INIT State (Again)**:
    - If the lease expires (without renewal) or no server responds, the client's **expiration timer** (set to 100% of the lease time) triggers, returning it to the **INIT** state to request a new IP address.

DHCP Timers

- **Renewal Timer**: 50% of lease time (triggers entry to RENEWING state).
- **Rebinding Timer**: 75% of lease time (triggers entry to REBINDING state).
- **Expiration Timer**: 100% of lease time (client returns to INIT to get a new IP address).

Network Address Resolution (NAT)

Network Address Translation (NAT) is a technology used to manage IP address shortages by enabling multiple devices on a local network to share a single or limited number of public IP addresses. Here's a breakdown of NAT's role and benefits:

1. **The Problem with Limited IP Ranges**:
    - ISPs provide small blocks of IP addresses to businesses or households. However, if the business or household needs more addresses (due to growth or increased

device usage), the ISP may not be able to accommodate because adjacent IP addresses may already be assigned to others.
    - Not every device on a local network needs simultaneous internet access. For example, a business with 20 computers may find that only a maximum of 4 need access at any given time.
2. **How NAT Solves This Problem**:
    - **Private IP Addresses for Internal Use**: NAT allows a local network to use private IP addresses (such as those in the 10.0.0.0, 172.16.0.0, or 192.168.0.0 blocks) for devices communicating within the local network.
    - **Public IP Addresses for Internet Access**: A small number of public IP addresses (e.g., those provided by the ISP) are shared by multiple devices to access the Internet through NAT.
3. **How NAT Works**:
    - The NAT-capable router has one public IP address for communication with the internet and multiple private IP addresses for devices on the local network.
    - When a device on the local network requests internet access, the NAT router translates the private IP address into the router's public IP address, creating a **mapping table** to keep track of active connections.
    - **Translation Table**: NAT maintains a table mapping each internal device's private IP and port to an external IP and port, allowing multiple internal devices to share the same public IP address while still keeping connections unique.
4. **Virtual Private Network (VPN) Compatibility**:
    - NAT supports VPNs by allowing secure and private communication within the local network, even when some devices communicate with the global internet.
5. **Types of NAT**:
    - **Static NAT**: Maps a single private IP to a single public IP.
    - **Dynamic NAT**: Maps a private IP to any available public IP from a pool.
    - **Port Address Translation (PAT)**: Also known as "overloading," it allows multiple devices to share one public IP by assigning different port numbers to each connection.
6. **Advantages of NAT**:
    - **Conserves IP Addresses**: NAT reduces the number of public IP addresses needed, which is useful with the limited IPv4 address pool.
    - **Security**: NAT hides internal IP addresses from external networks, adding a layer of security.

Overview of Address Translation

- **Functionality of NAT**:
    - NAT (Network Address Translation) is a method used to modify IP address information in packet headers while in transit across a routing device.

- It primarily allows multiple devices on a local network to share a single public IP address for Internet communication.

Outgoing Packets

- **Process**:
  - When a device within a private network sends a packet to the Internet, the packet is routed through the NAT router.
  - The NAT router modifies the packet:
    - It replaces the **source address** (the private IP address of the sending device) with its own **global NAT address** (the public IP address assigned by the ISP).
  - This allows the packet to be routed properly through the Internet.

Incoming Packets

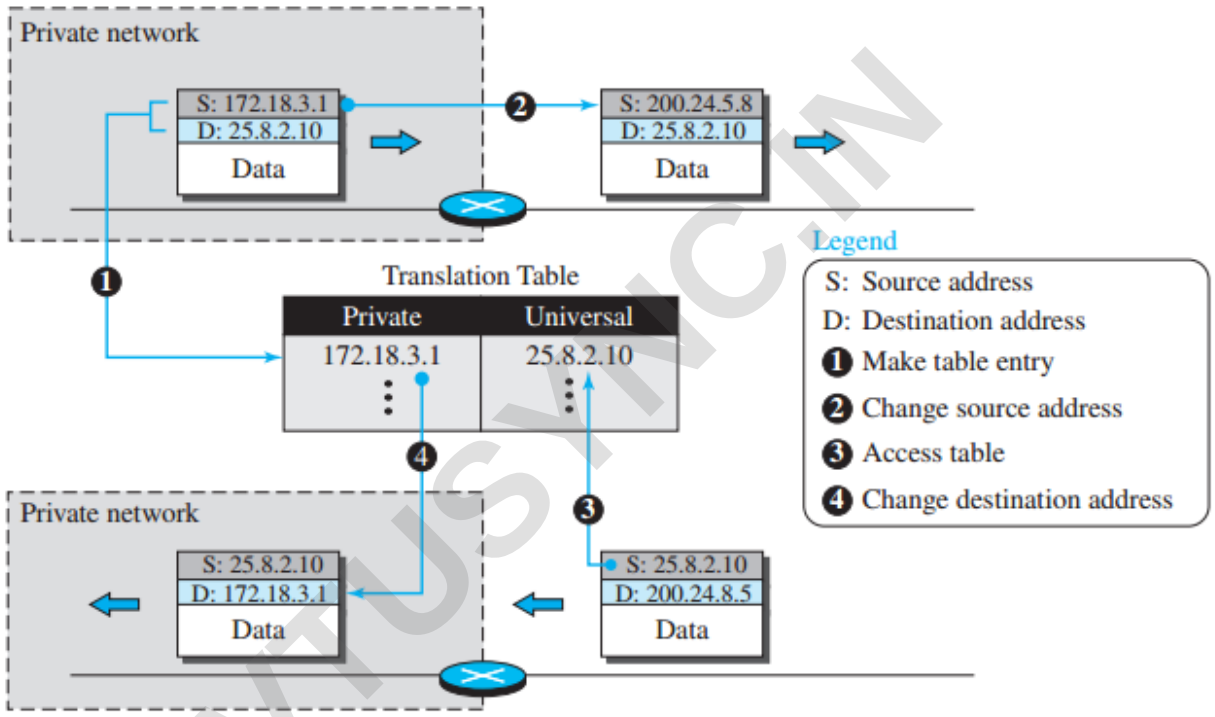
- **Process**:
  - Incoming packets from the Internet are also processed by the NAT router.
  - The NAT router changes the packet:
    - It replaces the **destination address** (the NAT router's global address) with the appropriate private IP address of the intended recipient within the local network.

Translation Table

- **Purpose**:
  - The translation table is essential for keeping track of the mappings between private and public addresses.
  - It solves the challenge of identifying which internal device should receive packets that are addressed to the public IP.
- **Structure**:
  - In its simplest form, the translation table contains two columns:
    - **Private Address**: The internal IP address of a device in the local network.
    - **External Address**: The corresponding public IP address that the device used when communicating with the Internet.
- **Operation**:
  - When a packet is sent out:
    - The NAT router records the **destination address** of the outgoing packet in the translation table along with the modified **source address**.
  - When a response packet arrives:

- The NAT router checks the **source address** of the incoming packet (the public address).
- It looks up this address in the translation table to find the corresponding **private address**.
- The router then modifies the packet's destination address to route it to the correct internal device.

Figure 18.31 *Translation*



Overview of NAT with a Pool of IP Addresses

- **Challenge with Single Global Address**:
  - Using only one global address in NAT limits the number of private-network hosts that can communicate with the same external host simultaneously.
  - This can create connectivity issues in scenarios where multiple devices need to access the same external resource.

Pool of Global Addresses

- **Functionality**:
  - To overcome the limitation of a single global address, a NAT router can utilize a **pool of global addresses**.
  - For example, instead of just one address (e.g., 200.24.5.8), a NAT router can manage multiple addresses (e.g., 200.24.5.8, 200.24.5.9, 200.24.5.10, and 200.24.5.11).
- **Benefits**:
  - With a pool of global addresses, multiple private-network hosts can communicate with the same external host simultaneously.
  - Each private address can pair with a unique global address, effectively allowing up to four private hosts to connect to the same external server at the same time.

Limitations of Using a Pool

- **Connection Restrictions**:
  - Even with a pool of addresses, the NAT router imposes some limitations:
    - No more than the number of global addresses in the pool can be used to connect to the same destination simultaneously (in this case, a maximum of four connections).
    - A private-network host cannot access multiple external server programs (e.g., HTTP and TELNET) simultaneously due to the shared global address.
    - Two private-network hosts cannot access the same external server program (e.g., both trying to access an HTTP server) at the same time if they require the same global address.

Using Both IP Addresses and Port Addresses

- **Many-to-Many Relationships**:
  - To facilitate a many-to-many relationship between private-network hosts and external server programs, more detailed information must be included in the NAT translation table.
  - The translation table can expand from two columns to five, incorporating:
    - **Private Address**: The internal IP address of the private-network host.
    - **Global Address**: The external IP address assigned by the NAT router.
    - **Source Port**: The port number used by the private-network host for the outgoing connection.
    - **Destination Port**: The port number used by the external server for the service being accessed.

        - **Transport Layer Protocol**: Identifies the protocol used (e.g., TCP, UDP) for the connection.
- **Example Scenario**:
    - If two hosts (172.18.3.1 and 172.18.3.2) need to access the same HTTP server (25.8.3.2):
        - The translation table can differentiate the two requests based on their source ports.
        - This eliminates ambiguity and allows both hosts to maintain separate connections to the external server without conflict.

Table 18.1 *Five-column translation table*

Private address	*Private port*	*External address*	*External port*	*Transport protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
⋮	⋮	⋮	⋮	⋮

1.5 IPv6 Datagram

Transitioning from IPv4 to IPv6 involves changing the packet format to accommodate a larger address space. The designers aimed to address IPv4's limitations while implementing new features.

Key features in IPv6

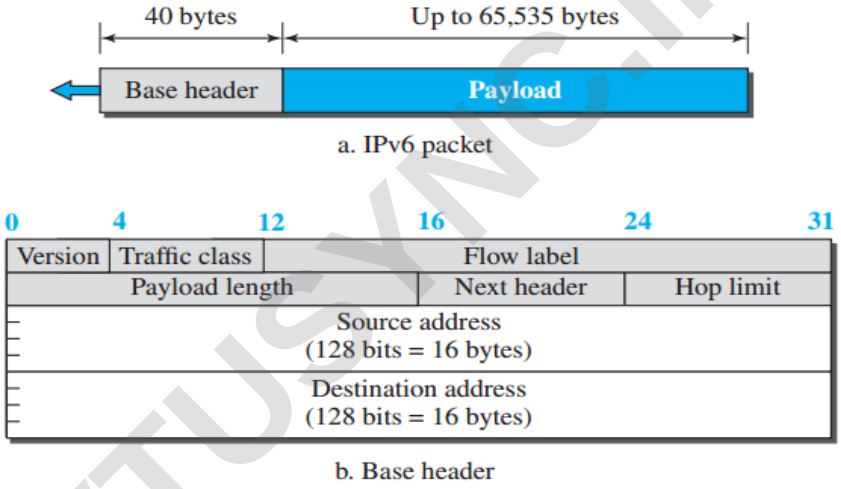
1. **Better Header Format**
    - **Structure**: IPv6 uses a new header format that separates options from the base header.
    - **Benefits**: This separation simplifies routing by allowing routers to process packets more quickly, as most options are not checked during routing.
2. **New Options**
    - **Functionality**: Introduces new options that enhance the protocol's capabilities, allowing for more advanced networking functions.
3. **Allowance for Extension**
    - **Future Readiness**: IPv6 is designed to be extensible, facilitating the integration of new technologies and applications as they emerge.
4. **Support for Resource Allocation**
    - **Traffic Class & Flow Label**: Replaces the IPv4 type-of-service field.

- **Special Handling**: These new fields enable the source to request prioritized handling for certain types of traffic, which is crucial for real-time applications like audio and video streaming.

5. **Enhanced Security**
   - **Encryption & Authentication**: Built-in options for encryption and authentication improve the confidentiality and integrity of data packets.
   - **Secure Communications**: These security features provide a more robust framework for safe data transmission.

Packet Format

Figure 22.6 *IPv6 datagram*



The IPv6 packet is shown in Figure 22.6. Each packet is composed of a base header followed by the payload. The base header occupies 40 bytes, whereas payload can be up to 65,535 bytes of information. The description of fields follows.

1. Version

- **Definition**: A 4-bit field that specifies the version number of the IP protocol.
- **Value for IPv6**: The version is set to 6.

2. Traffic Class

- **Field Size**: 8 bits.
- **Purpose**: Used to differentiate between different types of payloads with varying delivery requirements.

- **Replacement**: This field replaces the type-of-service field from IPv4.

3. Flow Label

- **Field Size**: 20 bits.
- **Purpose**: Designed to provide special handling for specific flows of data.
- **Details**: More information will be discussed later regarding its usage.

4. Payload Length

- **Field Size**: 2 bytes.
- **Purpose**: Specifies the length of the IP datagram's payload, excluding the header.
- **Note**: In IPv6, the base header length is fixed at 40 bytes, so only the payload length needs to be defined.

5. Next Header

- **Field Size**: 8 bits.
- **Purpose**: Indicates the type of the first extension header (if present) or the type of data following the base header.
- **Comparison**: Similar to the protocol field in IPv4; more details will be covered when discussing the payload.

6. Hop Limit

- **Field Size**: 8 bits.
- **Purpose**: Functions like the TTL (Time to Live) field in IPv4, limiting the number of hops a packet can make.

7. Source and Destination Addresses

- **Source Address**: A 16-byte (128-bit) address identifying the original source of the datagram.
- **Destination Address**: A 16-byte (128-bit) address identifying the intended recipient of the datagram.

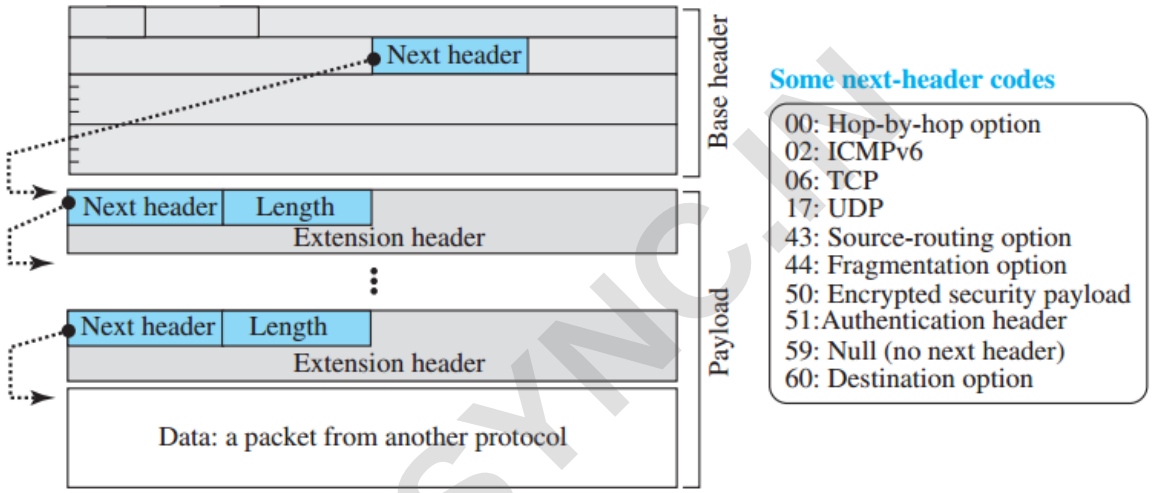
8. Payload

- **Comparison to IPv4**: The payload field in IPv6 has a different format and meaning compared to IPv4, accommodating new types of data and extension headers.

Payload Structure

- **Definition**: In IPv6, the payload consists of zero or more extension headers followed by the data from higher-layer protocols (e.g., UDP, TCP).
- **Extension Headers**: Unlike IPv4, where options are part of the main header, IPv6 treats options as separate extension headers.

Figure 22.7 *Payload in an IPv6 datagram*



Extension Headers

- **Flexibility**: The payload can contain multiple extension headers, allowing for customization based on specific requirements.
- **Mandatory Fields**:
  1. **Next Header**: Indicates the type of the next header (which can be another extension header or the data payload).
  2. **Length**: Specifies the length of the extension header in bytes.

Next Header Field

- **Field Values**: Each next header field value (code) defines the type of the next header:
  - **Types**: This includes options like hop-by-hop options, source routing options, etc.
  - **Final Header**: The last next header field in the chain specifies the protocol type of the payload data (e.g., UDP, TCP).

Concept of Flow and Priority in IPv6

The Internet Protocol (IP) was originally designed as a **connectionless protocol**, meaning each packet operates independently, with no connection between sender and receiver. Over time, there has been a trend toward using IP as a **connection-oriented protocol** to improve efficiency and guarantee certain types of service. In IPv4, the MPLS (Multiprotocol Label Switching) technology is used to simulate connection-oriented behavior by encapsulating packets with labels. However, IPv6 integrates a **flow label** directly into its datagram format, allowing IPv6 to achieve a similar effect natively.

Flow Label in IPv6

In IPv6, the flow label is a unique identifier for a sequence of packets, known as a **flow**, that share common characteristics. For example, packets within the same flow typically travel the same path, use the same resources, and may have similar security or processing requirements. The flow label allows routers to efficiently manage packets with different requirements. Routers that handle flow labels have a **flow label table** where each active flow label has a corresponding entry.

1. **Flow Label Table**:
    - The table contains entries with the specific services required by each flow.
    - When a packet arrives, the router consults this table based on the packet's flow label. This bypasses the need for repeated routing calculations, as the flow label table directly provides the necessary information, such as the **next hop address**.
2. **Flow Label for Priority and Real-Time Data**:
    - Flow labels improve efficiency by allowing routers to **quickly forward packets** based on pre-established flows.
    - They are also crucial for **real-time audio and video transmissions**, which require resources such as **high bandwidth** and **large buffers** to avoid delays. With IPv6 flow labels, these resources can be reserved in advance to meet the demands of real-time data.
3. **Supporting Protocols**:
    - While the flow label identifies the flow, it does not contain all information needed for the router's flow label table entries. Instead, other protocols or options like **hop-by-hop options**, **RTP (Real-Time Transport Protocol)**, and **RSVP (Resource Reservation Protocol)** are used to provide additional context and enable resource reservations.

Fragmentation and Reassembly

In IPv6, fragmentation and reassembly processes are handled differently than in IPv4, with a few key distinctions aimed at optimizing network performance.

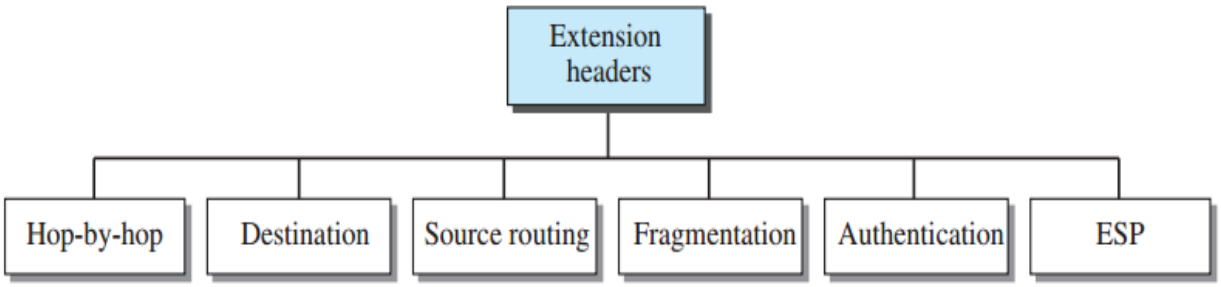
1. **Source-Only Fragmentation**:
    - **IPv6 fragmentation is handled solely by the source**, not by intermediate routers. This change reduces the processing load on routers, enhancing their efficiency and speed.
    - When a source needs to send a large packet, it is responsible for determining whether fragmentation is necessary. If the packet exceeds the Maximum Transmission Unit (MTU) of the path, the source fragments it before sending.
2. **Router Processing**:
    - **Routers do not fragment packets in IPv6**. Fragmenting packets at a router requires considerable processing, such as recalculating the values of fragmentation-related fields. By removing this requirement, IPv6 reduces router processing demands and allows packets to be processed more swiftly.
3. **Reassembly at the Destination**:
    - Fragmented packets are **reassembled only at the destination**. This approach reduces the complexity and workload on intermediate routers, which only forward packets without performing reassembly.
4. **MTU Checks and ICMPv6 Messages**:
    - Routers check the size of each packet against the MTU of the network they are about to forward it to. If a packet is too large, the router **drops the packet** and sends an **ICMPv6 "packet-too-big"** error message back to the source. This informs the source that it must fragment the packet or reduce its size to fit the MTU.

Extension Header

An IPv6 packet is made of a base header and some extension headers. The length of the base header is fixed at 40 bytes. To give more functionality to the IP datagram, the base header can be followed by up to six **extension headers.**

Many of these headers are options in IPv4. Six types of extension headers have been defined. These are hop-by-hop option, source routing, fragmentation, authentication, encrypted security payload, and destination option (see Figure 22.8).

Figure 22.8 *Extension header types*



Hop-by-Hop Option

The **hop-by-hop option** is designed for scenarios where information needs to be processed by every router along the datagram's path. This is typically used for control functions, debugging, and packet management. Some defined hop-by-hop options include:

- **Pad1**: A 1-byte option added for alignment purposes to ensure certain options start at specific bits within a 32-bit word.
- **PadN**: Similar to Pad1 but used when more than 1 byte of padding is needed for alignment.
- **Jumbo Payload**: Extends payload capacity beyond the typical IPv6 limit of 65,535 bytes, enabling larger datagrams if necessary.

Destination Option

The **destination option** is used when information should be read only by the destination node. Routers along the way do not access this information, keeping it private for the destination. This option has the same format as the hop-by-hop option and currently includes:

- **Pad1 and PadN**: Similar to the hop-by-hop options, these padding bytes are used for alignment in the destination option header.

Source Routing

The **source routing extension header** allows the sender to specify specific intermediate nodes or addresses that the packet must visit before reaching its destination. This combines the functionalities of IPv4's strict and loose source routing options, enabling flexible routing paths for specific applications.

Fragmentation

IPv6 fragmentation functions similarly to IPv4 but with key differences in where fragmentation occurs:

- Only the **original source** is allowed to fragment IPv6 packets, which reduces the load on routers.
- The source uses **Path MTU Discovery** to determine the smallest MTU on the packet's path and fragments the packet accordingly.
- If Path MTU Discovery is not available or used, the source fragments the packet to a size of 1280 bytes or smaller, as this is the minimum MTU for networks connected to the Internet.

Authentication

The **authentication extension header** ensures that a packet originates from a genuine sender and maintains data integrity. This prevents impersonation and verifies that data has not been altered en route, securing the communication between sender and receiver.

Encrypted Security Payload (ESP)

The **ESP extension header** provides confidentiality for data in transit by encrypting the payload, protecting it from eavesdropping. This extension helps ensure secure transmission over potentially untrusted networks. Each of these headers allows IPv6 to be flexible, secure, and suitable for a wide variety of applications while keeping the base header lightweight.

Comparison of IPv4 Options and IPv6 Extension Headers

IPv6 introduces a new approach to handling packet options by using **extension headers** instead of embedding options within the main header as in IPv4. Here's a comparison of specific options between IPv4 and IPv6:

1. **Padding Options**:
    - **IPv4**: Utilizes **no-operation** and **end-of-option** fields for alignment.
    - **IPv6**: Replaces these with **Pad1** (1 byte) and **PadN** (for 2+ bytes) to manage alignment within extension headers.
2. **Record Route Option**:
    - **IPv4**: Includes a **record route** option, but it was seldom used.
    - **IPv6**: This option is removed due to limited use.
3. **Timestamp Option**:
    - **IPv4**: Includes a **timestamp** option for tracking packet times along the route, but it was also rarely used.

- **IPv6**: This option is not implemented.

4. **Source Route Option**:
   - **IPv4**: Has a **source route option**, allowing the sender to specify the route.
   - **IPv6**: Implements this as the **source route extension header**, allowing more flexible source-routing control.
5. **Fragmentation**:
   - **IPv4**: Handles fragmentation within the base header, allowing both the source and routers to fragment packets.
   - **IPv6**: Moves fragmentation control to the **fragmentation extension header**, where only the original source handles fragmentation.
6. **Authentication**:
   - **IPv4**: Lacks built-in support for authentication.
   - **IPv6**: Adds a new **authentication extension header** to validate the sender's identity and ensure data integrity.
7. **Encrypted Security Payload (ESP)**:
   - **IPv4**: Has no built-in option for encryption.
   - **IPv6**: Introduces the **ESP extension header** for encrypting data, enhancing confidentiality and protecting against eavesdropping.

IPv6's extension headers streamline packet processing, enhance security, and add flexibility, while removing outdated or rarely-used options from IPv4. This modular approach makes IPv6 better suited to handle diverse network demands.