UNIT IV - NETWORK LAYER

• Introduction to Network Layer services • IPv4 datagram format • DHCP • ICMP • NAT • Routing Algorithms: Link state, Distance vector, Path vector • Routing Protocols: OSPF, RIP • IP addressing methods • Subnetting & Super netting • Protocols: IP: ARP, RARP, DHCP

1. NETWORK LAYER SERVICES

- The network layer in the TCP/IP protocol suite is responsible for the host-to-host delivery of datagrams.
- It provides services to the transport layer and receives services from the data-link layer.
- The network layer translates the logical addresses into physical addresses
- It determines the route from the source to the destination and also manages the traffic problems such as switching, routing and controls the congestion of data packets.
- The main role of the network layer is to move the packets from sending host to the receiving host.

Services provided by network layer are

PACKETIZING

- The first duty of the network layer is definitely packetizing.
- This means encapsulating the payload (data received from upper layer) in a network-layer packet at the source and decapsulating the payload from the network-layer packet at the destination.
- The network layer is responsible for delivery of packets from a sender to a receiver without changing or using the contents.

ROUTING AND FORWARDING

Routing

- The network layer is responsible for routing the packet from its source to the destination.
- The network layer is responsible for finding the best one among these possible routes.
- The network layer needs to have some specific strategies for defining the best route.
- Routing is the concept of applying strategies and running routing protocols to create the decision-making tables for each router.
- These tables are called as routing tables.

Forwarding

- Forwarding can be defined as the action applied by each router when a packet arrives at one of its interfaces.
- The decision-making table, a router normally uses for applying this action is called the forwarding table.
- When a router receives a packet from one of its attached networks, it needs to forward the packet to another attached network.

ERROR CONTROL

- The network layer in the Internet does not directly provide error control.
- It adds a checksum field to the datagram to control any corruption in the header, but not in the whole datagram.
- This checksum prevents any changes or corruptions in the header of the datagram.
- The Internet uses an auxiliary protocol called ICMP, that provides some kind of error control if the datagram is discarded or has some unknown information in the header.

FLOW CONTROL

- Flow control regulates the amount of data a source can send without overwhelming the receiver.
- The network layer in the Internet, however, does not directly provide any flow control.
- The datagrams are sent by the sender when they are ready, without any attention to the readiness of the receiver.
- Flow control is provided for most of the upper-layer protocols that use the services of the network layer, so another level of flow control makes the network layer more complicated and the whole system less efficient.

CONGESTION CONTROL

- Another issue in a network-layer protocol is congestion control.
- Congestion in the network layer is a situation in which too many datagrams are present in an area of the Internet.
- Congestion may occur if the number of datagrams sent by source computers is beyond the capacity of the network or routers.
- In this situation, some routers may drop some of the datagrams.

SECURITY

- Another issue related to communication at the network layer is security.
- To provide security for a connectionless network layer, we need to have another virtual level that changes the connectionless service to a connection-oriented service. This virtual layer is called as called IPSec (IP Security).

2.IPV4 ADDRESSES

- The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the Internet address or IP address.

- Internet Protocol version 4 (IPv4) is the fourth version in the development of the Internet Protocol (IP) and the first version of the protocol to be widely deployed.
- IPv4 is described in IETF publication in September 1981.
- The IP address is the address of the connection, not the host or the router. An IPv4 address is a 32-bit address that uniquely and universally defines the connection .
- If the device is moved to another network, the IP address may be changed.
- IPv4 addresses are unique in the sense that each address defines one, and only one, connection to the Internet.
- If a device has two connections to the Internet, via two networks, it has two IPv4 addresses.
- Pv4 addresses are universal in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

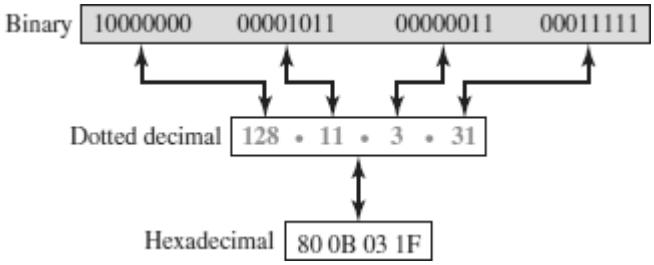
IPV4 ADDRESS SPACE

- IPv4 defines addresses has an address space.
- An address space is the total number of addresses used by the protocol.
- If a protocol uses *b* bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1).
- IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than four billion).
- 4 billion devices could be connected to the Internet.

IPV4 ADDRESS NOTATION

There are three common notations to show an IPv4 address:

(i) binary notation (base 2), (ii) dotted-decimal notation (base 256), and
(ii) hexadecimal notation (base 16).



In *binary notation,* an IPv4 address is displayed as 32 bits. To make the address more readable, one or more spaces are usually inserted between bytes (8 bits).

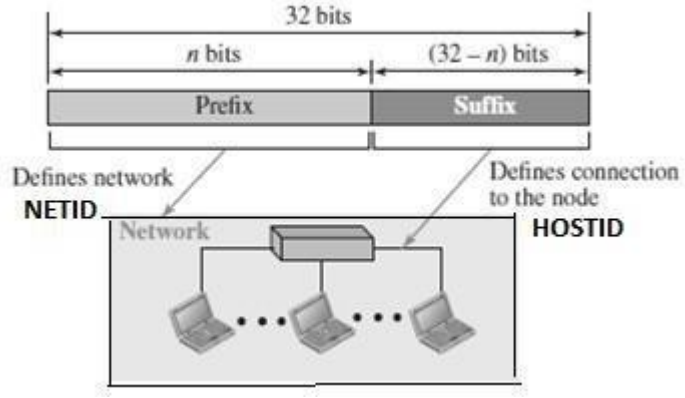
In *dotted-decimal notation,IPv4 addresses are* usually written in decimal form with a decimal point (dot) separating the bytes. Each number in the dotted-decimal notation is between 0 and 255.

In hexadecimal notation, each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits. This notation is often used in network programming.

HIERARCHY IN IPV4 ADDRESSING

- In any communication network that involves delivery, the addressing system is hierarchical.
- A 32-bit IPv4 address is also hierarchical, but divided only into two parts.

- The first part of the address, called the *prefix*, defines the network(Net ID); the second part of the address, called the *suffix*, defines the node (Host ID).
- The prefix length is *n* bits and the suffix length is (32- □*n*) bits.



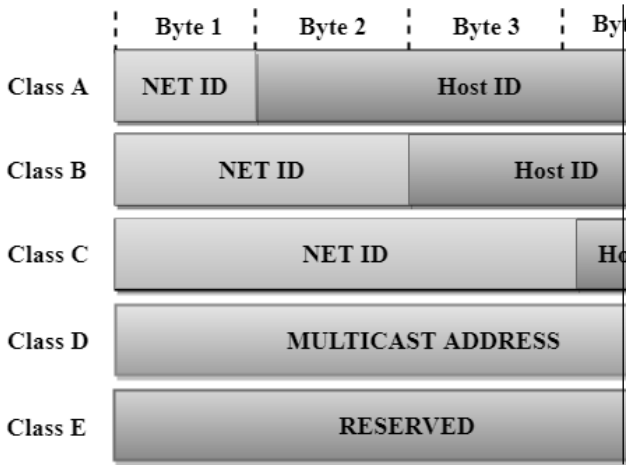
- A prefix can be fixed length or variable length.
- The network identifier in the IPv4 was first designed as a fixed-length prefix.
- This scheme is referred to as classful addressing.
- The new scheme, which is referred to as classless addressing, uses a variable-length network prefix.

CATEGORIES OF IPV4 ADDRESSING

- There are two broad categories of IPv4 Addressing techniques.
- They are
  - Classful Addressing
  - Classless Addressing

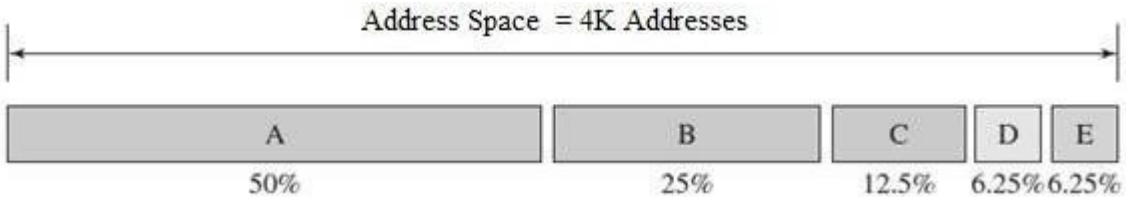
CLASSFUL ADDRESSING

- An IPv4 address is 32-bit long(4 bytes).
- An IPv4 address is divided into sub-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

Classful Network Architecture



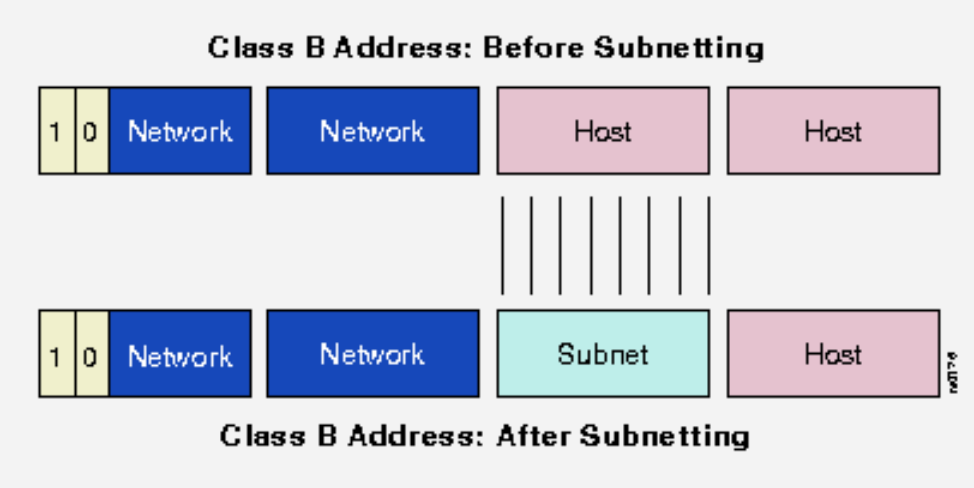
Class	Higher bits	NET ID bits	HOST ID bits	No. of Networks	No.of hosts per network	Range
A	0	8	24	2^7	2^{24}	0.0.0.0 **to** 127.255.255.255
B	10	16	16	2^{14}	2^{16}	128.0.0.0 **to** 191.255.255.255
C	110	24	8	2^{21}	2^8	192.0.0.0 **to** 223.255.255.255
D	1110	Not Defined	Not Defined	Not Defined	Not Defined	224.0.0.0 **to** 239.255.255.255
E	1111	Not Defined	Not Defined	Not Defined	Not Defined	240.0.0.0 **to** 255.255.255.255

Class A

- In Class A, an IP address is assigned to those networks that contain a large number of hosts.
- The network ID is 8 bits long.
- The host ID is 24 bits long.
- In Class A, the first bit in higher order bits of the first octet is always set to 0 and the remaining 7 bits determine the network ID.
- The 24 bits determine the host ID in any network.
- The total number of networks in Class A = 2^7 = 128 network address
- The total number of hosts in Class A = 2^{24} - 2 = 16,777,214 host address

Class B

- In Class B, an IP address is assigned to those networks that range from small-sized to large-sized networks.
- The Network ID is 16 bits long.
- The Host ID is 16 bits long.
- In Class B, the higher order bits of the first octet is always set to 10, and the remaining14 bits determine the network ID.
- The other 16 bits determine the Host ID.
- The total number of networks in Class B = 2^{14} = 16384 network address
- The total number of hosts in Class B = 2^{16} - 2 = 65534 host address



Class C

- In Class C, an IP address is assigned to only small-sized networks.
- The Network ID is 24 bits long.
- The host ID is 8 bits long.
- In Class C, the higher order bits of the first octet is always set to 110, and the remaining 21 bits determine the network ID.
- The 8 bits of the host ID determine the host in a network.
- The total number of networks = 2^{21} = 2097152 network address
- The total number of hosts = 2^{8} - 2 = 254 host address

Class D

- In Class D, an IP address is reserved for multicast addresses.
- It does not possess subnetting.
- The higher order bits of the first octet is always set to 1110, and the remaining bits determines the host ID in any network.

Class E

- In Class E, an IP address is used for the future use or for the research and development purposes.
- It does not possess any subnetting.
- The higher order bits of the first octet is always set to 1111, and the remaining

bits determines the host ID in any network.

Address Depletion in Classful Addressing

- The reason that classful addressing has become obsolete is address depletion.
- Since the addresses were not distributed properly, the Internet was faced with the problem of the addresses being rapidly used up.
- This results in no more addresses available for organizations and individuals that needed to be connected to the Internet.
- To understand the problem, let us think about class A.
- This class can be assigned to only 128 organizations in the world, but each organization needs to have a single network with 16,777,216 nodes .
- Since there may be only a few organizations that are this large, most of the addresses in this class were wasted (unused).
- Class B addresses were designed for midsize organizations, but many of the addresses in this class also remained unused.
- Class C addresses have a completely different flaw in design. The number of addresses that can be used in each network (256) was so small that most companies were not comfortable using a block in this address class.
- Class E addresses were almost never used, wasting the whole class.

Advantage of Classful Addressing

- Although classful addressing had several problems and became obsolete, it had one advantage.
- Given an address, we can easily find the class of the address and, since the prefix length for each class is fixed, we can find the prefix length immediately.
- In other words, the prefix length in classful addressing is inherent in the address; no extra information is needed to extract the prefix and the suffix.

Subnetting and Supernetting

- To alleviate address depletion, two strategies were proposed and implemented:
  (i) Subnetting and (ii) Supernetting.

Subnetting

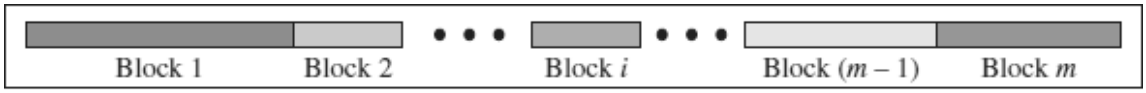
- In subnetting, a class A or class B block is divided into several subnets.
- Each subnet has a larger prefix length than the original network.
- For example, if a network in class A is divided into four subnets, each subnet has a prefix of $n_{sub} = 10$.
- At the same time, if all of the addresses in a network are not used, subnetting allows the addresses to be divided among several organizations.

CLASSLESS ADDRESSING

- In 1996, the Internet authorities announced a new architecture called **classless addressing.**
- In classless addressing, variable-length blocks are used that belong to no classes.
- We can have a block of 1 address, 2 addresses, 4 addresses, 128 addresses, and

so on.

- In classless addressing, the whole address space is divided into variable length blocks.
- The prefix in an address defines the block (network); the suffix defines the node (device).
- Theoretically, we can have a block of 2^0, 2^1, 2^2, ... 2^{32} addresses.
- The number of addresses in a block needs to be a power of 2. An organization can be granted one block of addresses.

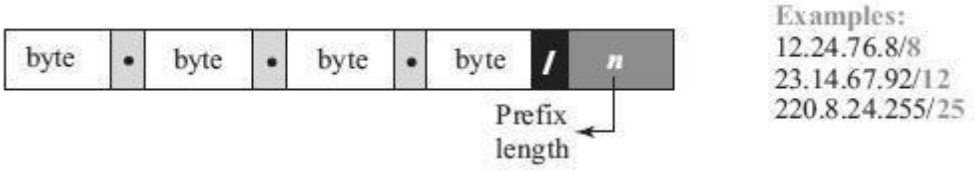


Address space

- The prefix length in classless addressing is variable.
- We can have a prefix length that ranges from 0 to 32.
- The size of the network is inversely proportional to the length of the prefix.
- A small prefix means a larger network; a large prefix means a smaller network.
- The idea of classless addressing can be easily applied to classful addressing.
- An address in class A can be thought of as a classless address in which the prefix length is 8.
- An address in class B can be thought of as a classless address in which the prefix is 16, and so on. In other words, classful addressing is a special case of classless addressing.

Notation used in Classless Addressing

- The notation used in classless addressing is informally referred to as *slash notation* and formally as ***classless interdomain routing*** or ***CIDR.***



- For example , 192.168.100.14 **/24** represents the IP address 192.168.100.14 and, its subnet mask 255.255.255.0, which has 24 leading 1-bits.

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 subblocks and grants the subblocks to its customers.

Supernetting

Supernetting is a technique used to combine multiple IP networks (subnets) into a larger network, often called a **supernet**. It helps in reducing the number of entries in routing tables by consolidating several smaller network addresses into a single address range. Supernetting is the opposite of **subnetting**, where a large network is divided into

smaller subnets.
Supernetting is mainly used in scenarios like **Classless Inter-Domain Routing (CIDR)** to create larger address blocks and efficiently manage IP address allocation, reducing the need for multiple routes in routers and enhancing routing efficiency.

Key Concepts in Supernetting

1. **CIDR Notation**:
    - In CIDR, IP addresses are written with a suffix that specifies the number of bits used for the network portion (e.g., 192.168.0.0/22).
    - In supernetting, the CIDR suffix decreases to encompass more addresses, combining multiple network blocks into a single block.
2. **Supernet Mask**:
    - Supernetting uses a mask that is shorter than the default subnet mask, allowing for a broader range of IP addresses.
    - For instance, a default Class C network (/24) might use a /22 mask to combine multiple Class C networks.
3. **Requirements for Supernetting**:
    - Networks being combined must be **contiguous** (consecutive IP ranges).
    - The network addresses should have **binary alignment** for aggregation.

How Supernetting Works

When supernetting, multiple smaller IP blocks (networks) are combined to form a larger block. Supernetting involves reducing the number of bits in the subnet mask, which results in an increased address range. For example, four contiguous Class C networks can be combined into a single supernet by reducing the subnet mask to /22 (from /24 for each individual Class C network).

1. **Identify the IP Ranges**:
    - Ensure that the networks are contiguous. Supernetting cannot work if the IP blocks are scattered or non-sequential.
2. **Convert IP Addresses to Binary**:
    - Convert the starting and ending IPs of the range into binary to determine the shared bits.
3. **Determine the Supernet Mask**:
    - Identify the number of leading bits that remain the same across all networks.
    - The length of this common prefix determines the new supernet mask.
4. **Calculate the Supernet Address**:
    - The network address for the supernet is determined by setting the remaining host bits (non-common bits) to zero.

Example of Supernetting: Combining Four Class C Networks

Let's walk through a supernetting example to combine four contiguous Class C networks into a single supernet.

Objective: Combine the following four Class C networks into a supernet:

- 192.168.4.0/24
- 192.168.5.0/24
- 192.168.6.0/24
- 192.168.7.0/24

Steps:

1. **Convert the Network Addresses to Binary**:

Convert each network address to binary (only the last octet differs in this case):

- 192.168.4.0 = 11000000.10101000.00000100.00000000
- 192.168.5.0 = 11000000.10101000.00000101.00000000
- 192.168.6.0 = 11000000.10101000.00000110.00000000
- 192.168.7.0 = 11000000.10101000.00000111.00000000

2. **Identify the Common Prefix**:

Examine the binary representations to find the common prefix:
Copy code
11000000.10101000.000001xx.00000000

- The common bits extend up to the first 22 bits (from 192.168.4.0 to 192.168.7.0).
- So, the common prefix for this supernet is 192.168.4.0/22.

3. **Calculate the Supernet Mask**:

Since the first 22 bits are common, the **supernet mask** becomes /22. This mask covers four Class C networks.

- **Binary of /22 Mask**: 11111111.11111111.11111100.00000000
- **Decimal Representation**: 255.255.252.0

4. **Calculate the Range of IPs in the Supernet**:

With a /22 mask, the IP address range of the supernet 192.168.4.0/22 will span from:

- **Starting IP**: 192.168.4.0
- **Ending IP**: 192.168.7.255

This range covers all addresses from 192.168.4.0 to 192.168.7.255, effectively combining the four original networks into one.

5. **Result**:

The four Class C networks have now been combined into a single **supernet**:

- **Supernet Address**: 192.168.4.0
- **Subnet Mask**: 255.255.252.0 or /22 in CIDR notation.
- **IP Range**: 192.168.4.0 to 192.168.7.255

Verification with an Example IP

To verify if an IP falls within this supernet, apply the /22 subnet mask and check if it lies within the range:

Example IP: 192.168.6.1

1. **Binary Representation** of 192.168.6.1:
   - 11000000.10101000.00000110.00000001
2. **Apply the /22 Mask**:
   - Masked with 255.255.252.0, the network portion of 192.168.6.1 matches the supernet 192.168.4.0/22, confirming that 192.168.6.1 is within the supernet range.

Benefits of Supernetting

1. **Reduced Routing Table Entries**:
   - Instead of listing each individual network, routers can use a single entry for the supernet, improving efficiency and speed in routing decisions.
2. **Efficient IP Address Management**:
   - Supernetting allows better utilization of IP addresses by allocating larger, aggregated address blocks.
3. **Scalability**:
   - Supernetting helps manage larger network architectures by reducing the complexity of IP addressing schemes.

Summary

Supernetting is a powerful technique in CIDR that aggregates multiple contiguous subnets into a larger address block, reducing the complexity and size of routing tables. By identifying a common prefix and using a shorter subnet mask, supernetting enables network administrators to manage IP addresses and routes more efficiently.

Special Addresses in IPv4

- There are five special addresses that are used for special purposes: *this-host* address, *limited-broadcast*address, *loopback* address, *private* addresses, and *multicast* addresses.

This-host Address

- ✓ The only address in the block **0.0.0.0**/**32** is called the *this-host* address.
- ✓ It is used whenever a host needs to send an IP datagram but it does not know its own address to use as the source address.

Limited-broadcast Address

- ✓ The only address in the block **255.255.255.255**/**32** is called the *limited-broadcast* address.
- ✓ It is used whenever a router or a host needs to send a datagram to all devices in a network.
- ✓ The routers in the network, however, block the packet having this address as the destination;the packet cannot travel outside the network.

Loopback Address

- ✓ The block **127.0.0.0**/**8** is called the *loopback* address.
- ✓ A packet with one of the addresses in this block as the destination address never leaves the host; it will remain in the host.

Private Addresses

- ✓ Four blocks are assigned as private addresses: 10.0.0.0/**8**, 172.16.0.0/**12**, 192.168.0.0/**16**, and 169.254.0.0/**16**.

Multicast Addresses

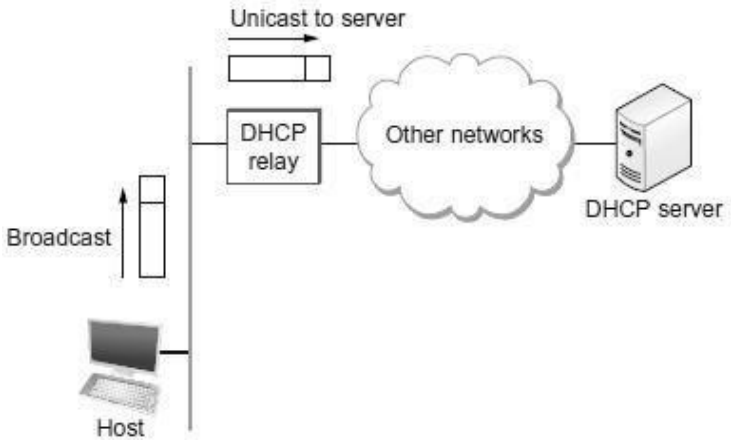
- ✓ The block 224.0.0.0/**4** is reserved for multicast addresses.

---

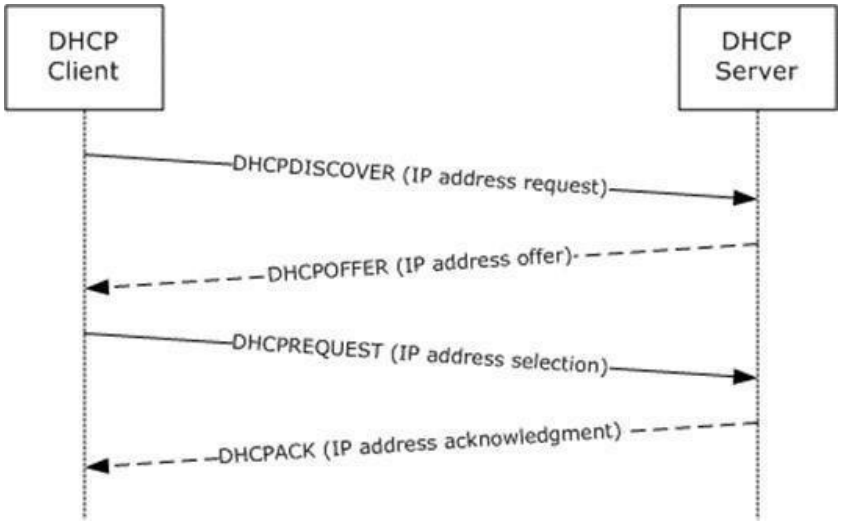
3. DHCP – DYNAMIC HOST CONFIGURATION PROTOCOL

- The dynamic host configuration protocol is used to simplify the installation and maintenance of networked computers.
- DHCP is derived from an earlier protocol called BOOTP.
- Ethernet addresses are configured into network by manufacturer and they are unique.
- IP addresses must be unique on a given internetwork but also must reflect the structure of the internetwork
- Most host Operating Systems provide a way to manually configure the IP information for the host
- **Drawbacks of manual configuration :**
    1. A lot of work to configure all the hosts in a large network
    2. Configuration process is error-prune

- It is necessary to ensure that every host gets the correct network number and that no two hosts receive the same IP address.
- For these reasons, automated configuration methods are required.
- The primary method uses a protocol known as the *Dynamic Host Configuration Protocol* (DHCP).
- The main goal of DHCP is to minimize the amount of manual configuration required for a host.
- If a new computer is connected to a network, DHCP can provide it with all the necessary information for full system integration into the network.
- DHCP is based on a client/server model.
- DHCP clients send a request to a DHCP server to which the server responds with an IP address
- DHCP server is responsible for providing configuration information to hosts.
- There is at least one DHCP server for an administrative domain.
- The DHCP server can function just as a centralized repository for host configuration information.
- The DHCP server maintains a pool of available addresses that it hands out to hosts on demand.



- A newly booted or attached host sends a DHCPDISCOVER message to a special IP address (255.255.255.255., which is an IP broadcast address.
- This means it will be received by all hosts and routers on that network.
- DHCP uses the concept of a *relay agent.* There is at least one relay agent on each network.
- DHCP relay agent is configured with the IP address of the DHCP server.
- When a relay agent receives a DHCPDISCOVER message, it unicasts it to the DHCP server and awaits the response, which it will then send back to the requesting client.



DORA in DHCP

Dynamic Host Configuration Protocol (DHCP) is a network management protocol used to automate the process of configuring devices on IP networks, allowing them to communicate effectively. One of the key processes in DHCP is known as **DORA**, which stands for **Discover, Offer, Request, and Acknowledge**. This is the sequence of messages exchanged between a client and a server during the DHCP leasing process.

1. Discover

- **Purpose**: The client (usually a computer or any network device) sends out a broadcast message to discover available DHCP servers on the network.
- **Message Type**: DHCP Discover (DHCPDISCOVER)
- **Process**:
  - When a device connects to a network, it initializes its network interface and sends a DHCP Discover message to the broadcast address (255.255.255.255).
  - This message contains the client's MAC address and a transaction ID to identify the session.

2. Offer

- **Purpose**: One or more DHCP servers respond to the client with an offer of configuration parameters.
- **Message Type**: DHCP Offer (DHCPOFFER)
- **Process**:
  - Upon receiving the Discover message, a DHCP server responds with a DHCPOFFER message, which includes:
    - An available IP address.
    - Subnet mask.
    - Default gateway.
    - Lease time (duration for which the IP address is valid).
    - Any additional configuration options (e.g., DNS servers).
  - This response is sent as a unicast (to the client's MAC address) or as a broadcast if the server does not know the client's address.

3. Request

- **Purpose**: The client selects one of the offers received and requests the offered parameters from the chosen DHCP server.
- **Message Type**: DHCP Request (DHCPREQUEST)
- **Process**:
  - The client sends a DHCPREQUEST message to the selected DHCP server, indicating its acceptance of the offered IP address.

- If multiple offers were received, the client can specify which offer it is accepting by including the server identifier (the IP address of the DHCP server).
- This message may also be broadcast to inform all other DHCP servers that the client has accepted one offer, so they can withdraw their offers.

4. Acknowledge

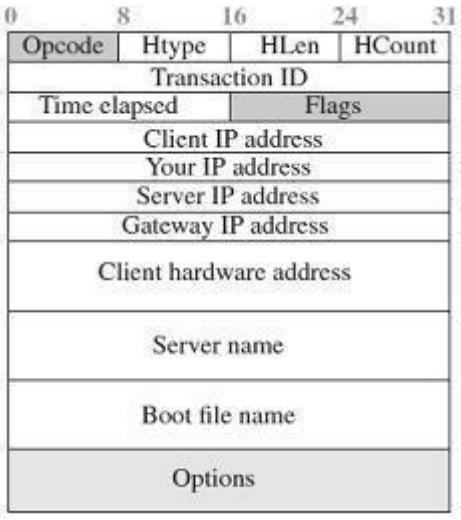
- **Purpose**: The DHCP server acknowledges the client's request and finalizes the lease of the IP address.
- **Message Type**: DHCP Acknowledgment (DHCPACK)
- **Process**:
  - Upon receiving the DHCPREQUEST message, the DHCP server confirms the allocation of the requested IP address by sending a DHCPACK message back to the client.
  - This message contains the final configuration parameters and confirms that the client can use the assigned IP address.
  - If the server cannot fulfill the request (e.g., the IP address is no longer available), it sends a DHCP NAK (Negative Acknowledgment) message instead.

Summary of DORA Process

Step	Message Type	Description
Discover	DHCPDISCOVER	Client broadcasts a discover message to find DHCP servers.
Offer	DHCPOFFER	DHCP server responds with an offer of an IP address and other configuration options.
Request	DHCPREQUEST	Client requests the offered IP address from the selected DHCP server.
Acknowledge	DHCPACK	DHCP server acknowledges the request and finalizes the lease by sending the DHCPACK message.

DHCP Message Format

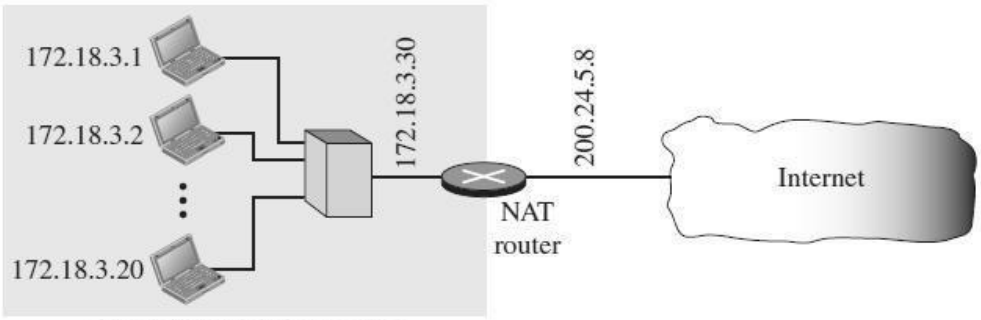
- A DHCP packet is actually sent using a protocol called the *User Datagram Protocol* (UDP).



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
Client IP address: Set to 0 if the client does not know it
Your IP address: The client IP address sent by the server
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

4. NETWORK ADDRESS TRANSLATION (NAT)

- A technology that can provide the mapping between the private and universal (external)addresses, and at the same time support virtual private networks is called as **Network Address Translation (NAT).**
- The technology allows a site to use a set of private addresses for internal communication and a set of global Internet addresses (at least one) for communication with the rest of the world.
- The site must have only one connection to the global Internet through a NAT-capable router that runs NAT software.



Site using private addresses

- The private network uses private addresses.
- The router that connects the network to the global address uses one private address and one global address.
- The private network is invisible to the rest of the Internet; the rest of the Internet sees only the NAT router with the address 200.24.5.8.

Types of NAT

- Two types of NAT exists .
  - (a) One-to-one translation of IP addresses
  - (b) One-to-many translation of IP addresses

Address Translation

- All of the outgoing packets go through the NAT router, which replaces the source address in the packet with the global NAT address.
- All incoming packets also pass through the NAT router, which replaces the destination address in the packet (the NAT router global address) with the appropriate private address.

Translation Table

- There may be tens or hundreds of private IP addresses, each belonging to one specific host.
- The problem arises when we want to translate the source address to an external address. This is solved if the NAT router has a translation table.

Translation table with two columns

✓ A translation table has only two columns: the private address and the external address (destination address of the packet).

✓ When the router translates the source address of the outgoing packet, it also makes note of the destination address—where the packet is going.

✓ When the response comes back from the destination, the router uses the source address of the packet (as the external address) to find the private address of the packet.

Two-column translation table

Private address	*External address*
172.18.3.1	25.8.3.2
172.18.3.2	25.8.3.2
⋮	⋮

Translation table with five columns

✓ To allow a many-to-many relationship between private-network hosts and external server programs, we need more information in the translation table.

✓ If the translation table has five columns, instead of two, that include the source and destination port addresses and the transport-layer protocol, the ambiguity is eliminated.

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
⋮	⋮	⋮	⋮	⋮

5. FORWARDING OF IP PACKETS

- Forwarding means to deliver the packet to the next hop (which can be the final destination or the intermediate connecting device).
- Although IP protocol was originally designed as a connectionless protocol, today the tendency is to use IP as a connection-oriented protocol based on the label attached to an IP datagram .
- When ***IP is used as a connectionless protocol***, forwarding is based on the ***destination address*** of the IP datagram.
- When the ***IP is used as a connection-oriented protocol***, forwarding is based on the ***label*** attached to an IP datagram.

FORWARDING BASED ON DESTINATION ADDRESS

➢ This is a traditional approach.

➢ In this case, forwarding requires a host or a router to have a forwarding table.

➢ When a host has a packet to send or when a router has received a packet to be forwarded, it looks at this table to find the next hop to deliver the packet to.

➢ The main points in forwarding of IP Packets(datagram) are the following:

- Every IP Packets contains the IP address of the destination host.
- The network part of an IP address uniquely identifies a single physical network that is part of the larger Internet.
- All hosts and routers that share the same network part of their address are connected to the same physical network and can thus communicate with each other by sending frames over that network.
- Every physical network that is part of the Internet has at least one router that, by definition, is also connected to at least one other physical network; this router can exchange packets with hosts or routers on either

network.

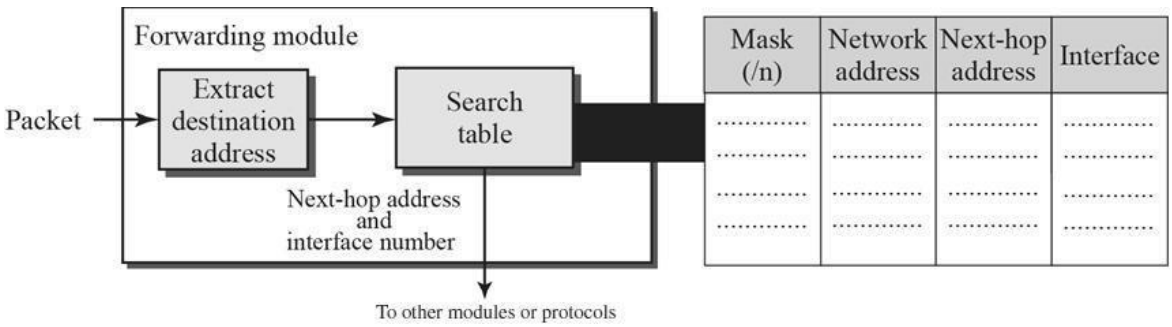
- Forwarding IP Packets can therefore be handled in the following way.
  - A Packets is sent from a source host to a destination host, possibly passing through several routers along the way.
  - Any node, whether it is a host or a router, first tries to establish whether it is connected to the same physical network as the destination.

- To do this, it compares the network part of the destination address with the network part of the address of each of its network interfaces. (Hosts normally have only one interface, while routers normally have two or more, since they are typically connected to two or more networks.)
- If a match occurs, then that means that the destination lies on the same physical network as the interface, and the packet can be directly delivered over that network that has a reasonable chance of getting the packet closer to its destination.
- If there is no match, then the node is not connected to the same physical network as the destination node, then it needs to send the packet to a router.
- In general, each node will have a choice of several routers, and so it needs to pick the best one, or at least one that has a reasonable chance of getting the datagram closer to its destination.
- The router that it chooses is known as the *next hop* router.
- The router finds the correct next hop by consulting its forwarding table. The forwarding table is conceptually just a list of (NetworkNum, NextHop) pairs.
- There is also a default router that is used if none of the entries in the table matches the destination's network number.
- All Packets destined for hosts not on the physical network to which the sending host is attached will be sent out through the default router.

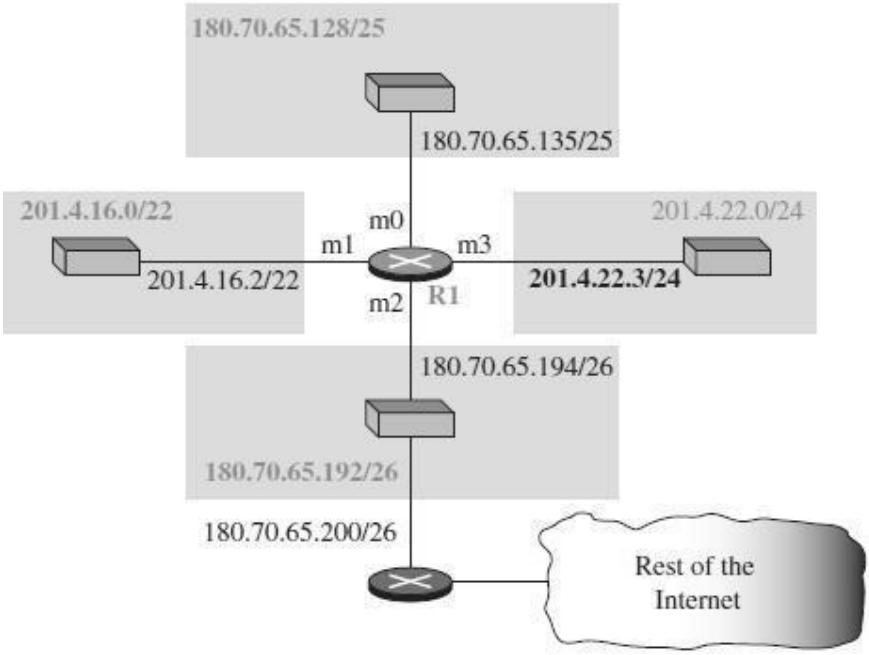
Forwarding Algorithm

```
if (NetworkNum of destination = NetworkNum of one of my interfaces) then
   deliver packet to destination over that interface
else
   if (NetworkNum of destination is in my forwarding table) then
      deliver packet to NextHop router
   else
      deliver packet to default router
```

Simplified Forwarding Module



- The job of the forwarding module is to search the table, row by row.
- In each row, the *n* leftmost bits of the destination address (prefix) are kept and the rest of the bits (suffix) are set to 0s.
- If the resulting address (*network address*), matches with the address in the first column, the information in the next two columns is extracted; otherwise the search continues. Normally, the last row has a default value in the first column, which indicates all destination addresses that did not match the previous rows.
- Routing in classless addressing uses another principle, **longest mask matching.**
- This principle states that the forwarding table is sorted from the longest mask to the shortest mask.
- In other words, if there are three masks, /27, /26, and /24, the mask /27 must be the first entry and /24 must be the last.



- Let us make a forwarding table for router R1 using the configuration as given in the figure above

Forwarding table for router R1

Network address/mask	*Next hop*	*Interface*
180.70.65.192/26	—	m2
180.70.65.128/25	—	m0
201.4.22.0/24	—	m3
201.4.16.0/22	—	m1
Default	180.70.65.200	m2

- When a packet arrives whose leftmost 26 bits in the destination address match the bits in the first row, the packet is sent out from interface m2.
- When a packet arrives whose leftmost 25 bits in the address match the bits in the second row, the packet is sent out from interface m0, and so on.
- The table clearly shows that the first row has the longest prefix and the fourth row has the shortest prefix.

- The longer prefix means a smaller range of addresses; the shorter prefix means a larger range of addresses.

6.NETWORK LAYER PROTOCOLS : IP, ICMPV4

- The main protocol Internet Protocol is responsible for packetizing, forwarding, and delivery of a packet at the network layer.
- The Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery.

IP - INTERNET PROTOCOL

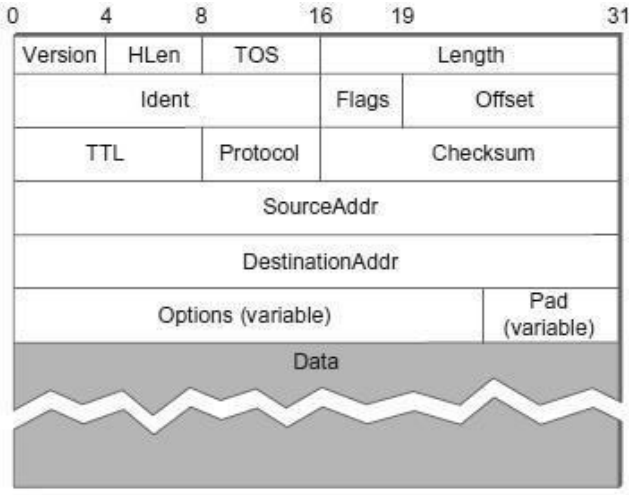
- The Internet Protocol is the key tool used today to build scalable, heterogeneous internetworks.
- IP runs on all the nodes (both hosts and routers) in a collection of networks IP defines the infrastructure that allows these nodes and networks to function as a single logical internetwork.

IP SERVICE MODEL

- Service Model defines the host-to-host services that we want to provide
- The main concern in defining a service model for an internetwork is that we can provide a host-to-host service only if this service can somehow be provided over each of the underlying physical networks.
- The Internet Protocol is the key tool used today to build scalable, heterogeneous internetworks.
- The **IP service model** can be thought of as having **two parts**:
  - A ***GLOBAL ADDRESSING SCHEME*** - which provides a way to identify all hosts in the internetwork
  - A ***DATAGRAM DELIVERY MODEL*** – A connectionless model of data delivery.

IP PACKET FORMAT / IP DATAGRAM FORMAT

- A key part of the IP service model is the type of packets that can be carried.
- The IP datagram consists of a header followed by a number of bytes of data.



FIELD	DESCRIPTION
Version	Specifies the version of IP. Two versions exists – IPv4 and IPv6.
HLen	Specifies the length of the header
TOS ***(***Type of Service)	An indication of the parameters of the quality of service desired such as Precedence, Delay, Throughput and Reliability.
Length	Length of the entire datagram, including the header. The maximum size of an IP datagram is 65,535(2^{10})bytes
Ident (Identification)	Uniquely identifies the packet sequence number. Used for fragmentation and re-assembly.
Flags	Used to control whether routers are allowed to fragment a packet. If a packet is fragmented , this flag value is 1.If not, flag value is 0.
Offset (Fragmentation offset)	Indicates where in the datagram, this fragment belongs. The fragment offset is measured in units of 8 octets (64 bits). The first fragment has offset zero.
TTL (Time to Live)	Indicates the maximum time the datagram is allowed to remain in the network. If this field contains the value zero, then the datagram must be destroyed.
Protocol	Indicates the next level protocol used in the data portion of the datagram
Checksum	Used to detect the processing errors introduced into the packet
Source Address	The IP address of the original sender of the packet.
Destination Address	The IP address of the final destination of the packet.
Options	This is optional field. These options may contain values for options such as Security, Record Route, Time Stamp, etc
Pad	Used to ensure that the internet header ends on a 32 bit boundary. The padding is zero.

The IPv4 header consists of several fields that define the characteristics and properties of the packet. Below is a detailed explanation of each field:

1. Version

- **Description**: A 4-bit field that indicates the IP version used.
- **Common Versions**:
  - IPv4
  - IPv6
- **Value**: In the case of IPv4, this field always contains the decimal value **4**.

Notes:

- Datagrams belonging to different versions have different structures and are parsed differently.
- IPv4 datagrams are parsed by version-4 parsers, while IPv6 datagrams are parsed by version-6 parsers.

2. Header Length (HLEN)

- **Description**: A 4-bit field that contains the length of the IP header in 32-bit words.
- **Purpose**: Helps in determining the start of the actual data.

Minimum and Maximum Header Length:

- **Minimum**: 20 bytes (5 rows × 4 bytes)
- **Maximum**: 60 bytes (20 bytes base header + 40 bytes options)

Concept of Scaling Factor:

- **Calculation**:
  - Header length = HLEN field value × 4 bytes
- **Examples**:
  - HLEN = 5 → Header length = 5 × 4 = 20 bytes
  - HLEN = 10 → Header length = 10 × 4 = 40 bytes
  - HLEN = 15 → Header length = 15 × 4 = 60 bytes

Notes:

- **Differences**: The header length and the HLEN field value are different. The range for the HLEN value is always [5, 15], while the actual header length is [20, 60].
- When solving questions, if the value lies in [5, 15], it must be the HLEN field value.

3. Type of Service (TOS)

- **Description**: An 8-bit field used for Quality of Service (QoS) to mark the datagram for specific treatment.

4. Total Length

- **Description**: A 16-bit field containing the total length of the datagram (header + data).
- **Calculation**:
  - Total length = Header length + Payload length
- **Minimum Total Length**: 20 bytes (20 bytes header + 0 bytes data)
- **Maximum Total Length**: 65535 bytes (maximum value of a 16-bit word)

5. Identification

- **Description**: A 16-bit field used for identifying fragments of an original IP datagram.
- **Purpose**: Each fragmented datagram is assigned the same identification number, which is useful for reassembling fragmented datagrams.

6. DF Bit (Do Not Fragment)

- **Description**: Indicates whether the datagram can be fragmented.
- **Value**:
  - **0**: Allows fragmentation.
  - **1**: Prohibits fragmentation. If fragmentation is required, the datagram is discarded, and an error message is sent to the sender.

7. MF Bit (More Fragments)

- **Description**: Indicates whether there are more fragments following the current one.
- **Value**:
  - **0**: This is the last fragment or the only fragment.
  - **1**: More fragments are following; set for all fragments except the last one.

8. Fragment Offset

- **Description**: A 13-bit field indicating the position of a fragmented datagram in the original unfragmented IP datagram.
- **Calculation**:
  - Fragment offset = Number of data bytes ahead of the current fragment in the original

datagram.

- **Scaling Factor**: A scaling factor of 8 is used for the fragment offset (fragment offset field value = Fragment Offset / 8).

Need for Scaling Factor:

- The maximum amount of data that can be sent in the payload field is $2^{16} – 20$ bytes.
- In a worst-case scenario, the last fragmented datagram could contain only 1 byte, making the fragment offset $2^{16} – 21$, which exceeds the 13-bit representation limit.

9. Time to Live (TTL)

- **Description**: An 8-bit field indicating the maximum number of hops a datagram can take before being discarded.
- **Purpose**: Prevents infinite looping in routing.
- **Decrement**: The TTL is decremented by 1 at each hop and when reaching the destination.
- **Action on Zero**: If TTL reaches zero before reaching the destination, the datagram is discarded.

Notes:

- At intermediate devices, TTL must be greater than zero to proceed further, while at the destination, TTL can be zero or greater.

10. Protocol

- **Description**: An 8-bit field that specifies the next-level protocol to the network layer at the destination.
- **Common Protocol Numbers**:
  - ICMP: 1
  - IGMP: 2
  - TCP: 6
  - UDP: 17

Purpose:

- The protocol number allows routers to prioritize which datagrams to retain during buffer congestion, based on protocol reliability.

11. Header Checksum

- **Description**: A 16-bit field containing a checksum value for the entire header.
- **Purpose**: Used for error checking of the header.
- **Verification**: The checksum is compared at each hop; if mismatched, the datagram is discarded.
- **Update**: Routers update this field when modifying the datagram header.

Important Note:

- The header checksum only includes the IP header; errors in the data field are handled by the encapsulated protocol.

12. Source IP Address

- **Description**: A 32-bit field containing the logical address of the sender of the datagram.

13. Destination IP Address

- **Description**: A 32-bit field containing the logical address of the receiver of the datagram.

14. Options

- **Description**: A variable-length field ranging from 0 to 40 bytes.
- **Uses**:
  - Record route

- Source routing
- Padding

1. Record Route:

- Records the IP addresses of the routers through which the datagram passes.
- **Maximum Addresses**: Up to 9 IPv4 router addresses can be recorded.

2. Source Routing:

- Specifies the route that the datagram must take to reach its destination.
- **Types**: Loose or strict source routing.

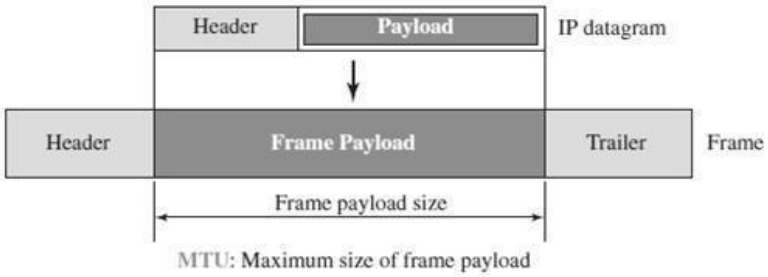
3. Padding:

- Involves adding dummy data to fill unused space to conform to standard sizes.
- **Example**: If the header length is 30 bytes, 2 bytes of dummy data are added to make it 32 bytes.

IP DATAGRAM - FRAGMENTATION AND REASSEMBLY

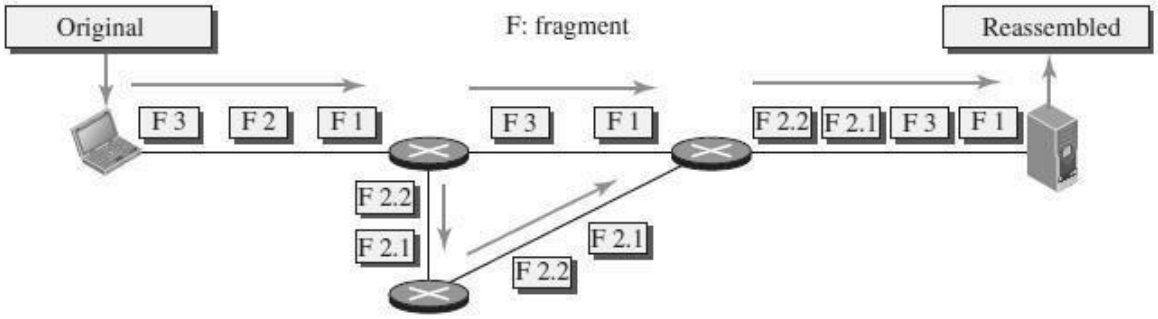
Fragmentation:

- Every network type has a ***maximum transmission unit*** (MTU), which is the largest IP datagram that it can carry in a frame.

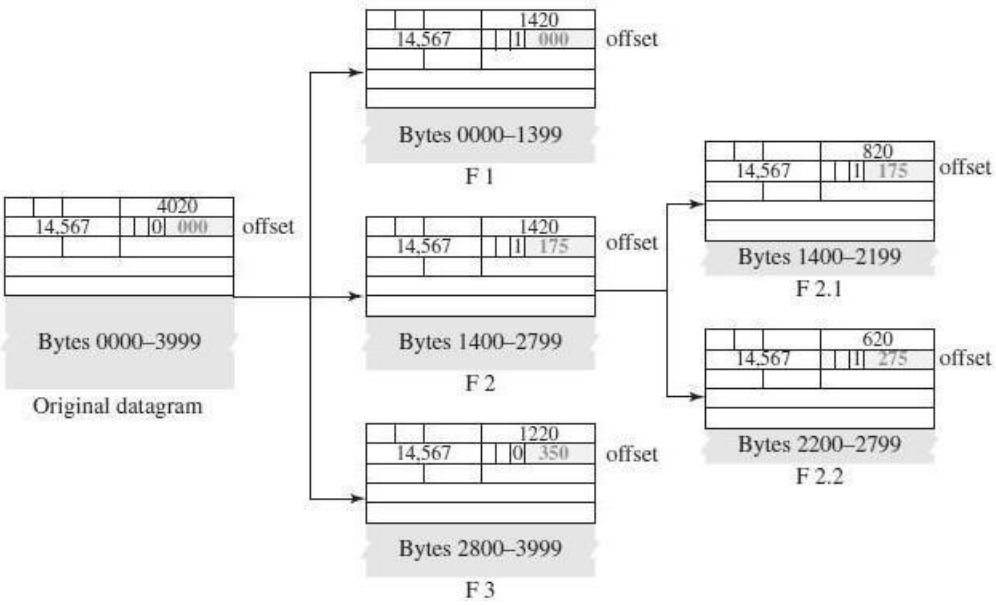


- Fragmentation of a datagram will only be necessary if the path to the destination includes a network with a smaller MTU.
- When a host sends an IP datagram,it can choose any size that it wants.
- Fragmentation typically occurs in a router when it receives a datagram that it wants to forward over a network that has an MTU that is smaller than the received datagram.
- Each fragment is itself a self-contained IP datagram that is transmitted over a sequence of physical networks, independent of the other fragments.
- Each IP datagram is re-encapsulated for each physical network over which it travels.
- For example , if we consider an Ethernet network to accept packets up to 1500 bytes long.
- This leaves two choices for the IP service model:
  - Make sure that all IP datagrams are small enough to fit inside one packet on any network technology
  - Provide a means by which packets can be fragmented and reassembled when they are too big to go over a given network technology.
- Fragmentation produces smaller, valid IP datagrams that can be readily reassembled into the original datagram upon receipt, independent of the order of their arrival.

Example:



- The original packet starts at the client; the fragments are reassembled at the server.
- The value of the identification field is the same in all fragments, as is the value of the flags field with the more bit set for all fragments except the last.
- Also, the value of the offset field for each fragment is shown.
- Although the fragments arrived out of order at the destination, they can be correctly reassembled.



- The value of the offset field is always relative to the original datagram.
- Even if each fragment follows a different path and arrives out of order, the final destination host can reassemble the original datagram from the fragments received (if none of them is lost) using the following strategy:
    1) The first fragment has an offset field value of zero.
    2) Divide the length of the first fragment by 8. The second fragment has an offset value equal to that result.
    3) Divide the total length of the first and second fragment by 8. The third fragment has an offset value equal to that result.
    4) Continue the process. The last fragment has its M bit set to 0.
    5) Continue the process. The last fragment has a *more* bit value of 0.

Reassembly:

- Reassembly is done at the receiving host and not at each router.
- To enable these fragments to be reassembled at the receiving host, they all carry the same identifier in the Ident field.

- This identifier is chosen by the sending host and is intended to be unique among all the datagrams that might arrive at the destination from this source over some reasonable time period.
- Since all fragments of the original datagram contain this identifier, the reassembling host will be able to recognize those fragments that go together.
- For example, if a single fragment is lost, the receiver will still attempt to reassemble the datagram, and it will eventually give up and have to garbage-collect the resources that were used to perform the failed reassembly.
- Hosts are now strongly encouraged to perform "path MTU discovery," a process by which fragmentation is avoided by sending packets that are small enough to traverse the link with the smallest MTU in the path from sender to receiver.

Key Terms

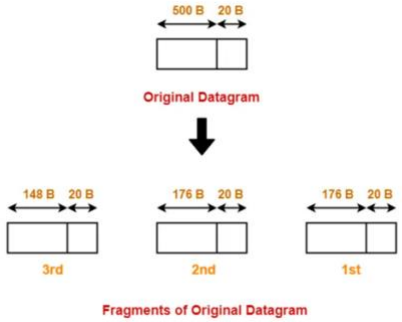
- **MTU (Maximum Transmission Unit)**: The largest size of a packet that can be transmitted over a network.
- **DF Bit (Don't Fragment)**: A flag in the IP header that indicates whether the packet can be fragmented.
- **MF Bit (More Fragments)**: A flag indicating if more fragments are following.
- **Fragment Offset**: Indicates the position of a fragment in the original datagram.

Example Discussed in Class:

Example 1:

Scenario

- **Host A**: Located in Network X, MTU = 520 bytes
- **Host B**: Located in Network Y, MTU = 200 bytes
- **Datagram Details from Host A**:
  - Header length = 20 bytes
  - Payload length = 500 bytes
  - Total length = 520 bytes
  - DF bit = 0 (Fragmentation allowed)



Steps of Fragmentation by Router

Step 1: Examine the Datagram

- **Router Action**: Examines the received datagram.
  - Size of the datagram = 520 bytes
  - Destination network Y has MTU = 200 bytes
  - DF bit is set to 0, indicating fragmentation is allowed.

Conclusion: The datagram size exceeds the MTU, and fragmentation is necessary.

Step 2: Determine Fragment Size

- **Router Decision**:
  - Maximum total length of each fragment = MTU = 200 bytes
  - Header size = 20 bytes

- Maximum data in each fragment = 200 - 20 = 180 bytes

Rule for Fragment Size:

- Each fragment must be as large as possible but less than or equal to MTU and a multiple of 8.
- **Chosen Fragment Size**:
  - Maximum data per fragment = 176 bytes (largest multiple of 8 less than 180).

Step 3: Create Fragments

- The original datagram (520 bytes) is split into three fragments:
  - **First Fragment**: 176 bytes of data
  - **Second Fragment**: 176 bytes of data
  - **Third Fragment**: 148 bytes of data

Header Information for Each Fragment:

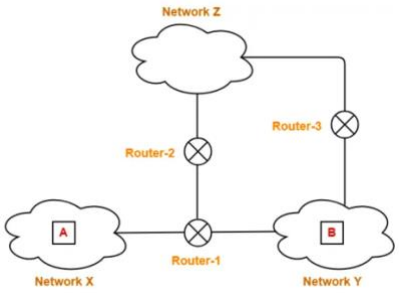
1. **First Fragment**:
   - Header length field value = 20/4=520 / 4 = 520/4=5
   - Total length field value = 176+20= 196
   - MF bit = 1 (more fragments follow)
   - Fragment offset field value = 0
   - Header checksum recalculated.
   - Identification number = Same as original datagram.
2. **Second Fragment**:
   - Header length field value = 20/4=520 / 4 = 520/4=5
   - Total length field value = 176+20=196
   - MF bit = 1
   - Fragment offset field value = 176/8=22
   - Header checksum recalculated.
   - Identification number = Same as original datagram.
3. **Third Fragment**:
   - Header length field value = 20/4=520 / 4 = 520/4=5
   - Total length field value = 148+20=168
   - MF bit = 0 (last fragment)
   - Fragment offset field value = (176+176)/8=44(176 + 176) / 8 = 44(176+176)/8=44
   - Header checksum recalculated.
   - Identification number = Same as original datagram.

Transmission: The router transmits all three fragments to Host B.

Step 4: Reassembly at Destination

- **Receiver Action**: The destination host (Host B) receives the three fragments.
- **Reassembly Algorithm**: Applies an algorithm to combine all fragments to reconstruct the original datagram.

Example 2:



Initial Scenario

- **Host A** sends a datagram to **Host B**.
- **Original Datagram Characteristics**:
  - Header length = 20 bytes
  - Payload length = 500 bytes
  - Total length = 520 bytes
  - DF bit = 0 (Fragmentation allowed)

Fragmentation by Router-1

- **Router-1** divides the datagram into three fragments:
  1. **First Fragment**: 176 bytes of data
  2. **Second Fragment**: 176 bytes of data
  3. **Third Fragment**: 148 bytes of data

Journey of the Second Fragment

- **Second Fragment Characteristics**:
  - Header length = 20 bytes
  - Payload length = 176 bytes
  - Total length = 196 bytes
  - DF bit = 0

Router-2 Processing

Step 1: Examination

- **Router-2** receives the second fragment.
- **Determination**:
  - Size of the datagram = 196 bytes
  - Destination network Z has MTU = 110 bytes
  - DF bit is set to 0.

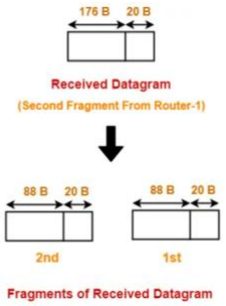
Conclusion: The size of the datagram exceeds the MTU, requiring further fragmentation.

Step 2: Determine Fragment Size

- **Router-2 Decision**:
  - Maximum total length of any fragment = MTU = 110 bytes.
  - Header size = 20 bytes.
  - Maximum data per fragment = 110 - 20 = 90 bytes.

Rule for Fragment Size:

- Router-2 opts to send a maximum of 88 bytes of data in each fragment (the largest multiple of 8 that is less than 90).



Step 3: Create New Fragments

- Router-2 creates two new fragments:
    1. **First Fragment**:
        - Data = 88 bytes
    2. **Second Fragment**:
        - Data = 88 bytes

Header Information for Each New Fragment:

1. **First Fragment**:
    - Header length field value = 20/4=520 / 4 = 520/4=5
    - Total length field value = 88+20=10888 + 20 = 10888+20=108
    - MF bit = 1 (more fragments follow)
    - Fragment offset field value = 176/8=22176 / 8 = 22176/8=22 (offset from the original datagram)
    - Identification number = Same as the original datagram.

Note: This fragment is not the first fragment of the original datagram; it is the first fragment of the datagram received by Router-2.

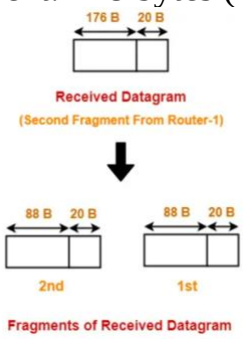
2. **Second Fragment**:
    - Header length field value = 20/4=520 / 4 = 520/4=5
    - Total length field value = 88+20=10888 + 20 = 10888+20=108
    - MF bit = 0 (last fragment)
    - Fragment offset field value = (176+88)/8=33(176 + 88) / 8 = 33(176+88)/8=33
    - Identification number = Same as the original datagram.

Note: This fragment is the last fragment of the datagram received by Router-2.

Transmission: Router-2 transmits both fragments, which reach the destination via Router-3.

Step 4: At the Destination

- The receiver obtains a total of 4 fragments (including those directly from Router-1):
    - 1st fragment: 176 bytes (from Router-1)
    - 2nd fragment: 88 bytes (first new fragment from Router-2)
    - 3rd fragment: 88 bytes (second new fragment from Router-2)
    - 4th fragment: 148 bytes (from Router-1)



Reassembly Algorithm

The receiver employs the following steps to reassemble all fragments:

1. Identify if the datagram is fragmented using the MF bit and Fragment Offset field.
2. Identify all fragments belonging to the same datagram using the identification field.
3. Identify the first fragment (fragment with offset value = 0).
4. Identify subsequent fragments using total length, header length, and fragment offset.
5. Repeat steps until the MF bit = 0.

Fragment Offset Calculation for the next subsequent fragment: Fragment Offset= (Payload length of the current fragment / 8) +
Offset field value of the current fragment

Fragmentation Overhead

Fragmentation increases overhead due to multiple IP headers being attached. The total overhead is calculated as:

Total Overhead=(Total number of fragmented datagrams−1)×size of IP header

Efficiency Calculation

Efficiency=Useful bytes transferred / Total bytes transferred

Or

Efficiency=Data without header / Data with header

Bandwidth Utilization or Throughput

Throughput=Efficiency × Bandwidth

Important Notes

1. **Source Side Fragmentation**: Fragmentation is avoided at the source due to wise segmentation by the transport layer.
2. **Route Variability**: Datagrams from the same source to the same destination may take different routes.
3. **Fragment Offset Values**:
   - Set to 0 for the first fragment.
   - MF bit set to 1 for all fragmented datagrams except the last.
4. **Unique Combinations of MF and Fragment Offset Values**:
   - **1, 0**: 1st Fragment
   - **1, !=0**: Intermediate Fragment
   - **0, !=0**: Last Fragment
   - **0, 0**: No Fragmentation
5. **Identification Field**: Same for all fragments of the original datagram to facilitate reassembly.
6. **Minimum Bandwidth Bottleneck**: When different intermediaries have different bandwidths, the minimum bandwidth must be considered for throughput calculations.
7. **Fragmentation Responsibility**: Handled by routers, while reassembly occurs only at the destination.
8. **Reassembly Challenges**: Fragments may take different paths, preventing

them from meeting at a router, necessitating reassembly only at the final destination.

9. **Re-fragmentation of Parent Fragments**: If a fragment is re-fragmented, the offset value for the first re-fragment is the same as its parent, and the MF bit value for the last re-fragment remains the same as its parent.

Question 1: Which fields of the IPv4 header get modified from source to destination?

Question 2: An IPv4 packet has arrived with the first 8 bits as shown: 01000010. The receiver discards the packet. Why?
Solution 2: There is an error in this packet. The 4 leftmost bits (0100) show the version, which is correct. The next 4 bits (0010) show an invalid header length (2 x 4 = 8). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

Question 3: In an IPv4 packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?
Solution 3: The HLEN value is 8, which means the total number of bytes in the header is 8 x 4, or 32 bytes. The first 20 bytes are the base header, and the next 12 bytes are the options.

Question 4: In an IPv4 packet, the value of HLEN is 5, and the value of the total length field is 0x0028. How many bytes of data are being carried by this packet?
Solution 4: The HLEN value is 5, which means the total number of bytes in the header is 5 x 4, or 20 bytes (no options). The total length is 40 bytes, which means the packet is carrying 20 bytes of data (40-20).

Question 5: A packet has arrived with an M bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?
Solution 5: If the M bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A non-fragmented packet is considered the last fragment.

Question 6: A packet has arrived with an M bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?
Solution 6: If the M bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset).

Question 7: A packet has arrived with an M bit value of 1 and a fragmentation offset value of 0. Is this the first fragment, the last fragment, or a middle fragment?
Solution 7: Because the M bit is 1, it is either the first fragment or the middle one. Because the offset value is 0, it is the first fragment.

Question 8: A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution 8: To find the number of the first byte, we multiply the offset value by 8. This means that the first-byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

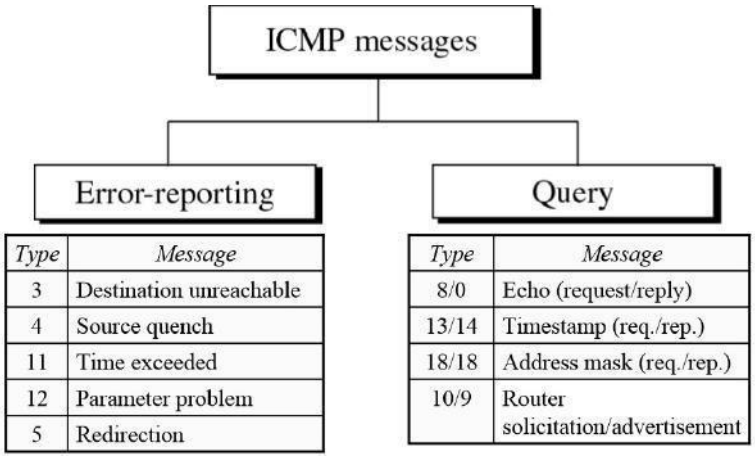
Question 9: A packet has arrived in which the offset value is 100, the value of HLEN is 5, and the value of the total length field is 100. What are the numbers of the first byte and the last byte?
Solution 9: The first-byte number is 100 x 8 = 800. The total length is 100 bytes, and the header length is 20 bytes (5 x 4), which means that there are 80 bytes in this datagram. If the first-byte number is 800, the last-byte number must be 879.

ICMPV4 - INTERNET CONTROL MESSAGE PROTOCOL VERSION 4

- ICMP is a network-layer protocol.
- It is a companion to the IP protocol.
- Internet Control Message Protocol (ICMP) defines a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully.

ICMP MESSAGE TYPES

- ICMP messages are divided into two broad categories: *error-reporting messages* and *query messages.*
- The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet.
- The query messages help a host or a network manager get specific information from a router or another host.



Type	*Message*
3	Destination unreachable
4	Source quench
11	Time exceeded
12	Parameter problem
5	Redirection

Type	*Message*
8/0	Echo (request/reply)
13/14	Timestamp (req./rep.)
18/18	Address mask (req./rep.)
10/9	Router solicitation/advertisement

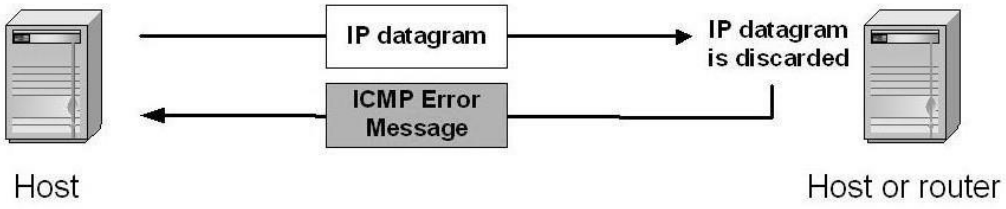
ICMP Error – Reporting Messages

- ICMP error messages report error conditions
- Typically sent when a datagram is discarded
- Error message is often passed from ICMP to the application program

- ***Destination Unreachable—***When a router *cannot route* a datagram, the datagram is discarded and sends a destination unreachable message to source host.
- ***Source Quench—***When a router or host discards a datagram due to *congestion*, it sends a source-quench message to the source host. This message acts as flow

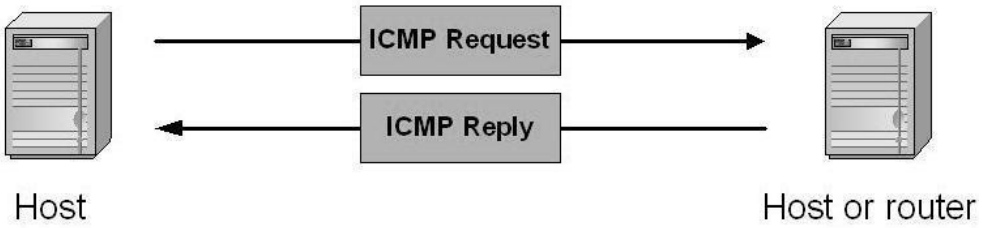
control.

- ***Time Exceeded—***Router discards a datagram when TTL field becomes 0 and a time exceeded message is sent to the source host.
- ***Parameter Problem—***If a router discovers ambiguous or *missing* value in any field of the datagram, it discards the datagram and sends parameter problem message to source.
- ***Redirection—***Redirect messages are sent by the default router to inform the source host to *update* its forwarding table when the packet is routed on a wrong path.



ICMP Query Messages

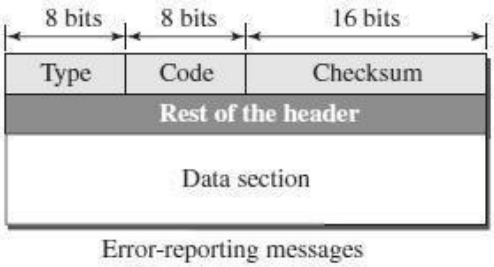
- Request sent by host to a router or host
- Reply sent back to querying host



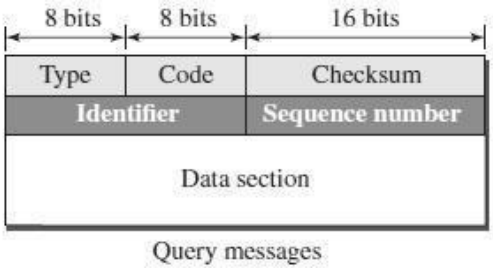
- ***Echo Request & Reply—***Combination of echo request and reply messages determines whether two systems communicate or not.
- ***Timestamp Request & Reply—***Two machines can use the timestamp request and reply messages to determine the round-trip time (RTT).
- ***Address Mask Request & Reply—***A host to obtain its subnet mask, sends an address mask request message to the router, which responds with an address mask reply message.
- ***Router Solicitation/Advertisement—***A host broadcasts a router solicitation message to know about the router. Router broadcasts its routing information with router advertisement message.

ICMP MESSAGE FORMAT

- An ICMP message has an 8-byte header and a variable-size data section.



Error-reporting messages



Query messages

Type	Defines the type of the message
Code	Specifies the reason for the particular message type
Checksum	Used for error detection
Rest of the header	Specific for each message type
Data	Used to carry information
Identifier	Used to match the request with the reply
Sequence Number	Sequence Number of the ICMP packet

ICMP DEBUGGING TOOLS

Two tools are used for debugging purpose. They are (1) Ping (2) Traceroute

Ping

- The *ping* program is used to find if a host is alive and responding.
- The source host sends ICMP echo-request messages; the destination, if alive, responds with ICMP echo-reply messages.
- The *ping* program sets the identifier field in the echo-request and echo-reply message and starts the sequence number from 0; this number is incremented by 1 each time a new message is sent.
- The *ping program* can calculate the round-trip time.
- It inserts the sending time in the data section of the message.
- When the packet arrives, it subtracts the arrival time from the departure time to get the round-trip time (RTT).

$ ping google.com

Traceroute or Tracert

- The *traceroute* program in UNIX or *tracert* in Windows can be used to trace the path of a packet from a source to the destination.
- It can find the IP addresses of all the routers that are visited along the path.
- The program is usually set to check for the maximum of 30 hops (routers) to be visited.
- The number of hops in the Internet is normally less than this.

$ traceroute google.com

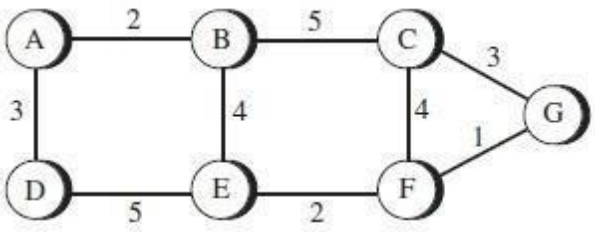
---

7. UNICAST ROUTING

- Routing is the process of selecting best paths in a network.
- In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables.
- Routing a packet from its source to its destination means routing the packet from a *source router* (the default router of the source host) to a *destination router* (the router connected to the destination network).
- The source host needs no forwarding table because it delivers its packet to the default router in its local network.
- The destination host needs no forwarding table either because it receives the packet from its default router in its local network.
- Only the intermediate routers in the networks need forwarding tables.

NETWORK AS A GRAPH

- The Figure below shows a graph representing a network.



- The nodes of the graph, labeled A through G, may be hosts, switches, routers, or networks.
- The edges of the graph correspond to the network links.
- Each edge has an associated *cost.*
- The basic problem of routing is to find the lowest-cost path between any two nodes, where the cost of a path equals the sum of the costs of all the edges that make up the path.
- This static approach has several problems:
  - ❖ It does not deal with node or link failures.
  - ❖ It does not consider the addition of new nodes or links.
  - ❖ It implies that edge costs cannot change.
- For these reasons, routing is achieved by running routing protocols among the nodes.
- These protocols provide a distributed, dynamic way to solve the problem of finding the lowest-cost path in the presence of link and node failures and changing edge costs.

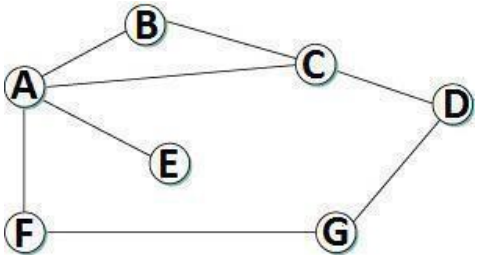
UNICAST ROUTING ALGORITHMS

- There are three main classes of routing protocols:
  1) **Distance Vector Routing Algorithm – Routing Information Protocol**
  2) **Link State Routing Algorithm – Open Shortest Path First Protocol**
  3) **Path-Vector Routing Algorithm - Border Gateway Protocol**

DISTANCE VECTOR ROUTING (DSR)
ROUTING INFORMATION PROTOCOL (RIP)
BELLMAN - FORD ALGORITHM

- Distance vector routing is *distributed*, i.e., algorithm is run on all nodes.
- Each node *knows* the distance (cost) to each of its directly connected neighbors.
- Nodes construct a *vector* (Destination, Cost, NextHop) and distributes to its neighbors.
- Nodes compute routing table of *minimum* distance to every other node via NextHop using information obtained from its neighbors.

Initial State



- In given network, *cost* of each link is 1 hop.
- Each node sets a distance of 1 (hop) to its *immediate* neighbor and cost to itself as 0.
- Distance for non-neighbors is marked as *unreachable* with value ∞ (infinity).
- For node *A*, nodes *B*, *C*, *E* and *F* are *reachable*, whereas nodes *D* and *G* are *unreachable*.

Destination	Cost	NextHop
A	0	A
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Node A's initial table

Destination	Cost	NextHop
A	1	A
B	1	B
C	0	C
D	1	D
E	∞	—
F	∞	—
G	∞	—

Node C's initial table

Destination	Cost	NextHop
A	1	A
B	∞	—
C	∞	—
D	∞	—
E	∞	—
F	0	F
G	1	G

Node F's initial table

- The initial table for all the nodes are given below

Initial Distances Stored at Each Node (Global View)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

- Each node *sends* its initial table (distance vector) to neighbors and receives their estimate.
- Node *A* sends its table to nodes *B*, *C*, *E* & *F* and receives tables from nodes *B*, *C*, *E* & *F*.
- Each node *updates* its routing table by comparing with each of its neighbor's table
- For each destination, Total Cost is computed as:
  - ***Total Cost*** = Cost (*Node* to *Neighbor*) + Cost (*Neighbor* to *Destination*)
- If Total Cost < Cost then
  - ***Cost*** = Total Cost and NextHop = *Neighbor*
- Node *A learns* from *C*'s table to reach node *D* and from *F*'s table to reach node *G*.
- Total Cost to reach node *D* via *C* = Cost (*A* to *C*) + Cost(*C* to *D*)

  Cost = 1 + 1 = 2.
  - Since 2 < ∞, entry for destination *D* in *A*'s table is changed to (*D*, 2, *C*)
  - Total Cost to reach node *G* via *F* = Cost(*A* to *F*) + Cost(*F* to *G*) = 1 + 1 = 2
  - Since 2 < ∞, entry for destination *G* in *A*'s table is changed to (*G*, 2, *F*)
- Each node builds *complete* routing table after few exchanges amongst its neighbors.

Node A's final routing table

Destination	Cost	NextHop
A	0	A
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

- System stabilizes when all nodes have complete routing information, i.e., **convergence.**
- Routing tables are exchanged *periodically or* in case of *triggered update*.
- The final distances stored at each node is given below:

Final Distances Stored at Each Node (Global View)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Updation of Routing Tables

There are two different circumstances under which a given node decides to send a routing update to its neighbors.

Periodic Update

- In this case, each node automatically sends an update message every so often, even if nothing has changed.
- The frequency of these periodic updates varies from protocol to protocol, but it is typically on the order of several seconds to several minutes.

Triggered Update

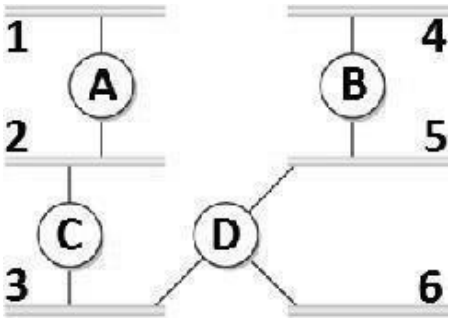
- In this case, whenever a node notices a link failure or receives an update from one of its neighbors that causes it to change one of the routes in its routing table.

- Whenever a node's routing table changes, it sends an update to its neighbors, which may lead to a change in their tables, causing them to send an update to their neighbors.

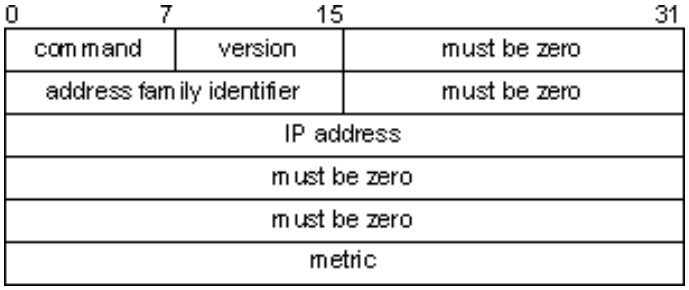
ROUTING INFORMATION PROTOCOL (RIP)

- RIP is an intra-domain routing protocol based on distance-vector algorithm.

Example



- Routers *advertise* the cost of reaching networks. Cost of reaching each link is 1 hop. For example, router *C* advertises to *A* that it can reach network 2, *3* at cost 0 (directly connected), networks *5*, *6* at cost 1 and network *4* at cost 2.
- Each router *updates* cost and next hop for each network number.
- Infinity is defined as 16, i.e., any route cannot have more than 15 hops. Therefore, RIP can be implemented on small-sized networks only.
- Advertisements are sent every 30 seconds or in case of triggered update.



- ***Command*** - It indicates the packet type.
  Value 1 represents a request packet. Value 2 represents a response packet.
- ***Version*** - It indicates the RIP version number. For RIPv1, the value is 0x01.
- ***Address Family Identifier*** - When the value is 2, it represents the IP protocol.
- ***IP Address*** - It indicates the destination IP address of the route. It can be the addresses of only the natural network segment.
- ***Metric*** - It indicates the hop count of a route to its destination.

Count-To-Infinity (or) Loop Instability Problem

- Suppose link from node *A* to *E* goes *down.*
  - Node *A* advertises a distance of ∞ to *E* to its neighbors
  - Node B receives periodic update from C before A's update reaches B
  - Node *B* updated by *C*, concludes that *E* can be reached in 3 hops via *C*
  - Node *B* advertises to *A* as 3 hops to reach *E*

❖ Node *A* in turn updates *C* with a distance of 4 hops to *E* and so on

- Thus nodes update each other until cost to *E* reaches *infinity*, i.e., *no convergence*.
- Routing table does not stabilize.
- This problem is called *loop instability* or *count to infinity*

Solution to Count-To-Infinity (or) Loop Instability Problem :

- *Infinity* is redefined to a small number, say 16.
- Distance between any two nodes can be 15 hops maximum. Thus distance vector routing *cannot be used* in large networks.
- When a node updates its neighbors, it does not send those routes it learned from each neighbor back to that neighbor. This is known as **split horizon**.
- **Split horizon with poison reverse** allows nodes to advertise routes it learnt from a node back to that node, but with a warning message.

Split Horizon with Poison Reverse

Definition: A routing technique used in distance-vector protocols to prevent routing loops.

Key Concepts:

1. **Split Horizon:** Prevents a router from advertising routes back to the neighbor from which it learned them, reducing the risk of loops.
2. **Poison Reverse:** When a route becomes unreachable, the router sends an update with an infinite metric (e.g., 16 in RIP) to inform the neighbor that the route is invalid.

Benefits:

- Avoids routing loops
- Speeds up network convergence

Example: If Router A learns a route to Network X from Router B, it won't advertise it back to Router B. If Network X becomes unreachable, Router A will inform Router B that the route is poisoned, ensuring both routers update their tables accordingly.

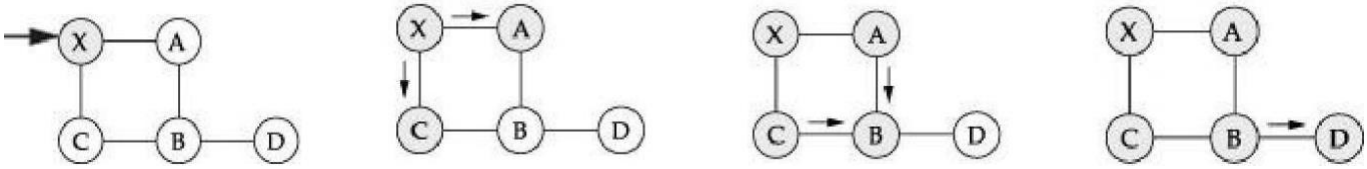
LINK STATE ROUTING (LSR)
OPEN SHORTEST PATH PROTOCOL (OSPF)
DIJKSTRA'S ALGORITHM

- Each node knows *state* of link to its neighbors and *cost.*
- Nodes create an update packet called *link-state packet* (LSP) that contains:
  - ID of the node
  - List of neighbors for that node and associated cost
  - 64-bit Sequence number
  - Time to live
- Link-State routing protocols rely on two mechanisms:
  - ***Reliable flooding*** of link-state information to all other nodes
  - ***Route calculation*** from the accumulated link-state knowledge

Reliable Flooding

- Each node *sends* its LSP out on each of its directly connected links.
- When a node receives LSP of another node, checks if it has an LSP already for that node.

- If not, it stores and forwards the LSP on all other links except the incoming one.
- Else if the received LSP has a *bigger* sequence number, then it is stored and forwarded. Older LSP for that node is *discarded*.
- Otherwise discard the received LSP, since it is not latest for that node.
- Thus recent LSP of a node eventually *reaches* all nodes, i.e., reliable *flooding*.



(a) (b) (c) (d)

- Flooding of LSP in a small network is as follows:
  - When node *X* receives *Y*'s LSP (*fig a*), it floods onto its neighbors *A* and *C* (*fig b*)
  - Nodes *A* and *C* forward it to *B*, but does not sends it back to *X* (*fig c*).
  - Node *B* receives two copies of LSP with same sequence number.
  - Accepts one LSP and forwards it to *D* (*fig d*). Flooding is complete.
- LSP is generated either *periodically* or when there is a *change* in the topology.

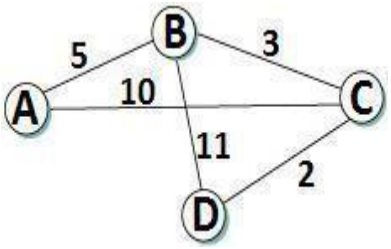
Route Calculation

- Each node knows the entire topology, once it has LSP from every other node.
- Forward search algorithm is used to compute routing table from the received LSPs.
- Each node maintains two lists, namely Tentative and Confirmed with entries of the form (Destination, Cost, NextHop).

DIJKSTRA'S SHORTEST PATH ALGORITHM (FORWARD SEARCH ALGORITHM)

1. Each host maintains two lists, known as ***Tentative*** and ***Confirmed***
2. Initialize the Confirmed list with an entry for the Node (Cost = 0).
3. Node just added to Confirmed list is called Next. Its LSP is examined.
4. For each neighbor of Next, calculate cost to reach each neighbor as Cost (Node to Next) + Cost (Next to Neighbor).
   a. If Neighbor is neither in Confirmed nor in Tentative list, then add (Neighbor, Cost, NextHop) to Tentative list.
   b. If Neighbor is in Tentative list, and Cost is less than existing cost, then replace the entry with (Neighbor, Cost, NextHop).
5. If Tentative list is empty then *Stop*, otherwise move *least* cost entry from Tentative list to Confirmed list. Go to *Step 2*.

Example :

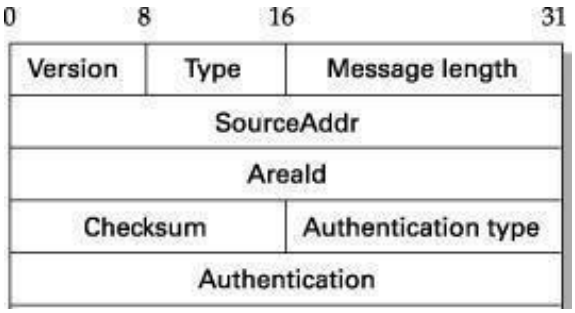


Step	Confirmed	Tentative	Comments
1	(D,0,–)		Since D is the only new member of the confirmed list, look at its LSP.
2	(D,0,–)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better than anything else on either list, so put it on Tentative list; same for C.
3	(D,0,–) (C,2,C)	(B,11,B)	Put lowest-cost member of Tentative (C) onto Confirmed list. Next, examine LSP of newly confirmed member (C).
4	(D,0,–) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
5	(D,0,–) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to Confirmed, then look at its LSP.
6	(D,0,–) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative entry.
7	(D,0,–) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative (A) to Confirmed, and we are all done.

OPEN SHORTEST PATH FIRST PROTOCOL (OSPF)

- OSPF is a non-proprietary widely used link-state routing protocol.
- OSPF Features are:
  - **Authentication—**Malicious host can collapse a network by advertising to reach every host with cost 0. Such disasters are averted by authenticating routing updates.
  - **Additional hierarchy—**Domain is partitioned into areas, i.e., OSPF is more scalable.
  - **Load balancing—**Multiple routes to the same place are assigned same cost. Thus traffic is distributed evenly.

Link State Packet Format



- ***Version*** **—** represents the current version, i.e., 2.
- ***Type*** **—** represents the type (1–5) of OSPF message.
  - T*ype 1* - "hello" message, T*ype 2* - request, T*ype 3* – send ,
  - T*ype 4* - acknowledge the receipt of link state messages ,
  - T*ype 5* - reserved
- ***SourceAddr*** **—** identifies the sender
- ***AreaId*** **—** 32-bit identifier of the area in which the node is located
- ***Checksum*** **—** 16-bit internet checksum
- ***Authentication type*** **—** 1 (simple password), 2 (cryptographic authentication).
- ***Authentication*** **—** contains password or cryptographic checksum

Difference Between Distance-Vector And Link-State Algorithms

Distance vector Routing	*Link state Routing*
Each node talks only to its directly connected neighbors, but it tells them everything it has learned (i.e., distance to all nodes).	Each node talks to all other nodes, but it tells them only what it knows for sure (i.e., only the state of its directly connected links).

Feature	Distance Vector Routing	Link State Routing
Algorithm Used	Bellman-Ford Algorithm	Dijkstra's Algorithm
Routing Information	Each router sends its routing table to its neighbors.	Each router sends information about its immediate neighbors to all routers in the network.
Routing Updates	Periodic or triggered updates. Updates are based on information from neighbors (local knowledge).	Triggered by a change in the network topology. Routers have knowledge of the entire network (global knowledge).
Knowledge of Network	Routers only know the distance to destination and next hop (local knowledge).	Routers have complete knowledge of the network topology (global knowledge).
Convergence Time	Moderate convergence; may experience persistent routing loops and count-to-infinity issues.	Faster convergence with only transient loop problems.
Bandwidth Usage	No flooding; small packets and local sharing require less bandwidth. Less traffic overall.	More bandwidth is required for flooding and sending large link-state packets. More network traffic overall.
Scalability	Suitable for smaller networks due to slower convergence and the risk of loops.	More scalable and suitable for larger networks due to faster convergence.
Complexity	Less computationally intensive, simpler to implement, less CPU utilization.	More complex and computationally demanding, high CPU utilization due to global network knowledge.
Example Protocols	RIP (Routing Information Protocol), IGRP (Interior Gateway Routing Protocol)	OSPF (Open Shortest Path First), IS-IS (Intermediate System to Intermediate System)
Handling of Network Changes	Slower to adapt to network changes.	Quickly adapts to changes in the network topology.
Route Calculation	Based on least hops to the destination.	Based on least cost to the destination.
Packet Updates	Full routing tables are updated and sent to neighbors using broadcasts.	Only link states are updated and sent using multicasts.
Risk of Loops	Higher risk of persistent routing loops.	Only transient loop problems, less prone to loops due to complete network knowledge.
Updates Contain	Distance vectors, which include the next hop and distance metrics.	Link-state packets, containing information about all neighbors and their link states.
Flooding Mechanism	No flooding; updates are sent only to neighbors.	Requires network-wide flooding of link-state advertisements.
Traffic Load	Less traffic due to fewer updates.	More network traffic compared to Distance Vector Routing.

PATH VECTOR ROUTING (PVR)
BORDER GATEWAY PROTOCOL (BGP)

- Path-vector routing is an asynchronous and distributed routing algorithm.
- The Path-vector routing is not based on least-cost routing.
- The best route is determined by the source using the policy it imposes on the route.
- In other words, the source can control the path.
- Path-vector routing is not actually used in an internet, and is mostly designed to

route a packet between ISPs.

Spanning Trees

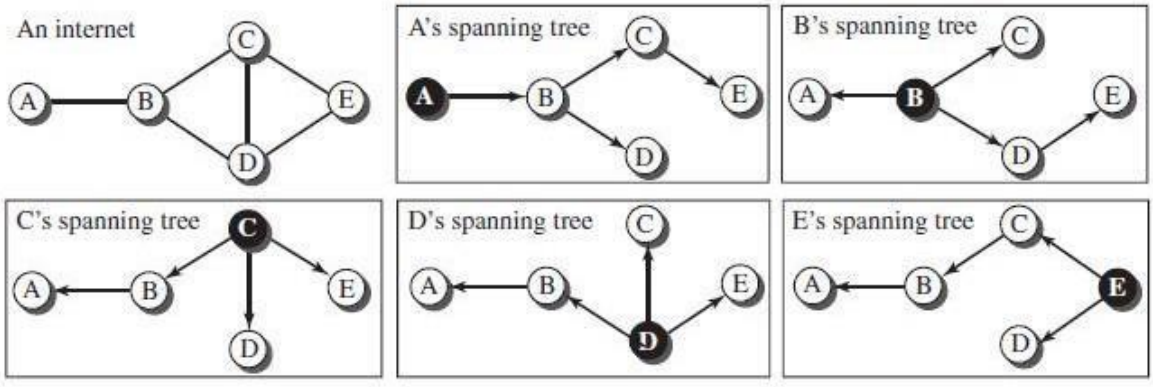
- In path-vector routing, the path from a source to all destinations is determined by the *best* spanning tree.
- The best spanning tree is not the least-cost tree.
- It is the tree determined by the source when it imposes its own policy.
- If there is more than one route to a destination, the source can choose the route that meets its policy best.
- A source may apply several policies at the same time.
- One of the common policies uses the minimum number of nodes to be visited. Another common policy is to avoid some nodes as the middle node in a route.
- The spanning trees are made, gradually and asynchronously, by each node. When a node is booted, it creates a *path vector* based on the information it can obtain about its immediate neighbor.
- A node sends greeting messages to its immediate neighbors to collect these pieces of information.
- Each node, after the creation of the initial path vector, sends it to all its immediate neighbors.
- Each node, when it receives a path vector from a neighbor, updates its path vector using the formula

$$\text{Path}(x, y) = \text{best}\ \{\text{Path}(x, y), [(x + \text{Path}(v, y)]\} \quad \text{for all } v\text{'s in the internet.}$$

- The policy is defined by selecting the *best* of multiple paths.
- Path-vector routing also imposes one more condition on this equation.
- If Path (**v**, ***y***) includes ***x***, that path is discarded to avoid a loop in the path.
- In other words, ***x*** does not want to visit itself when it selects a path to ***y***.

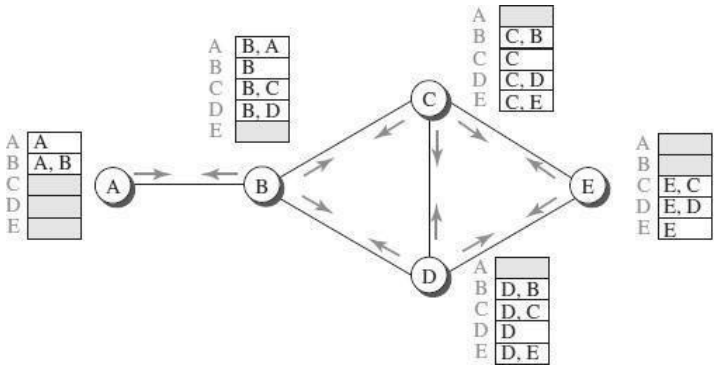
Example:

- The Figure below shows a small internet with only five nodes.
- Each source has created its own spanning tree that meets its policy.
- The policy imposed by all sources is to use the minimum number of nodes to reach a destination.
- The spanning tree selected by A and E is such that the communication does not pass through D as a middle node.
- Similarly, the spanning tree selected by B is such that the communication does not pass through C as a middle node.



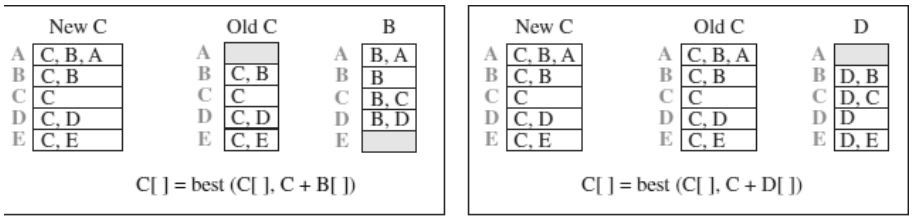
Path Vectors made at booting time

- The Figure below shows all of these path vectors for the example.
- Not all of these tables are created simultaneously.
- They are created when each node is booted.
- The figure also shows how these path vectors are sent to immediate neighbors after they have been created.



Updating Path Vectors

- The Figure below shows the path vector of node C after two events.
- In the first event, node C receives a copy of B's vector, which improves its vector: now it knows how to reach node A.
- In the second event, node C receives a copy of D's vector, which does not change its vector.
- The vector for node C after the first event is stabilized and serves as its forwarding table.



Event 1: C receives a copy of B's vector

Event 2: C receives a copy of D's vector

Address Resolution Protocol (ARP)

1. Overview

ARP is a protocol that maps an IP address to a physical machine address (MAC address) in a local area network (LAN). Operating at the **Link Layer** in the OSI model, ARP enables direct communication between devices in the same network segment.

2. ARP Functionality and Working

Primary Function: ARP translates an IP address to a MAC address, ensuring devices in the same network can locate and communicate with each other. When a device wants to send a packet within the same network, it needs the MAC address of the destination.

Working Steps:

1. **Initiating Communication**: A device (host or router) initiates a communication request to a destination IP address on the local network. If it doesn't already know the MAC address associated with that IP, it triggers an ARP request.
2. **ARP Request**:
   - The device broadcasts an **ARP request packet** on the network, asking, "Who has this IP address?"
   - This packet includes:
     - Source IP and MAC address of the requesting device.
     - Target IP address (the address it wants the MAC for).

        - Target MAC address is left empty.
3. **Broadcast**: The ARP request is sent to all devices in the local network segment.
4. **ARP Response**:
    - The device with the matching IP address responds by sending an **ARP reply packet** directly to the requester.
    - This reply contains the MAC address associated with the target IP address.
5. **Caching**: The requester updates its **ARP cache** with the IP-MAC mapping. The cached entry is saved for a period to avoid repeated requests for the same address.
6. **Data Transmission**: With the resolved MAC address, the device can now transmit data directly to the intended destination.

Step-by-Step Working of ARP

1. **Initiating Communication**:
    - Suppose Device A with IP address 192.168.1.2 wants to communicate with Device B, which has IP 192.168.1.3.
    - Device A knows the IP address of Device B but doesn't know its MAC address.
    - To communicate directly at the Link Layer, Device A needs Device B's MAC address.
2. **ARP Request Creation**:
    - Device A constructs an **ARP request packet** with the following fields:
        - **Sender IP Address**: IP of Device A (192.168.1.2).
        - **Sender MAC Address**: MAC address of Device A (e.g., AA:BB:CC:DD:EE:01).
        - **Target IP Address**: IP address of Device B (192.168.1.3).
        - **Target MAC Address**: Left empty because Device A does not yet know Device B's MAC address.
3. **Broadcasting the ARP Request**:
    - Device A sends the ARP request as a **broadcast** frame with the destination MAC address set to FF:FF:FF:FF:FF:FF.
    - Broadcasting ensures that all devices on the same local network segment receive this request.
4. **Receiving the ARP Request**:
    - All devices on the network segment receive the broadcasted ARP request and examine it.
    - Each device checks if the **Target IP Address** field matches its own IP address.
    - Only Device B, with IP 192.168.1.3, recognizes that it is the intended recipient of the request.
5. **Creating the ARP Reply**:
    - Device B generates an **ARP reply packet** with the following fields:
        - **Sender IP Address**: IP of Device B (192.168.1.3).
        - **Sender MAC Address**: MAC address of Device B (e.g., AA:BB:CC:DD:EE:02).
        - **Target IP Address**: IP address of Device A (192.168.1.2).
        - **Target MAC Address**: MAC address of Device A (e.g., AA:BB:CC:DD:EE:01).
6. **Unicast Response to Device A**:
    - Device B sends the ARP reply as a **unicast** packet directly to Device A using Device A's MAC address (AA:BB:CC:DD:EE:01).
7. **Updating ARP Cache**:
    - Device A receives the ARP reply and extracts Device B's MAC address from the packet.
    - Device A then stores this IP-to-MAC mapping (e.g., 192.168.1.3 mapped to AA:BB:CC:DD:EE:02) in its **ARP cache**.
    - This cache allows Device A to remember Device B's MAC address, avoiding repeated ARP requests for the same IP.
8. **Data Transmission**:
    - With the MAC address of Device B now known, Device A can send packets directly to Device B at the Link Layer.

3. ARP Packet Structure

Field	Description
Hardware Type	Specifies the hardware protocol (Ethernet = 1)
Protocol Type	Specifies the protocol address type (IPv4 = 0x0800)
Hardware Size	Length of the MAC address (6 bytes)
Protocol Size	Length of the IP address (4 bytes)
Opcode	1 for ARP request, 2 for ARP reply
Sender MAC Address	MAC address of the request sender
Sender IP Address	IP address of the request sender
Target MAC Address	MAC address of the target (empty in requests)
Target IP Address	IP address of the target device

4. **Types of ARP**
- **Proxy ARP**: Allows a router to respond to ARP requests on behalf of another device outside the local network.
- **Gratuitous ARP**: A device sends an ARP request with its own IP and MAC to update other devices' ARP caches.

Reverse Address Resolution Protocol (RARP)

1. Overview

RARP maps a known MAC address to an IP address. Diskless workstations, which may lack an IP address configuration upon booting, typically use RARP to request an IP from a server within the network.

2. RARP Functionality and Working

Primary Function: RARP enables a device with a known MAC address to request an IP address from a network server.

Working Steps:
1. **Device Boot-Up**: A diskless workstation or device that doesn't know its IP address sends a RARP request with its MAC address after booting up.
2. **RARP Request**:
    - The device broadcasts a **RARP request** on the local network, asking, "What IP address corresponds to this MAC address?"
    - This packet contains:
        - Device's MAC address as the sender.
        - Sender IP address field is left blank, as the device doesn’t know its IP.
3. **RARP Server Response**:
    - A RARP server on the network listens for RARP requests and checks its configuration to find a matching IP address.
    - The server then responds with a **RARP reply**, assigning an IP address to the requester.
4. **IP Configuration**: The requester configures itself with the provided IP address, enabling network communication.

Step-by-Step Working of RARP
1. **Device Boot-Up Without an IP Address**:
    - A device (e.g., a diskless workstation) on a network boots up without an IP address configuration.
    - It only knows its MAC address (e.g., AA:BB:CC:DD:EE:01).
    - To communicate, it needs to obtain an IP address.
2. **RARP Request Creation**:
    - The device creates a **RARP request packet** with the following fields:
        - **Sender MAC Address**: MAC address of the device (e.g., AA:BB:CC:DD:EE:01).
        - **Sender IP Address**: Empty, since the device does not have an IP address.
        - **Target MAC Address**: Set to FF:FF:FF:FF:FF:FF (broadcast).
        - **Target IP Address**: Left empty as well, as this is the information the device is requesting.
3. **Broadcasting the RARP Request**:
    - The device sends the RARP request as a **broadcast** frame with the destination MAC address set to FF:FF:FF:FF:FF:FF.
    - Broadcasting ensures that all devices on the network, including the RARP server, receive the request.
4. **RARP Server Receives the Request**:

- A RARP server on the network (a server specifically configured to handle RARP requests) receives the broadcasted request.
- The RARP server checks the **Sender MAC Address** to identify the requester and searches its internal database or configuration to find a corresponding IP address for this MAC.

5. **Creating the RARP Reply**:
   - Once the RARP server finds the IP address assigned to the MAC address of the requester, it creates a **RARP reply packet** with the following fields:
     - **Sender IP Address**: IP address assigned to the requesting device (e.g., 192.168.1.2).
     - **Sender MAC Address**: MAC address of the RARP server.
     - **Target IP Address**: Empty.
     - **Target MAC Address**: MAC address of the requesting device (AA:BB:CC:DD:EE:01).
6. **Unicast Response to the Device**:
   - The RARP server sends the RARP reply as a **unicast** frame directly to the MAC address of the requesting device.
7. **Device Receives IP Configuration**:
   - The requesting device receives the RARP reply packet and extracts its assigned IP address from the **Sender IP Address** field in the packet.
   - The device configures itself with this IP address, enabling it to communicate on the network with other IP-based devices.

3. RARP Packet Structure

Field	Description
Hardware Type	Specifies the hardware protocol (Ethernet = 1)
Protocol Type	Specifies the protocol address type (IPv4 = 0x0800)
Hardware Size	Length of the MAC address (6 bytes)
Protocol Size	Length of the IP address (4 bytes)
Opcode	3 for RARP request, 4 for RARP reply
Sender MAC Address	MAC address of the request sender
Sender IP Address	Not filled in RARP request
Target MAC Address	MAC address of the RARP server
Target IP Address	Not filled in RARP request

Key Points on FF:FF:FF:FF:FF:FF Broadcast Address in ARP and RARP

- **ARP**:
  - ARP uses FF:FF:FF:FF:FF:FF to broadcast a request, asking "Who has this IP address?"
  - The broadcast ensures that all devices in the network segment receive the request, allowing the device with the matching IP to respond.
- **RARP**:
  - RARP also uses FF:FF:FF:FF:FF:FF for broadcasting the initial request, asking for "What IP address is assigned to this MAC address?"
  - This broadcast mechanism enables the RARP server to receive the request and reply with the correct IP address assignment.

5. Key Differences between ARP and RARP

Feature	ARP	RARP
Purpose	Resolves IP address to MAC address	Resolves MAC address to IP address
Usage	Used by all devices in the local network	Used by diskless devices to obtain an IP
Broadcast	ARP requests are broadcasted	RARP requests are broadcasted
Server Requirement	Does not require a server	Requires a RARP server