LOGICAL ADDRESSING

Logical addressing is implemented by network layer.

Logical addressing is a Global Addressing scheme.

Data-link layer handles the addressing problem locally, but if packets passes the network boundary there is a need for logical addressing system to help distinguish source and destination systems.

The network layer adds a header to the packet coming from the upper layer that includes the logical addresses of the sender and receiver.

There are 2 types of addressing mechanisms are present:

1. IPv4 (IPversion4)
2. IPv6 (IP version6)

IPv4 ADDRESSES

An **IPv4** address is a **32-bit** address that **uniquely** and **universally** defines the connection of a device to the Internet.

Unique: Two devices on the Internet can never have the same address at the same time.

Universal: The addressing system must be accepted by any host that wants to be connected to the Internet.

Address Space

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

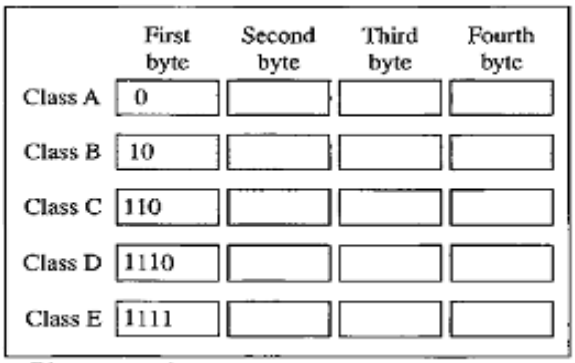
Notations

There are two notations to show an IPv4 address: Binary and Dotted-Decimal Notation.

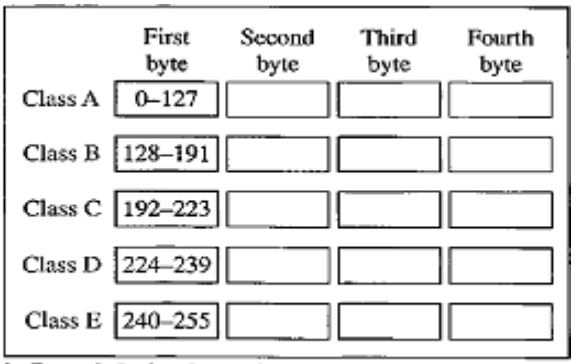
Binary	Dotted-Decimal
• IPv4 address is displayed as 32 bits. Each octet is often referred to as a byte. • It is a 4 byte address Ex: 10000000 00001011 00000011 00011111	• Internet addresses are written in decimal form with a dot separating the bytes. • Each number in dotted-decimal notation is a value ranging from 0 to255. Ex: 128.11.3.31

CLASSFUL ADDRESSING

- Initially IPv4 used the concept of Classful addressing.
- In classful addressing, the address space is divided into five classes: A, B, C, D, and E.
- If the address is given in binary notation, the first few bits can immediately tell us the class of the address.
- If the address is given in decimal-dotted notation, the first byte defines the class.



a. Binary notation



b. Dotted-decimal notation

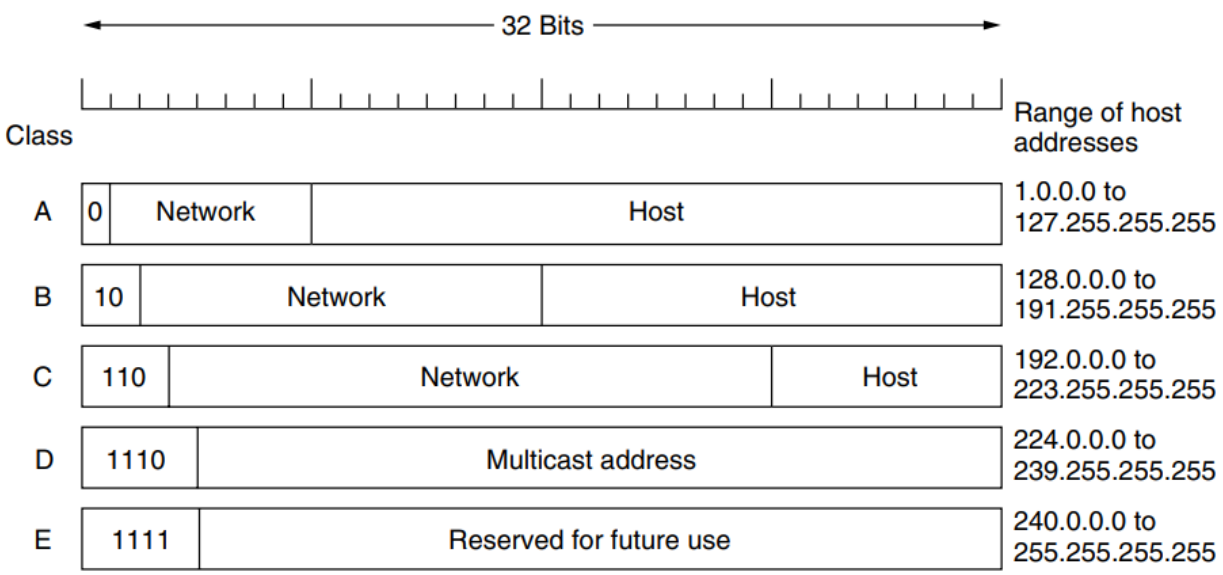
Classes and Blocks

- Each class is divided into a fixed number of blocks.
- Size of the each block is also fixed.

Class	No of Blocks	Block Size	Application
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved

Purpose of classes:

- **Class A** addresses were designed for large organizations with a large number of attached hosts or routers but most of the addresses in class A were wasted and were not used.
- **Class B** addresses were designed for midsize organizations with tens of thousands of attached hosts or routers.
- **Class C** addresses were designed for small organizations with a small number of attached hosts or routers.
- **Class D** addresses were designed for multicasting, each address in class D is used to define one group of hosts on the Internet.
- **Class E** addresses were reserved for future use.


32 Bits
Class
Range of host addresses
A
0
Network
Host
1.0.0.0 to 127.255.255.255
B
10
Network
Host
128.0.0.0 to 191.255.255.255
C
110
Network
Host
192.0.0.0 to 223.255.255.255
D
1110
Multicast address
224.0.0.0 to 239.255.255.255
E
1111
Reserved for future use
240.0.0.0 to 255.255.255.255

Netid and Hostid

- In classful addressing, an IP address in class A, B, or C is divided into Network id and Host id.

Mask or Default Mask

- A default mask is a 32-bit number made of contiguous 1's followed by contiguous0's.
- The mask can help us to find the Netid and the Hostid.
- For example, the mask for a class A address has eight 1s, which means the first 8 bits of any address in class A define the Netid; the next 24 bits define the Hostid.

The masks for classes A, B, and C are:

Class	Binary	Dotted-Decimal	CIDR
A	**11111111** 00000000 00000000 00000000	**255**.0.0.0	**/8**
B	**11111111 11111111** 00000000 00000000	**255.255**.0.0	**/16**
C	**11111111 11111111 11111111** 00000000	**255.255.255**.0	**/24**

Subnetting

- Subnetting is a process of dividing a large block into smaller contiguous groups and assigns each group to smaller networks (subnets) or share a part of the addresses with neighbors.
- Subnetting increases the number of 1's in the mask.

Supernetting

- In supernetting, an organization can combine several blocks to create a larger range of addresses. Supernetting decreases the number of 1's in the mask.

Example: an organization that needs 1000 addresses can be granted four contiguous class C blocks. The organization can then use these addresses to create one super network.

Address Depletion

- The number of available IPv4 addresses is decreasing as the number of internet users are increasing.
- We have run out of class A and B addresses, and a class C block is too small for most midsize organizations.
- One solution that has alleviated the problem is the idea of **Classless Addressing**.

CLASSLESS ADDRESSING

Purpose

- Classless addressing was designed and implemented to overcome address depletion and give more organizations access to the Internet.
- In this scheme, there are no classes, but the addresses are still granted in blocks.

Address Blocks

- In classless addressing, when a small or large entity, needs to be connected to

the Internet, it is granted a block of addresses.

- The size of the block (the number of addresses) varies based on the nature and size of the entity.

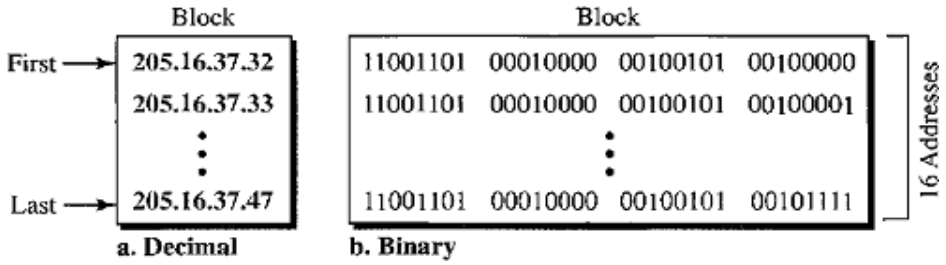
Example:

- For a large organization may be given thousands of addresses
- For a house two addresses are sufficient
- An Internet service provider may be given hundreds of thousands based on the number of customers it may serve.

Restrictions on classless address blocks

1. The addresses in a block must be **contiguous**, one after another.
2. The number of addresses in a block must be a power of 2 (1, 2, 4, 8, ...).
3. The first address must be evenly divisible by the number of addresses.

Consider the below figure for classless addressing that shows a block of addresses, in both binary and dotted-decimal notation, granted to a small business that needs 16 addresses.



It satisfies all 3 restrictions:

- The addresses are contiguous.
- The number of addresses is a power of 2 (16 =2^4).
- The first address is divisible by 16. The first address, when converted to a decimal number, is 3,440,387,360, which when divided by 16 results in215,024,210.

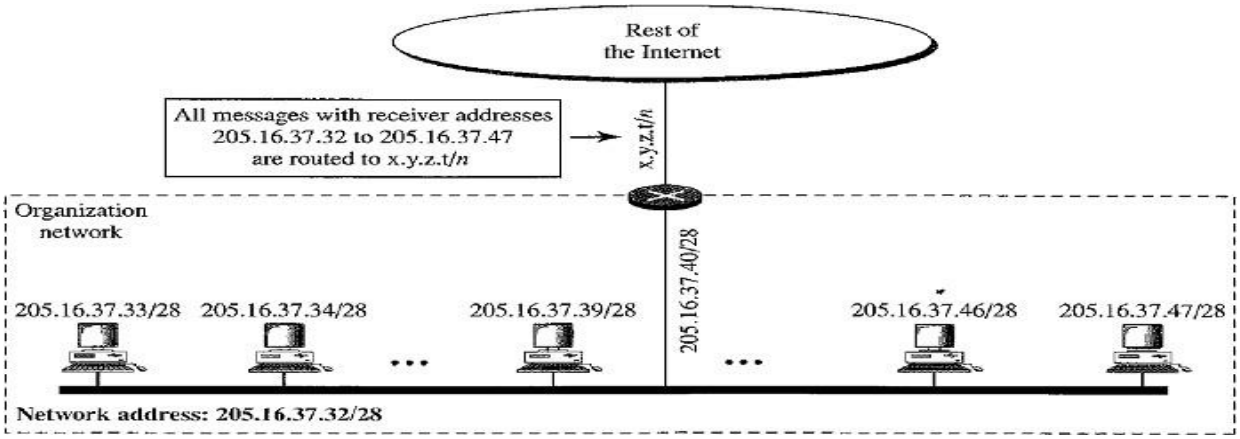
Mask

A mask is a 32-bit number in which the ***n*** leftmost bits are 1's and the **32 - *n*** rightmost bits are 0's, where ***n= 0 to32.***

In 1Pv4 addressing, a block of addresses can be defined as **x.y.z.t/n** in which **x.y.z.t** defines one of the addresses and the ***/n*** defines the mask. ***/n*** is called as CIDR notation.

- **First Address** in the block can be found by setting the rightmost **32 - *n*** bits to0's.
- **Last Address** in the block can be found by setting the rightmost **32 - *n*** bits to1's.
- **Number of Addresses** in the block can be found by using the formula 2^{32-n}.

- **Network Address** is the first address in the block and defines the organization network. Usually the first address is used by routers to direct the message sent to the organization from the outside.



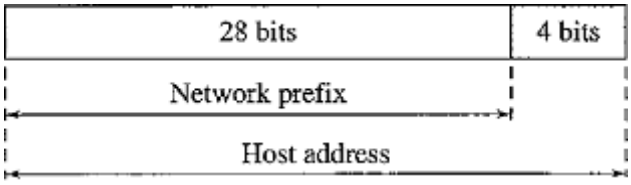
Example: **205.16.37.39/28** or **11001101 00010000 00100101 00100111**

- First address: 11001101 000100000100101 0010000 or 205.16.37.**32**
- Last address: 11001101 00010000 001001010010 1111 or205.16.37.**47**
- Number of Addresses: $\mathbf{2^{32-28} = 2^4=16.}$
- Network Address (First Address) 11001101 000100000100101 0010000 or205.16.37.32

Netid and Hostid

- The ***n*** leftmost bits of the address **x.y.z.t/n** define the **network address** or **prefix**.
- The **(32 – *n*)** rightmost bits define the particular **suffix** or **host address** (computer or router) connected to the network.

205.16.37.39/28 or **11001101 00010000 00100101 0010 0111**



Network Address Translation (NAT)

- As the number of home users and small business users are increasing day by day it is not possible to give each and every user to one IPv4 address due to shortage of IPv4 addresses.
- In order to overcome this problem the developers designed the concepts of private IP address and Network Address Translation(NAT).
- **NAT** enables a user to have a large set of addresses internally (private IP addresses) and a small set of addresses externally (public IP addresses).

The Internet authorities have reserved three sets of addresses as private addresses:

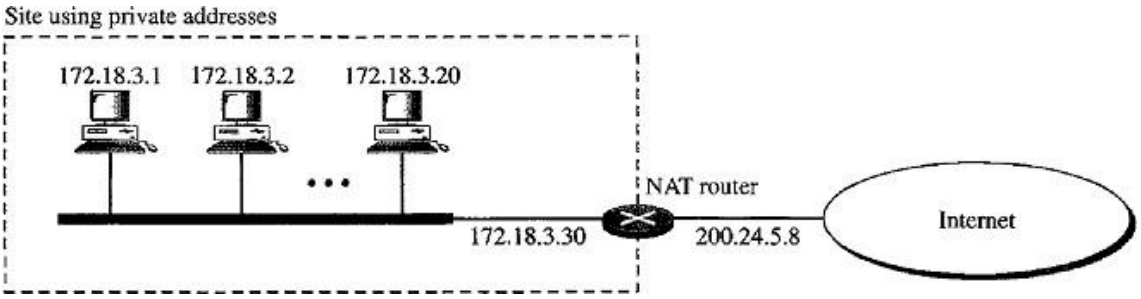
Range	Total
10.0.0.0 to 10.255.255.255	2^{24}
172.16.0.0 to 172.31.255.255	2^{20}
192.168.0.0 to 192.168.255.255	2^{16}

Any organization can use an address out of this set without permission from the Internet authorities.

- They are unique inside the organization, but they are not unique globally. No router will forward a packet that has one of these addresses as the destination address.

Example:

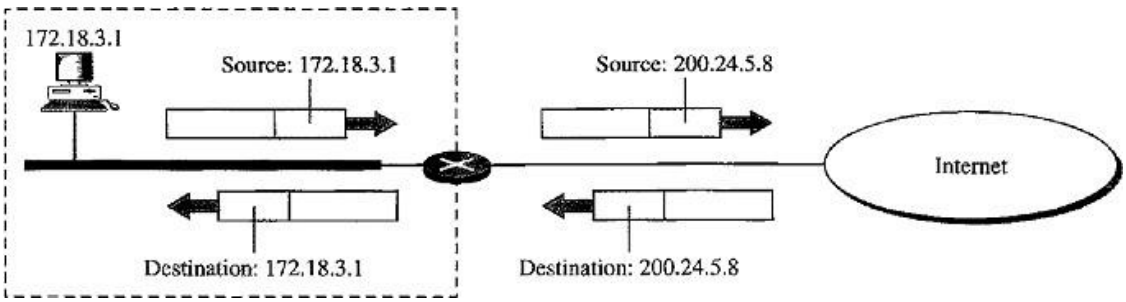
Consider the below figure describes the private network with private addresses:



- The NAT router has one public address **200.24.5.8,** it is a global address.
- The internal devices having addresses from **172.18.3.1** to **172.18.3.30** these are local addresses.

Address Translation

- Alltheoutgoingpacketsgothroughthe NATrouter,whichreplacesthe**sourceaddress** 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 are two types of translation tables:

1. Two Column translation table (Using one IP address)
2. Five column translation table (Using IP addresses and Port Numbers)

Two Column Translation Table

- It contains two columns: Private Address and External Address.
- In this strategy, communication must always be initiated by the private network.
- 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.

Translation Table

Private	External
172.18.3.1	25.8.2.10
…	….
….	….

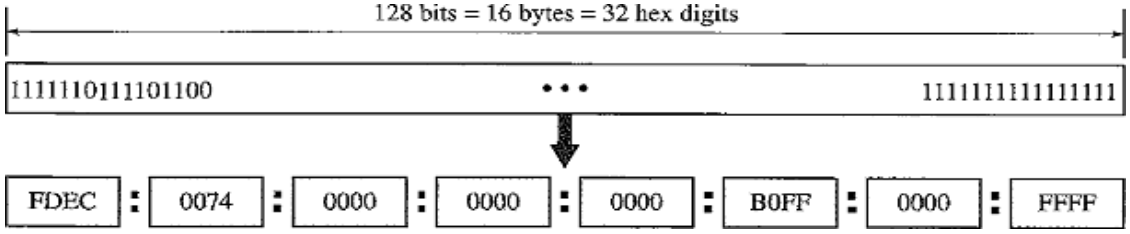
IPv6 ADDRESSES (Internetworking Protocol version6)

Why IPv6?

- In order to overcome the problems of address depletion.
- It eliminates the concept of NAT and Private Addresses.
- There is no need for classless addressing and DHCP.

Structure

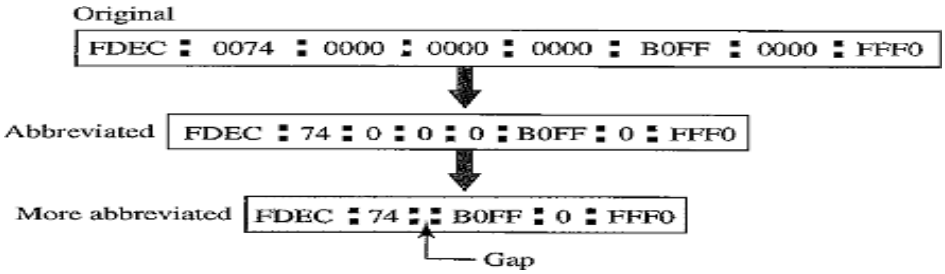
IPv6 specifies hexadecimal colon notation (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F).
An IPv6 address consists of 16 bytes (octets); it is 128 bits long. 128 bits is divided into eight sections. Each of 4 hex digits separated by a colon.



Abbreviation

- Hexa decimal format of IPv6 is very long and many of the digits are zeros.
- We can abbreviate this address as the leading zeros of a section (four digits between two colons) can be omitted.

Note: Only leading zeros are omitted not
trailing zeros. Example:



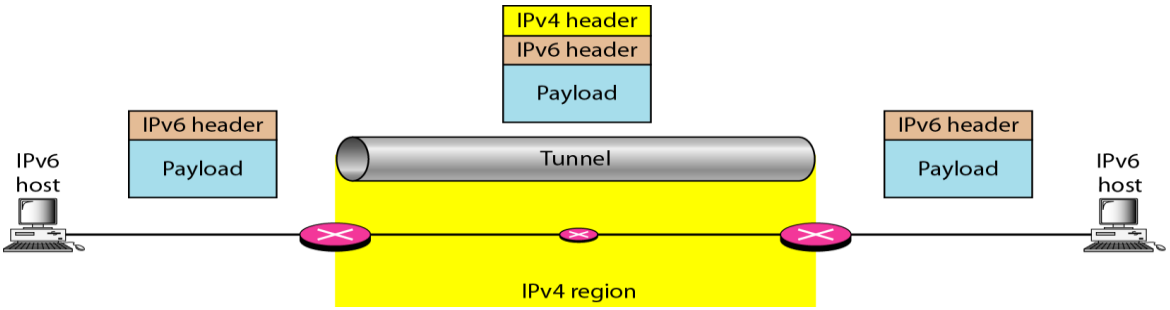
- In the above example: 0074 written as 74, 0000 written as 0.
- If there are consecutive sections consisting of zeros only. We can remove the zeros altogether and replace them with a double semicolon.

Address Space

- IPv6 has 2^{128} addresses are available. It is a much larger address space than IPv4

Tunneling

- Tunneling is a strategy used when two computers using IPv6 want to communicate with each other and the packet must pass through a region that usesIPv4.
- To pass through this region, the packet must have an IPv4address.
- So the IPv6 packet is encapsulated in an IPv4 packet when it enters the region, and it leaves its capsule when it exits the region.
- It seems as if the IPv6 packet goes through a tunnel at one end and emerges at the other end. The IPv4 packet is carrying an IPv6 packet as data, the protocol value is set to41.



IPv4 Delivery Mechanism

- IPv4 delivery mechanism is used in TCP/IP protocols.
- IPv4 is an unreliable and connectionless datagram protocol.
- If reliability is important, IPv4 must be paired with a reliable protocol such as TCP.

Datagram

Packets in the IPv4 layer are called datagrams.

A datagram is a variable-length packet consisting of two parts:

1. Header
2. Data

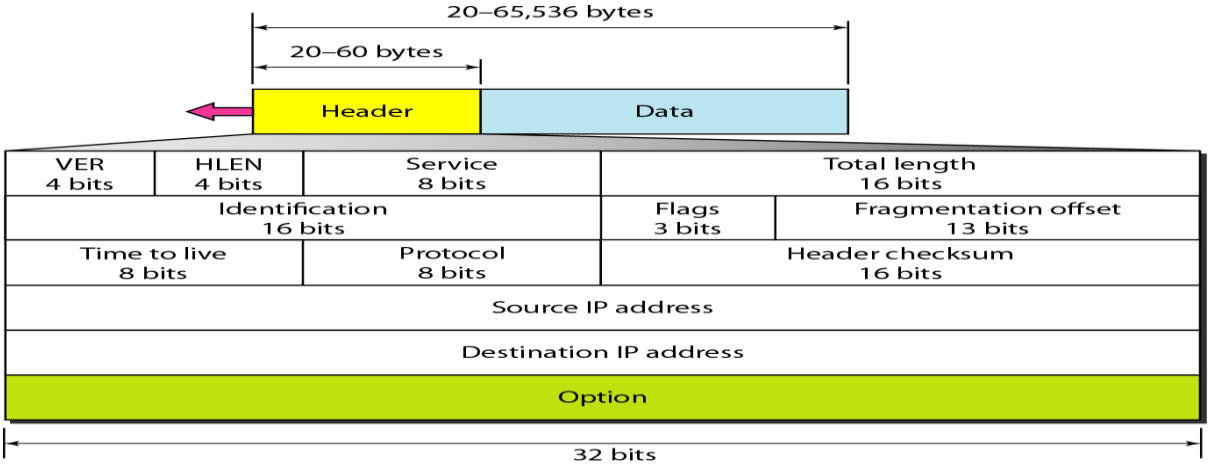
The header is 20 to 60 bytes in length and contains information essential to routing and delivery.

1. Version (VER) – 4bits

- It defines the version of the IPv4 protocol. Currently 4th version of IPv4 is using.

2. Header length (HLEN) – 4bits

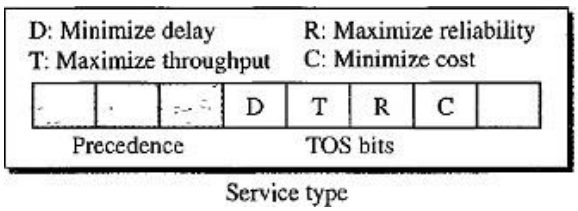
- It defines the total length of the datagram header in 4-bytewords.
- The length of the header is variable between 20 and 60bytes.
- When there are no options, the header length is 20 bytes, and the value of this field is 5 (5x 4 =20).
- When the option field is at its maximum size, the value of this field is 15 (15 x 4 = 60).



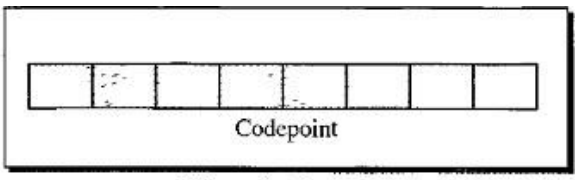
3. Services

IETF has changed the interpretation and name of this 8-bit field.

Previously this field is called **Service Type** but now the name changed to **Differentiated Services**.



Service type



Differentiated services

Service Type

In this interpretation, the first 3 bits are called precedence bits. The next 4 bits are called type of service (TOS) bits, and the last bit is not used.

i. Precedence

- It is a 3-bit subfield ranging from 0 to 7 (000 to111).
- The precedence defines the priority of the datagram in issues such as congestion.
- If a router is congested and needs to discard some datagrams, those datagrams with lowest precedence are discarded first.

ii. Type of Service(TOS)

It is a 4-bit subfield with each bit having a special meaning. Out of 4 bits only one bit will have the value of 1.

TOS Bits	Description
0000	Normal (default)
0001	Minimize Cost
0010	Maximize Reliability
0100	Maximize Throughput
1000	Minimize Delay

Differentiated Services

In this interpretation, the first 6 bits make up the code-point subfield, and the last 2 bits are not used. The code-point subfield can be used in two different ways:

a. When the 3 rightmost bits are 0's, the 3 leftmost bits are interpreted the same as the precedence bits in the service type interpretation.

b. When the 3 rightmost bits are not all Os, the 6 bits define 64 services based on the priority assignment by the Internet or local authorities.

Category	**Code-point**	**Assigning Authority**	**No of service types**	**Numbers**
1	XXXXX0	Internet	32	0,2,4,6,8,……60,62
2	XXXX11	Local	16	3,7,11,15,……59,63
3	XXXX01	Temporary or Experiment	16	1,5,9,13,17...……,61

4. Total length

This is a 16-bit field that defines the total length (**header plus data**) of the IPv4 datagram in bytes. Total length of IPv4 is 65,535 (2^{16}-1).

Length of data = Total length - header length

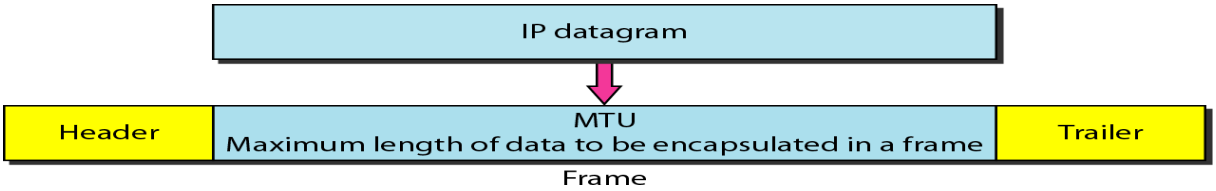
FRAGMENTATION

- A datagram is fragmented if it is too large for a network to carry it.

Maximum Transfer Unit (MTU)

- the Total size of the datagram must be less than this maximum size.

- The maximum length of IPV4 datagram is 65,535bytes.



- If the length of the datagram exceeds the MTU then the datagram must be fragmented to make it possible to pass through the networks.
- When a datagram is fragmented, each fragment has its own header with only few fields are changed. Remaining fields are copied by all fragments.

Fields Related to Fragmentation: Identification, Flag, Offset.

Identification (16 bits)

- When a datagram is fragmented, all fragments have the same identification number the same as the original datagram. All fragments having the same identification value must be assembled into one datagram.
- The identification number helps the destination in reassembling the datagram.

Flags (3 bits)

The first bit is **Reserved**.

The second bit is called the **Do Not Fragment** bit.

- If its value is 1, the machine must not fragment the datagram.
- If its value is 0, the datagram can be fragmented if necessary.

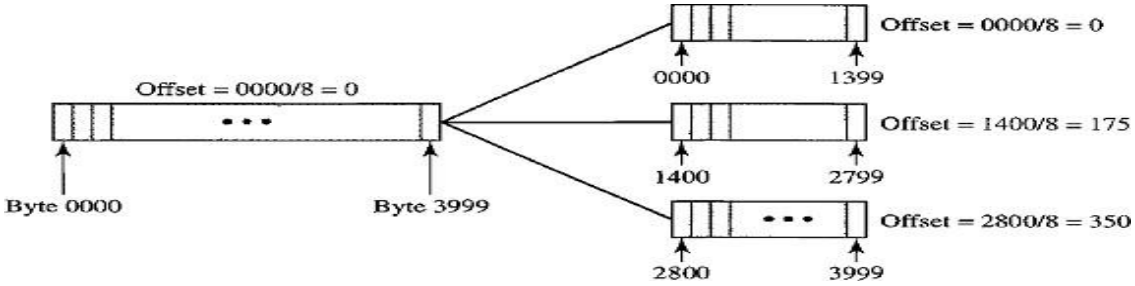
The third bit is called the **More Fragment** bit.

- If its value is 1, it means the datagram is not the last fragment.
- If its value is 0, it means this is the last or only fragment.

Fragmentation offset (13 bits)

It shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes.

Example: Consider the below figure shows a datagram with a data size of 4000 bytes fragmented into three fragments.



The bytes in the original datagram are numbered 0 to 3999.

Fragment Number	Range	Offset Value
First	0-1399	0/8=0
Second	1400-2799	1400/8=175
Third	2800-3999	2800/8=350

Time To Live - TTL (8 bits)

A datagram has a limited lifetime in its travel through an internet. This field can be used in two ways:

i. This field was originally designed to hold a timestamp, which was decremented by each visited router. The datagram was discarded when the value became zero.

ii. This field is used mostly to control the maximum number of hops (routers) visited by the datagram. Each router that processes the datagram decrements this number by 1. The router discards the datagram, if **TTL=0**.
When a source host sends the datagram, it stores a number in TTL field. This value is approximately 2 times the maximum number of routes between any two hosts.

Protocol (8 bits)

- This field defines the higher-level protocol that uses the services of the IPv4layer.
- An IPv4 datagram can encapsulate data from several higher-level protocols such as TCP, UDP, ICMP, and IGMP.
- This field specifies the final destination protocol to which the IPv4 datagram is delivered.

Checksum (16 bits)

The checksum in the IPv4 packet covers only the header, not the data. There are two reasons:

i. All higher-level protocols that encapsulate data in the IPv4 datagram have a checksum field that covers the whole packet. The checksum for the IPv4 datagram does not have to check the encapsulated data.

ii. The header of the IPv4 packet changes with each visited router, but the data do not changes. So the checksum includes only the part that has changed.

Options

Options are not required for a datagram. They can be used for network testing and debugging.

End of Option: An end-of-option option is a 1-byte option used for padding at the end of the option field.
Record Route: A record route option is used to record the Internet routers that handle the datagram.
It can list up to nine router addresses. It can be used for debugging and management purposes.

Strict Source Route: A strict source route option is used by the source to predetermine a route for the datagram as it travels through the Internet. The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput.
If the datagram visits a router that is not on the list, the datagram is discarded and an error message is issued.

Loose Source Route: A loose source route option is similar to the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.

Timestamp: A timestamp option is used to record the time of datagram processing by a router.

Source Address (32 bits) & Destination Address (32 bits)

- These two fields define the IPv4 address of the Source and Destination respectively.
- These fields must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

Disadvantages of IPv4

1. Despite all short-term solutions, such as subnetting, classless addressing, and NAT, address depletion is still a long-term problem in the Internet.
2. The Internet must accommodate real-time audio and video transmission. This type of transmission requires minimum delay strategies and reservation of resources not provided in the IPv4 design.
3. The Internet must accommodate encryption and authentication of data for some applications. No encryption or authentication is provided byIPv4.

IPV6 DELIVERY MECHANISM

- IPV6 is introduced to overcome the deficiencies ofIPv4.
- IPv6 is also called as IPng (Internetworking Protocol next generation).
- In IPv6, the Internet protocol was extensively modified to accommodate the growth of the Internet. Packet format, Length of IP address, ICMP, IGMP, ARP, RARP, RIP routing protocol are also modified inIPv6.

Advantages of IPv6

- **Larger address space** An IPv6 address is 128 bits long whereas IPv4 is 32-bitaddress.
- **Better header format** IPv6 uses a new header format in which options are separated from the base header. when options are needed it is inserted between the base header and the upper-layer data.
- **New options** IPv6 has new options to allow for additional functionalities.
- **Allowance for extension** IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.
- **Resource Allocation** In IPv6 the type-of-service field has been removed, but a mechanism called ***flow label*** has been added to enable the source to request special handling of the packet. This mechanism can be used to support traffic such as real-time audio and video.
- **More Security** The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

Packet Format

In IPv6 each packet is composed of a mandatory base header followed by the payload.
The **Payload** consists of two parts:

- Optional extension headers
- Data from an upper layer

The **Base Header** occupies 40 bytes, whereas the extension headers and data from the upper layer contain up to 65,535 bytes of information.

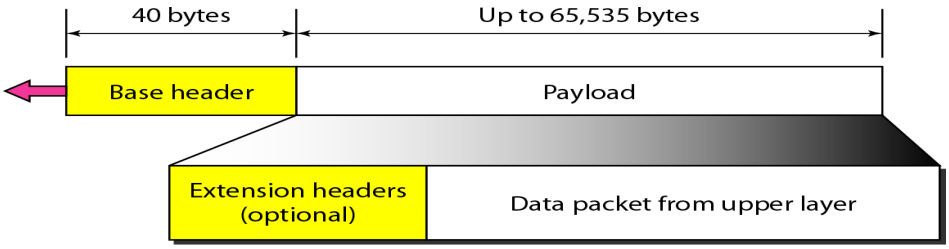
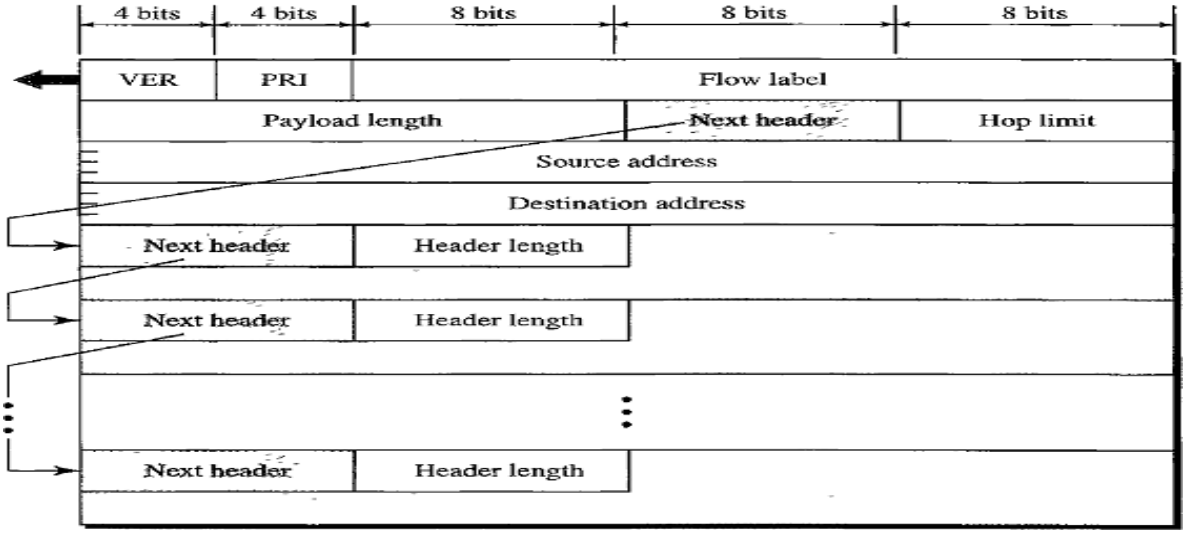


Fig: IPv6 Datagram header and payload

Base Header

Fields in IPv6 datagram are:



- **Version (4-bit)**

  This field defines the version number of the IP. For IPv6, the value is 6.

- **Priority(4-bit)**

  The priority field defines the priority of the packet with respect to traffic congestion.

- **Flow label (24-bit or 3Byte)**

  Flow label field that is designed to provide special handling for a particular flow of data.

- **Payload length (16 bit or 2Byte)**

  Payload length field defines the length of the IP datagram excluding the base header.

- **Next header(8-bit)**

  The next header is an 8-bit field defining the header that follows the base header in the datagram. The next header is either optional extension headers used by IP or the header of TCP or UDP encapsulated packet.

  Note: This field in IPv4 is called the *protocol.*

- **Hop limit (8 bit)**

  Hop limit field serves the same purpose as the TTL field in IPv4

- **Source address (128-bit or 16 Byte) and Destination Address (128 bit or 16Byte)**

  The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram.

  The destination address field is a 16-byte (128-bit) Internet address that usually identifies the final destination of the datagram. If source routing is used, this field contains the address of the next router.

Next Header codes for IPv6:

Code	Next Header
0	Hop-by-hop option
2	ICMP
6	TCP
17	UDP
43	Source Routing
44	Fragmentation
50	Encrypted security payload
51	Authentication
59	Null (No Next Header)
60	Destination Option

Priority

The priority field of the IPv6 packet defines the priority of each packet with respect to other packets from the same source.

Example: If one of two consecutive datagrams must be discarded due to congestion, the datagram with the lower **packet priority** will be discarded.

IPv6 divides traffic into two broad categories:

i. Congestion-Controlled
ii. Non congestion-controlled

Congestion-Controlled Traffic

When there is congestion a source adapts itself to slowdown the traffic.

Example: TCP uses sliding window protocol can easily respond to the traffic.

Congestion-controlled data are assigned priorities from 0 to 7

Priority	Meaning	Description
0	No specific traffic	Priority 0 is assigned to a packet when the process does not define a priority.
1	Background data	defines data that are usually delivered in the background. Ex: Delivery of the news.
2	Unattended data traffic	If the user is not waiting (attending) for the data to be received, the packet will be given a priority of 2. Ex: Email
3	Reserved	
4	Attended bulk data traffic	A protocol that transfers data while the user is waiting to receive the data is given a priority of 4 Ex: FTP and HTTP
5	Reserved	
6	Interactive traffic	Protocols that need user interaction are assigned 6. Ex: TELNET
7	Controlled traffic	Routing Protocols are given Highest Priority 7. Ex: OSPF, RIP, SNMP

Non congestion-Controlled Traffic

- The source does not adapt itself to congestion. It is a type of traffic that expects minimum delay. Priority numbers from 8 to 15 are assigned to Non congestion-controlled traffic.
- In this traffic Discarding of packets is not desirable and Retransmission in most cases is impossible.

Examples: Real-time audio and video.

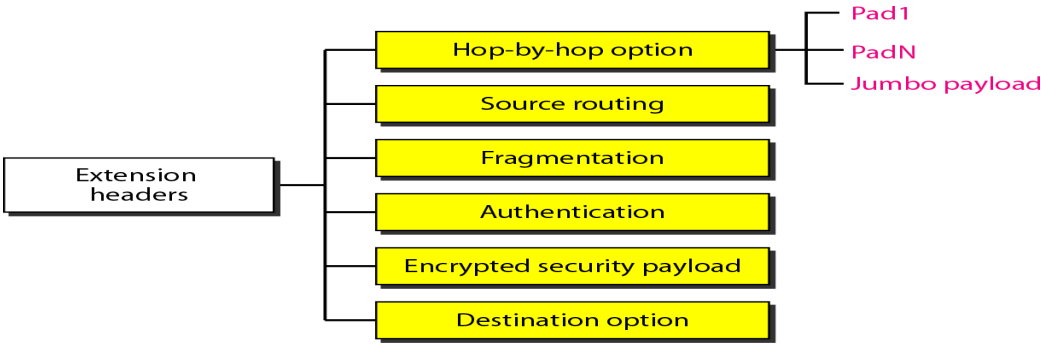
- **Priority 15:** It is given to data containing **Less Redundancy** (low-fidelity audio or video)
- **Priority 8:** It is given to data containing M**ore Redundancy** (high-fidelity audio or video)

Flow Label

- A sequence of packets, sent from a particular source to destination that needs special handling by routers is called a F**low of packets.**
- The combination of the source address and the value of the **Flow Label** uniquely define a flow of packets.
- To a router, a flow is a sequence of packets that share the same characteristics such as traveling the same path, using the same resources, having the same kind of security etc.
- A router that supports the handling of flow labels has a flow label table. The table has an entry for each active flow label.
- Each entry defines the services required by the corresponding flow label.
- When a router receives a packet it consults the flow label table instead of consulting the routing table and going through a routing algorithm to define the address of the next hop, it can easily look in a flow label table for the next hop.
- This mechanism speed up the processing of a packet by a router.

Extension Headers

To give greater functionality to the IP datagram, the base header can be followed by up to six types of extension headers.



Hop-by-Hop Option

The hop-by-hop option is used when the source needs to pass information to all routers visited by the datagram.

Only three options have been defined: Pad l, Pad N, and jumbo payload.

- The Pad l option is 1 byte long and is designed for 1 byte alignment purposes.
- Pad N is used when 2 or more bytes is needed for alignment.
- The jumbo payload option is used to define a payload longer than 65,535bytes.

Source Routing

- The source routing extension header combines the concepts of the strict source route and the loose source route options ofIPv4.

Fragmentation

- In IPv4, the source or a router is required to fragment if the size of the datagram is larger than the MTU of the network over which the datagram travels.
- In IPv6, only the original source can fragment. A source must use a path MTU discovery technique to find the smallest MTU supported by any network on the path.
- The source then fragments using this knowledge.

Authentication

- The authentication extension header has a dual purpose: it validates the message sender and ensures the integrity of data.

Encrypted Security Payload (ESP)

- ESP is an extension that provides confidentiality and guards against eavesdropping.

Destination Option

- It is used when the source needs to pass information to the destination only.
- Intermediate routers are not permitted access to this information.

Comparison between IPv4 Options and IPv6 Extension Headers

1. The no-operation and end-of-option options in IPv4 are replaced by Pad1 and Pad N options in IPv6.
2. The record route option is not implemented in IPv6 because it was not used.
3. The timestamp option is not implemented because it was not used.
4. The source route option is called the source route extension header inIPv6.
5. The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header inIPv6.
6. The authentication and Encrypted Security Payload extension headers are new inIPv6.

ADDRESS MAPPING

The internet is made of a combination of physical networks connected by internetworking devices such as routers. A packet starting from a source host may pass through several different physical networks before finally reaching the destination host. The hosts and routers are recognized at the **network level** by their **logical (IP) addresses**, while at the **physical level**, theyare recognized by their **physical (MAC) addresses.**

Thus delivery of a packet to a host or a router requires **two levels** of **addressing**: **logical (IP)** and **physical (MAC)**.

We need to be able to map a logical address to its corresponding physical address and vice-versa.These can be done by using either static or dynamic mapping.

Static mapping

Static mapping involves the creation of a table that associates a logical address with a physical address.

Limitations:

- A machine could change its NIC (Network Interface Card), resulting in a new physicaladdress.
- In some LANs, such as LocalTalk, the physical address changes every time the computeris turned on.
- A mobile computer can move from one physical network to another, resulting in achange in its physical address.

To implement these changes, a static mapping table must be updated periodically. This overhead could affect network performance.

Dynamic mapping

In such mapping each time a machine knows one of the two addresses (logical or physical), itcan use a protocol to find the other one.

Mapping Logical to Physical Address: ARP

ARP stands for **Address Resolution Protocol** which is one of the most important protocols of the Network layer in the OSI model. ARP finds the physical address, also known as Media Access Control (MAC) address, of a host from its known IP address Figure 21.2.

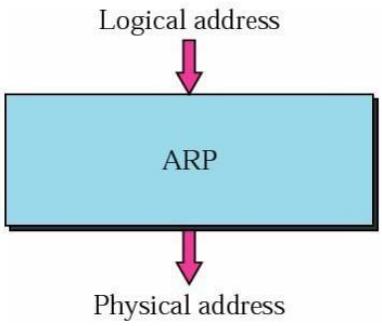


Figure 21.2: ARP Mapping

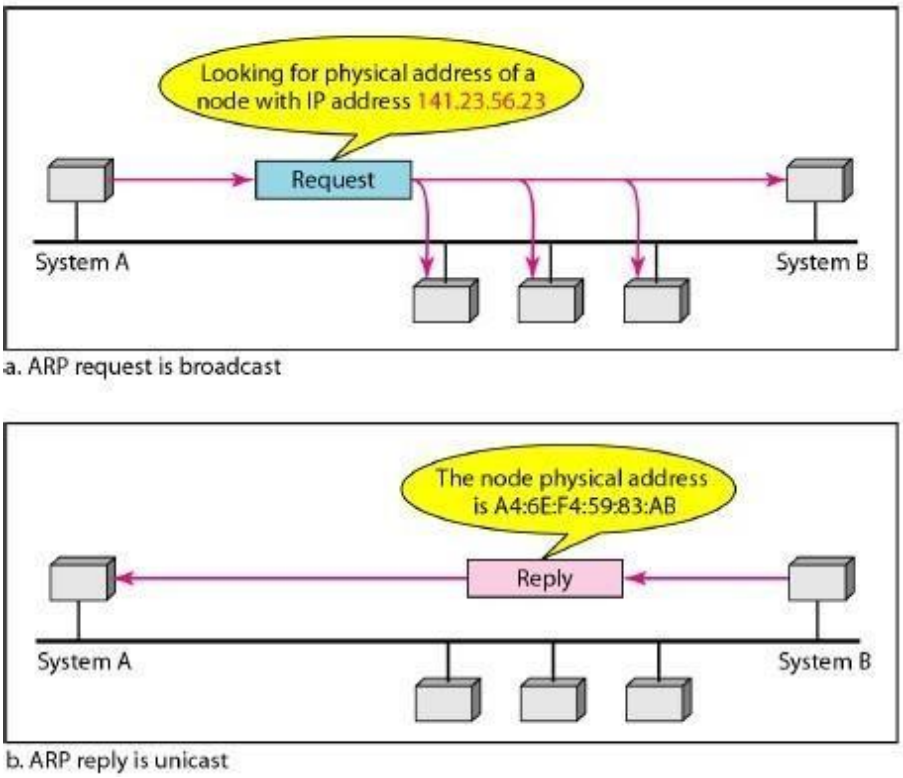


Figure 21.3 ARP operation

Following **steps** are involved in logical to physical address mapping:

a. The host or the router sends an **ARP query packet**. The ARP query packet includes the physical and IP addresses of the sender and the IP address of the receiver. As the sender does not know the physical address of the receiver, the **ARP query is broadcast over the network** (see Figure 21.3).

b. Every host or router on the network receives and processes the ARP query packet, but only the intended recipient recognizes its IP address and sends back an **ARP response packet**.

c. The ARP response packet contains the recipient's IP and physical addresses. The ARP response packet is **unicast directly to the inquirer** (host/router) by using the physical address received in the query packet.

Example: (Figure 21.3)The system on the left (A) has a packet that needsto be delivered to another system (B) with IP address 141.23.56.23.
System A needs to pass the packet to its data link layer for the actual delivery, but it does not know the physical address of the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of 141.23.56.23. This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 21.3b.
System B sends an ARP reply packet that includes its physical address.

Now system A can send all the packets it has for this destination by using the physical address it received.

Cache Memory

Using ARP is inefficient if system A needs to broadcast an ARP request for each IP packet it needs to send to system B. ARP can be useful if the ARP reply is cached (kept in cache memory for a while) because a system normally sends several packets to the same destination. A system that receives an ARP reply stores the mapping in the cache memory and keeps it for 20 to 30 minutes unless the space in the cache is exhausted. Before sending an ARP request, the system first checks its cache to see if it can find the mapping.

ARP Packet Format

Figure 21.4 shows the format of an ARP packet.

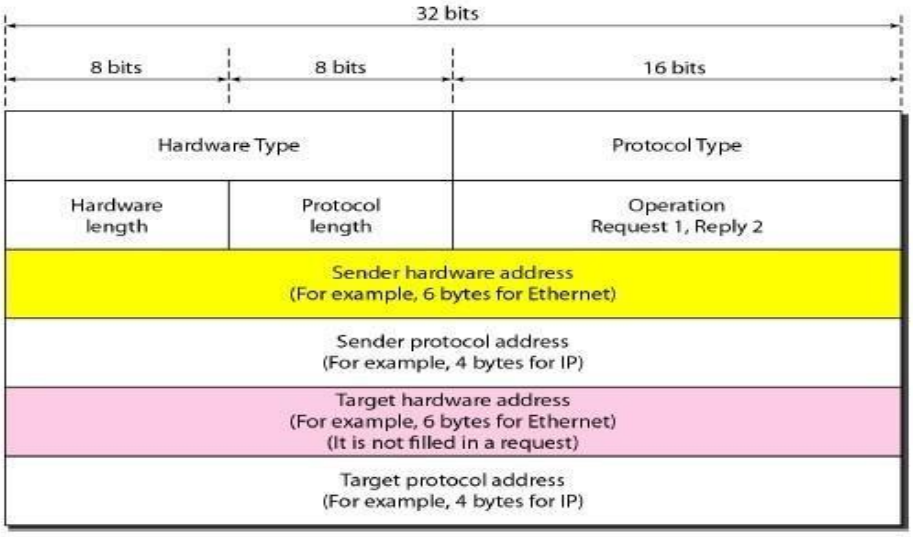


Figure 21.4 ARP packet

The fields are as follows:

a. **Hardware type.** This is a 16-bit field defining the type of the network on which ARP is running. Each LAN has been assigned an integer based on its type. For **example**, Ethernet is given type 1. ARP can be used on any physical network.

b. **Protocol type.** This is a 16-bit field defining the protocol. For example, the value of this field for the IPv4 protocol is 080016, ARP can be used with any higher-level protocol.

c. **Hardware length**. This is an 8-bit field defining the length of the physical address in bytes. For example, for Ethernet the value is 6.

d. **Protocol length**. This is an 8-bit field defining the length of the logical address in bytes. For example, for the IPv4 protocol the value is 4.

e. **Operation**. This is a 16-bit field defining the type of packet. Two packet types are defined: ARP request (1) and ARP reply (2).

f. **Sender hardware address**. This is a variable-length field defining the physical address of the sender. For example, for Ethernet this field is 6 bytes long.

g. **Sender protocol address**. This is a variable-length field defining the logical (for example, IP) address of the sender. For the IP protocol, this field is 4 bytes long.

h. **Target hardware address**. This is a variable-length field defining the physical address of the target. For example, for Ethernet this field is 6 bytes long. For an ARP request message, this field is alI 0s because the sender does not know the physical address of the target.

i. **Target protocol address**. This is a variable-length field defining the logical (for example, IP) address of the target. For the IPv4 protocol, this field is 4 bytes long.

Encapsulation

An ARP packet is encapsulated directly into a data link frame. For example, in Figure 21.5 an ARP packet is encapsulated in an Ethernet frame. Note that the type field indicates that the data carried by the frame are an ARP packet.

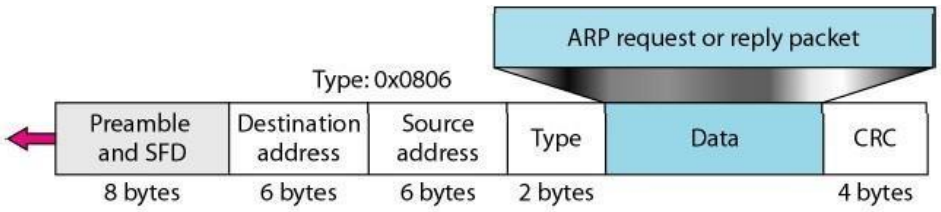


Figure 21.5 Encapsulation of ARP packet

ARP Operation

Let us see how ARP functions on a typical internet. First we describe the steps

involved. Then we discuss the four cases in which a host or router needs to use ARP. These are the steps involved in an ARP process:

1. The sender knows the IP address of the target. We will see how the sender obtains thisshortly.

2. IP asks ARP to create an ARP request message, filling in the sender physical address, thesender IP address, and the target IP address. The target physical address field is filled with 0s.

3. The message is passed to the data link layer where it is encapsulated in a frame by using the physical address of the sender as the source address and the physical broadcast address as the destination address.

4. Every host or router receives the frame. Because the frame contains a broadcast destination address, all stations remove the message and pass it to ARP. All machines except the one targeted drop the packet. The target machine recognizes its IP address.

5. The target machine replies with an ARP reply message that contains its physical address. The message is unicast.

6. The sender receives the reply message. It now knows the physical address of the targetmachine.

7. The IP datagram, which carries data for the target machine, is now encapsulated in a frameand is unicast to the destination.

Proxy ARP

A proxy ARP is an ARP that acts on behalf of a set of hosts. Whenever a router running a proxy ARP receives an ARP request looking for the IP address of one of these hosts, the router sends an ARP reply announcing its own hardware (physical) address. After the router receives the actual IP packet, it sends the packet to the appropriate host or router. Let us give an example.In Figure 21.8 the ARP installed on the right-hand host will answer only to an ARP request witha target IP address of 141.23.56.23.

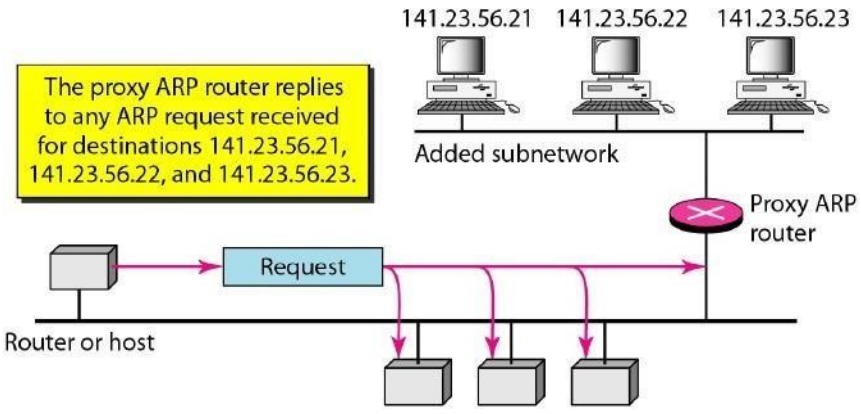


Figure 21.8 Proxy

However, the administrator may need to create a subnet without changing the whole system to recognize subnetted addresses. One solution is to add a router running a proxy ARP. In this case, the router acts on behalf of all the hosts installed on the subnet. When it receives an ARP request with a target IP address that matches the address of one of its host (141.23.56.21, 141.23.56.22, or 141.23.56.23), it sends an ARP reply and announces its hardware address as the target hardware address. When the router receives the IP packet, it sends the packet to the appropriate host.

Mapping Physical to Logical Address: RARP, BOOTP, and DHCP

There are occasions in which a **host knows its physical address, but needs to know its logical address** Figure 21.9. This may happen in **two cases**:

Case 1: *A diskless station is just booted*. The station can find its physical address by checking its interface, but it does not know its IP address.

Case 2: *An organization does not have enough IP addresses to assign to each station*; it needs to assign IP addresses on demand. The station can send its physical address and ask for a short time lease.

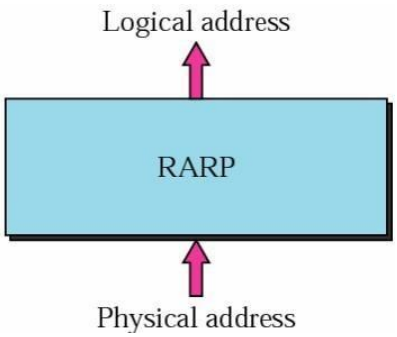


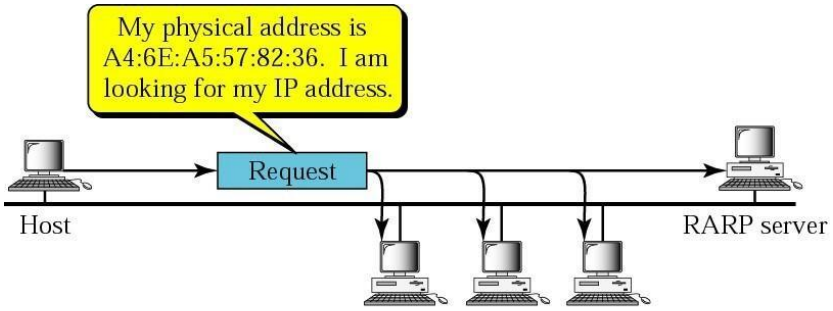
Figure 21.9: RARP Mapping

RARP

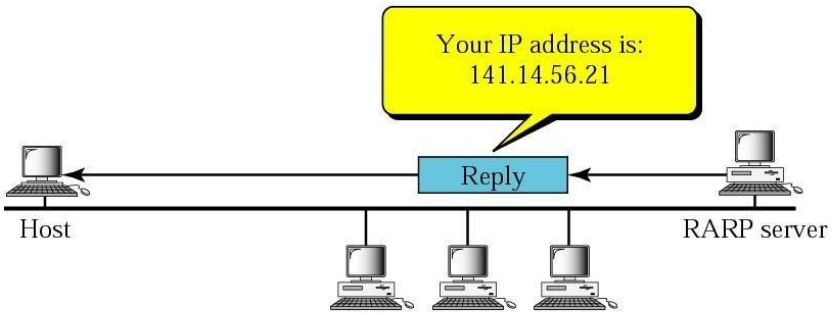
Reverse Address Resolution Protocol (RARP) finds the logical address for a machine that knows only its physical address. RARP Operation

RARP operation is displayed in Figure 21.10

a. A **RARP request** is created and **broadcast** on the local network.
b. Another machine on the local network that knows all the IP addresses will respond with a **RARP reply**.
c. The requesting machine must be running a **RARP client** program; the responding machine must be running a **RARP server** program.



a. RARP request is broadcast



b. RARP reply is unicast

Figure 21.10 RARP Operation

RARP Packet Format & Encapsulation

The format of the RARP packet is the same as the ARP packet format as displayed in Figure 21.4, except that the Operation field. It's value is 3 for RARP request message and 4 for RARP reply message.

An RARP packet is also encapsulated directly into a data link frame just like ARP packet as displayed in Figure 21.5.

Limitations of RARP:

- As broadcasting is done at the data link layer. The physical broadcast address, all 1's inthe case of Ethernet, does not pass the boundaries of a network.
- This means that if an administrator has several networks or several subnets, it needs toassign a RARP server for each network or subnet.
- This is the reason that **RARP is almost obsolete**.
- Two protocols, BOOTP and DHCP, are replacing RARP.

BOOTP

The Bootstrap Protocol (BOOTP) is a client/server based protocol at application layer, designed to **provide physical address** to **logical address mapping**. The administrator may put the client and the server on the same network or on different networks, as shown in Figure 21.11a and Figure 21.11b respectively. **BOOTP** messages are **encapsulated** in a **UDP packet**, and the UDP packet itself is encapsulated in an **IP packet**, as shown in Figure 21.12.

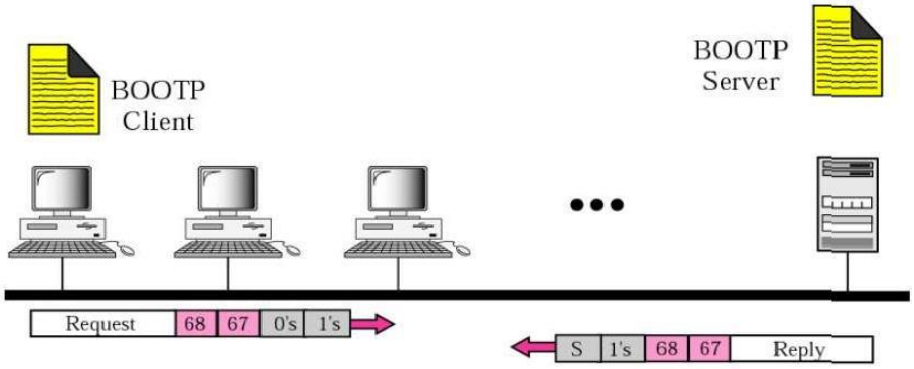


Figure 21.11a BOOTP client and server on the same network

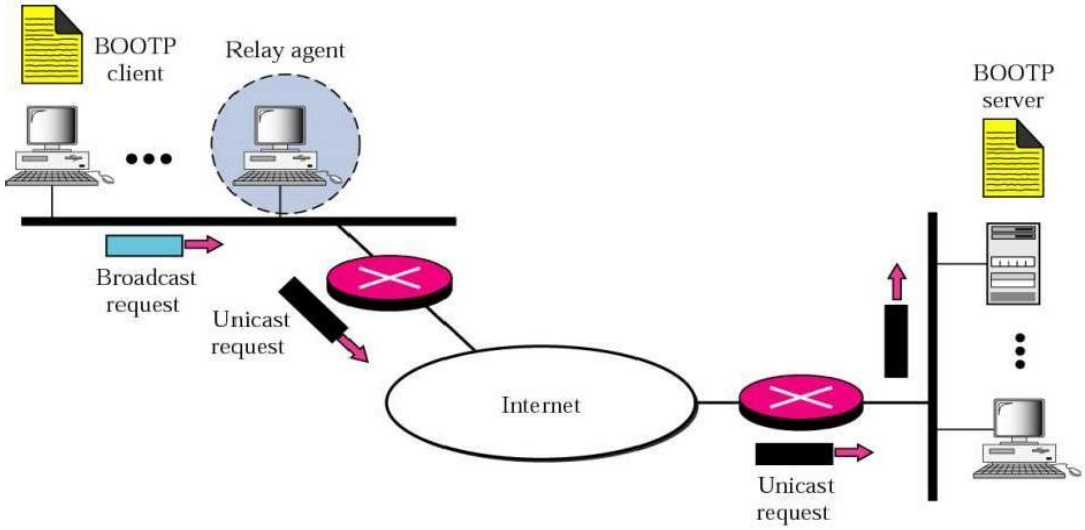


Figure 21.11b BOOTP client and server on the same and different network

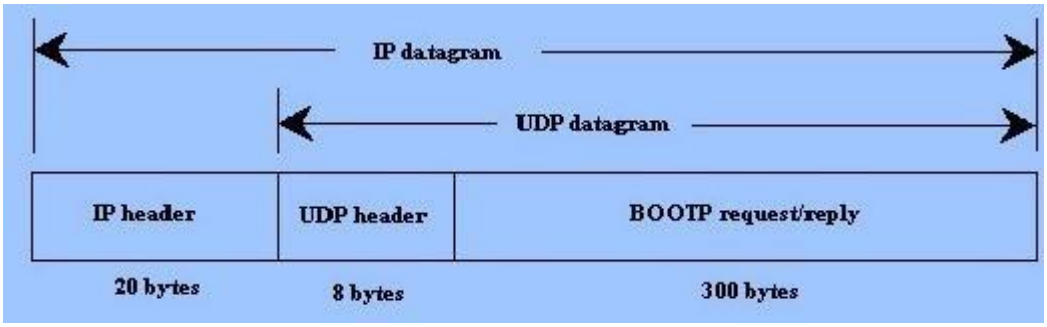


Figure 21.12 BOOTP data Encapsulation

BOOTP Operation

There are two cases of BOOTP operation described below:

Case 1: Client and server on same network (*Figure 21.11a)*

1. When a BOOTP client is started, it has no IP address, so it broadcasts a message containing its MAC address onto the network. This message is called a "BOOTP request," and it is picked up by the BOOTP server, which replies to the client with the following information that the client needs:
    a. The client's IP address, subnet mask, and default gateway address.
    b. The IP address and host name of the BOOTP server.
    c. The IP address of the server that has the boot image, which the client needs to load its operating system.
2. When the client receives this information from the BOOTP server, it configures and initializes its TCP/IP protocol stack, and then connects to the server on which the boot image is shared.

Case 2 : Client and server on different networks(Figure 21.11b**)**

1. If the server exists on some distant network the BOOTP request is broadcast because the client does not know the IP address of the server.
2. The client simply uses all as 0's the source address and all 1's as the destination address.
3. But a broadcast IP datagram cannot pass through any router. To solve the problem, there is a need for an intermediary.
4. One of the hosts in local network (or a router that can be configured to operate at the application layer) can be used as a relay . The host in this case is called a **relay agent**.
5. The relay agent knows the unicast address of a BOOTP server. When it receives this type of packet, it encapsulates the message in a unicast datagram and sends the request to the BOOTP server.
6. The packet, carrying a unicast destination address, is routed by any router and reaches the BOOTP server.
7. The BOOTP server knows the message comes from a relay agent because one of the fields in the request message defines the IP address of the relay agent.
8. BOOTP server sends a BOOTP reply message to the relay agent.
9. The relay agent, after receiving the reply, sends it to the BOOTP client.

4.1.2. Limitations of BOOTP

- BOOTP is **not a dynamic configuration** protocol.
- BOOTP cannot handle these situations because the binding between the physical and IP addresses is static and fixed in a table until changed by the administrator.

DHCP

The **Dynamic Host Configuration Protocol** (DHCP) has been devised to

provide static anddynamic address allocation that can be manual or automatic as required.

- **Static Address Allocation** In this capacity DHCP acts as BOOTP does. It is backward compatible with BOOTP, which means a host running the BOOTP client can request a static address from a DHCP server. A DHCP server has a database that statically binds physical addresses to IP addresses.
- **Dynamic Address Allocation** DHCP has a second database with a pool of available IPaddresses. This second database makes DHCP dynamic. When a DHCP client requests a temporary IP address, the DHCP server goes to the pool of available (unused) IP addresses and assigns an IP address for a negotiable period of time.

- When a DHCP client sends a DHCP request to a DHCP server, the server first checks its static database. If an entry with the requested physical address exists in the static database, the permanent IP address of the client is returned.
- On the other hand, if the entry does not exist in the static database, the server selects an IP address from the available pool, assigns the address to the client, and adds the entry to the dynamic database.
- The dynamic aspect of DHCP is needed when a host moves from network to network oris connected and disconnected from a network (as is a subscriber to a s rvice provider).
- DHCP provides temporary IP addresses for a limited time. The addresses assigned from the pool are temporary addresses.
- The DHCP server issues a lease for a specific time. When the lease expires, the clientmust either stop using the IP address or renew the lease.
- The server has the option to agree or disagree with the renewal. If the server disagrees,the client stops using the address.

DHCP Operation

DHCP provides an automated way to distribute and update IP addresses and other configuration information on a network. A DHCP server provides this information to a DHCP client through the exchange of a series of messages, known as the DHCP conversation or the DHCP transaction displayed in Figure 21.13.

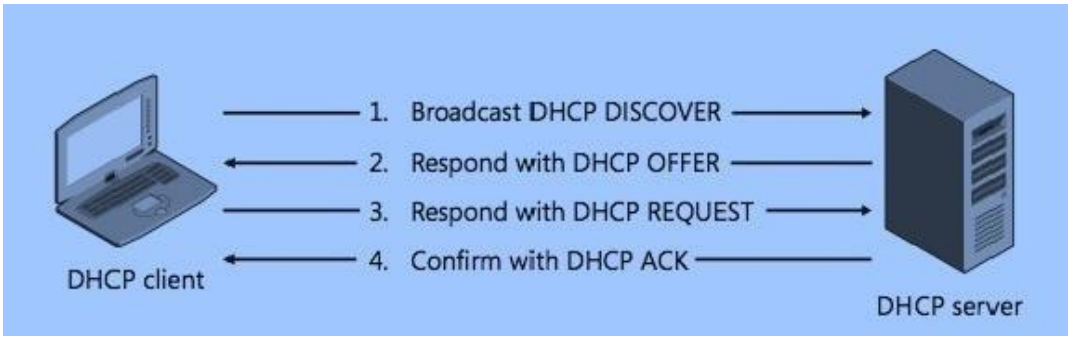


Figure 21.13:

DHCP OperationDHCP client goes through the

f ur step process:

1. A DHCP client sends a broadcast packet (**DHCP Discover**) to discover DHCP servers on the LAN segment.

2. The DHCP servers receive the **DHCP Discover** packet and respond with **DHCP Offer** packets, offering IP addressing information.

3. If the client receives the **DHCP Offer** packets from multiple DHCP servers, the first **DHCP Offer** packet is accepted. The client responds by broadcasting a **DHCP Request** packet, requesting network parameters from a single server.

4. The DHCP server approves the lease with a **DHCP Acknowledgement (DHCP Ack)** packet. The packet includes the lease duration and other configuration information.

1. ICMP

IP provides unreliable and connectionless datagram delivery. It was designed this way to make efficient use of network resources. The IP protocol is a best-effort delivery service that deliversa datagram from its original source to its final destination. However, **IP protocol has two deficiencies:** lack of error control and lack of assistance mechanisms.

- The IP protocol has no error-reporting or error-correcting mechanism.
- What happens if something goes wrong?
- What happens if a router must discard a datagram because it cannot find a router to thefinal destination, or because the time-to-live field has a zero value?
- What happens if the final destination host must discard all fragments of a datagrambecause it has not received all fragments within a predetermined time limit?

These are examples of situations where an error has occurred and the IP protocol has no built-in mechanism to notify the original host.

- The IP protocol also lacks a mechanism for host and management queries.
- A host sometimes needs to determine if a router or another host is alive.
- And sometimes a network administrator needs information from another host or router.

The Internet Control Message Protocol (ICMP) has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol.

1.1. Types of Messages

ICMP messages are divided into two broad categories: **Error-reporting messages and Querymessages**

The **error-reporting messages** report problems that a router or a host (destination) mayencounter when it processes an IP packet.

The **query messages**, which occur in pairs, help a host or a network manager get specificinformation from a router or another host.

1.2. Message Format

An ICMP message has an **8-byte header** and a **variable-size data section**. Although the general format of the header is different for each message type, the first 4 bytes are common to all. As Figure 21.14 shows:

The first field, **ICMP type**, defines the type of the message.

The **code field** specifies the reason for the particular message type.

The last common field is the **checksum field** used for

securing ICMP header.The rest of the header is specific for

each message type.

The **data section** in error messages carries information for finding the original packetthat had the error.

In ICMP query messages, the data section carries extra information based on the type ofthe query.

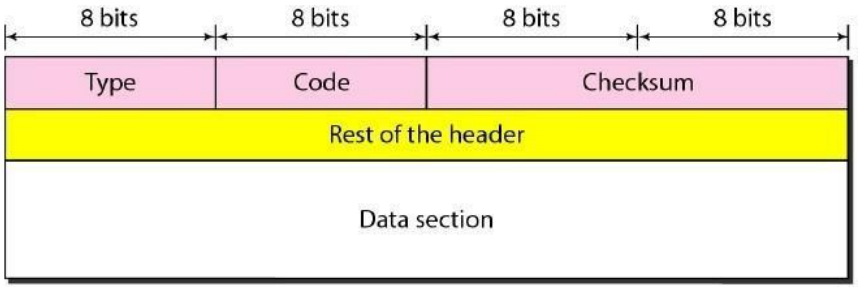


Figure 21.14 General format of ICMP messages

1.3. ICMP Encapsulation:

ICMP itself is a network layer protocol. However it's messages are not passed directly to datalink layer. Instead the messages are first encapsulated inside IP datagrams before going to the lower layer (see Figure 21.15).

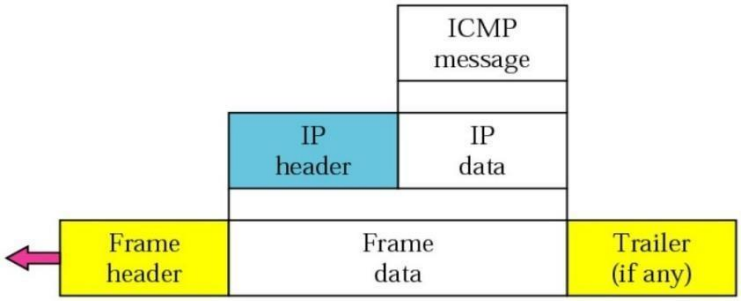


Figure 21.15 Contents of data field for the error messages

1.4. Error Reporting Messages

One of the main responsibilities of ICMP is to report errors.

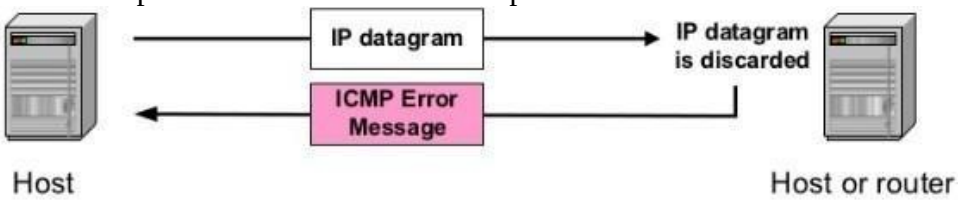


Figure 21.16 ICMP Error Reporting Message

Error messages are typically sent when a datagram is discarded due to some error asdisplayed in Figure 12.16.

Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses.

Five types of errors are handled: *destination unreachable, source quench, timeexceeded,parameter problems, and redirection* (see Figure 21.17).

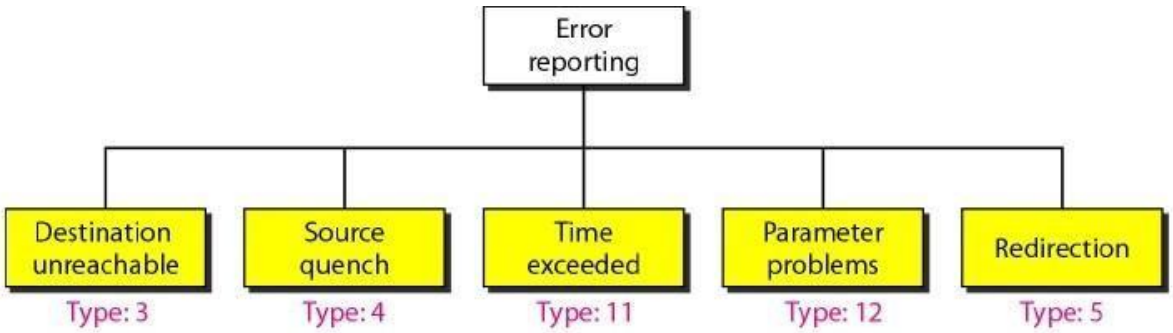


Figure 21.17 Error-reporting messages

a. Destination Unreachable

When a router cannot route a datagram or a host cannot deliver a datagram, the datagram is discarded and the router or the host sends a destination-unreachable message back to the source host that initiated the datagram.

b. Source Quench

- **The source-quench message in ICMP was designed to add a kind of flow control to the IP.**
- When a router or host discards a datagram due to congestion**, it sends a source- quench message to the sender of the datagram**. This message has **two purposes**.
- **First**, it informs the source that the datagram has been discarded.
- **Second**, it warns the source that there is congestion somewhere in the path and that the source should slow down (quench) the sending process.

c. Time Exceeded

The time-exceeded message is generated in **two cases**:

Case1: As routers use routing tables to find the next hop (next router) that must receive the packet. If there are errors in one or more routing tables, a packet can travel in a loopor a cycle, going from one router to the next or visiting a series of routers endlessly. Each datagram contains a field called *time to live* that controls this situation. When a datagram visits a router, the value of this field is decremented by 1. When the time-to-live value reaches 0, after decrementing, the router discards the datagram. However, when the datagram is discarded, a time-exceeded message must be sent by the router to the original source.

Case2: A time-exceeded message is also generated when not all fragments that make upa message arrive at the destination host within a certain time limit.

d. Parameter Problem

Any **ambiguity in the header** part of a datagram can create serious problems as the datagram travels through the Internet. If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter-problem message back to the source.

e. Redirection

- This concept of redirection is shown in Figure 21.18. Host A wants to send a datagram to host B.

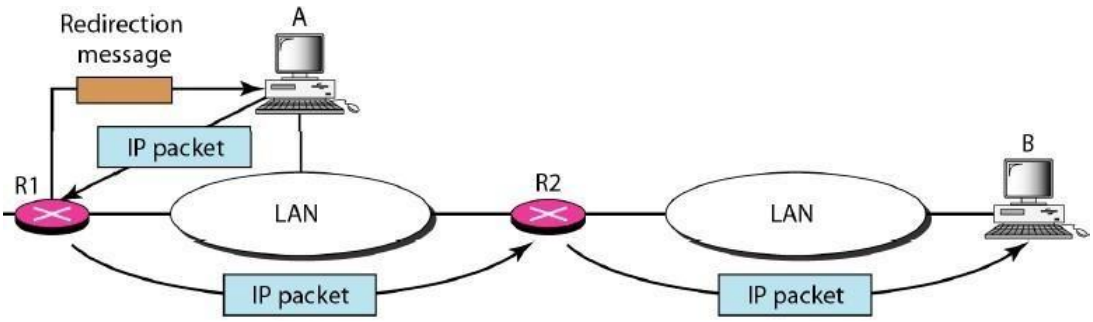


Figure 21.18 Redirection concept

- Router R2 is obviously the most efficient routing choice, but host A did notchoose router R2. The datagram goes to R1 instead.
- Router R1, after consulting its table, finds that the packet should have gone toR2.
- It sends the packet to R2 and, at the same time, sends a redirection message tohost A.
- Host A's routing table can now be updated.

1.5. ICMP Query Messages

In addition to error reporting, ICMP can diagnose some network problems. This is accomplished through the query messages, a group of **four different pairs of messages**, as shown in Figure 21.19.

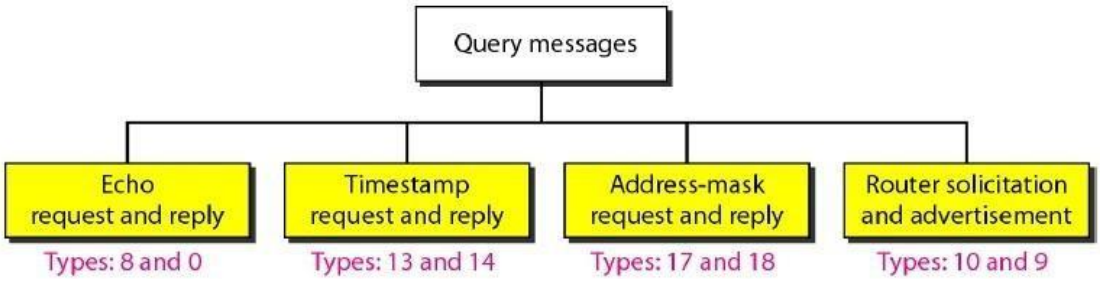


Figure 21.19 Query messages

In this type of ICMP message, a node sends a ICMP request message that is answered ina specific format as ICMP reply by the destination node, depicted in Figure 21.20.

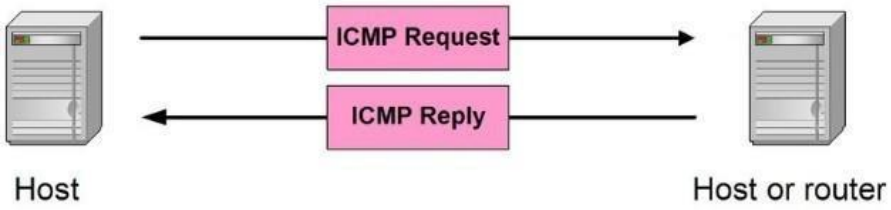


Figure 21.20 ICMP Query Message

A query message is encapsulated in an IP packet, which in turn is encapsulated in a data link layer frame.

However, in this case, no bytes of the original IP are included in the message, as shownin Figure 21.21.

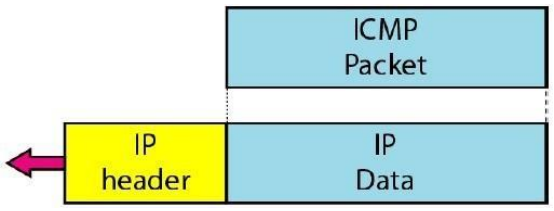


Figure 21.21 Encapsulation of ICMP query messages

a. Echo Request and Echo Reply

The echo-request and echo-reply messages are designed for diagnostic purposes. The combination of echo-request and echo-reply messages determines whether two systems (hosts or routers) can communicate with each other Figure 21.22. It also confirms that the intermediate routers are receiving, processing, and forwarding IP datagrams.

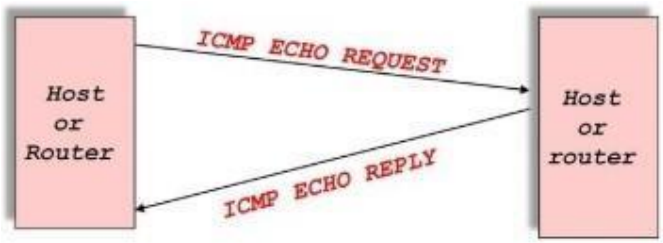


Figure 21.22 ICMP Echo Request and Echo Reply

Today, most systems provide a version of the ***ping* command** that can create a series (instead of just one) of echo-request and echo-reply messages, providing statistical information. We can use the *ping* program to find if a host is alive and responding.

b. Timestamp Request and Timestamp Reply

Two machines (hosts or routers) can use the timestamp request and timestamp reply messages to determine the round-trip time needed for an IP datagram to travel between them. It can also be used to synchronize the clocks in two machines.

c. Address-Mask Request and Address-Mask Reply

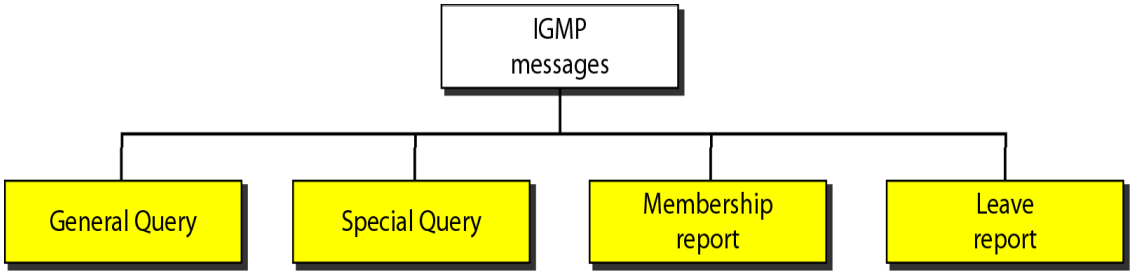
A host may know its IP address, but it may not know the corresponding mask. For example, a host may know its IP address as 159.31.17.24, but it may not know that the corresponding mask is /24. To obtain its mask, a host sends an address-mask-request message to a router on the LAN. If the host knows the address of the router, it sends the request directly to the router. If it does not know, it broadcasts the message. The router receiving the address-mask-request message responds with an address-mask-reply message, providing the necessary mask for the host. This can be applied to its full IP address to get its subnet address.

d. Router Solicitation and Router Advertisement

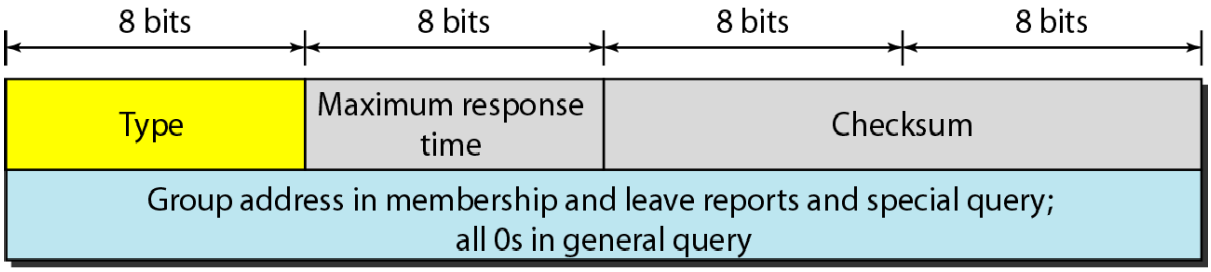
The router-solicitation and router-advertisement messages can help a host to check whether the neighboring routers are alive and functioning. A host can broadcast (or multicast) a router-solicitation message. The router or routers that receive the solicitation message broadcast their routing information using the router-advertisement message. A router can also periodically send router-advertisement messages even if no host has solicited.

IGMP:

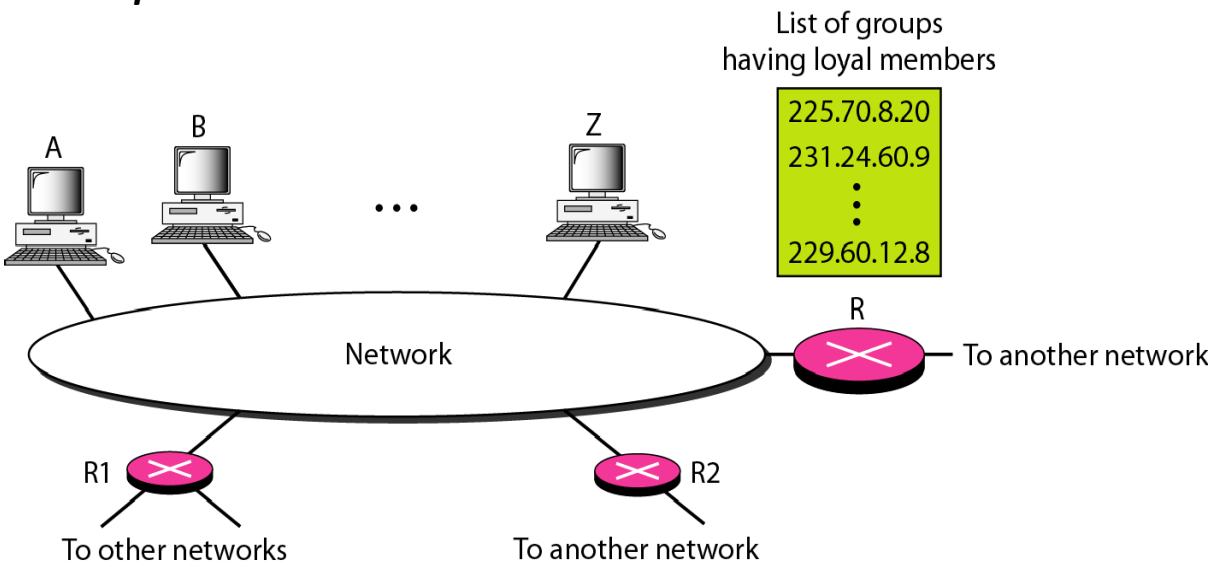
The IP protocol can be involved in two types of communication: unicasting and multicasting. The Internet Group Management Protocol (IGMP) is one of the necessary, but not sufficient, protocols that is involved in multicasting.. IGMP is a companion to the IP protocol.



- The Internet Group Management Protocol (IGMP) is a protocol that allows several devices to share one IP address so they can all receive the same data.
- IGMP is a network layer protocol used to set up multicasting on networks that use the Internet Protocol version 4 (IPv4).
- Specifically, IGMP allows devices to join a multicasting group.
- Multicasting is when a group of devices all receive the same messages or packets.
- Multicasting works by sharing an IP address between multiple devices. Any network traffic directed at that IP address will reach all devices that share the IP address, instead of just one device.
- This is much like when a group of employees all receive company emails directed at a certain email alias.
- Membership reports: Devices send these to a multicast router in order to become a member of a multicast group.
- "Leave group" messages: These messages go from a device to a router and allow devices to leave a multicast group.
- General membership queries: A multicast-capable router sends out these messages to the entire connected network of devices to update multicast group membership for all groups on the network.
- Group-specific membership queries: Routers send these messages to a specific multicast group, instead of the entire network.



IGMP operation:



Distance Vector Routing

- In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. The term **Vector** means a **Table**.
- In this protocol each node maintains a table of minimum distances to every node.
- The table at each node also guides the packets to the desired node by showing the next hop in the route.
- The table for node A shows how we can reach to any node from node A.
- Ex: From node A the least cost to reach node E is 6. The route passes through C.

To	Cost	Next
A	0	-
B	5	-
C	2	-
D	3	-
E	6	C

- The Distance Vector Routing protocol follows these basic steps:

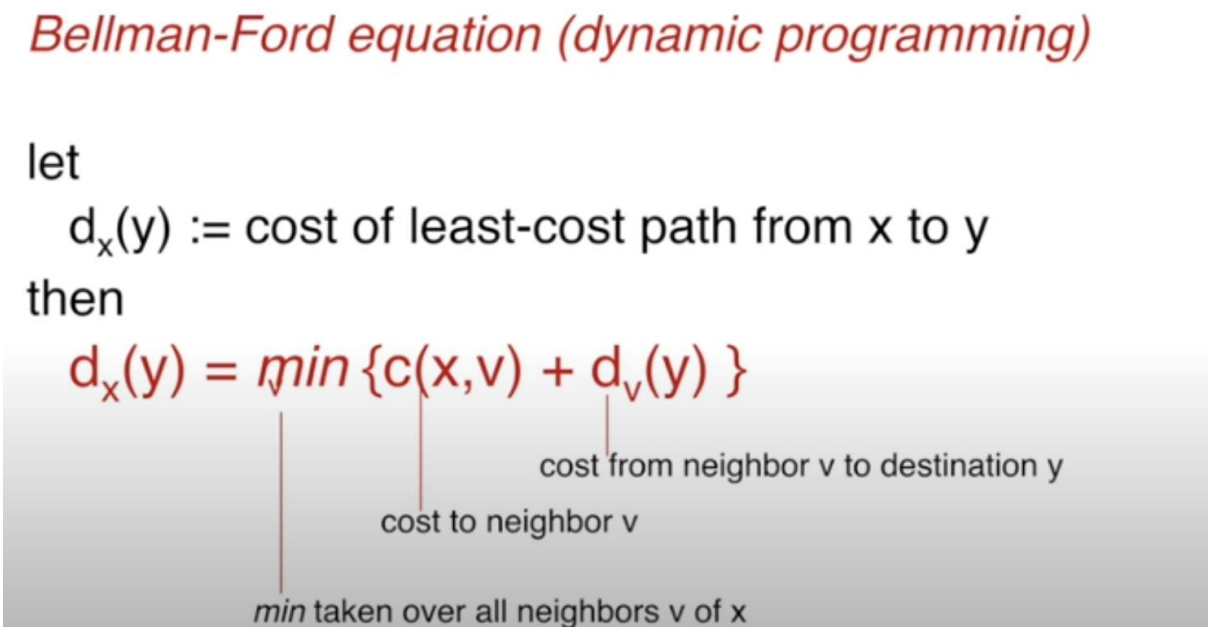
1. Initialization: Each router in the network is configured with its own distance vector table, which lists the distance (in terms of hop count or other metrics) to every other network in the inter-network.

2. Sending distance vectors: Each router sends its distance vector table to its immediate neighbors.(Sharing)

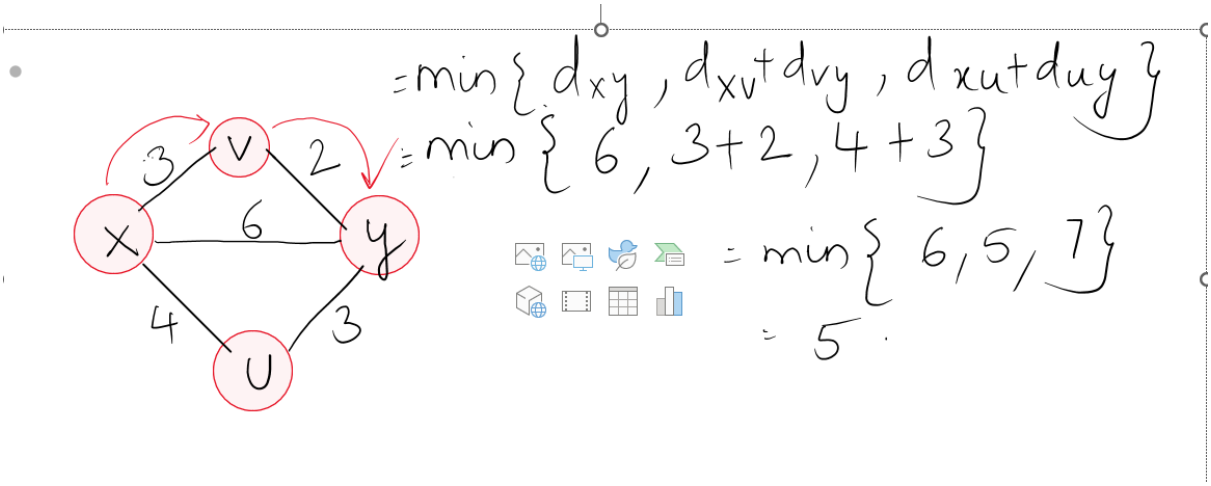
3. Updating distance vectors: Upon receiving a distance vector from a neighbor, a router updates its own table by comparing the distance to a network via its neighbor with the distance currently listed in its own table. If the new distance is shorter, the router updates its table with the new information.

4. Calculating best path: Using the information in its distance vector table, each router calculates the best path to each network and updates its routing table accordingly.

- **5. Periodic updates:** The sending and updating distance vectors are repeated periodically, typically every 30 seconds. This allows the network to adapt to changes in topology quickly.
- **6. Convergence:** The process of sending and updating distance vectors continues until all routers in the network have the most up-to-date information and have converged on the best path to each network.

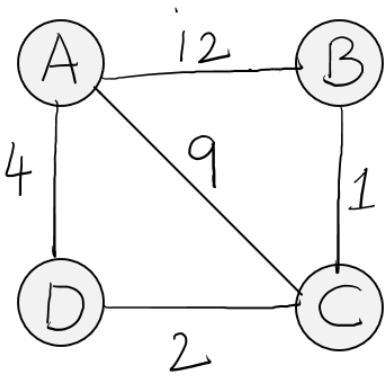


Example:



Example:

1. Initialization



A's table

To	Cost	Next
A	0	A
B	12	B
C	9	C
D	4	D

B's table

To	Cost	Next
A	12	A
B	0	B
C	1	C
D	∞	-

C's table

To	Cost	Next
A	9	A
B	1	B
C	0	C
D	2	D

D's table

To	Cost	Next
A	4	A
B	∞	-
C	2	C
D	0	D

2. Updation:
   A receives distance vectors from B, C, and D.
   It updates its table using their neighbors table.
   A reaches B initially with a cost of 12.
   After getting the neighbors tables A updates its table using Bellmans Ford Equation.
   Example:

A

To	Cost	Next
A	0	A
B	10	C
C	6	D
D	4	D

B

To	Cost	Next
A	10	C
B	0	B
C	1	C
D	3	C

C

To	Cost	Next
A	6	D
B	1	B
C	0	C
D	2	D

D

To	Cost	Next
A	4	A
B	3	C
C	2	C
D	0	D

All routers use the same process to update their tables.

3. Final Solution

A

To	Cost	Next
A	0	A
B	7	D,C
C	6	D
D	4	D

B

To	Cost	Next
A	7	C,D
B	0	B
C	1	C
D	3	C

C

To	Cost	Next
A	6	D
B	1	B
C	0	C
D	2	D

D

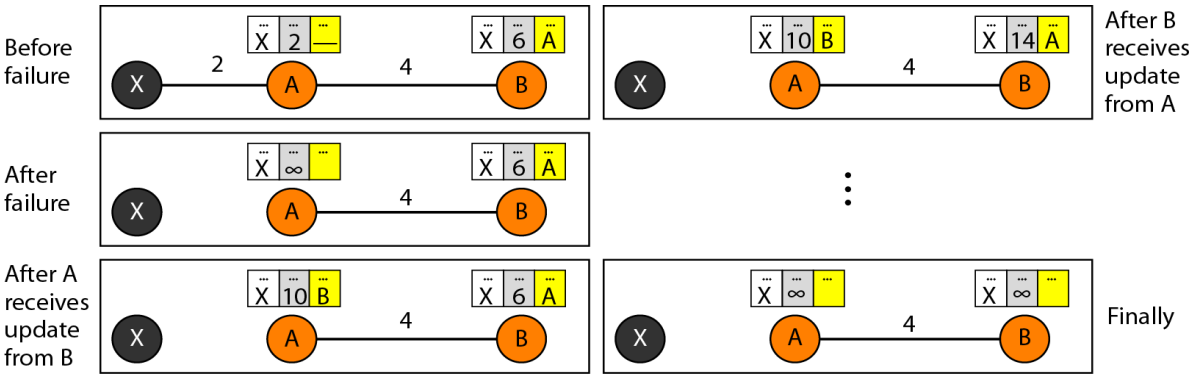
To	Cost	Next
A	4	A
B	3	C
C	2	C
D	0	D

Count to Infinity Problems Two-Node Loop Instability

A problem with distance vector routing is instability, which means that a network using this protocol can become unstable.
Consider the above figure:
It shows a system with 3 nodes: X, A and B.

- At the beginning, both nodes A and B know how to reach node X.
- But suddenly, the link between A and X fails.
- Node A changes its table.

If A can send its table to B immediately there will be no problem because B can identify by looking at the table value ∞.



Problem is: The system becomes unstable if B sends its routing table to A before receiving A's routing table.

- Node A receives the update and, assuming that B has found a way to reach X and immediately updates its routing table.
- Based on the triggered update strategy, A sends its new update to B.
- Now B thinks that something has been changed around A and updates its routing table.
- The cost of reaching X increases gradually until it reaches infinity.
- At this moment, both A and B know that X cannot be reached.

During this counting to infinity the system is not stable:

- Node A thinks that the route to X is via B.

- Node B thinks that the route to X is via A.
- If A receives a packet destined for X, it goes to B and then comes back to A.
- If B receives a packet destined for X, it goes to A and comes back to B.
- Packets bounce between A and B, creating a two-node loop problem.

Link State Routing (LSR):

In Link State Routing, Each node in the domain has the entire topology of the domain –

- List of nodes and links
- How they are connected including the type
- Cost (metric)
- Condition of the links (up or down)

Building Routing Tables

In link state routing, four sets of actions are required to ensure that each node has the routing table showing the least-cost node to every other node.

1. Creation of the states of the links by each node, called the Link State Packet(LSP).
2. Dissemination (Distribution) of LSPs to every other router called **Flooding**. The flooding can be done in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

Creation of Link State Packet (LSP)

A link state packet can carry a large amount of information such as the node identity, the list of links, a sequence number, and age etc.

- Node identity and the List of links are needed to make the topology.
- Sequence number facilitates flooding and distinguishes new LSPs from old ones.
- Age prevents old LSP's from remaining LSP's in the domain for a longtime.

Flooding of LSP's

After a node has prepared an LSP, it must be disseminated to all other nodes, not only to its neighbors. The process is called flooding.

Flooding will be done based on the following:

1. The creating node sends a copy of the LSP out of each interface.
2. A node that receives an LSP compares it with the copy it may already have.

Each and every LSP will be given a Sequence number at the time of their creation. Comparison of sequence numbers determines which LSP is older and which LSP is latest. If the newly arrived LSP is older than the one it already has, then the node discards the LSP.

Formation of Shortest Path Tree: Dijkstra Algorithm

- After receiving all LSPs, each node will have a copy of the whole topology.
- The topology is not sufficient to find the shortest path to every other node; a shortest path tree is needed.
- A tree is a graph of nodes and links, where one node is called Root.
- A shortest path tree is a tree in which the path between the root and every other node is the shortest.
- The Dijkstra's algorithm creates a shortest path tree from a graph.

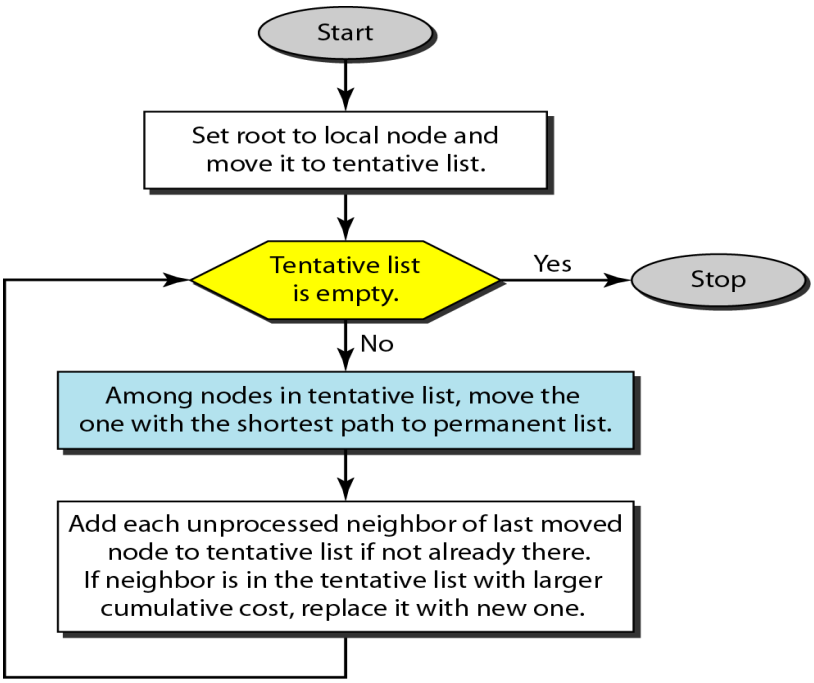
The algorithm divides the nodes into two sets:

1. Tentative nodes

2. Permanent nodes

Dijkstra's algorithm finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent.

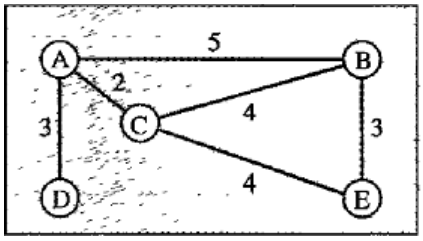
Flow Chart of Dijksta's Algorithm



Example:

Consider the below graph with five nodes: A,B,C,D,E.

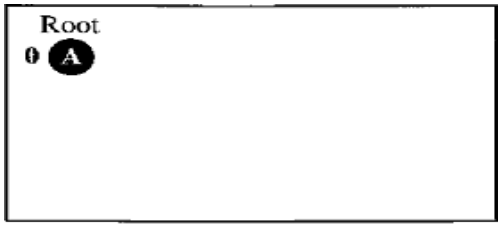
- Apply the Dijkstra's algorithm to node A.
- To find the shortest path in each step, we need the cumulative cost from the root to each node, which is shown next to the node.



Topology

At the end of each step, we show the permanent (filled circles) and the tentative (open circles) nodes and lists with the cumulative costs.

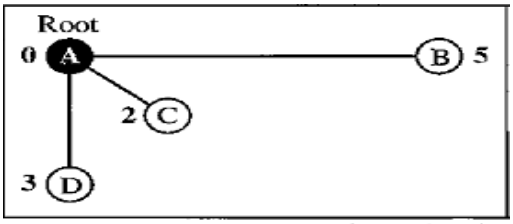
Step 1: We make node A the root of the tree and move it to the tentative list. Our two lists are Permanent list: **Empty** Tentative list: **A(0)**



1. Set root to A and move A to tentative list.

Step 2: Node A has the shortest cumulative cost from all nodes in the tentative list. We move A to the permanent list and add all neighbors of A to the tentative list. Our new lists are

Permanent list: A(0) Tentative list: B(5),



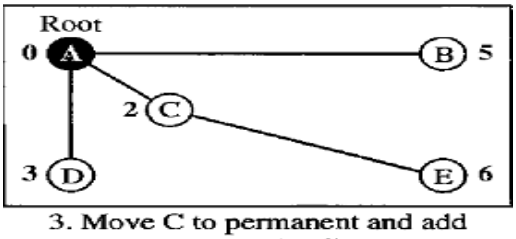
2. Move A to permanent list and add B, C, and D to tentative list.

C(2), D(3)

Step 3: Node C has the shortest cumulative cost from all nodes in the tentative list.

- We move C to the permanent list.
- Node C has three neighbors, but node A is already processed, which makes the unprocessed neighbors just B and E.
- However, B is already in the tentative list with a cumulative cost of 5.
- Node A could also reach node B through C with a cumulative cost of 6.
- Since 5 is less than 6, we keep node B with a cumulative cost of 5 in the tentative list and do not replace it.

Our new lists are: Permanent list: A(0),C(2) Tentative list: B(5),



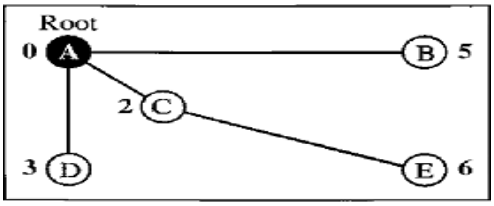
3. Move C to permanent and add E to tentative list.

D(3),E(6).

Step 4: Node D has the shortest cumulative cost of all the nodes in the tentative list.

- We move D to the permanent list. Node D has no unprocessed neighbor to be added to the tentative list.

Our new lists are: Permanent List: A(0),C(2),D(3) Tentative List: B(5),E(6).

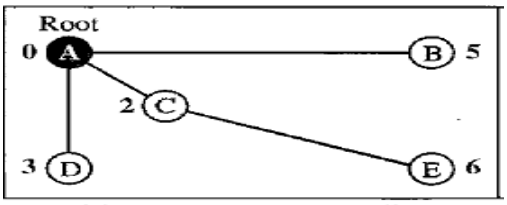


4. Move D to permanent list.

Step 5: Node B has the shortest cumulative cost of all the nodes in the tentative list.

- We move B to the permanent list. We need to add all unprocessed neighbors of B to the tentative list (i.e. just node E).
- E(6) is already in the list with a smaller cumulative cost.
- The cumulative cost to node E, as the neighbor of B, is 8. We keep node E(6) in the tentative list.

Our new lists are: Permanent list: A(0), B(5),C(2),D(3) Tentative list: E(6)

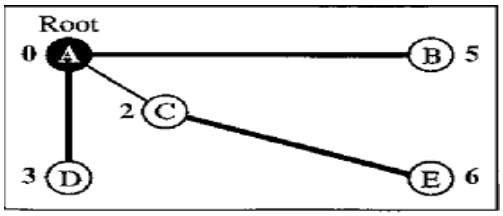


5. Move B to permanent list.

Step 6: Node E has the shortest cumulative cost from all nodes in the tentative list.

- Move E to the permanent list. Node E has no neighbor. Now the tentative list is empty.
- We stop the process here. The shortest path tree is ready for graph ABCDE.

The finalists are: Permanent list: A(0), B(5), C(2), D(3), E(6) Tentative list: Empty



6. Move E to permanent list (tentative list is empty).

Calculation of Routing Table from Shortest Path Tree

- Each node uses the shortest path tree protocol to construct its routing table.
- The routing table shows the cost of reaching each node

from the root. The below table shows routing table for Node A.

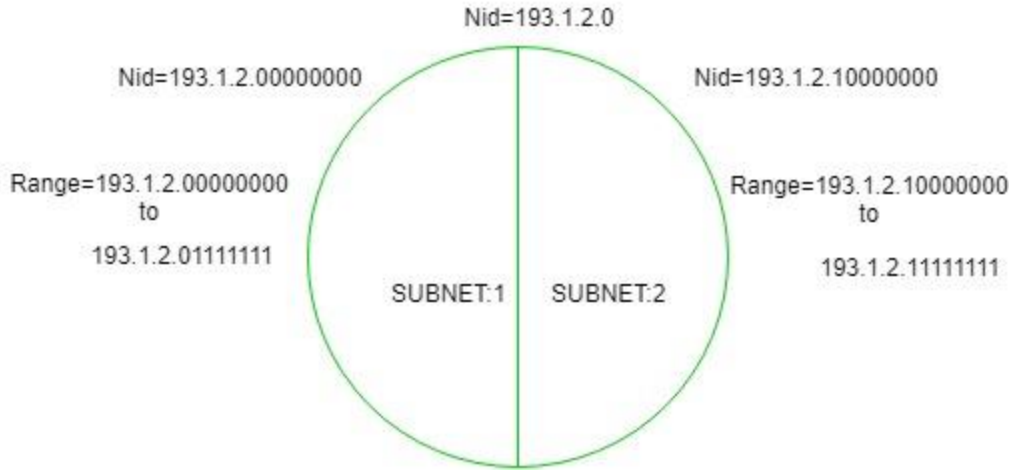
Node	Cost	Next Router
A	0	-
B	5	-
C	2	-
D	3	-
E	6	C

When a bigger network is divided into smaller networks, to maintain security, then that is known as Subnetting. So, maintenance is easier for smaller networks. For example, if we consider a class A address, the possible number of hosts is 2^{24} for each network, it is obvious that it is difficult to maintain such a huge number of hosts, but it would be quite easier to maintain if we divide the network into small parts.

Uses of Subnetting

1. Subnetting helps in organizing the network in an efficient way which helps in expanding the technology for large firms and companies.
2. Subnetting is used for specific staffing structures to reduce traffic and maintain order and efficiency.
3. Subnetting divides domains of the broadcast so that traffic is routed efficiently, which helps in improving network performance.
4. Subnetting is used in increasing network security.

The network can be divided into two parts: To divide a network into two parts, you need to choose one bit for each Subnet from the host ID part.



In class C the first 3 octets are network bits so it remains as it is.

- **For Subnet-1:** The first bit which is chosen from the host id part is zero and the range will be from (193.1.2.00000000 till you get all 1's in the host ID part i.e, 193.1.2.01111111) except for the first bit which is chosen zero for subnet id part.

Thus, the range of subnet 1 is: **193.1.2.0 to 193.1.2.127**

```
Subnet id of Subnet-1 is : 193.1.2.0
The direct Broadcast id of Subnet-1 is: 193.1.2.127
The total number of hosts possible is: 126 (Out of 128,
2 id's are used for Subnet id & Direct Broadcast id)
The subnet mask of Subnet- 1 is: 255.255.255.128
```

- **For Subnet-2:** The first bit chosen from the host id part is one and the range will be from (193.1.2.100000000 till you get all 1's in the host ID part i.e, 193.1.2.11111111).

Thus, the range of subnet-2 is: **193.1.2.128 to 193.1.2.255**

```
Subnet id of Subnet-2 is : 193.1.2.128
The direct Broadcast id of Subnet-2 is: 193.1.2.255
The total number of hosts possible is: 126 (Out of 128,
2 id's are used for Subnet id &  Direct Broadcast id)
The subnet mask of Subnet- 2 is: 255.255.255.128
The best way to find out the subnet mask of a subnet
is to set the fixed bit of host-id to 1 and the rest to 0.
```

Finally, after using the subnetting the total number of usable hosts is reduced from 254 to 252.

Note:

1. To divide a network into four (2_2) parts you need to choose two bits from the host id part for each subnet i.e, (00, 01, 10, 11).
2. To divide a network into eight (2_3) parts you need to choose three bits from the host id part for each subnet i.e, (000, 001, 010, 011, 100, 101, 110, 111) and so on.
3. We can say that if the total number of subnets in a network increases the total number of usable hosts decreases.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Function to convert an IP address to a 32-bit integer
unsigned int ipToInt(char* ip) {
   unsigned int a, b, c, d;
   sscanf(ip, "%u.%u.%u.%u", &a, &b, &c, &d);
   return (a << 24) | (b << 16) | (c << 8) | d;
}

// Function to convert a 32-bit integer to an IP address
void intToIp(unsigned int ip, char* buffer) {
   sprintf(buffer, "%u.%u.%u.%u", (ip >> 24) & 0xFF, (ip >> 16) & 0xFF, (ip >> 8) &
0xFF, ip & 0xFF);
}

// Function to calculate the subnet mask from a prefix length
unsigned int calculateSubnetMask(int prefixLength) {
   return prefixLength == 0 ? 0 : ~((1 << (32 - prefixLength)) - 1);
}

int main() {
   char ip[16];
```

```
    int prefixLength, newPrefixLength;
    unsigned int subnetMask, newSubnetMask, ipInt;
    char buffer[16];

    // Input IP address and prefix length
    printf("Enter IP address (e.g., 192.168.1.0): ");
    scanf("%s", ip);
    printf("Enter current prefix length (e.g., 24): ");
    scanf("%d", &prefixLength);

    // New prefix length for creating two subnets
    newPrefixLength = prefixLength + 1;

    // Convert IP address to integer
    ipInt = ipToInt(ip);

    // Calculate original subnet mask and new subnet mask
    subnetMask = calculateSubnetMask(prefixLength);
    newSubnetMask = calculateSubnetMask(newPrefixLength);

    // Calculate the number of hosts per subnet
    int hostsPerSubnet = (1 << (32 - newPrefixLength)) - 2; // subtract 2 for
network and broadcast addresses

    printf("\nNumber of subnets: 2\n");
    printf("Number of hosts per subnet: %d\n", hostsPerSubnet);

    // Generate subnets
    for (int i = 0; i < 2; i++) {
        unsigned int subnetNetwork = (ipInt & subnetMask) | (i << (32 -
newPrefixLength));
        unsigned int subnetBroadcast = subnetNetwork | ~newSubnetMask;
        unsigned int firstHost = subnetNetwork + 1;
        unsigned int lastHost = subnetBroadcast - 1;

        printf("\nSubnet %d:\n", i + 1);
        printf("Network Address: ");
        intToIp(subnetNetwork, buffer);
        printf("%s\n", buffer);

        printf("Broadcast Address: ");
        intToIp(subnetBroadcast, buffer);
```

```
        printf("%s\n", buffer);

        printf("Subnet Mask: ");
        intToIp(newSubnetMask, buffer);
        printf("%s\n", buffer);

        printf("First Host: ");
        intToIp(firstHost, buffer);
        printf("%s\n", buffer);

        printf("Last Host: ");
        intToIp(lastHost, buffer);
        printf("%s\n", buffer);
    }

    return 0;
}
```

INPUT:

```
Enter IP address (e.g., 192.168.1.0): 193.1.2.0
Enter current prefix length (e.g., 24): 24
```

OUTPUT:

```
Number of subnets: 2
Number of hosts per subnet: 126

Subnet 1:
Network Address: 193.1.2.0
Broadcast Address: 193.1.2.127
Subnet Mask: 255.255.255.128
First Host: 193.1.2.1
Last Host: 193.1.2.126

Subnet 2:
Network Address: 193.1.2.128
Broadcast Address: 193.1.2.255
Subnet Mask: 255.255.255.128
First Host: 193.1.2.129
Last Host: 193.1.2.254

--------------------------------
```

LAB EXPERIMENT -4

AIM: Write a C program to Implement distance vector routing algorithm for obtaining routing tables at each node.

Distance Vector Routing:

- **Nodes and Distance Vectors**: The program initializes a set of nodes and their distance vectors.
- **Bellman-Ford Algorithm**: This is used to update the distance vectors until they stabilize.
- **Routing Table Construction**: Each node maintains a routing table with the next hop and the distance to each destination.

Explanation:

1. **Initialization**:
   - The `initialize` function sets up the distance vectors and routing tables for each node.
   - If there's a direct link between two nodes, the initial distance is set to the cost of that link, and the next hop is set to the destination node itself. Otherwise, the distance is set to `INF` (representing no direct link), and the next hop is set to `-1`.
2. **Distance Vector Routing Algorithm**:
   - The `distanceVectorRouting` function uses the Bellman-Ford algorithm to update the distance vectors and next hops.
   - The algorithm iteratively updates the distance vectors until no further updates are necessary, indicating convergence.
3. **Printing the Routing Tables**:
   - The `printRoutingTable` function displays the routing table for each node, showing the destination, next hop, and distance.
4. **Main Function**:
   - The `main` function reads the number of nodes and the cost matrix from the user, initializes the data structures, runs the distance vector routing algorithm, and prints the routing tables.

CODE:

```
#include <stdio.h>

#include <stdlib.h>



#define INF 9999

#define MAX_NODES 10
```

```
// Function to initialize distance vector and routing table
void initialize(int numNodes, int costMatrix[MAX_NODES][MAX_NODES], int distVector[MAX_NODES][MAX_NODES], int nextHop[MAX_NODES][MAX_NODES]) {
    for (int i = 0; i < numNodes; i++) {
        for (int j = 0; j < numNodes; j++) {
            distVector[i][j] = costMatrix[i][j];
            if (costMatrix[i][j] != INF && i != j) {
                nextHop[i][j] = j;
            } else {
                nextHop[i][j] = -1;
            }
        }
    }
}

// Function to print routing table for each node
void printRoutingTable(int numNodes, int distVector[MAX_NODES][MAX_NODES], int nextHop[MAX_NODES][MAX_NODES]) {
    for (int i = 0; i < numNodes; i++) {
        printf("Routing table for node %d:\n", i);
        printf("Destination\tNext Hop\tDistance\n");
        for (int j = 0; j < numNodes; j++) {
            if (distVector[i][j] == INF) {
                printf("%d\t\t-\t\tINF\n", j);
            } else {
                printf("%d\t\t%d\t\t%d\n", j, nextHop[i][j], distVector[i][j]);
            }
        }
```

```
        printf("\n");
    }
}

// Function to implement Distance Vector Routing algorithm
void distanceVectorRouting(int numNodes, int costMatrix[MAX_NODES][MAX_NODES], int distVector[MAX_NODES][MAX_NODES], int nextHop[MAX_NODES][MAX_NODES]) {
    int updated;

    do {
        updated = 0;
        for (int i = 0; i < numNodes; i++) {
            for (int j = 0; j < numNodes; j++) {
                for (int k = 0; k < numNodes; k++) {
                    if (distVector[i][k] + distVector[k][j] < distVector[i][j]) {
                        distVector[i][j] = distVector[i][k] + distVector[k][j];
                        nextHop[i][j] = nextHop[i][k];
                        updated = 1;
                    }
                }
            }
        }
    } while (updated);
}

int main() {
    int numNodes, costMatrix[MAX_NODES][MAX_NODES];
    int distVector[MAX_NODES][MAX_NODES];
```

```
    int nextHop[MAX_NODES][MAX_NODES];

    printf("Enter the number of nodes: ");
    scanf("%d", &numNodes);

    printf("Enter the cost matrix (use %d for INF):\n", INF);
    for (int i = 0; i < numNodes; i++) {
        for (int j = 0; j < numNodes; j++) {
            scanf("%d", &costMatrix[i][j]);
        }
    }

    initialize(numNodes, costMatrix, distVector, nextHop);
    distanceVectorRouting(numNodes, costMatrix, distVector, nextHop);
    printRoutingTable(numNodes, distVector, nextHop);

    return 0;
}
```

INPUT:

```
Enter the number of nodes: 4
Enter the cost matrix (use 9999 for INF):
0       12      9       4
12      0       1       9999
9       1       0       2
4       9999    2       0
```

CHECK THE OUTPUT FOR THE ABOVE INPUT.