

# CS-206 Computer Networks

## SYLLABUS

Introduction to Internetworking and TCP/IP.

**Addressing and Routing:** Logical Addressing- IPv4 Addresses, IPv6 Addresses. Internet protocol- Internetworking, IPv4, IPv6, transition from IPv4 to IPv6. Address Mapping- ARP, RARP, BOOTP, DHCP, Error Reporting- ICMP. Multicasting-IGMP. Routing- Delivery, Forwarding, Intra and Inter-domain routing, Unicast Routing Protocols-Distance Vector Routing, Link State Routing, Path Vector Routing. Multicast Routing protocols.

**TCP and UDP:** Process to process delivery- Client/Server Paradigm, Multiplexing and Demultiplexing, Connectionless Versus Connection-Oriented Service, Reliable Versus Unreliable. UDP- Well-Known Ports for UDP, User Datagram, UDP Operation, Use of UDP.TCP- TCP Services, TCP Features, Segment, A TCP Connection, Flow Control, Error Control. Congestion Control- Network performance, Open loop congestion control, Closed loop congestion control, Congestion control in TCP, Quality of Service.

**Network Applications:** DNS- Name space, Distribution of name space, DNS in the Internet, resolution, DDNS. Remote logging- TELNET, Electronic Mail- SMTP, POP, IMAP, File Transfer- FTP, WWW, HTTP, Network Management: SNMP .

**Network Security:** Security services- message confidentiality, message integrity, Message authentication, Digital signature, Entity authentication, Key management- Symmetric, Asymmetric. Security in the Internet: IPSec, TLS, PGP, VPN and Firewalls.

# Suggested Readings:

1. **B. A. Forouzan, Data Communications and Networking, TMH .**
2. **A. S. Tanenbaum, Computer Networks, PHI.**
3. **A. Forouzan, TCP/IP Protocol Suite, 4th Edition, McGraw Hill, 2010.**
4. **D E. Comer, Internetworking with TCP/IP Principles, Protocols and Architecture, Pearson Education.**

# **Lecture 1**

## **Introduction to Internetworking and TCP/IP**

**Dr. Vandana Kushwaha**

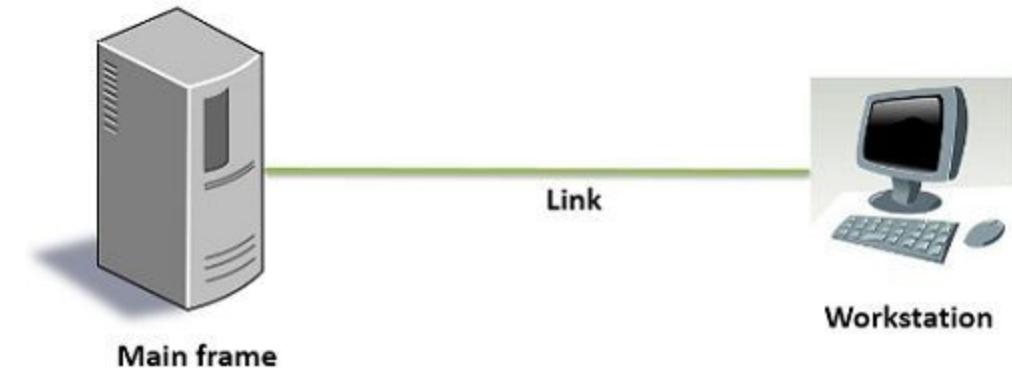
Department of Computer Science  
Institute of Science, BHU, Varanasi

# Outline of the Lecture

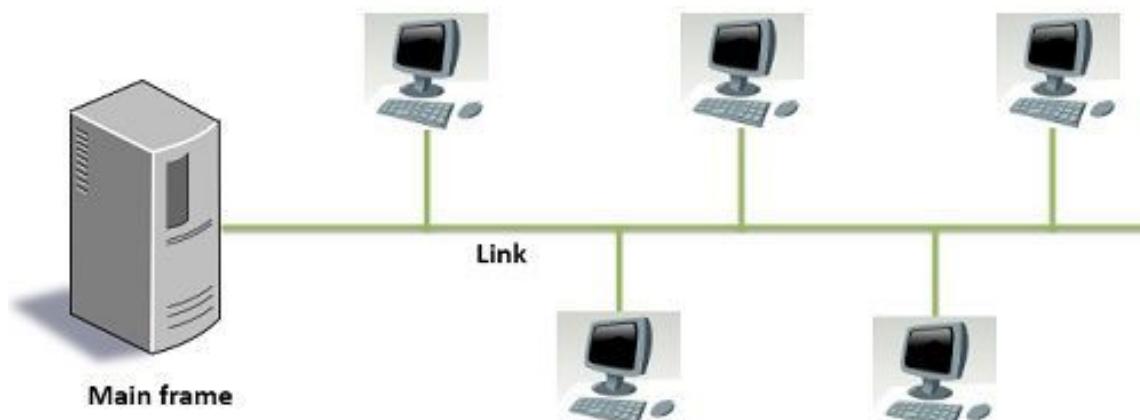
- Why Internetworking?
- Internetworking issues.
- Internetworking devices.
- TCP/IP Model

# Possible communication approaches

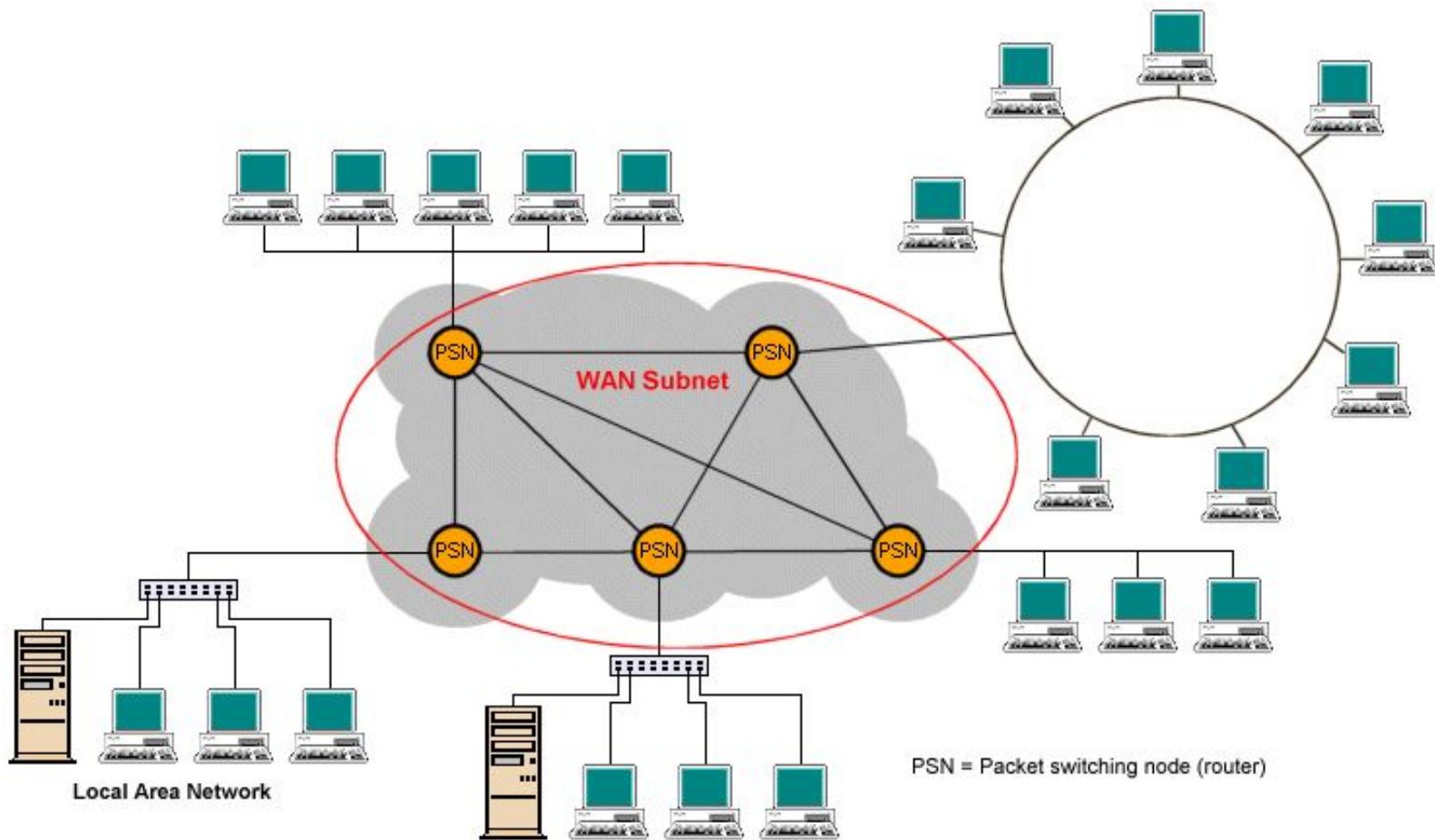
- **Point-to-point:** two stations can communicate with each other with the help of point-to-point links.
- **Multipoint broadcast network** like **LAN**, Satellite networks then the cellular telephone networks.
- **Switched communication network** which is also known as **WANs** as telephone network, X.25, frame relay.



Point-to-point Connection



Multipoint Connection



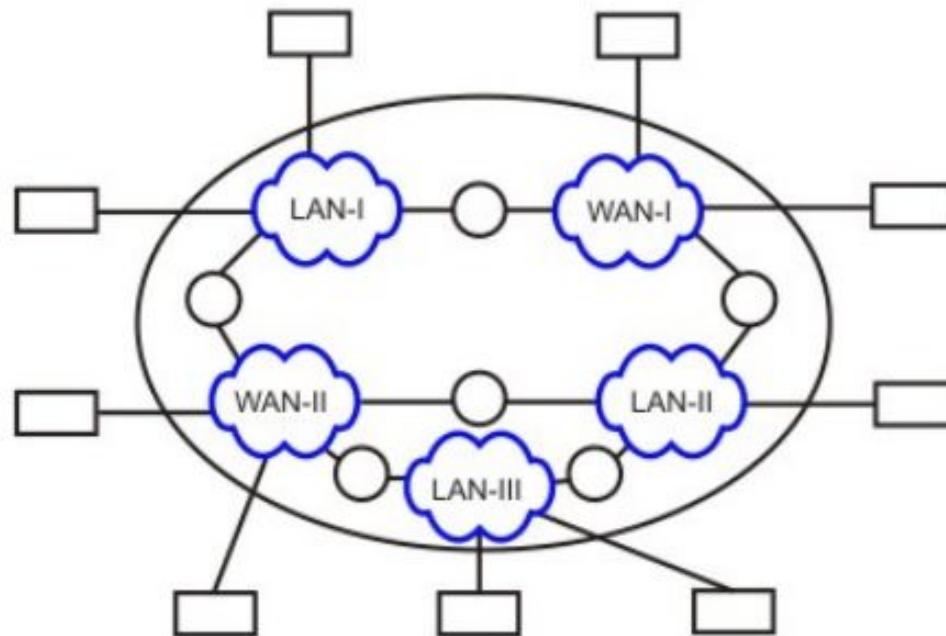
**Switched communication network (WAN)**

# Why Internetworking?

- A **point-to-point link** connects only two.
- **LAN technology** is designed to provide high speed communication over a **small geographical area**, e.g. A single Ethernet(LAN) can interconnect no more than 1024 hosts.
- **WAN technology** is designed to provide communication across different cities, countries and continents but their rate of data transfer is not very high.
- **Wireless networks** are limited by the range of their radios.
- It has been established that **isolated LAN and WAN** have **limited potential and usefulness**.
- Thus to build a **global network**, we need a way to **interconnect** these different types of links and multi-access networks.
- The concept of *interconnecting different types of networks to build a large, global network is the core idea of the Internet and is often referred to as internetworking.*

# What is internet?

- The basic objective of internet is to **connect individual heterogeneous network** of both LAN and WAN distributed across the world using suitable **hardware and software** in such a way that it gives the user the illusion of a single network.
- This **single virtual network** is widely known as internet which is essentially a **network of networks**.



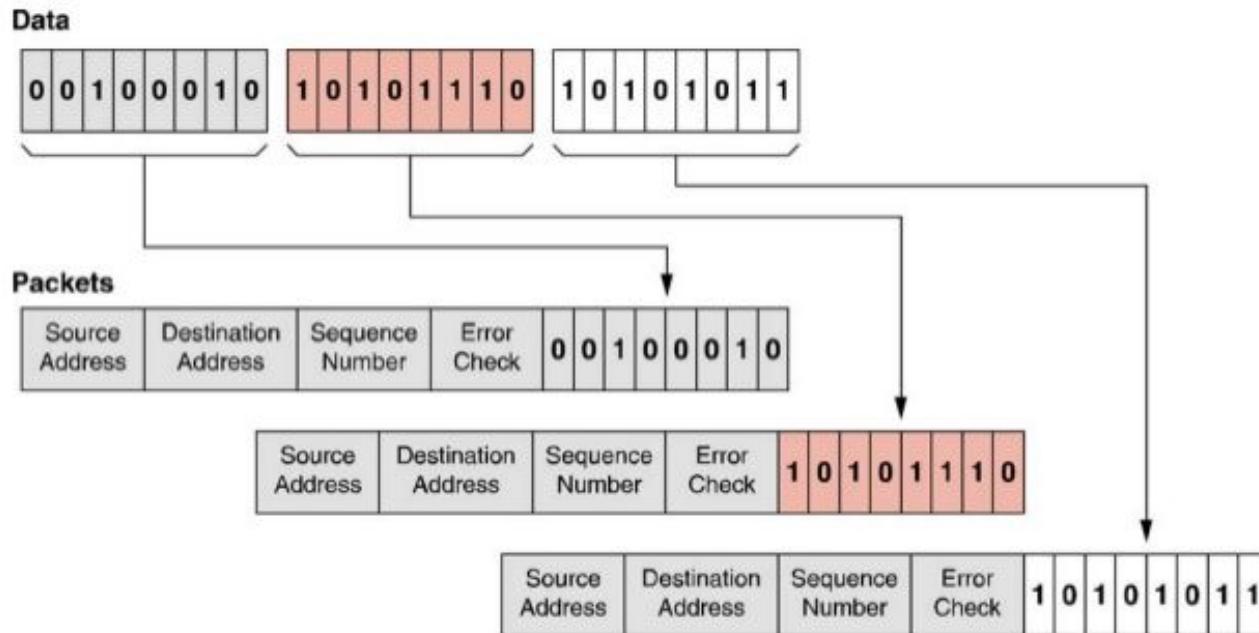
# Internetworking issues

- **Addressing:** Whenever two stations want to communicate with each other they must have some address.
- **Packetizing:** Apart from data other information like source address, destination address and various other things have to be put together in the form of a packet known as packetizing.
- **Fragmentation and Reassembling:** sometimes due to network constraint it may be necessary to fragment a particular packet into a number of packets as it goes through a network, then at the other end or in the destination node they have to be put together, they have to be reassembled.
- **Routing:** routing is required as the packet passes through a sequence of networks.
- **Flow Control, Error Control and Congestion Control:** required for reliability.
- **Security:** network security has to be maintained as the data passes through the network, security has to be maintained with the help of suitable encryption/decryption technique.

# **Addressing**

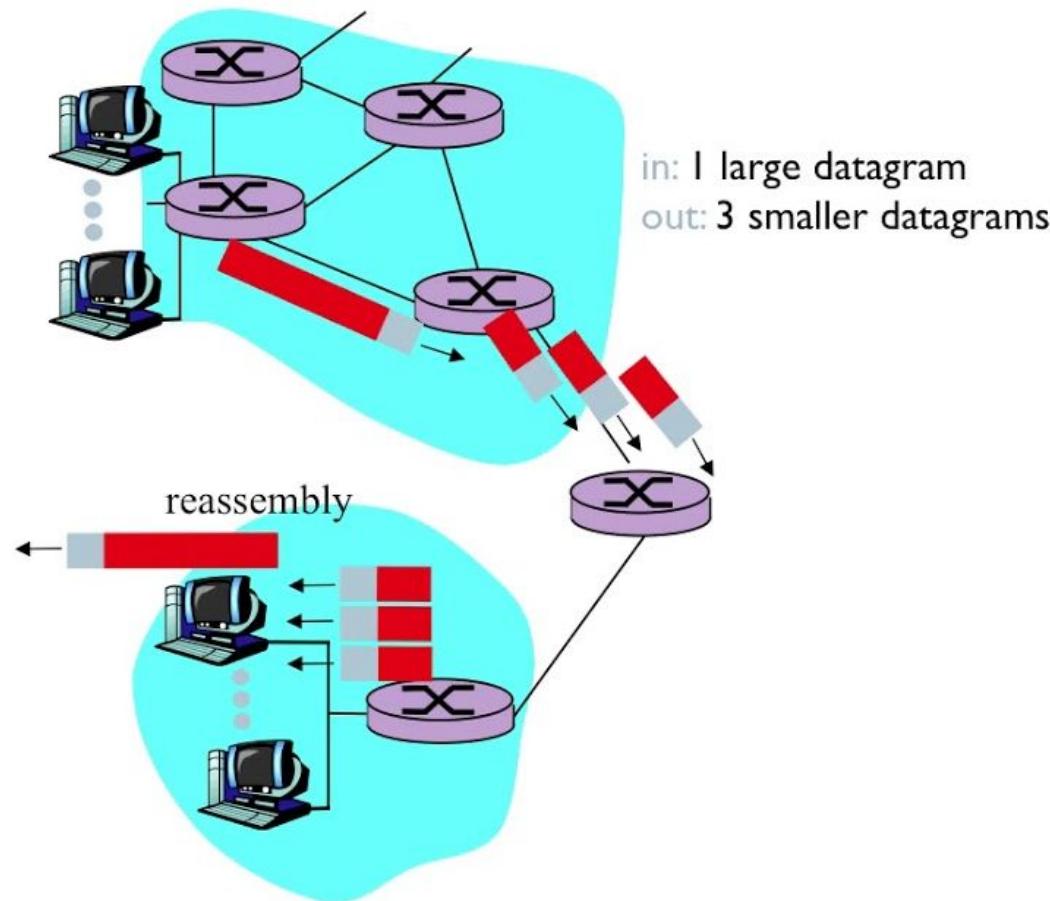
- **Physical(MAC)Addressing** at **Data Link Layer**
- **Logical(IP) Addressing** at **Network Layer**
- **Port Addressing** at **Transport Layer**

# Packetization



- Data stream is divided into 3 packets (8-bits each).
- Header information is added to the data portion.

# Fragmentation and Reassembling

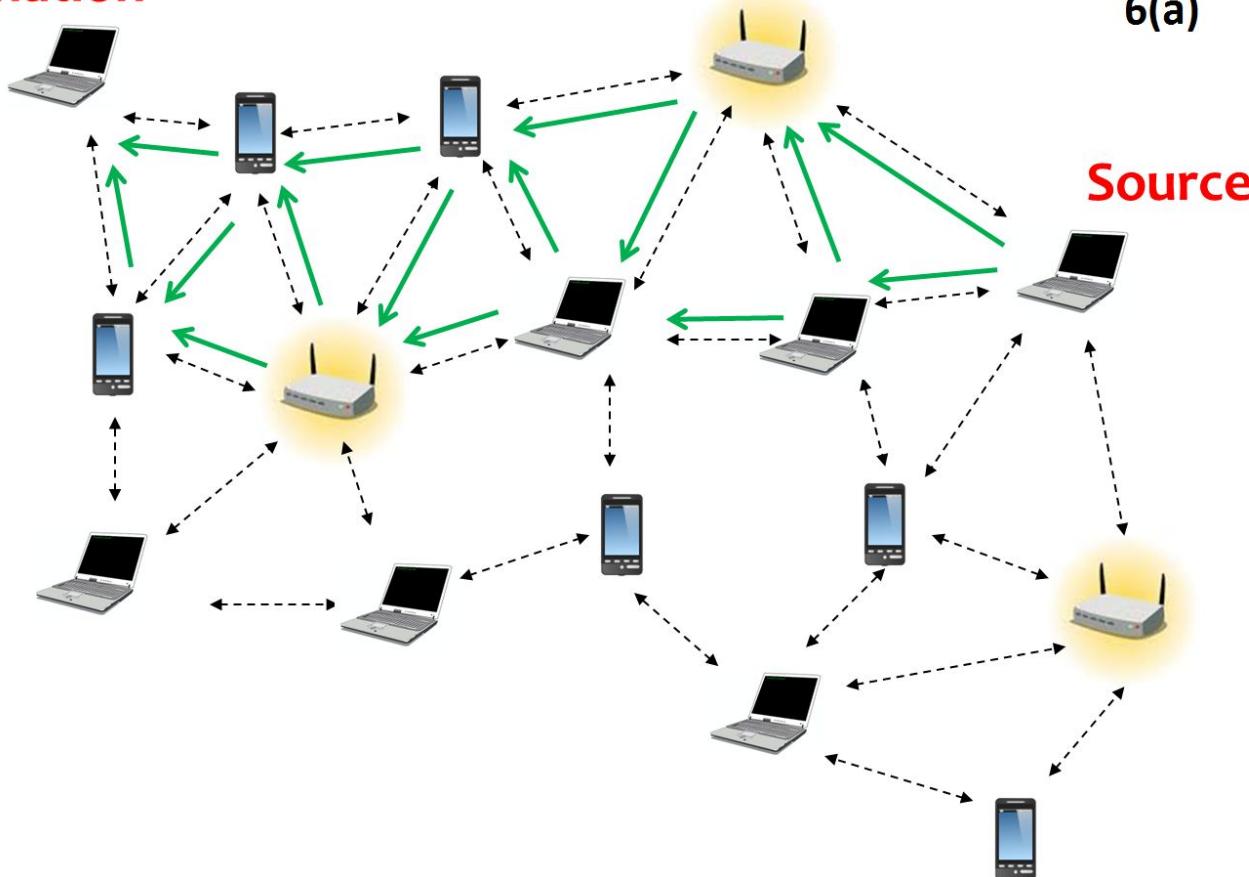


# Routing

Destination

6(a)

Source



# Internetworking devices

- **Hardware:**
  - **Repeater/Hub:** are essentially layer-1 relay, they act as some kind of relay.
  - **Bridge/Switch:** are layer-2 relay
  - **Router:** are layer-3 relay
  - **Gateway:** are layer-7 relay, can operate in all seven layers.
- **Software:**
  - **TCP/IP protocol suit** for Internetworking: There are four layers but TCP/IP essentially operates in two layers as TCP in the transport layer and IP in the network layer and it's a protocol suite for internetworking.
  - **Application Layer protocols:** there are application layer protocols which operate on top of TCP/IP like TELNET, FTP, SMTP, etc.

# **Lecture 0**

## **Introduction to Data Communication and Computer Network**

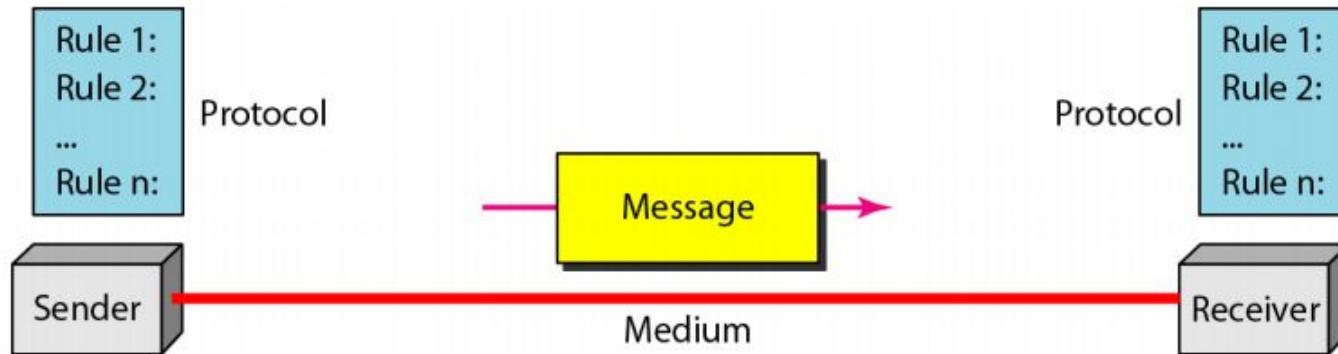
**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Data Communication

- **Data communication** is the exchange of data between two or more devices via some form of transmission medium such as a wired/wireless medium.
- **Fundamental characteristics** of effective data communications :
  - a. **Delivery**
  - b. **Accuracy**
  - c. **Timeliness**
    - a. **Delivery.** The system must deliver data to the correct destination.
    - b. **Accuracy.** The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable.
    - c. **Timeliness.** The system must deliver data in a timely manner. Data delivered late are useless. In the case of real time video and audio, timely delivery means delivering data as they are produced, in the same order that they are produced, and without significant delay.

# Components of Data Communications System



- **Sender.** The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.
- **Receiver.** The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.
- **Transmission medium.** The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.
- **Message.** The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
- **Protocol.** A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating, just as a person speaking French cannot be understood by a person who speaks only Japanese.

# Data Transmission Mode

Communication between two devices can be :

- a. Simplex
- b. Half-duplex
- c. Full-duplex.

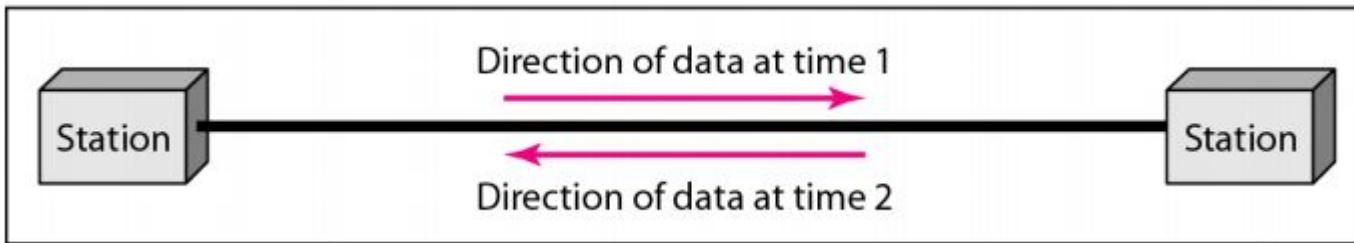
## a. Simplex:

In simplex mode, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit; the other can only receive. Keyboards and traditional monitors are examples of simplex devices.



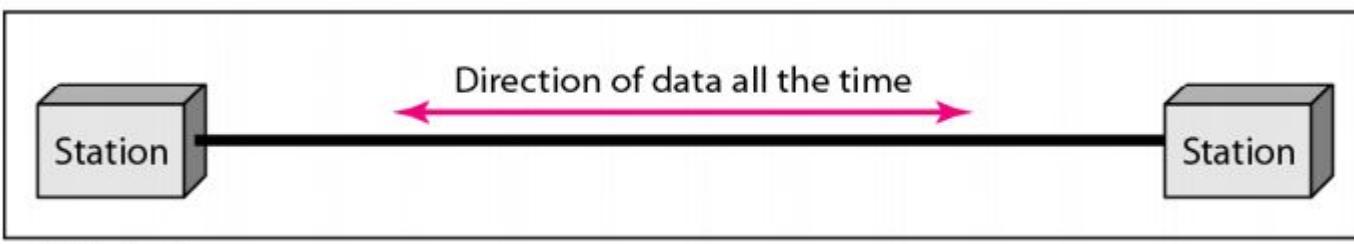
### b. Half-Duplex:

In half-duplex mode, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa. Walkie-talkies is a half-duplex system.



### c. Full-Duplex:

In full-duplex mode (also called duplex), both stations can transmit and receive simultaneously. One common example of full-duplex communication is the telephone network.

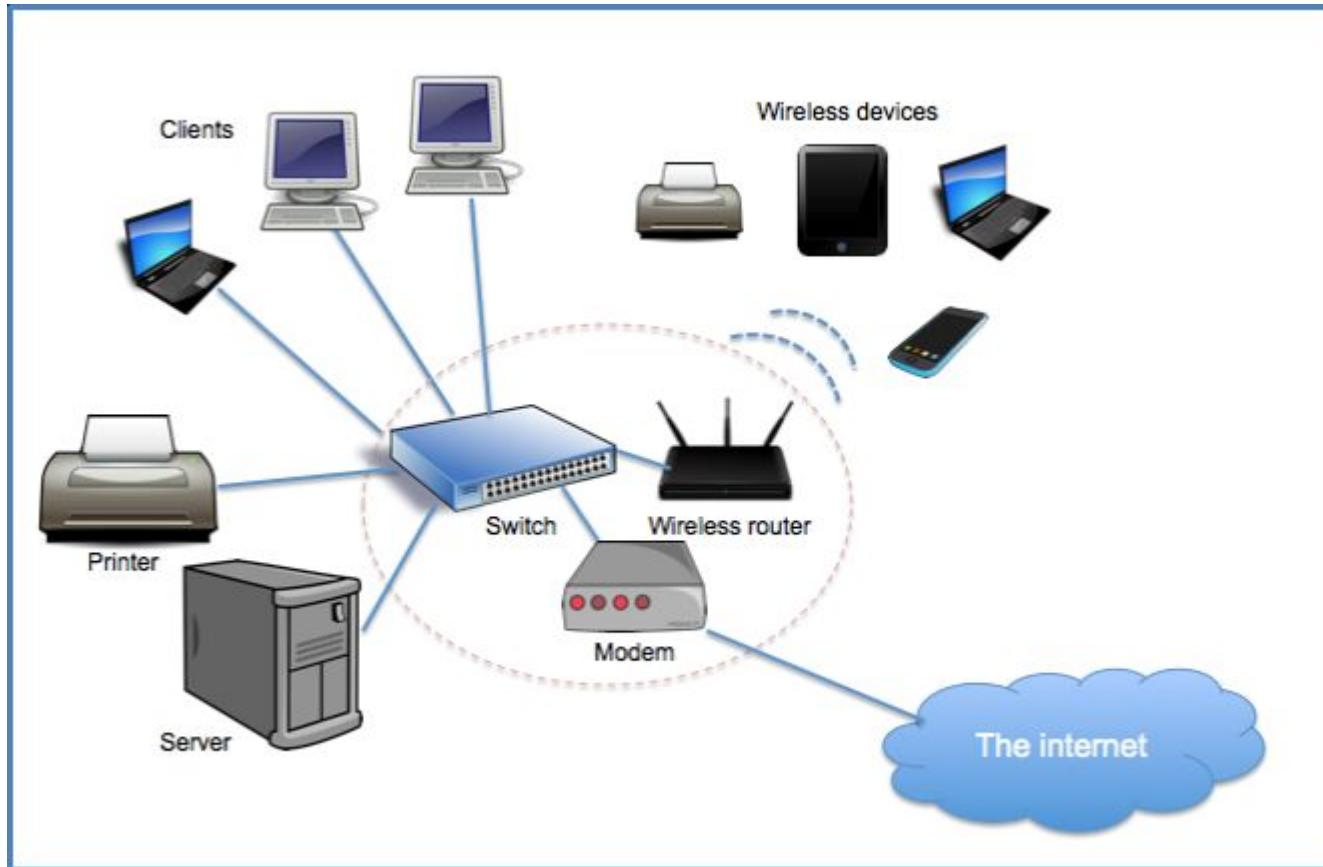


# Network

A **network** is a set of devices (often referred to as nodes/hosts/stations) connected by communication links.

A **node** can be a computer, printer, or any other device capable of sending and/or receiving data generated by other nodes on the network.

A **link** is a communications pathway that transfers data from one device to another.

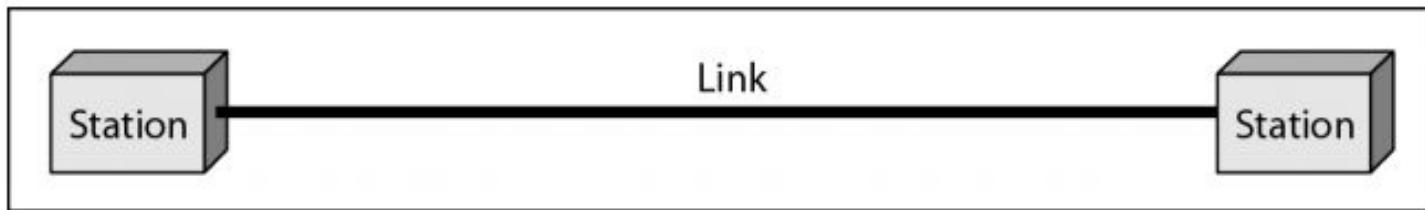


# Type of Connection

There are two possible types of connections:

- a. Point-to-point
- b. Multipoint.

- a. **Point-to-Point:** A point-to-point connection provides a dedicated link between two devices. The entire capacity of the link is reserved for transmission between those two devices.

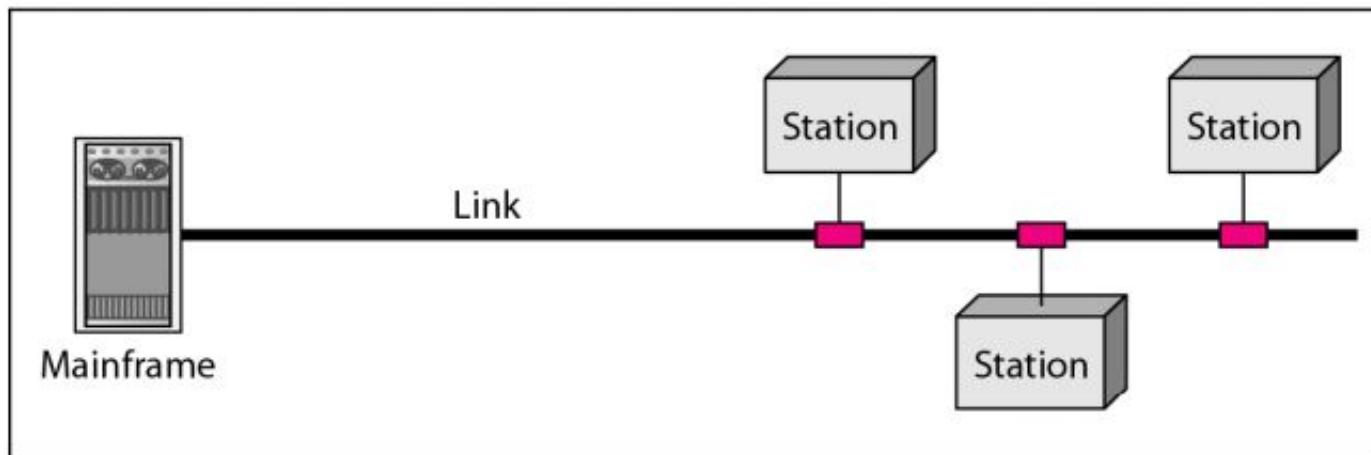


a. Point-to-point

# Type of Connection

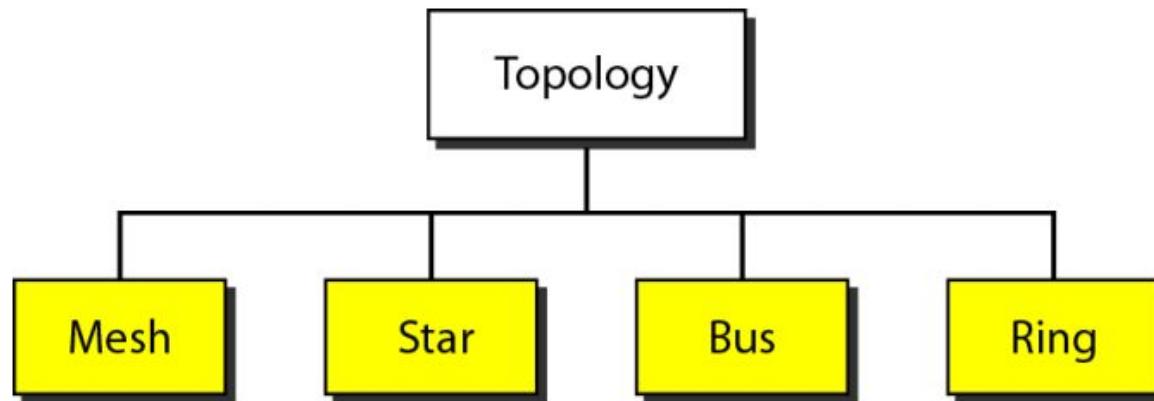
b. **Multipoint:** A multipoint connection is one in which more than two specific devices share a single link .

- In a multipoint environment, the capacity of the channel is shared either spatially or temporally.
- If several devices can use the link simultaneously, it is a spatially shared connection.
- If users must take turns, it is a timeshared connection.

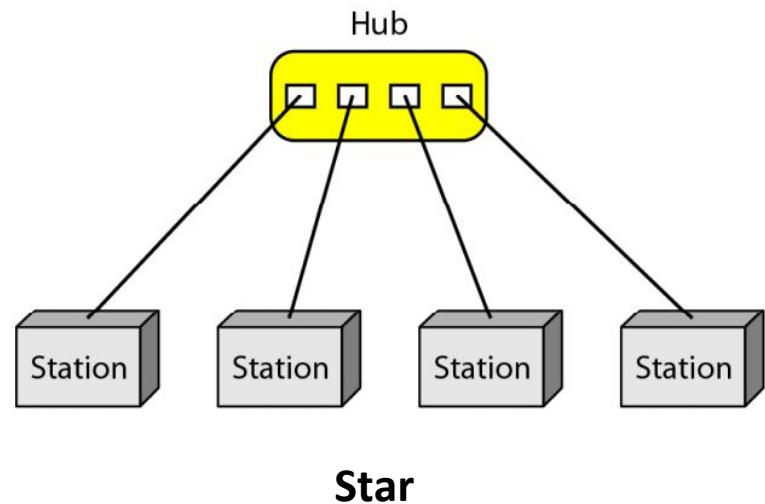
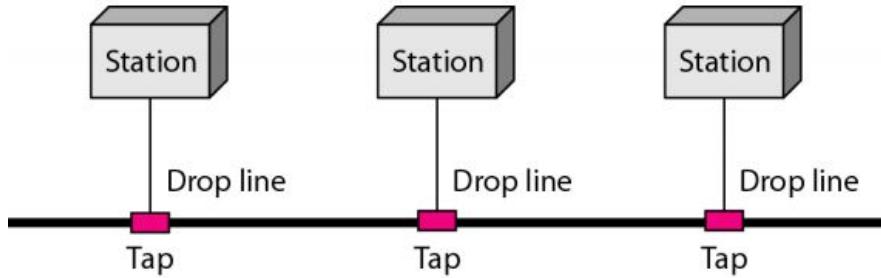
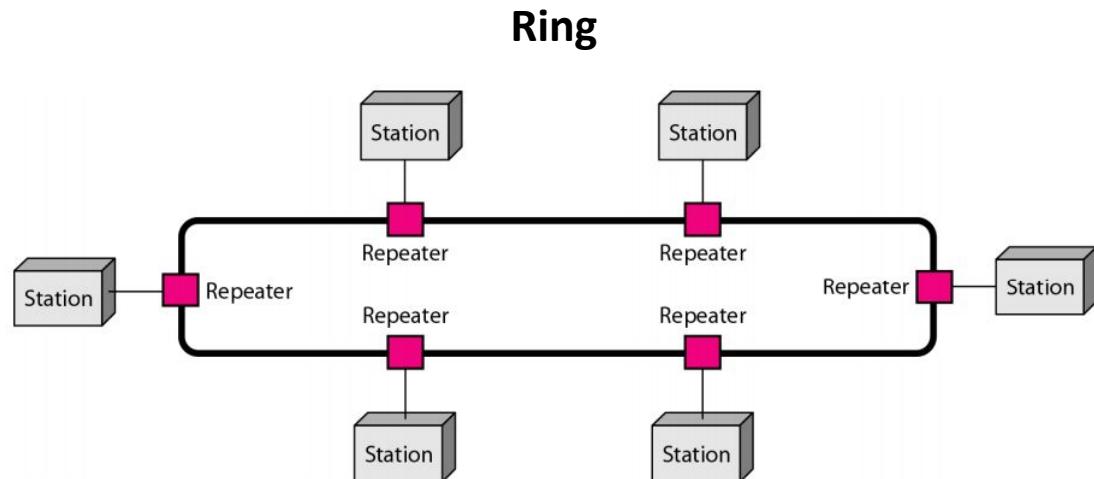
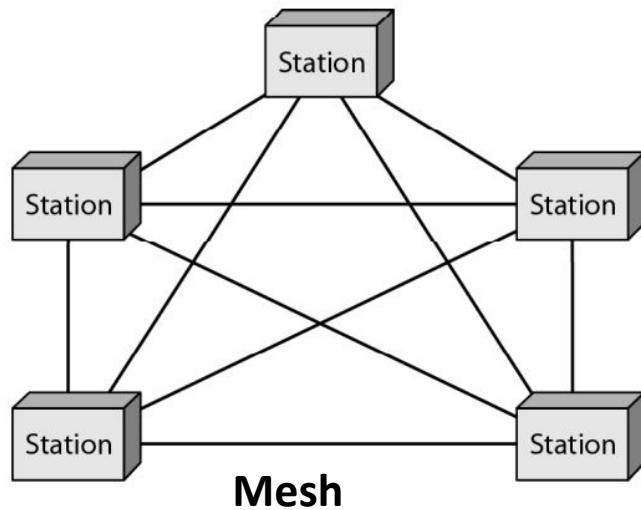


# Physical Topology

- The term physical topology refers to the way in which a network is laid out physically.
- The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another.
- There are **four** basic topologies possible: mesh, star, bus, and ring.



# Physical Topology



# Categories of Networks

- Networks are generally referring to two primary categories:
  - Local-area networks(LAN)
  - Wide-area networks(WAN)
- A **LAN** normally covers a small geographical area. A local area network (LAN) is usually privately owned and links the devices in a single office, building, or campus.
- A **WAN** provides long-distance transmission of data, image, audio and video information over large geographic areas that may comprise a country, a continent or even the whole world.

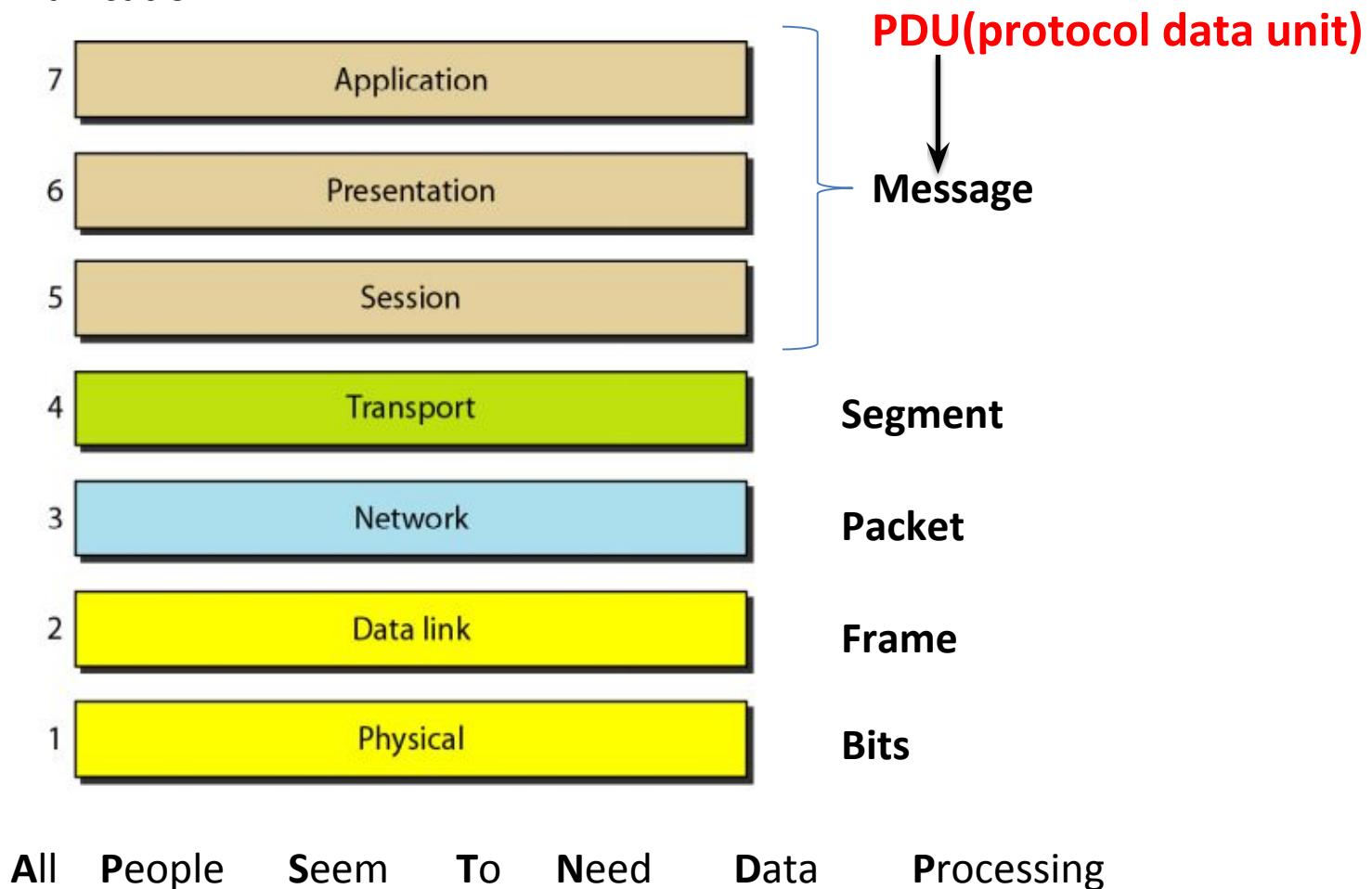
# Network Models:

## OSI Model

- The **International Standards Organization (ISO)** is a multinational body established in 1947, dedicated to worldwide agreement on international standards. *An ISO standard that covers all aspects of network communications is the **Open Systems Interconnection** model.* It was first introduced in the late **1970s**.
- An open system is a **set of protocols** that allows any two different systems to communicate regardless of their underlying architecture.
- The purpose of the OSI model is to show how to **facilitate communication between different systems** without requiring changes to the logic of the underlying hardware and software.
- The OSI model is not a protocol; it is a **conceptual model** for understanding and designing a network architecture.

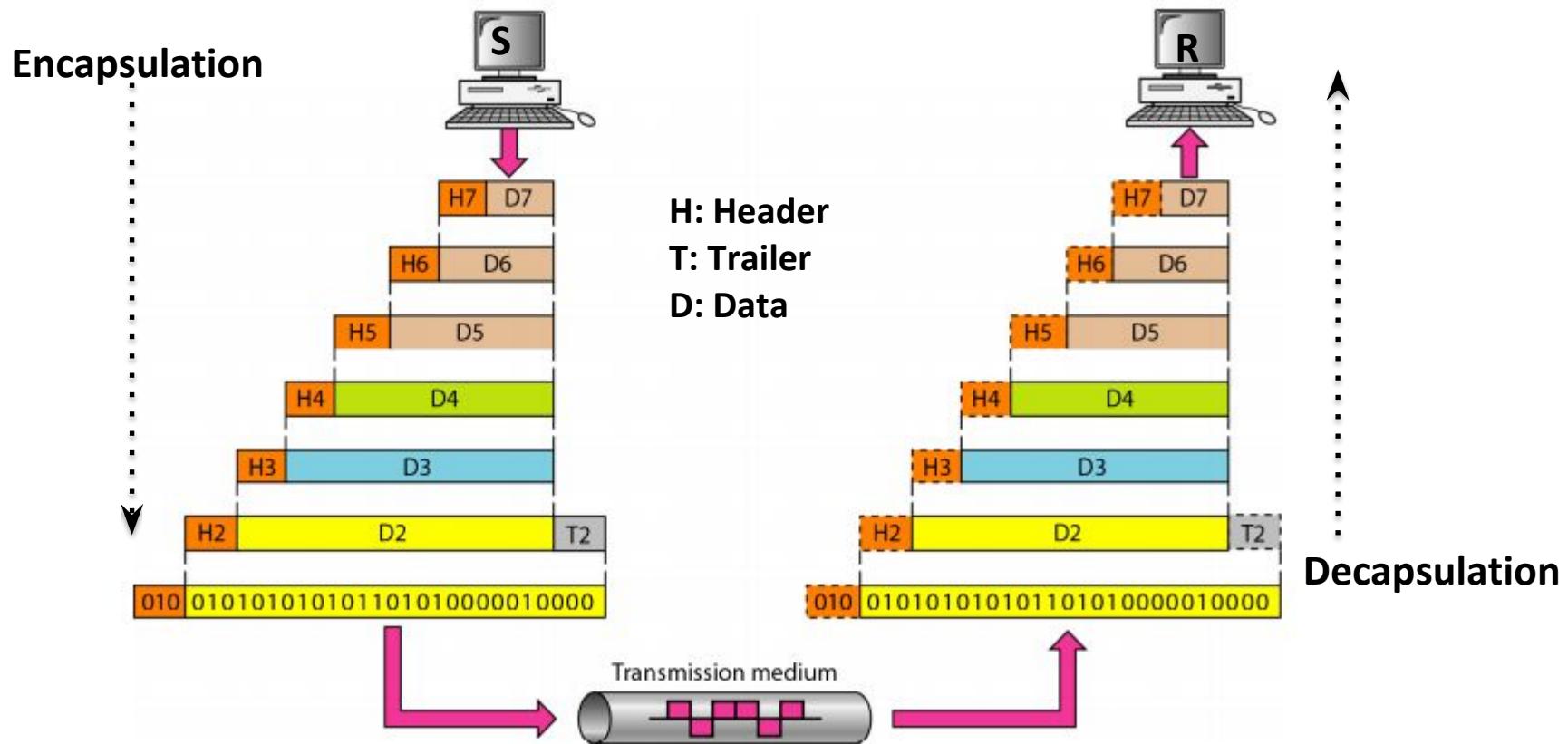
# OSI Model

- The OSI model is a **layered framework for the design of network systems** that allows communication between all types of computer systems. It consists of **seven** separate but related layers, each of which having different functionality to support data communication.



# Exchange of data using the OSI Model

The process starts at layer 7 (the application layer), then moves from layer to layer in descending, sequential order. At each layer, **a header**, or possibly **a trailer**, can be added to the data unit. Commonly, the trailer is added only at layer 2. When the formatted data unit passes through the physical layer (layer 1), it is changed into an electromagnetic signal and transported along a physical link.

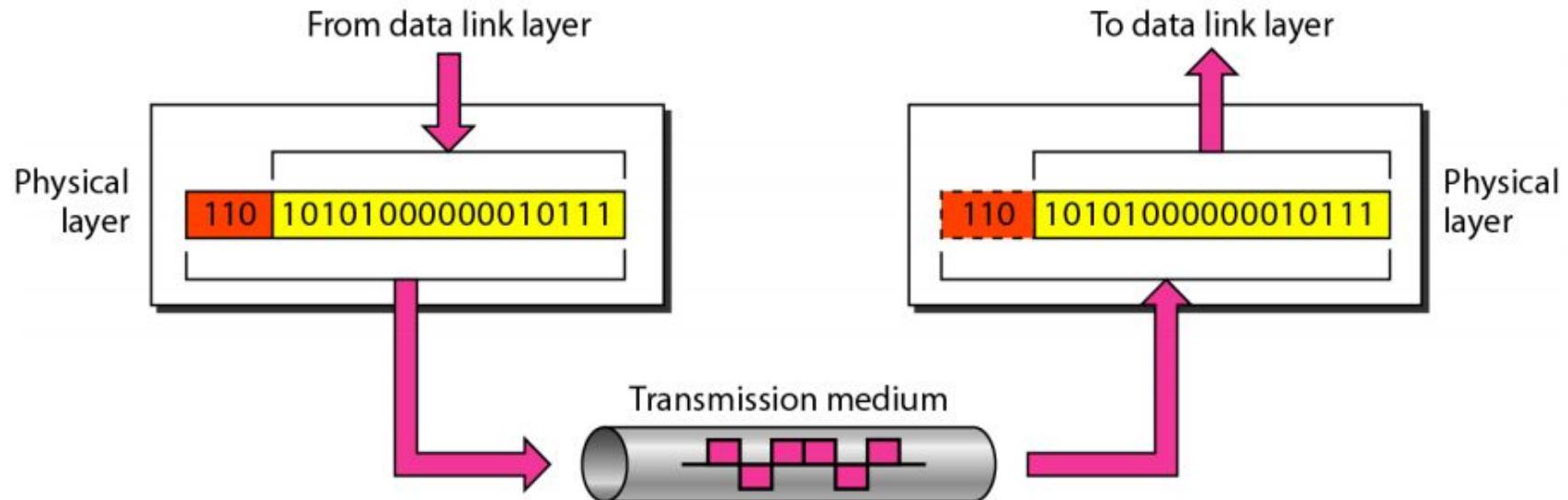


# Functionalities of Layers in OSI Model:

## 1. Physical Layer

It deals with the mechanical and electrical specifications of the interface and transmission medium.

**The physical layer is responsible for movements of individual bits from one hop (node) to the next.**

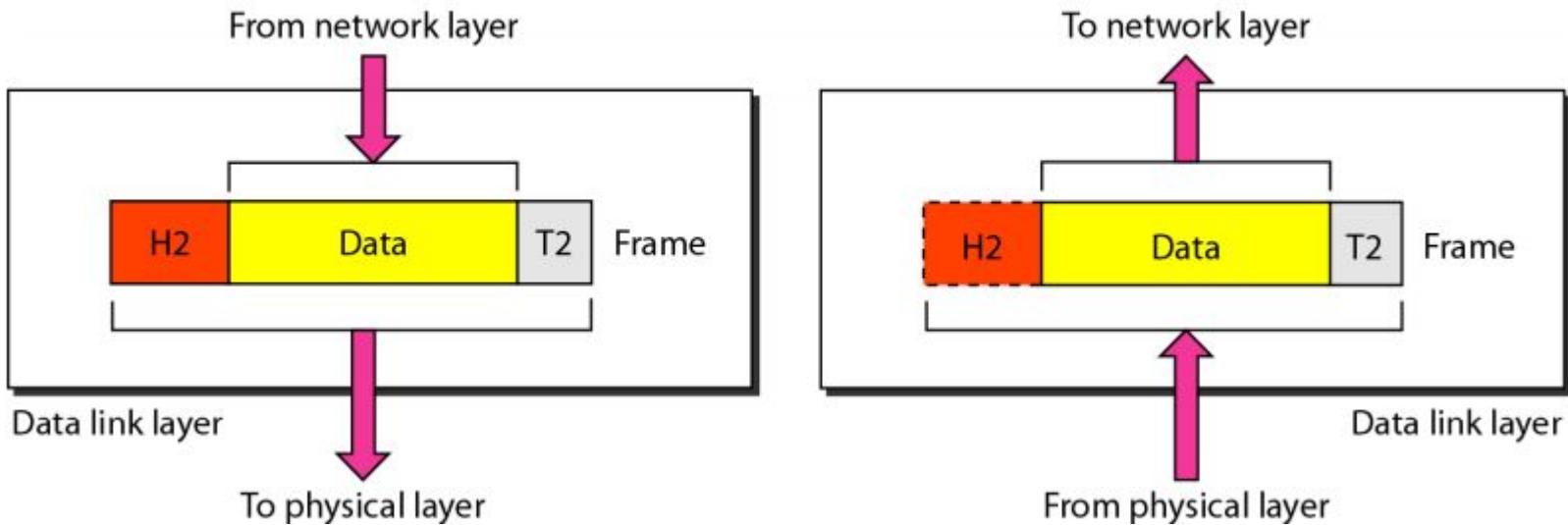


# Physical Layer Issues

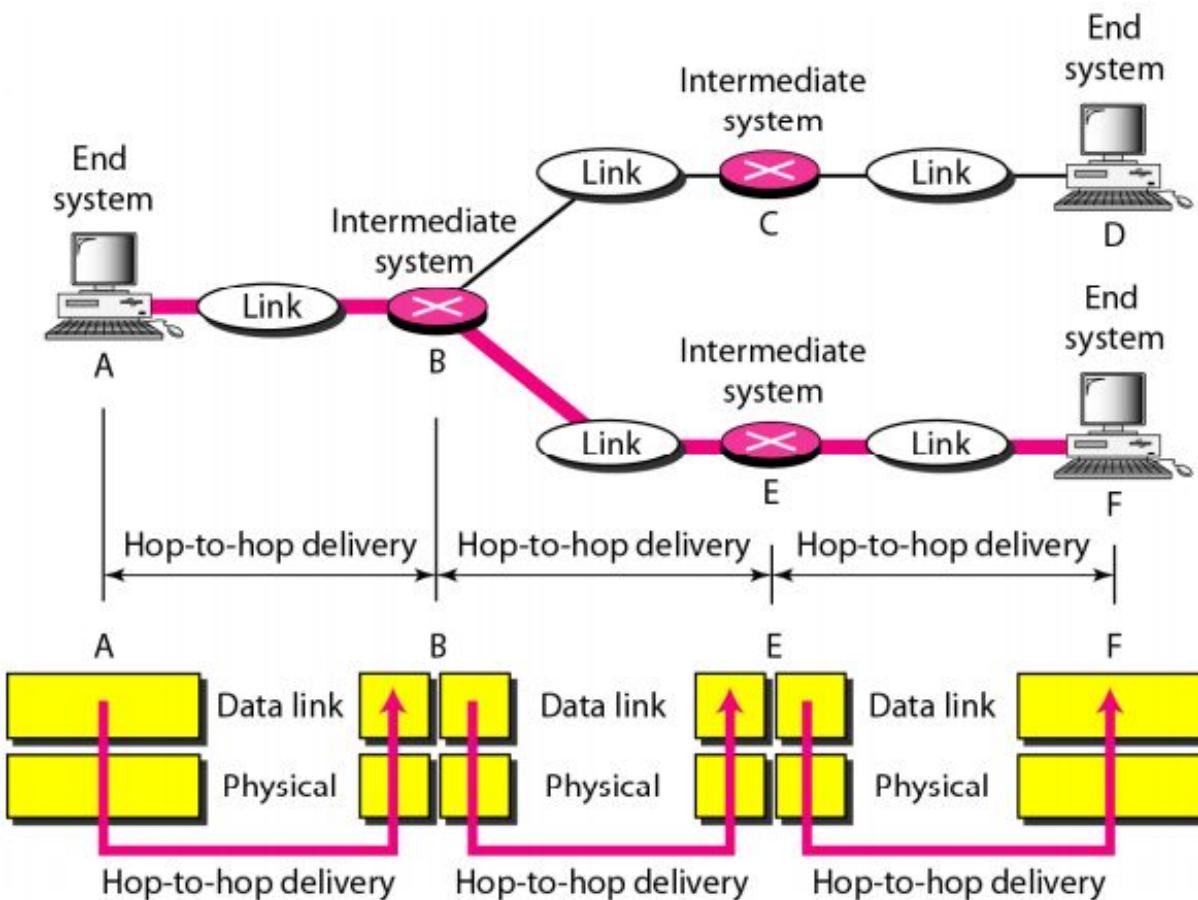
- a. **Physical characteristics of interfaces and medium.** The physical layer defines the characteristics of the interface between the devices and the transmission medium.
- b. **Representation of bits.** The physical layer data consists of a stream of bits (sequence of 0s or 1s) with no interpretation.
- c. **Data rate.** The transmission rate-the number of bits sent each second-is also defined by the physical layer.
- d. **Synchronization of bits.** The sender and receiver not only must use the same bit rate but also must be synchronized at the bit level.
- e. **Line configuration.** The physical layer is concerned with the connection of devices to the media; it may be point-to-point configuration or multi-point.
- f. **Physical topology.** The physical topology defines how devices are connected to make a network. Ex. Mesh, Star, Ring, Bus, or a hybrid topology.
- g. **Transmission mode.** The physical layer also defines the direction of transmission between two devices: simplex, half-duplex, or full-duplex.

# Data Link Layer

- The data link layer is responsible for moving frames from one hop (node) to the next.



# Hop-to-hop delivery



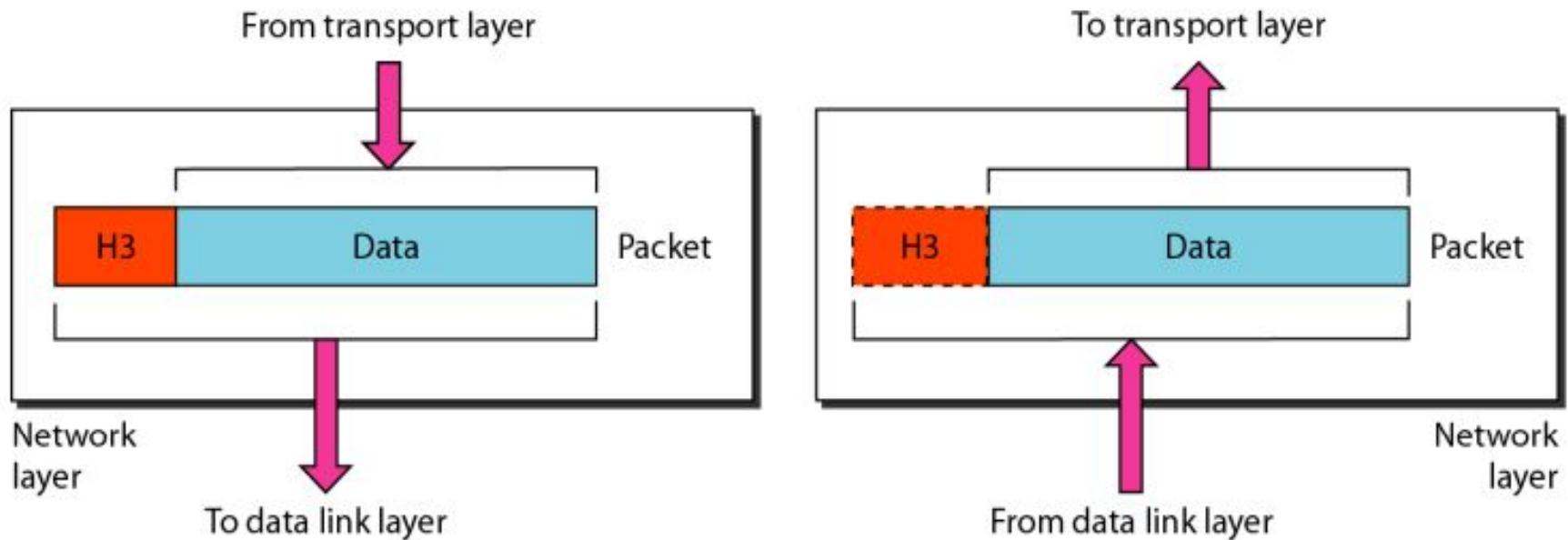
# Data Link Layer Issues

- 1. Framing.** The data link layer divides the stream of bits received from the network layer into manageable data units called frames.
- 2. Physical addressing.** If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender and/or receiver of the frame.
- 3. Flow control.** If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver.
- 4. Error control.** The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames.
- 5. Access control.** When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

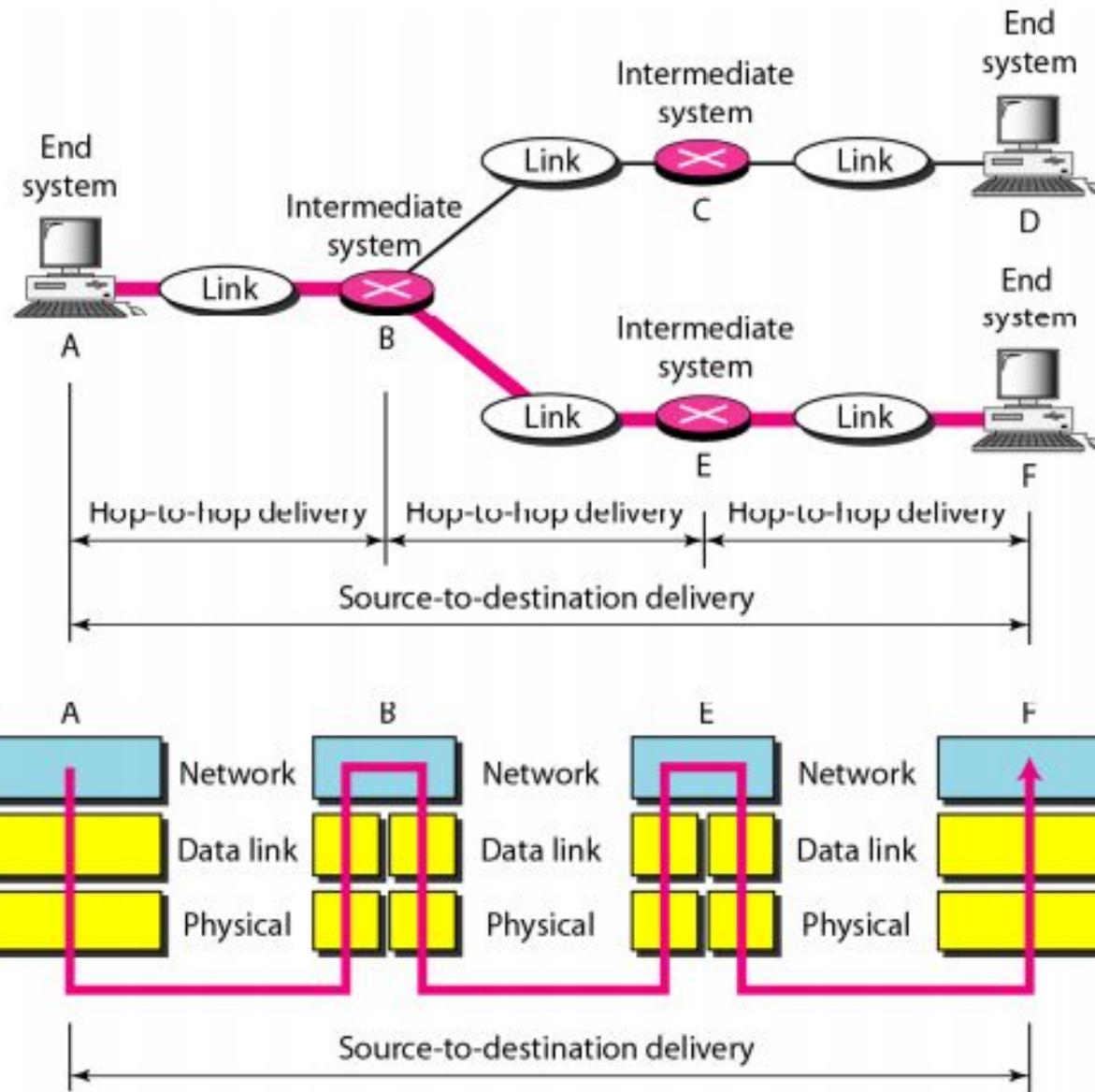
# Network Layer

The network layer is responsible for the source-to-destination delivery of a packet, possibly across multiple networks (links).

**The network layer is responsible for the delivery of individual packets from the source host to the destination host.**

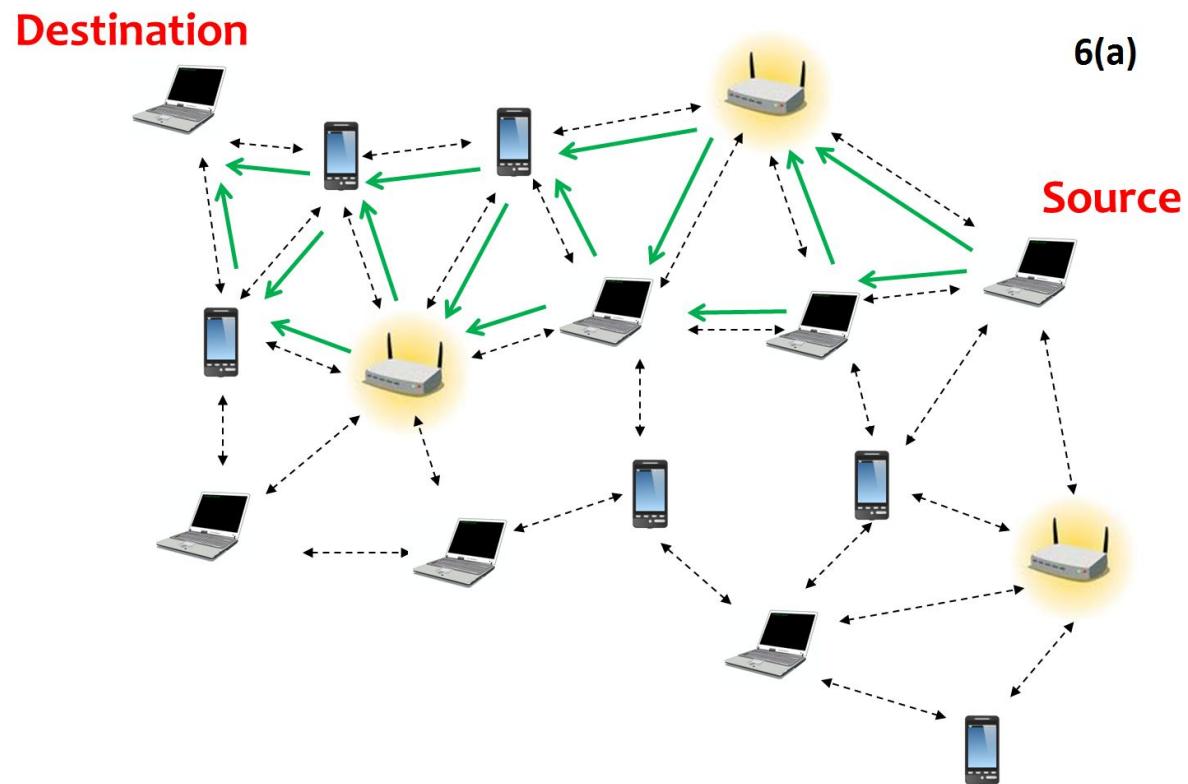


# Source-to-destination delivery



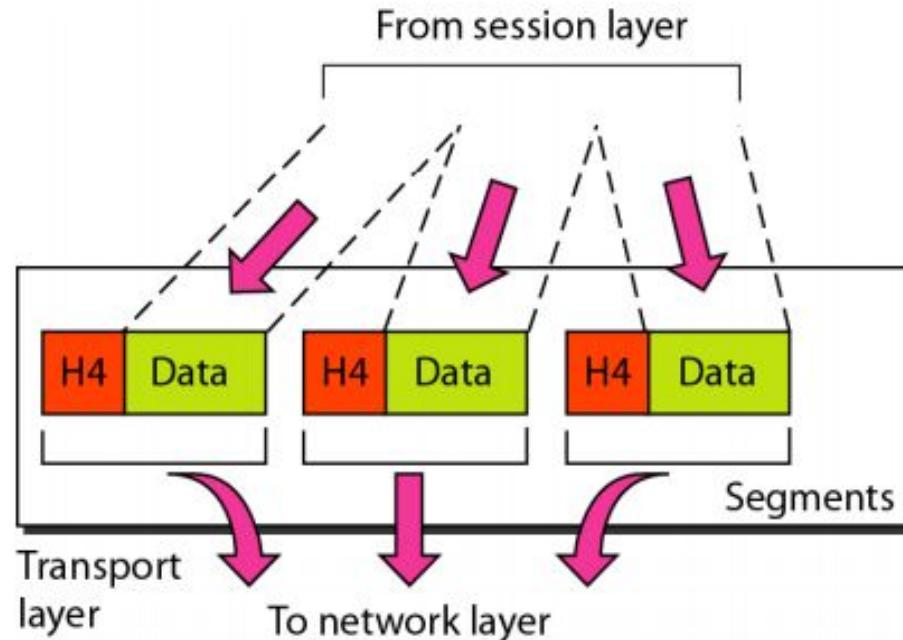
# Network Layer Issues

- 1. Logical (IP)addressing.** The network layer adds a header to the packet coming from the upper layer that, among other things, includes the logical addresses of the sender and receiver.
- 2. Routing.** When independent networks or links are connected to create internetworks or a large network, the connecting devices route or switch the packets to their final destination.

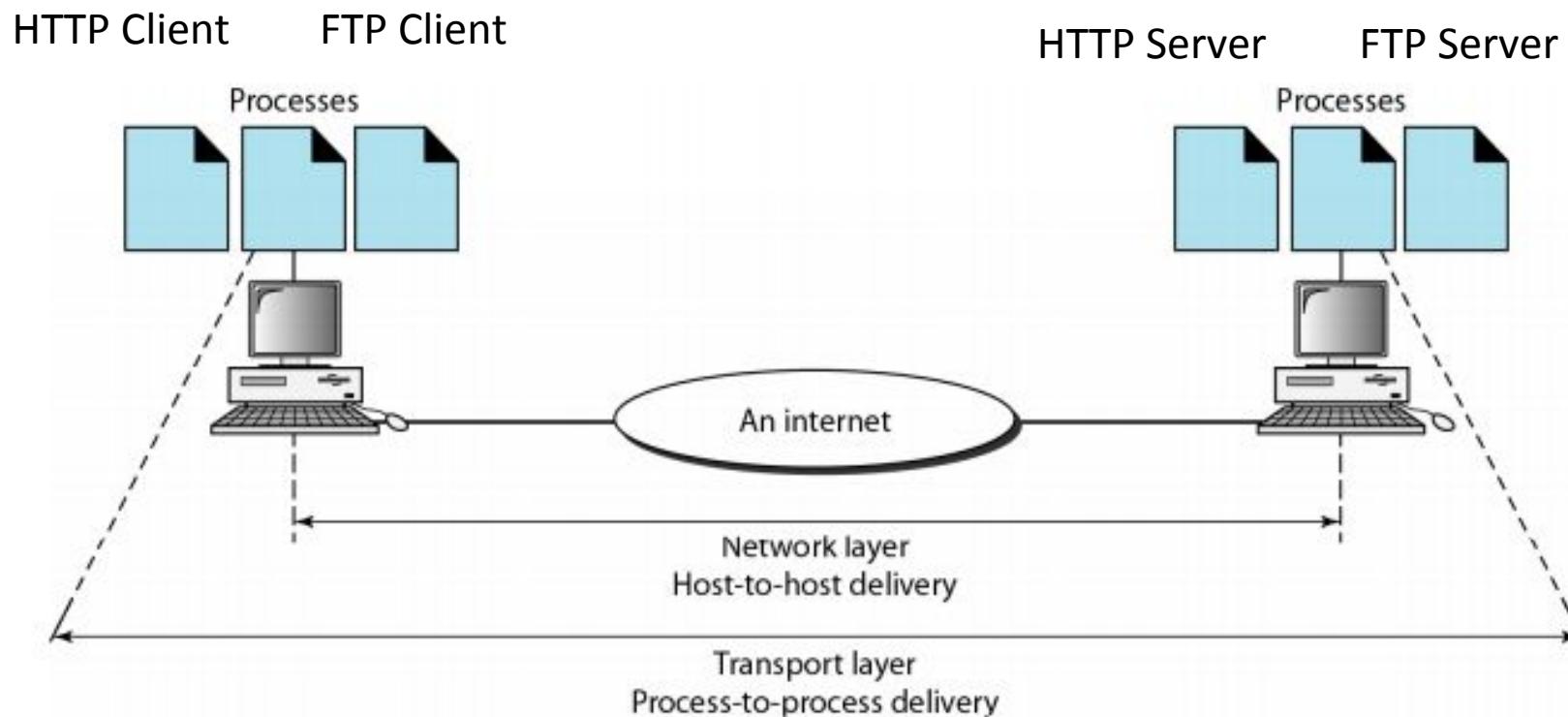


# Transport layer

- The transport layer is responsible for **process-to-process delivery of the entire message**.
- A process is an application program running on a host. E.g. FTP, HTTP.
- The transport layer ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source-to-destination level.

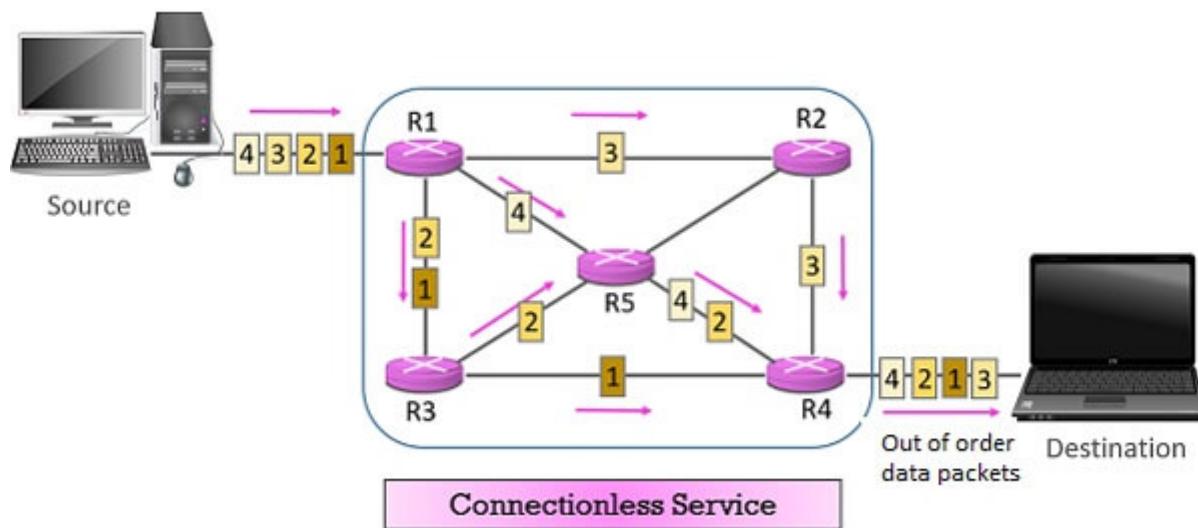
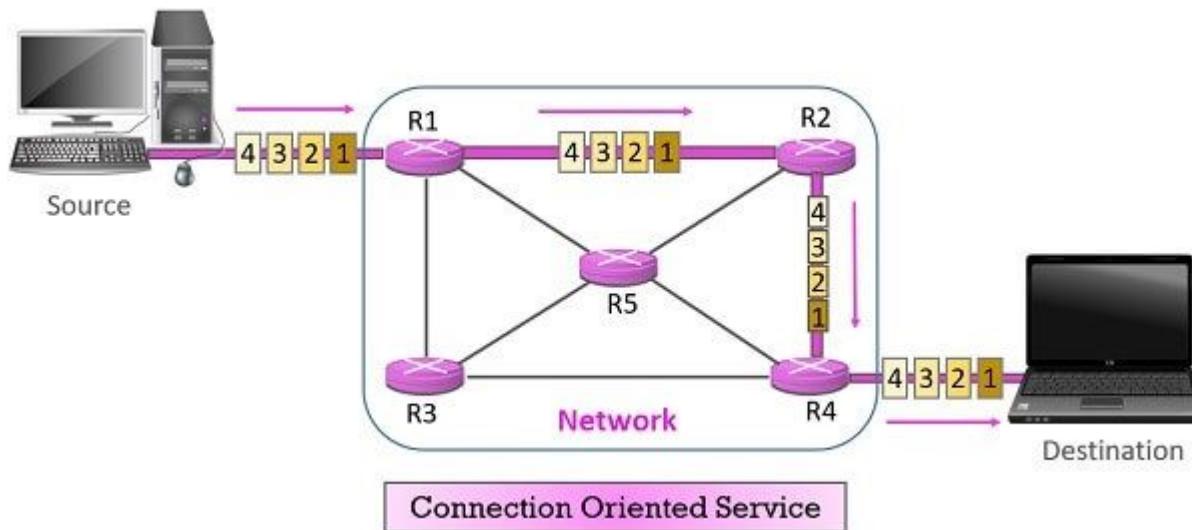


# Process-to-process delivery of a message



# Transport layer Issues

- a. **Port addressing.** Source-to-destination delivery means delivery not only from one computer to the next but also from a specific process (running program) on one computer to a specific process (running program) on the other.
- b. **Segmentation and reassembly.** A message is divided into transmittable segments, with each segment containing a sequence number. These numbers enable the transport layer to reassemble the message correctly upon arriving at the destination and to identify and replace packets that were lost in transmission.
- c. **Flow control.** The transport layer is responsible for flow control. However, flow control at this layer is performed end to end rather than across a single link.
- d. **Error control.** The transport layer is responsible for error control. However, error control at this layer is performed process-to-process rather than across a single link.
- e. **Congestion Control:** Due to increased network traffic packet may be delayed or lost. To resolve such situation congestion control techniques are used.
- f. **Connection control.** The transport layer can be either **connectionless** or **connection oriented**.
  - A **connectionless** transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine.
  - A **connection oriented** transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data are transferred, the connection is terminated.



# Session Layer

- The session layer is the **network dialog controller**. It **establishes, maintains, and synchronizes** the interaction among communicating systems.
  - a. **Dialog control.** The **session layer** tracks the dialogs between systems, which are also called sessions. This layer manages a session by initiating the opening and closing of sessions between end-user application processes.
  - b. **Synchronization.** The session layer allows a process to add checkpoints, or synchronization points, to a stream of data. Example: If a system is sending a file of 800 pages, adding checkpoints after every 50 pages is recommended. This ensures that 50 page unit is successfully received and acknowledged. This is beneficial at the time of crash as if a crash happens at page number 110; there is no need to retransmit 1 to 100 pages.

# Presentation Layer

The presentation layer is concerned with the syntax and semantics of the information exchanged between two systems.

**The presentation layer is responsible for translation, compression, and encryption.**

**a. Translation.** The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers, and so on. The information must be changed to common format(bit streams) before being transmitted. Ex- EBCDIC to ASCII conversion.

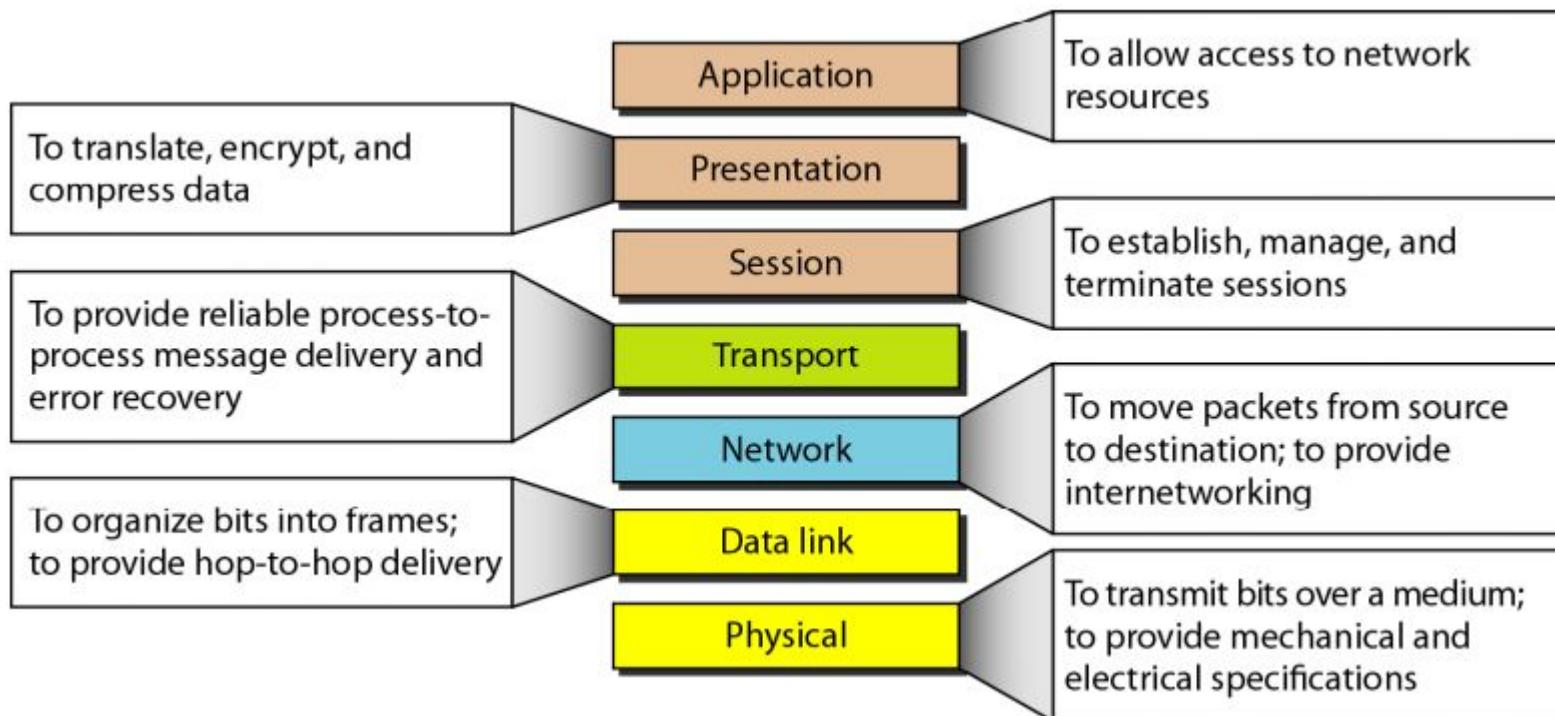
**c. Encryption.** A system must be able to ensure privacy. Encryption means that the sender transforms the original information to another form and sends the resulting message out over the network. Decryption reverses the original process to transform the message back to its original form.

**d. Compression.** Data compression reduces the number of bits contained in the information by this reduce the bandwidth requirement of the data. It is very important in the transmission of multimedia such as text, audio, and video.

# Application Layer

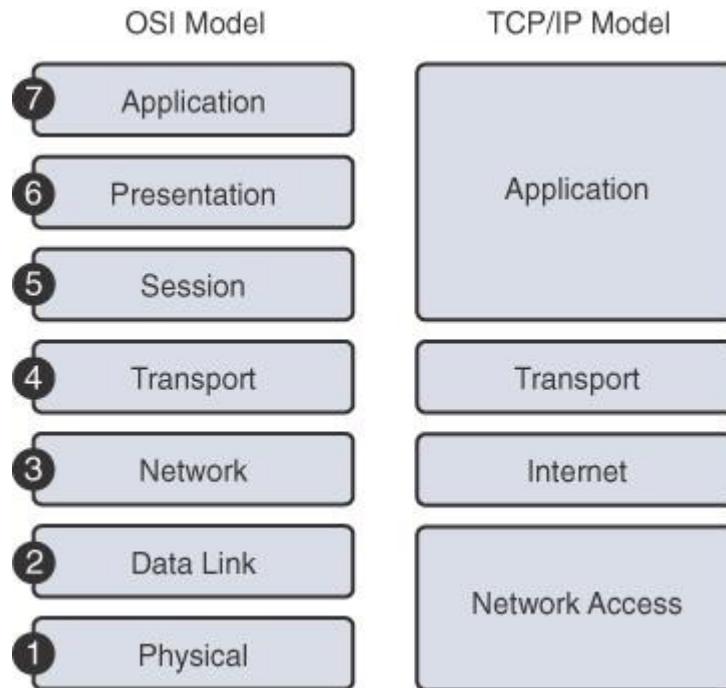
- The application layer enables the user, whether human or software, to access the network.
- It provides user interfaces and support for services such as electronic mail, remote file access and transfer and other types of distributed information services.
- A few examples of application layer protocols are the Hypertext Transfer Protocol (**HTTP**), File Transfer Protocol (**FTP**), Remote Login(**TELNET**), Simple Mail Transfer Protocol (**SMTP**), Simple Network Management Protocol(**SNMP**) and Domain Name System (**DNS**).

# Summary of layers



# TCP/IP Model

- The **OSI Model** we just looked at is just a reference/logical model.
- The TCP/IP protocol suite was developed prior to the OSI model.
- TCP/IP model, it was designed and developed by Department of Defense (DoD) in 1960s and is based on standard protocols.
- TCP/IP protocol suite is made of five layers: Network Access Layer(Physical+Data link), Internet Layer, Transport Layer and Application Layer.



# ADDRESSING

Three levels of addresses are used in an internet employing the TCP/IP protocols:

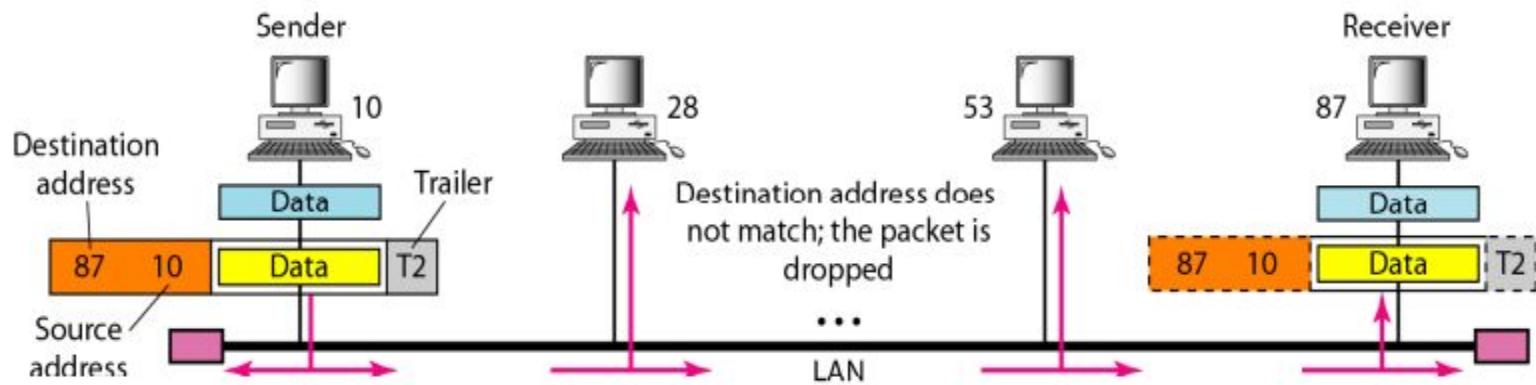
- 1. Physical (MAC) address at Data Link layer**
- 2. Logical (IP) address at Network Layer**
- 3. Port address at Transport Layer**

## **1. Physical(MAC) Addresses:**

The physical address, also known as the link address, is the address of a node as defined by its LAN or WAN. It is included in the frame used by the data link layer. It is the lowest-level address.

- The size and format of these addresses vary depending on the network.
- For example, **Ethernet** uses a **6-byte** (48-bit) physical address that is imprinted on the network interface card (NIC). LocalTalk (Apple), however, has a 1-byte dynamic address that changes each time the station comes up.

# Physical addresses Example



Numerals: Physical Address

# Logical (IP) Addresses

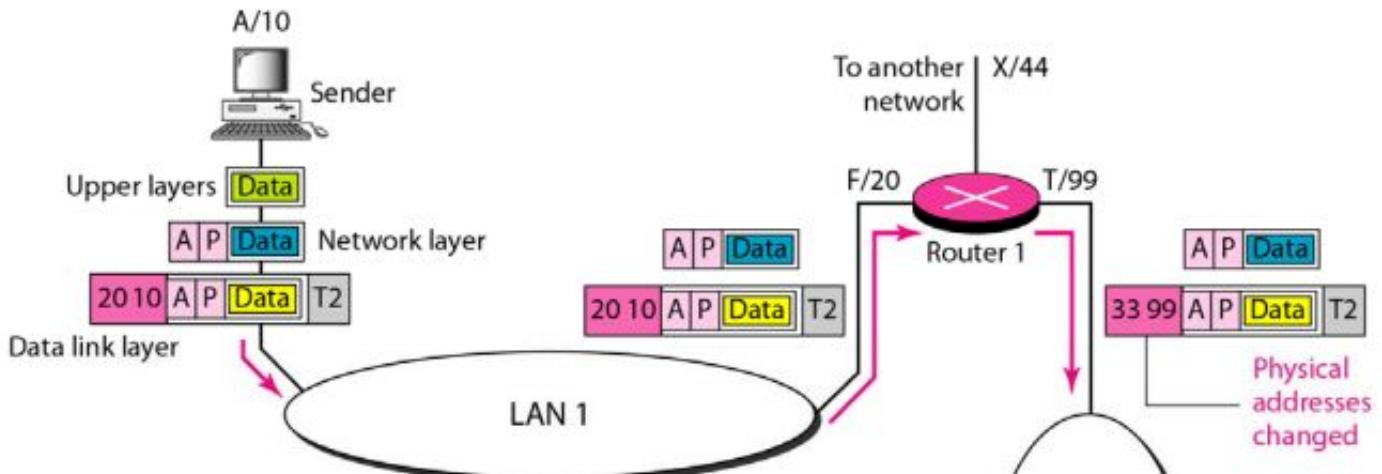
- Logical(IP)addresses are necessary for universal communications that are independent of underlying physical networks.
- Physical addresses are not adequate in an internetwork environment where different networks can have different address formats.
- A universal addressing system is needed in which each host can be identified uniquely, regardless of the underlying physical network.
- The logical addresses are designed for this purpose. A logical(IP) address in the Internet is currently a **32-bit address(IPv4)** that can uniquely define a host connected to the Internet.
- **128-bit IP address in IPv6.**

# Port Addresses

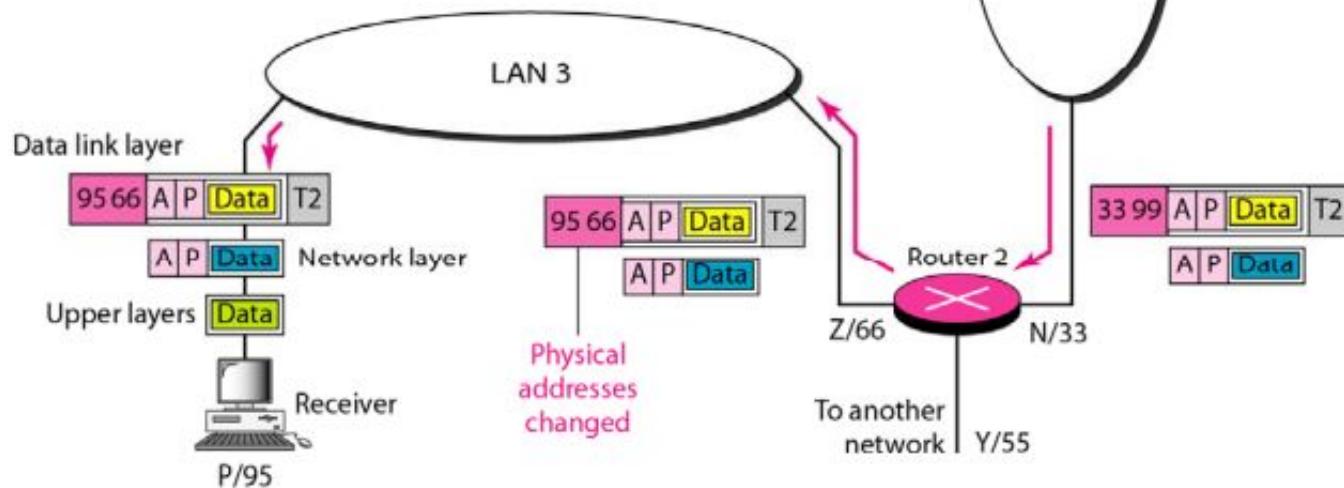
- The IP address and the MAC address are necessary for data to travel from a source to the destination host.
- However, arrival at the destination host is not the final objective of data communications on the Internet.
- Computers are devices that can run multiple processes at the same time. The end objective of Internet communication is a process communicating with another process.
- For example, computer **A** can communicate with computer **C** by using **TELNET**. At the same time, computer **A** communicates with computer **B** by using the File Transfer Protocol (**FTP**).
- For these processes to receive data simultaneously, we need a method to differentiate the different processes.
- In the **TCP/IP** architecture, the address assigned to a process is called a **port address**.
- A **port address** in TCP/IP is **16 bits** in length.

# Port Addresses/IP Addresses Example

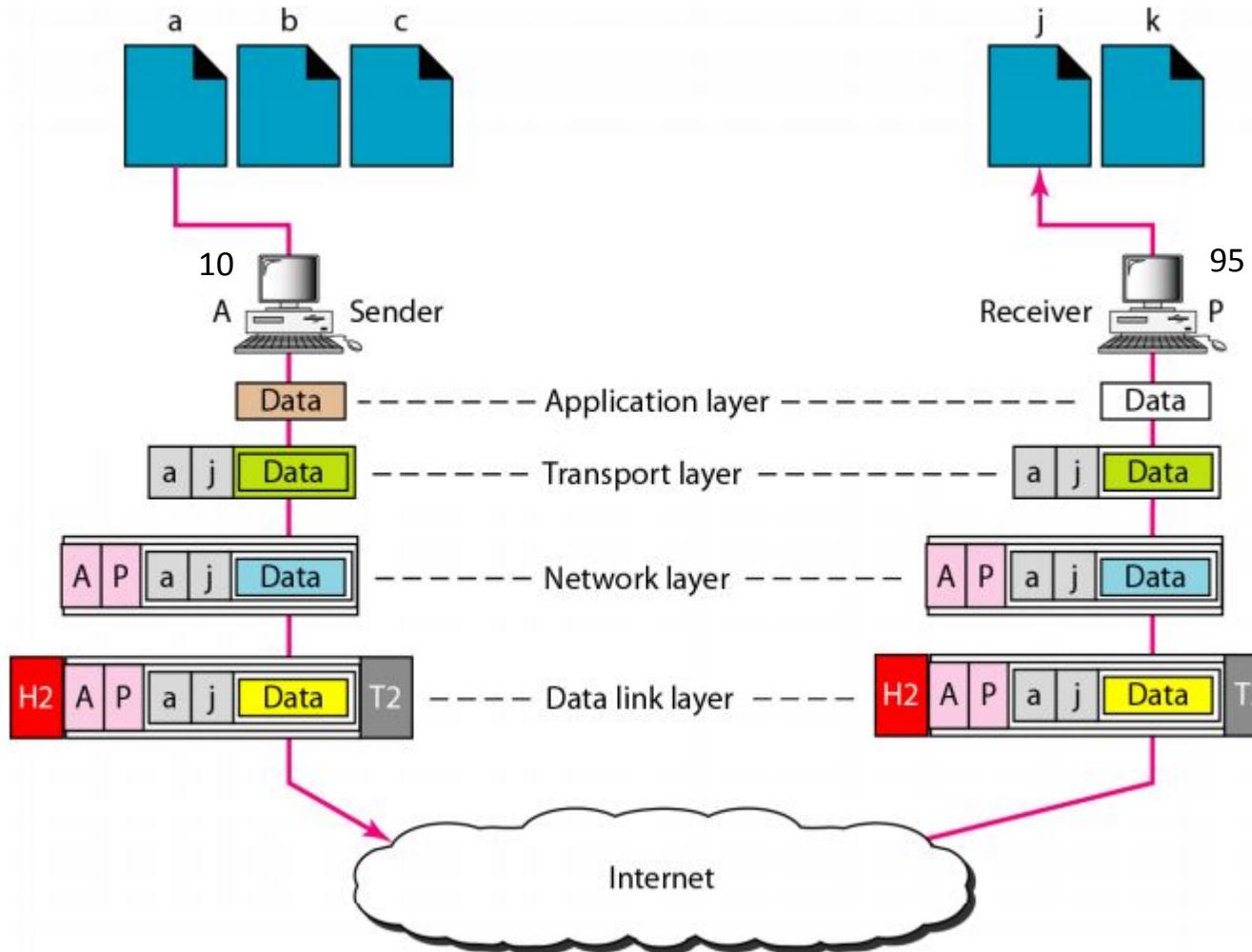
- Figure on next slide shows a part of an internet with two routers connecting three LANs. Each device computer or router has a pair of addresses (logical and physical) for each connection.
- The physical addresses will change from hop to hop, but the logical and port addresses usually remain the same.



Alphabets: Logical address  
Numerals: Physical address



# Port addresses



# **Lecture 2**

## **Logical Addressing- IPv4 Addresses**

**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Introduction

Communication at the network layer is **end-to-end(source to destination )**; a computer somewhere in the world needs to communicate with another computer somewhere else in the world. For this level of communication, we need a **global addressing scheme** ; we called this logical addressing or IP address.

## IPv4 ADDRESSES

- An **IPv4** address is a **32-bit** address.
- They are unique in the sense that each address defines one, and only one, connection to the Internet.
- Two devices on the Internet can never have the same address at the same time.
- But by using some strategies, an address may be assigned to a device for a time period and then taken away and assigned to another device.
- On the other hand, if a device operating at the network layer has ***m*** connections to the Internet, it needs to have ***m*** addresses.
- A router is such a device which needs as many IP addresses as the number of ports are there in it.

# IPv4 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)**.
- This means that, theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet. But the actual number is much less because of the restrictions imposed on the addresses.

# IPv4 Address Notations

There are **two** prevalent notations to show an IPv4 address:

- i. Binary notation and
- ii. Dotted decimal notation.

## i. Binary Notation

- In binary notation, the IPv4 address is displayed as 32 bits.
- Each octet is often referred to as a byte.
- So an IPv4 address referred to as a **32-bit address** or a **4-byte address**.
- The following is an example of an IPv4 address in binary notation:

**01110101 10010101 00011101 00000010**

# IPv4 Address Notations

## ii. Dotted-Decimal Notation

- To make the IPv4 address more **compact and easier to read**, Internet addresses are usually written in decimal form with a decimal point (dot) separating the bytes. The following is the dotted decimal notation of the previous address:

**117.149.29.2**

- Note that because each byte (octet) is 8 bits, each number in dotted-decimal notation is a value ranging from **0 to 255**.

# Types of IPv4 Addressing Schemes

There are two types of IPv4 addressing schemes:

- i. **Classful Addressing**
- ii. **Classless Addressing**

## i. Classful Addressing

- IPv4 addressing, at its inception, used the concept of classes. Although this scheme is becoming **obsolete**.
- In classful addressing, the address space is divided into **five classes**: A, B, C, D, and E.
- Each class occupies some part of the address space.
- 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.

# Finding the classes in binary and dotted-decimal notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

# Finding the classes in binary and dotted-decimal notation

## Example

Find the class of each address.

- a. 00000001 00001011 00001011 11101111
- b. 11000001 10000011 00011011 11111111
- c. 14.23.120.8
- d. 252.5.15.111

## Solution

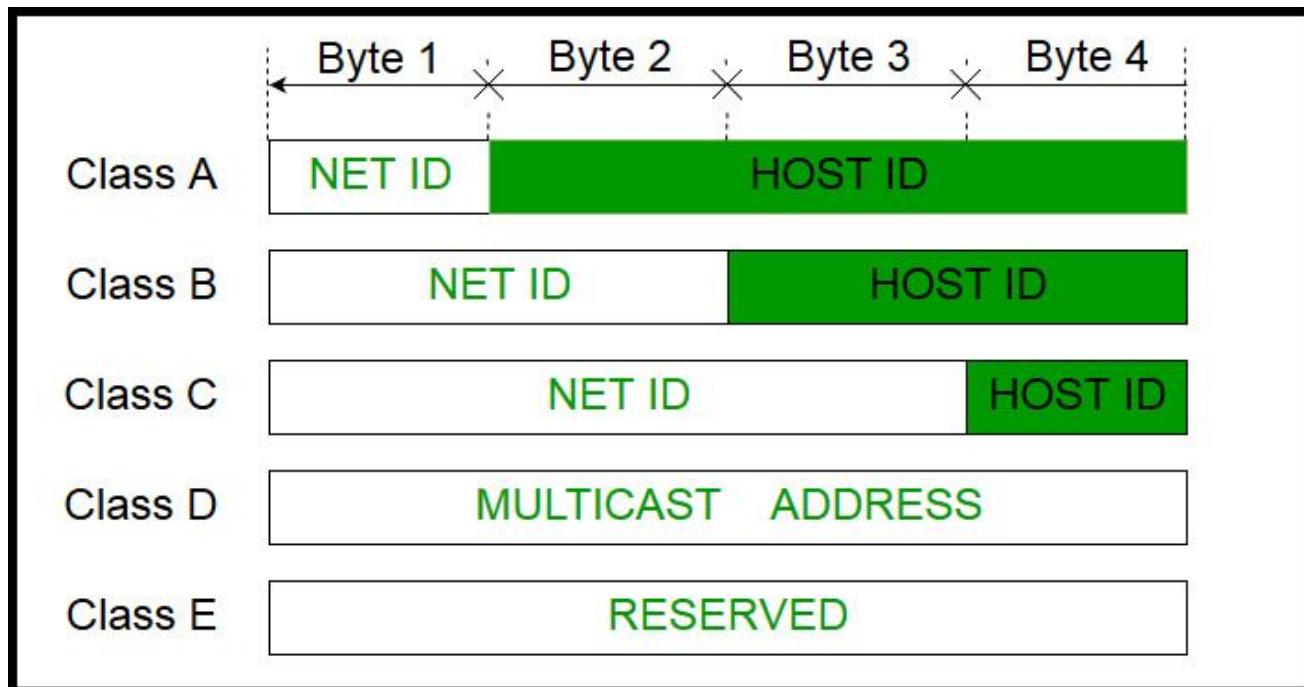
- a. The first bit is 0. This is a **class A** address.
- b. The first 2 bits are 1; the third bit is 0. This is a **class C** address.
- c. The first byte is 14 (between 0 and 127); the **class is A**.
- d. The first byte is 252 (between 240 and 255); the **class is E**.

# IPv4 address

Each IPv4 address is divided into two parts:

- **Network ID**
- **Host ID**

The class of IP address is used to determine the bits used for network ID and host ID and the number of total networks and hosts possible in that particular class.



# Classes and Blocks

In Classful addressing each class is divided into a fixed number of blocks with each block having a fixed size as shown in following Table.

Class	Number of Blocks	Block Size	Application
A	$2^7=128$	$2^{24}=16,777,216$	Unicast (large organizations )
B	$2^{14}=16,384$	$2^{16}=65,536$	Unicast (midsize organizations )
C	$2^{21}=2,097,152$	$2^8=256$	Unicast (small organizations )
D	1	$2^{28}=268,435,456$	Multicast
E	1	$2^{28}=268,435,456$	Reserved

# **Lecture 2**

## **Logical Addressing- IPv4 Addresses**

**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Introduction

Communication at the network layer is **end-to-end(source to destination )**; a computer somewhere in the world needs to communicate with another computer somewhere else in the world. For this level of communication, we need a **global addressing scheme** ; we called this logical addressing or IP address.

## IPv4 ADDRESSES

- An **IPv4** address is a **32-bit** address.
- They are unique in the sense that each address defines one, and only one, connection to the Internet.
- Two devices on the Internet can never have the same address at the same time.
- But by using some strategies, an address may be assigned to a device for a time period and then taken away and assigned to another device.
- On the other hand, if a device operating at the network layer has ***m*** connections to the Internet, it needs to have ***m*** addresses.
- A router is such a device which needs as many IP addresses as the number of ports are there in it.

# IPv4 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)**.
- This means that, theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet. But the actual number is much less because of the restrictions imposed on the addresses.

# IPv4 Address Notations

There are **two** prevalent notations to show an IPv4 address:

- i. Binary notation and
- ii. Dotted decimal notation.

## i. Binary Notation

- In binary notation, the IPv4 address is displayed as 32 bits.
- Each octet is often referred to as a byte.
- So an IPv4 address referred to as a **32-bit address** or a **4-byte address**.
- The following is an example of an IPv4 address in binary notation:

**01110101 10010101 00011101 00000010**

# IPv4 Address Notations

## ii. Dotted-Decimal Notation

- To make the IPv4 address more **compact and easier to read**, Internet addresses are usually written in decimal form with a decimal point (dot) separating the bytes. The following is the dotted decimal notation of the previous address:

**117.149.29.2**

- Note that because each byte (octet) is 8 bits, each number in dotted-decimal notation is a value ranging from **0 to 255**.

# Types of IPv4 Addressing Schemes

There are two types of IPv4 addressing schemes:

- i. **Classful Addressing**
- ii. **Classless Addressing**

## i. Classful Addressing

- IPv4 addressing, at its inception, used the concept of classes. Although this scheme is becoming **obsolete**.
- In classful addressing, the address space is divided into **five classes**: A, B, C, D, and E.
- Each class occupies some part of the address space.
- 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.

# Finding the classes in binary and dotted-decimal notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

# Finding the classes in binary and dotted-decimal notation

## Example

Find the class of each address.

- a. 00000001 00001011 00001011 11101111
- b. 11000001 10000011 00011011 11111111
- c. 14.23.120.8
- d. 252.5.15.111

## Solution

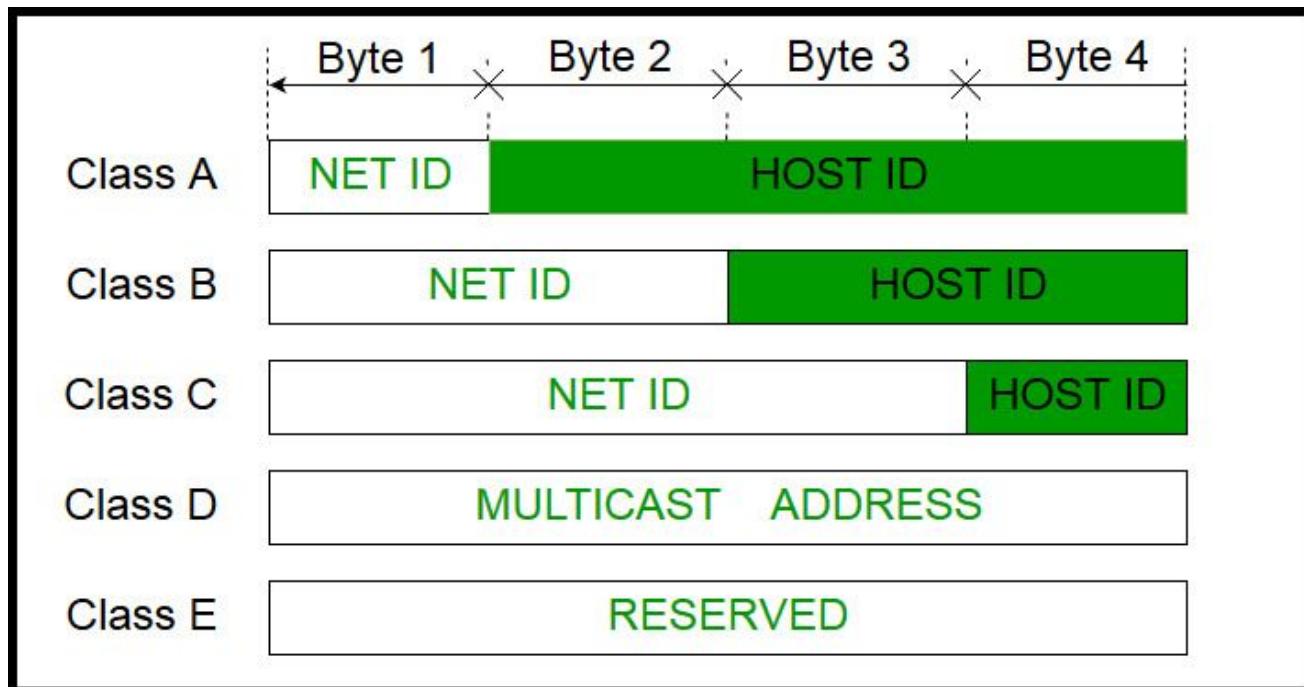
- a. The first bit is 0. This is a **class A** address.
- b. The first 2 bits are 1; the third bit is 0. This is a **class C** address.
- c. The first byte is 14 (between 0 and 127); the **class is A**.
- d. The first byte is 252 (between 240 and 255); the **class is E**.

# IPv4 address

Each IPv4 address is divided into two parts:

- **Network ID**
- **Host ID**

The class of IP address is used to determine the bits used for network ID and host ID and the number of total networks and hosts possible in that particular class.



# Classes and Blocks

In Classful addressing each class is divided into a fixed number of blocks with each block having a fixed size as shown in following Table.

Class	Number of Blocks	Block Size	Application
A	$2^7=128$	$2^{24}=16,777,216$	Unicast (large organizations )
B	$2^{14}=16,384$	$2^{16}=65,536$	Unicast (midsize organizations )
C	$2^{21}=2,097,152$	$2^8=256$	Unicast (small organizations )
D	1	$2^{28}=268,435,456$	Multicast
E	1	$2^{28}=268,435,456$	Reserved

# **Lecture 3.1**

## **Internet protocol – IPv4**

**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Internet Protocol version 4(IPv4)

The Internet Protocol version 4 (**IPv4**) is a Network Layer protocol in **TCP/IP** protocols suit.

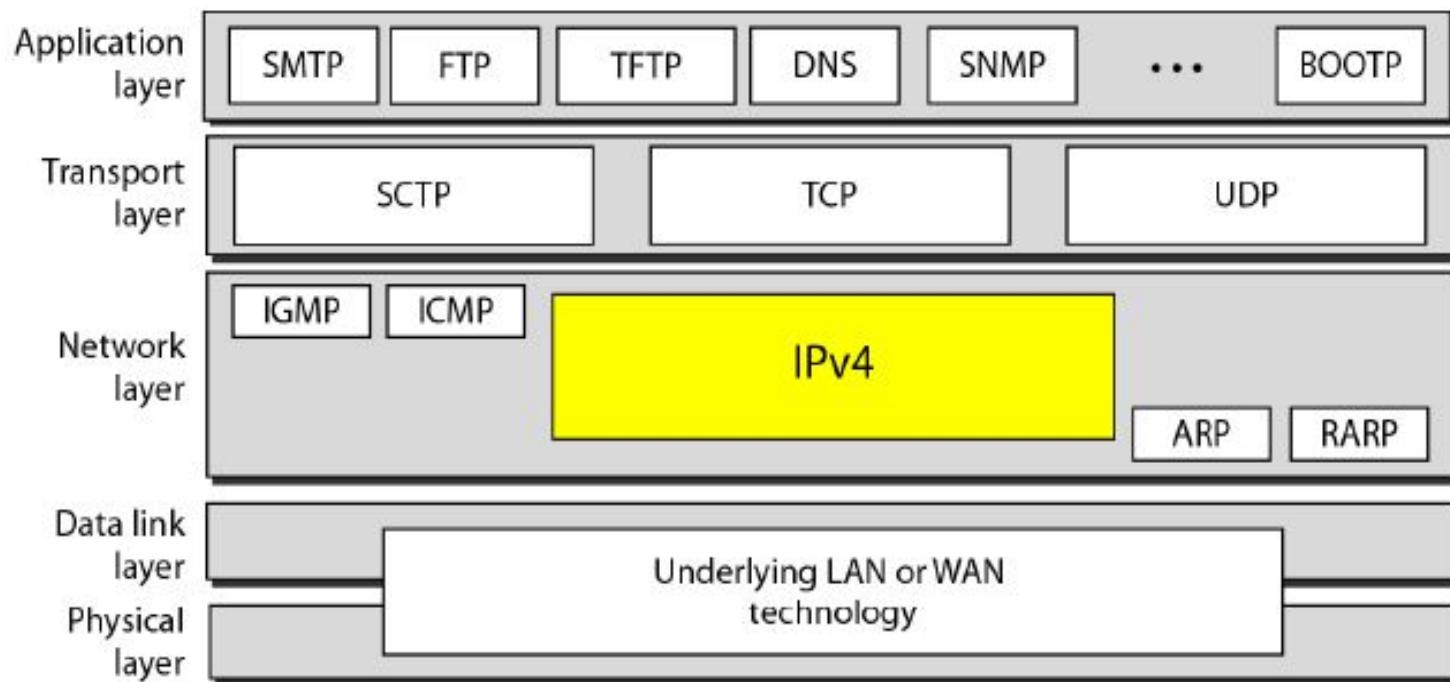
## Characteristics of IPv4 protocol

- **IPv4** is an **Unreliable** and **Connectionless Datagram protocol**- a **best-effort delivery service**.
- The term ***best-effort*** means that IPv4 provides **no error control or flow control** (except for error detection on the header).
- IPv4 **does its best** to get a transmission through to its destination, but with **no guarantees**.
- If **reliability is important**, IPv4 must be **paired with a reliable protocol at Transport Layer** such as TCP.

# **IPv4**

- IPv4 is also a **connectionless packet-switching** network that uses the **datagram approach**.
- This means that **each datagram is handled independently**, and each datagram **can follow a different route to the destination**.
- This implies that datagrams sent by the same source to the same destination could **arrive out of order**. Also, some could be **lost or corrupted** during transmission.
- IPv4 relies on a higher-level protocol like **TCP** to take care of all these problems.

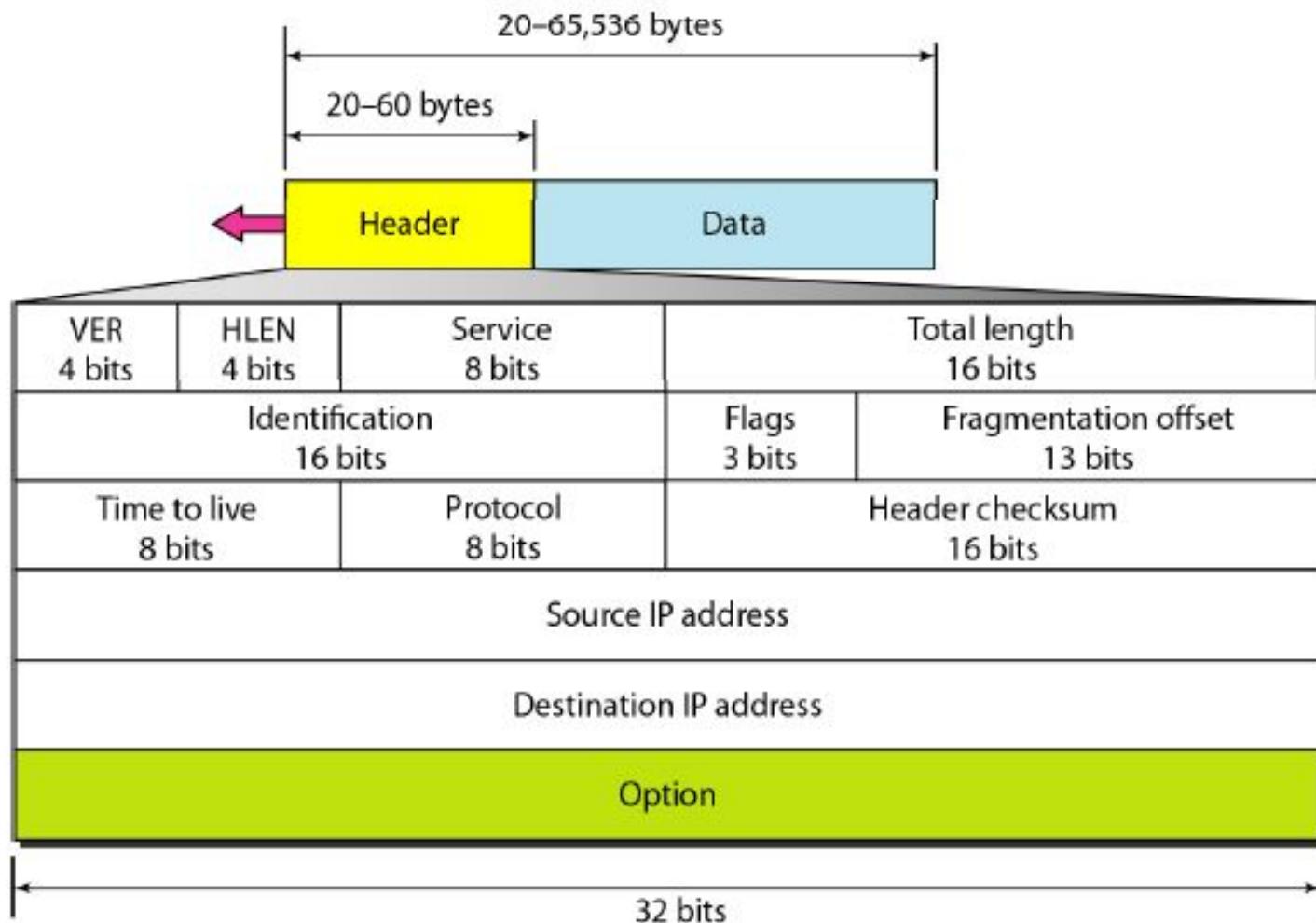
# Position of IPv4 in TCP/IP protocol suite



# IPv4 Datagram

- Packets in the IPv4 layer are called **datagrams**. Figure on next page shows the **IPv4 datagram format**. A datagram is a **variable-length** packet consisting of **two parts: header and data**.
- The **header** is **20 to 60 bytes in length** and contains information essential to **routing** and **delivery**. It is customary in **TCP/IP** to show the header in **4-byte sections**. A brief description of each field is in order:
  - a. **Version (VER)**. This **4-bit** field defines the **version** of the IPv4 protocol. Currently the version is **4**. However, version **6** (or **IPv6**) may totally replace version 4 in the future.
  - b. **Header length (HLEN)**. This **4-bit** field defines the **total length** of the **datagram header** in **4-byte words**. This field is needed because the length of the header is **variable** (between 20 and 60 bytes). When there are **no options**, the header length is **20 bytes**, and the value of this field is **5** ( $5 \times 4 = 20$ ). When the option field is at its **maximum size**, the value of this field is **15** ( $15 \times 4 = 60$ ).

# IPv4 Datagram



# Datagram

c. Services(8 bits). IETF has changed the interpretation and name of this 8-bit field.

This field, previously called **Service type**, is now called **Differentiated services**.

## 1. 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**.
- **Precedence** is a **3-bit** subfield ranging from 0 (**000** in binary) to 7 (**111** in binary).
- **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**.
- Some **datagrams** in the Internet are **more important** than others. For example, a datagram used for **network management** is much more **urgent and important**.

# Datagram

- **TOS bits** is a **4-bit** subfield with **each bit** having a **special meaning**.
- Although a bit can be either **0 or 1**, one and only one of the bits can have the value of **1** in each datagram.
- The bit patterns and their interpretations are given in Table below. With only 1 bit set at a time, we can have **five different types of services**.
- Application programs can request a specific type of service. The defaults for some applications are shown in Table.

<i>TOS Bits</i>	<i>Description</i>
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

# Default types of service

<i>Protocol</i>	<i>TOS Bits</i>	<i>Description</i>
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput

# Default types of service

- Interactive activities, activities requiring immediate attention, and activities requiring immediate response need **minimum delay**. e.g. TELNET.
- Those activities that send bulk data require **maximum throughput**. E.g. FTP(data)
- Management activities need **maximum reliability**. e.g. SNMP.
- Background activities need **minimum cost**. e.g. NNTP.

# Services: Differentiated Services

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

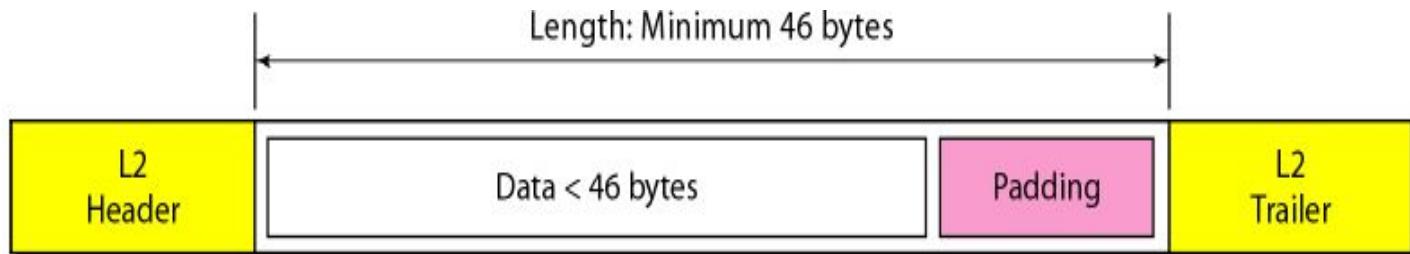
1. When the **3 rightmost bits are 0s**, the **3 leftmost bits are interpreted the same as the precedence bits** in the service type interpretation.
2. When the **3 rightmost bits are not all 0s**, the **6 bits define 64 services**(divided into three categories) :
  - The **first category** contains **32 service types**; the **second** and the **third** each contain **16 service types**.
  - The **first category** is assigned by the Internet authorities (IETF).
  - The **second category** be used by local authorities (organizations).
  - The **third category** is temporary and can be used for experimental purposes.

# Datagram

d. **Total length.** This is a **16-bit** field that defines the **total length (header plus data)** of the **IPv4 datagram in bytes**.

$$\text{Length of data} = \text{total length} - \text{header length}$$

- Since the field length is 16 bits, the **total length of the IPv4 datagram** is limited to **65,535 ( $2^{16} - 1$ ) bytes**, of which **20 to 60 bytes** are the header and the **rest is data** from the upper layer.
- If the size of an IPv4 datagram is **less than 46 bytes**, some **padding** will be added to meet this requirement. In this case, when a machine decapsulates the datagram, it needs to check the total length field to determine how much is really data and how much is padding.

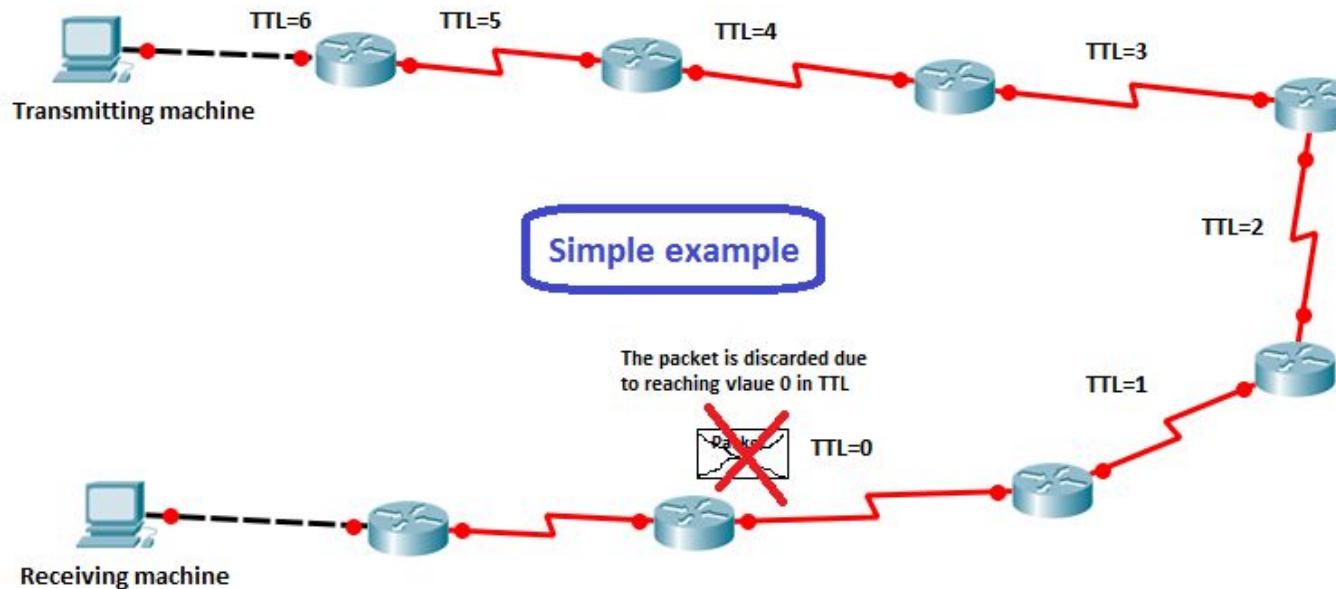


# Datagram

- e. **Identification.** This field is used in fragmentation (discussed in the next section).
- f. **Flags.** This field is used in fragmentation (discussed in the next section).
- g. **Fragmentation offset.** This field is used in fragmentation (discussed in the next section).
- h. **Time to live(TTL).** A datagram has a **limited lifetime** in its travel through an internet. This field is used mostly to **control the maximum number of hops** (routers) **visited** by the datagram.
  - When a source host sends the datagram, it stores a number in this field. This value is approximately **2 times the maximum number of hops between any two hosts**.
  - Each router that processes the datagram **decrements this number by 1**. If this value, after being decremented, is zero, the router **discards** the datagram.

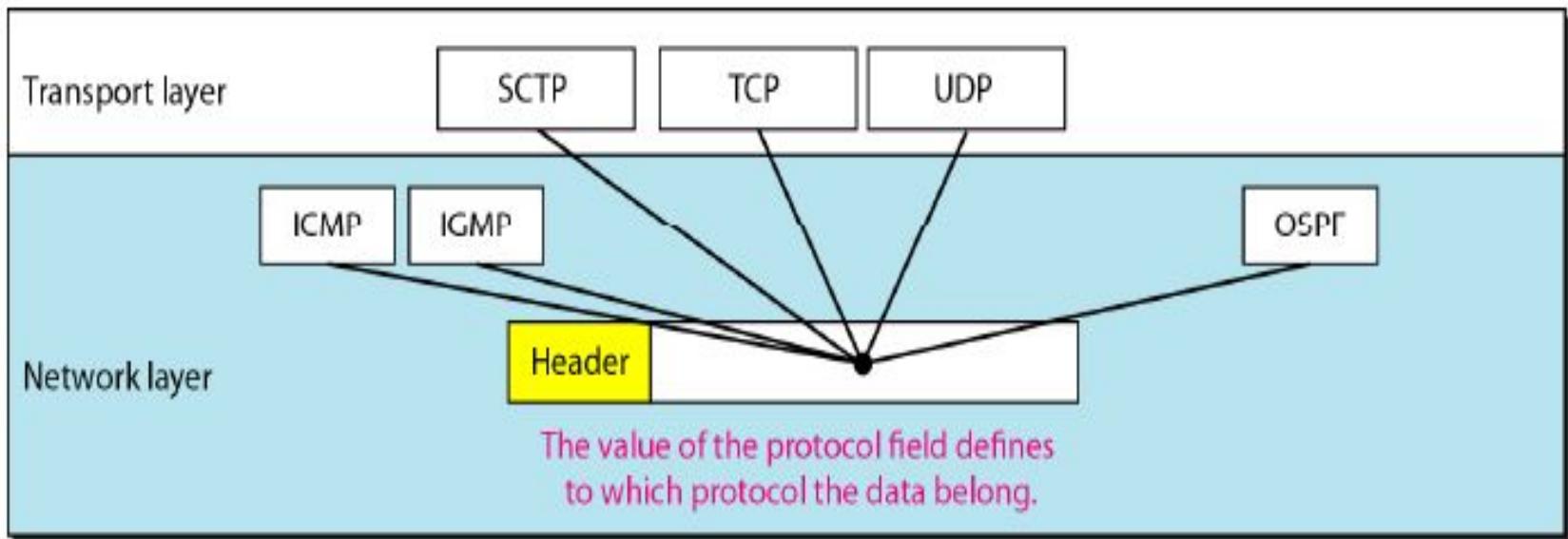
# Need for TTL

- This field is needed because routing tables in the Internet can become **corrupted**.
- A datagram may travel between two or more routers for a long time without ever getting delivered to the destination host. This field limits the lifetime of a datagram.
- Another use of this field is to **intentionally limit the journey of the packet**. For example, if the source wants to confine the packet to the local network, it can store 1 in this field. When the packet arrives at the first router, this value is decremented to 0, and the datagram is discarded.



# Datagram

- i. **Protocol.** This **8-bit** field defines the **higher-level protocol** that uses the services of the IPv4 layer. An IPv4 datagram can encapsulate data from several higher-level protocols such as **TCP, UDP, ICMP, and IGMP**.
- Since the **IPv4** protocol carries data from different other protocols, the value of this field helps the receiving network layer know to which protocol the data belong



# Datagram

- j. **Checksum.** The checksum is used to secure the IPv4 **header**.
- k. **Source address.** This **32-bit field** defines the **IPv4 address of the source**. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.
- l. **Destination address.** This **32-bit field** defines the **IPv4 address of the destination**. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.
- m. **Options.** The header of the IPv4 datagram is made of two parts: a **fixed part** and a **variable part**. The fixed part is 20 bytes long. The variable part comprises the **options** that can be a maximum of 40 bytes. They can be used for **network testing and debugging**.

# Examples

## Example 20.1

An IPv4 packet has arrived with the first 8 bits as shown:

01000010

The receiver discards the packet. Why?

### Solution

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 ( $2 \times 4 = 8$ ). The minimum number of bytes in the header must be 20. The packet has been **corrupted** in transmission.

## Example 20.2

In an IPv4 packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

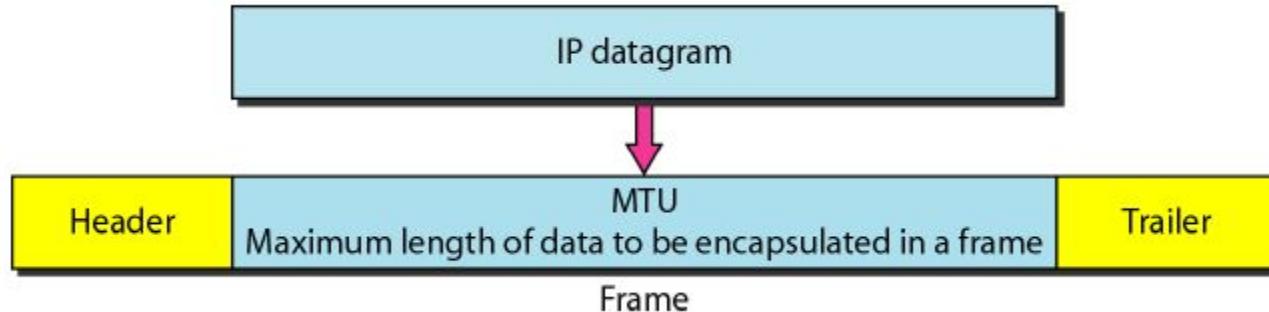
### Solution

The HLEN value is 8, which means the total number of bytes in the header is  $8 \times 4$ , or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

# Fragmentation

- A datagram can travel through different networks.
- Each router decapsulates the IPv4 datagram from the frame it receives, processes it, and then encapsulates it in another frame.
- The format and size of the received frame depend on the protocol used by the physical network through which the frame has just traveled.
- The format and size of the sent frame depend on the protocol used by the physical network through which the frame is going to travel.
- For example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.
- Each data link layer protocol has its own frame format in most protocols. One of the fields defined in the format is the maximum size of the data field.
- In other words, when a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network.

# MTU(Maximum Transferable Unit)



**The value of the MTU depends on the physical network protocol.** Table below shows the values for some protocols.

Protocol	MTU
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

# Fragmentation

- To make the IPv4 protocol independent of the physical network, the designers decided to make the maximum length of the IPv4 datagram equal to **65,535 bytes**.
- This makes transmission more efficient if we use a protocol with an MTU of this size. However, for other physical networks, we must divide the datagram to make it possible to pass through these networks. This is called **fragmentation**.
- When a datagram is **fragmented**, each fragment has its own header with most of the fields repeated, but with some changed.
- A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU. In other words, a datagram can be **fragmented several times before it reaches the final destination**.
- In IPv4, a datagram can be **fragmented by the source host or any router in the path** although there is a tendency to limit fragmentation only at the source.

# Fragmentation

- The **reassembly of the datagram**, however, is done **only by the destination host** because each fragment becomes an independent datagram.
- Whereas the fragmented datagram can travel through different routes, and we can never control or guarantee which route a fragmented datagram may take.
- All the fragments belonging to the same datagram should finally arrive at the destination host. So it is logical to do the **reassembly at the final destination**.
- When a datagram is fragmented, required parts of the header must be copied by all fragments. The option field may or may not be copied.
- The host or router that fragments a datagram must change the values of **three fields: flags, fragmentation offset and total length**. The rest of the fields must be copied.

# Fields Related to Fragmentation

The fields that are related to fragmentation and reassembly of an IPv4 datagram are:

- a. **identification**, b. **flags**, c. **fragmentation offset**.

## a. Identification:

This **16-bit field identifies a datagram originating from the source host.**

- The combination of the identification and source IPv4 address must uniquely define a datagram as it leaves the source host.
- When the IPv4 protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by 1.
- When a datagram is fragmented, the value in the identification field is copied to all fragments. In other words, all fragments have the same identification number, the same as the original datagram.
- The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value must be assembled into one datagram.

# Fields Related to Fragmentation

## b. Flags.

This is a **3-bit** field.



D: Do not fragment  
M: More fragments

- 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 it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host .
- If its **value is 0**, the datagram **can be fragmented** if necessary.
- The **third bit** is called the ***more fragment* bit**.
- If its **value is 1**, it means **the datagram is not the last fragment**; there are more fragments after this one.
- If its **value is 0**, it means this is **the last or only fragments**.

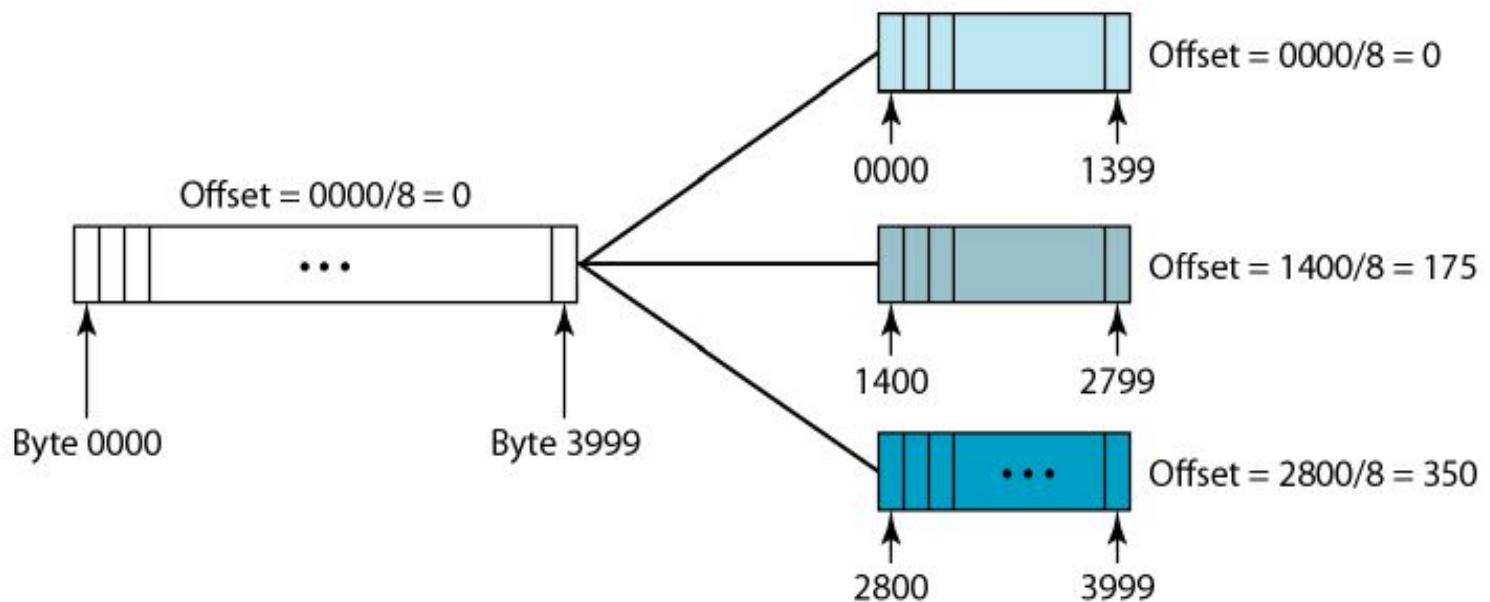
# Fields Related to Fragmentation

## c. Fragmentation offset.

This **13-bit** field 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**.
- Figure on next slide 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**.
- The **first fragment** carries bytes **0 to 1399**, the **offset** for this datagram is  $0/8 = 0$ .
- The **second fragment** carries bytes **1400 to 2799**; the **offset value** for this fragment is  $1400/8 = 175$ .
- Finally, the **third fragment** carries bytes **2800 to 3999**. The offset value for this fragment is  $2800/8 = 350$ .

# Fragmentation Example



# Fragmentation Example

## Example 1.

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

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.

## Example 2.

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

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).

# **Lecture 3.2**

## **Internet Protocol – IPv6**

**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Internet Protocol Version 6(IPv6)

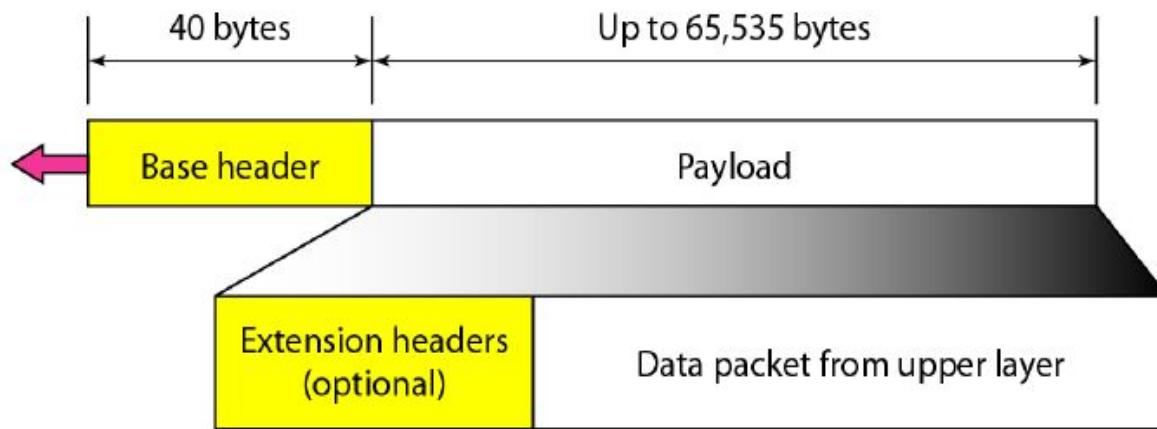
- IPv4 has some **deficiencies** that make it **unsuitable** for the **fast-growing Internet**.
  - Despite all short-term solutions, such as **classless addressing** and **NAT**, **address depletion** is still a long-term problem in the Internet.
  - The Internet must accommodate **real-time audio and video transmission**. This type of transmission requires **minimum delay** strategies and **reservation of resources** not provided in the IPv4 design.
  - The Internet must accommodate **encryption and authentication of data** for some applications. **No encryption or authentication** is provided by **IPv4**.
- To overcome these deficiencies, **IPv6** (Internetworking Protocol, version 6), also known as **IPng** (Internetworking Protocol, next generation), was proposed and is now a standard.

# Advantages of IPv6

- **Larger address space.** An IPv6 address is **128 bits** long. Compared with the 32-bit address of IPv4, this is a huge increase in the address space.
- **Better header format.** IPv6 uses a new header format in which **options are separated from the base header** and **inserted, when needed**, between the base header and the upper-layer data. This simplifies and speeds up the routing process because most of the options do not need to be checked by routers.
- **Allowance for extension.** IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.
- **Support for 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**.
- **Support for more security.** The **encryption and authentication** options in IPv6 provide **confidentiality and integrity** of the packet.

# IPv6 Packet Format

- Each IPv6 packet is composed of a **mandatory base header** followed by the **payload**.
- The payload consists of **two parts**:
  - **optional extension headers** and **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.

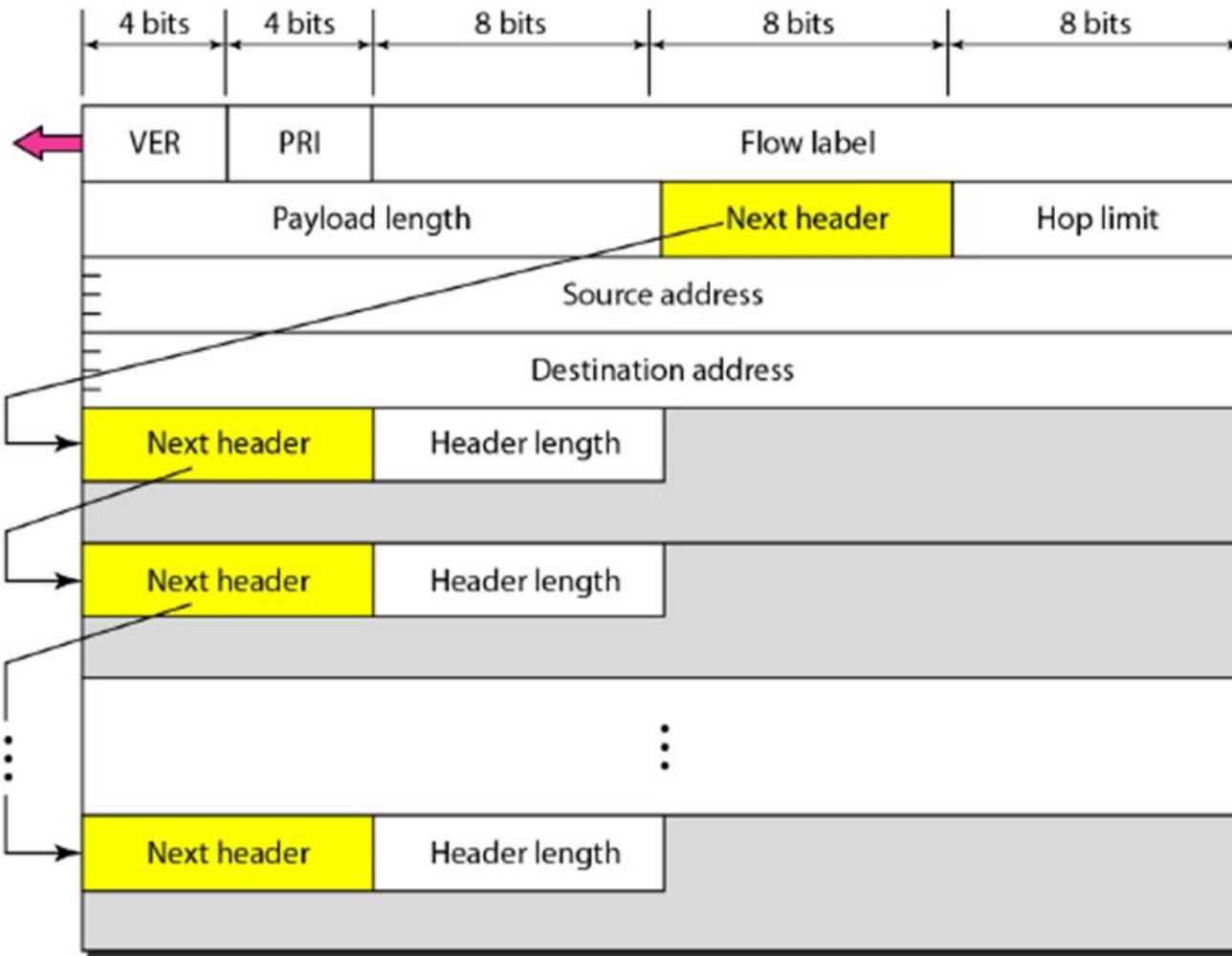


# IPv6 Base Header

Base header has **eight fields**. These fields are as follows:

- i. **Version.** This **4-bit** field defines the version number of the IP. For IPv6, the value is **6**.
- ii. **Priority.** The **4-bit** priority field defines the **priority of the packet** with respect to traffic congestion.
- iii. **Flow label.** The flow label is a **3-byte (24-bit)** field that is designed to provide special handling for a particular flow of data.
- iv. **Payload length.** The **2-byte** payload length field defines the **length of the IP datagram** excluding the base header.
- v. **Next header.** The next header is an **8-bit field** defining the header that **follows the base header** in the datagram. The next header is either one of the optional **extension headers used by IP** or the **header of an encapsulated packet such as UDP or TCP**. Each extension header also contains this field.

# IPv6 Base Header

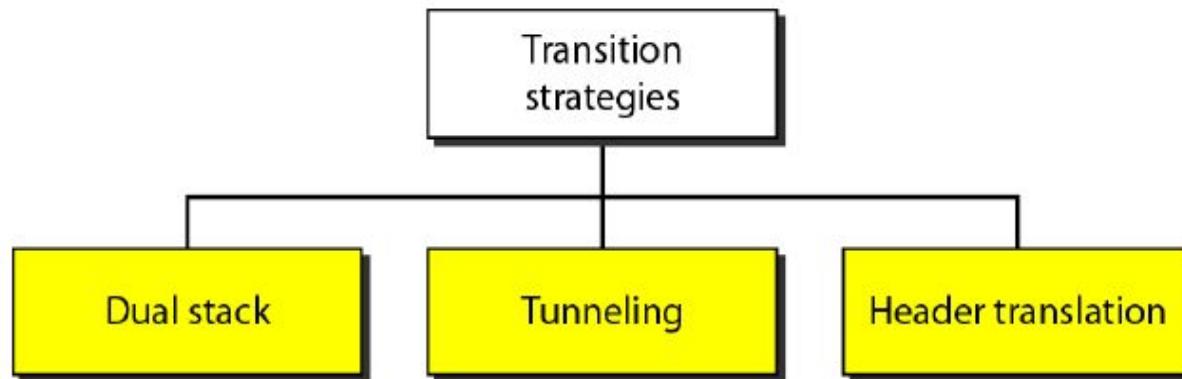


# IPv6 Base Header

- **Hop limit.** This **8-bit** hop limit field serves the same purpose as the **TTL** field in IPv4.
- **Source address.** The source address field is a **16-byte** (128-bit) Internet address that identifies the original source of the datagram.
- **Destination address.** The destination address field is a **16-byte** (128-bit) Internet address that usually identifies the final destination of the datagram.

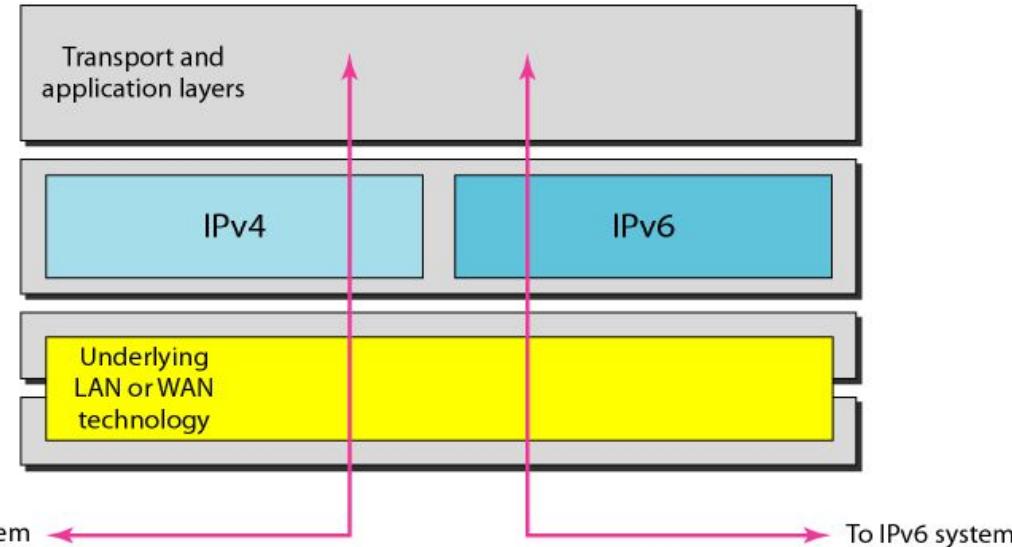
# TRANSITION FROM IPv4 TO IPv6

- Because of the **huge number of systems** on the **Internet**, the **transition from IPv4 to IPv6 cannot happen suddenly.**
- It takes a **considerable amount of time** before every system in the Internet can **move from IPv4 to IPv6**.
- The transition must be smooth to prevent any problems between IPv4 and IPv6 systems.
- **Three strategies** have been devised by the **IETF** to help the transition.



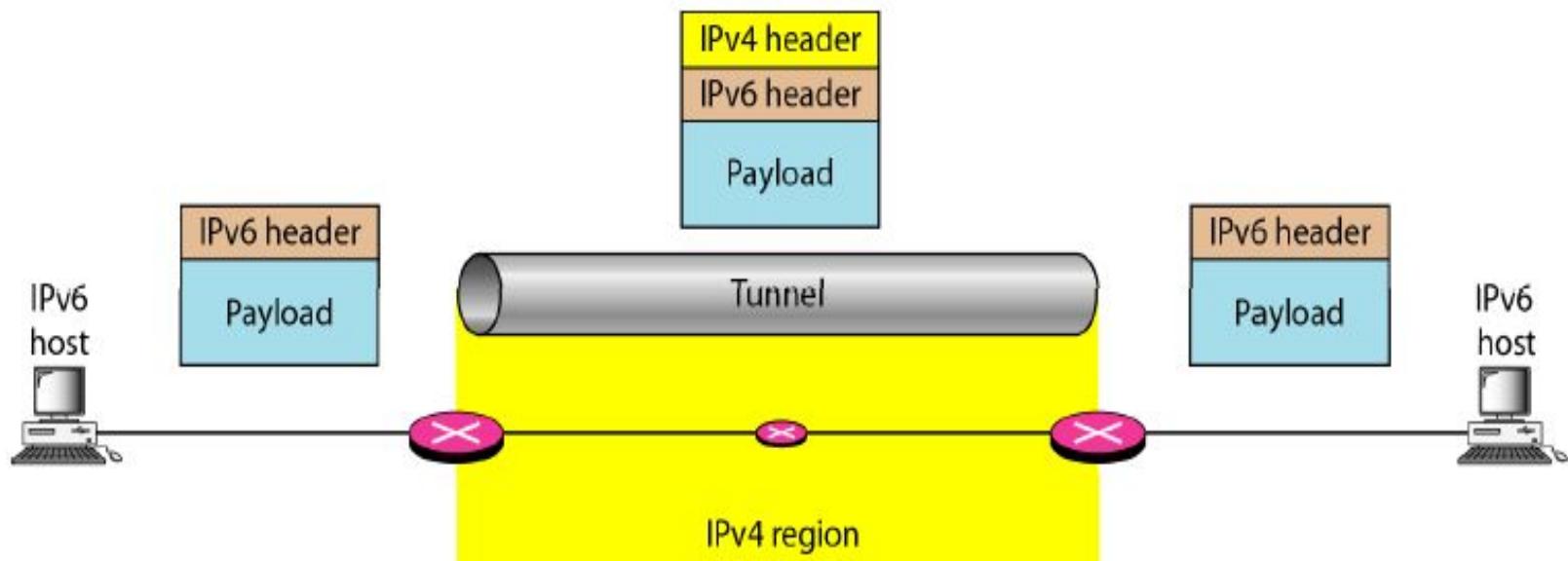
# Dual Stack

- It is recommended that all hosts, before migrating completely to version 6, have a **dual stack** of protocols.
- In other words, a station must run **IPv4 and IPv6 simultaneously until all the Internet uses IPv6**.
- To determine which version to use when sending a packet to a destination, the source host **queries the DNS**. If the DNS returns an IPv4 address, the source host sends an IPv4 packet. If the DNS returns an IPv6 address, the source host sends an IPv6 packet.



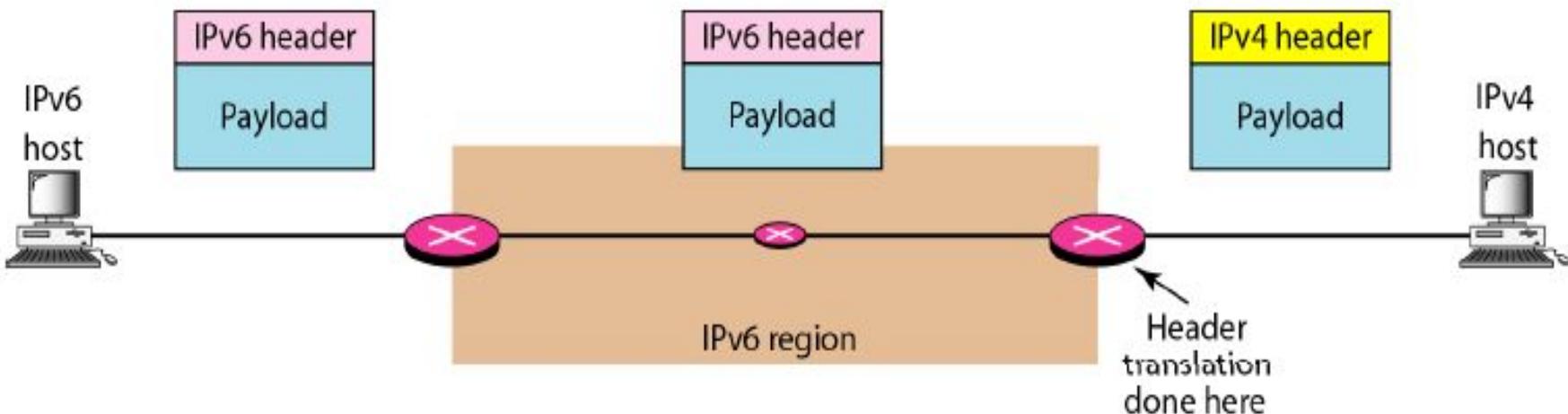
# Tunneling

- **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 uses IPv4**.
- To pass through this region, the **packet must have an IPv4 address**.
- 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.



# Header Translation

- Header translation is necessary when the **majority of the Internet has moved to IPv6** but **some systems still use IPv4**.
- The **sender wants to use IPv6, but the receiver does not understand IPv6**.
- **Tunneling does not work** in this situation because the packet must be in the IPv4 format to be understood by the receiver.
- In this case, the **header format must be totally changed** through **header translation**. The header of the IPv6 packet is converted to an IPv4 header.



# **Lecture 4.1**

## **Network Layer: Address Mapping Protocols**

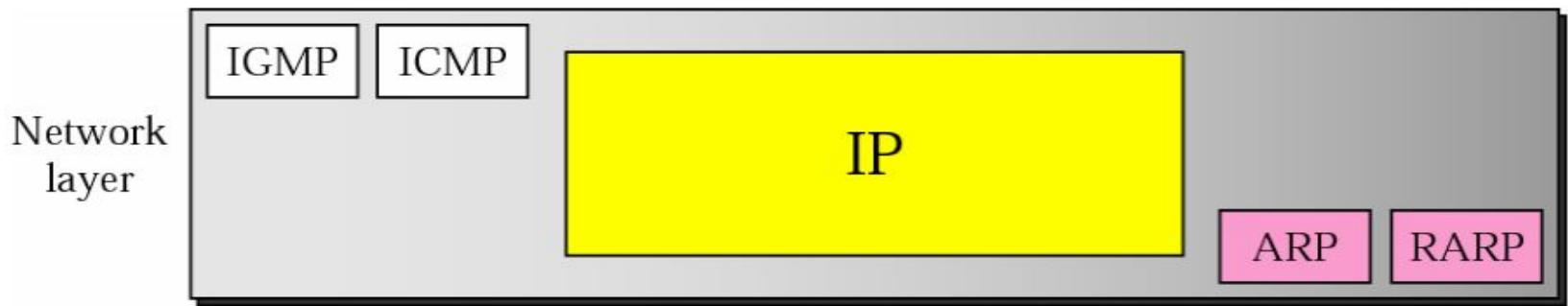
**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# INTRODUCTION

- IP (Internet Protocol) was designed as a **best-effort delivery protocol**, but it **lacks some features** such as **flow control** and **error control**. To make IP more **responsive** it takes help of other protocols.
- Protocols to create a **mapping a logical address to a physical address**: ARP (Address Resolution Protocol).
- Protocols to create a **reverse mapping** i.e. mapping a **physical address to a logical address**: RARP, BOOTP, and DHCP.
- Lack of flow and error control in the Internet Protocol has resulted in another protocol, **ICMP**. It reports congestion and some types of errors in the network or destination host.

# Address Mapping & Error Reporting Protocols



# ADDRESS MAPPING

- An 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, they are 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

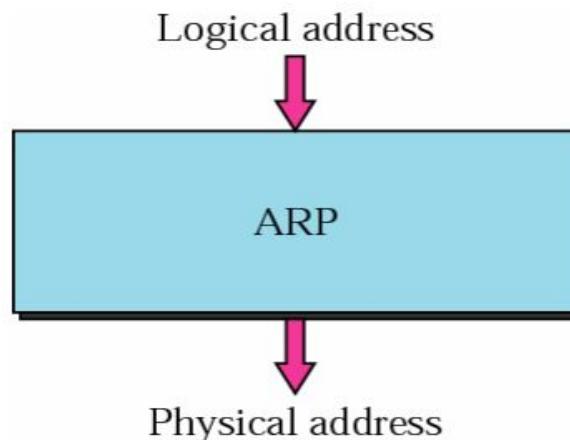
- Static mapping involves in the creation of a **table** that **associates a logical address with a physical address**.
- This table is stored in each machine on the network.
- Static mapping has some **limitations** because physical addresses may change in the following ways:
  - A machine could change its **NIC** (Network Interface Card), resulting in a **new physical address**.
  - In some LANs, such as **LocalTalk**, the **physical address changes** every time the computer is turned on.
  - A **mobile computer** can move from one physical network to another, resulting in a change in its physical address.

# Dynamic mapping

In such mapping each time a machine knows one of the two addresses (logical or physical), it can 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**.

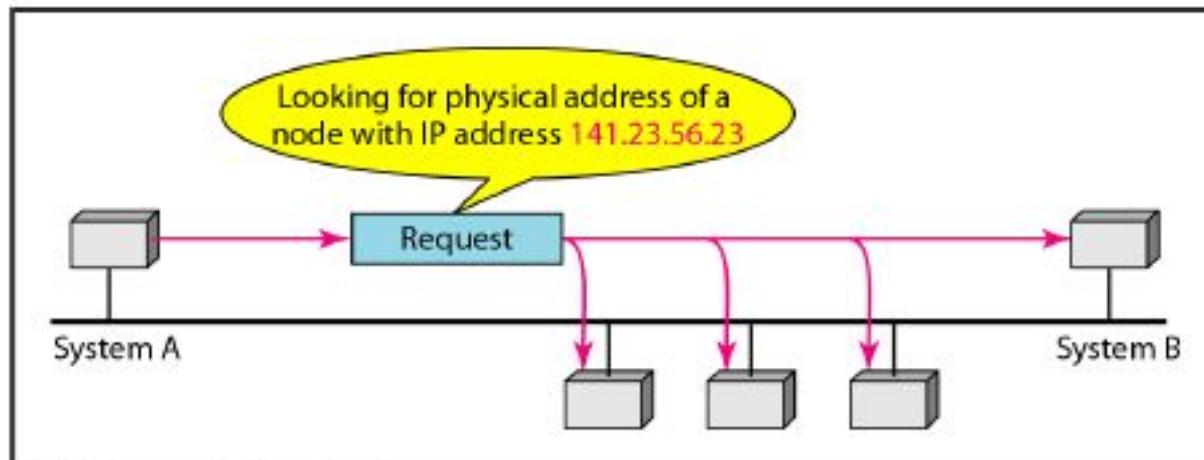


# Mapping Logical to Physical Address: ARP

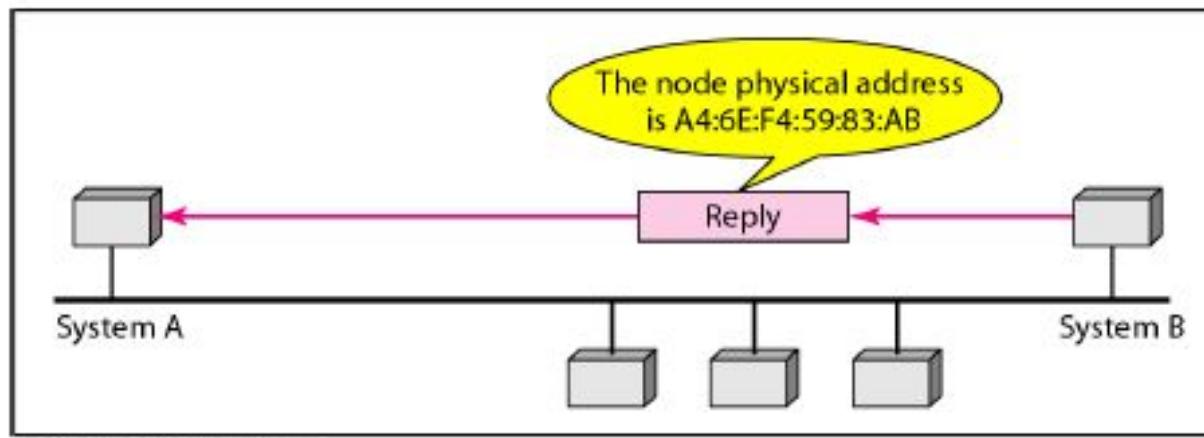
Following **steps** are involved in logical to physical address mapping:

- 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**.
- 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**.
- 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.

# Mapping Logical to Physical Address: ARP



a. ARP request is broadcast

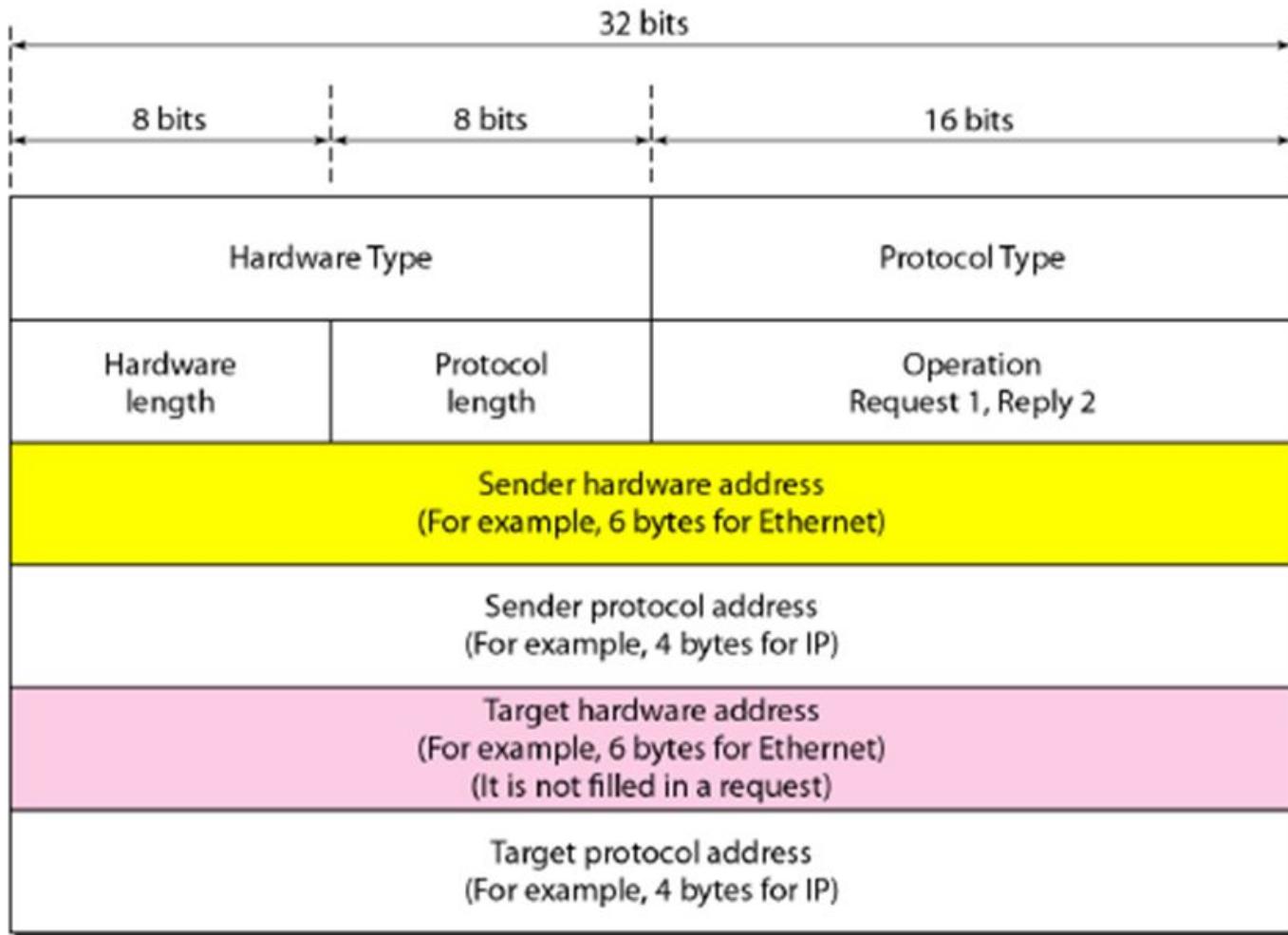


b. ARP reply is unicast

# ARP Cache

- 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



# ARP Packet Format

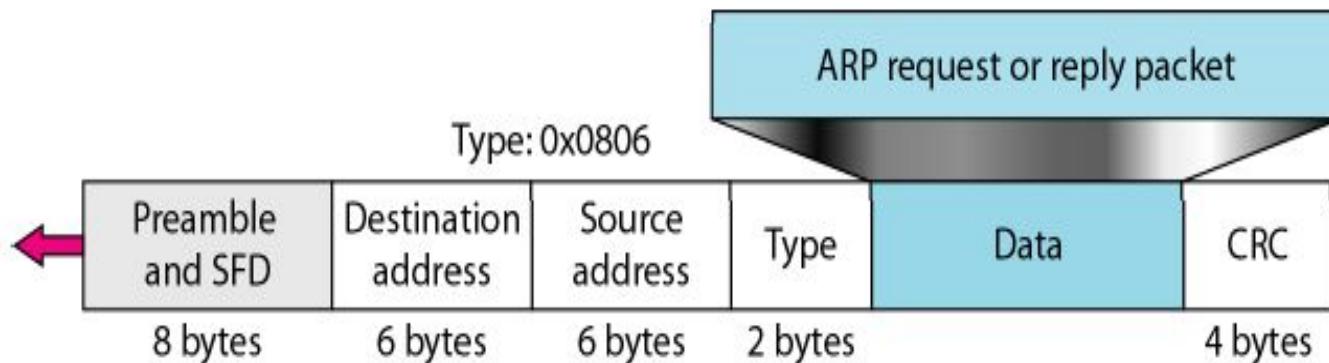
- a. **Hardware type.** This is a 16-bit field **defining the type of the network** on which ARP is running. Each LAN has been assigned an **integer** based on its type. For example, **Ethernet is given type 1**.
- b. **Protocol type.** This is a 16-bit field defining the **protocol**. For example, the 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)**.

# ARP Packet Format

- 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 **all 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 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.



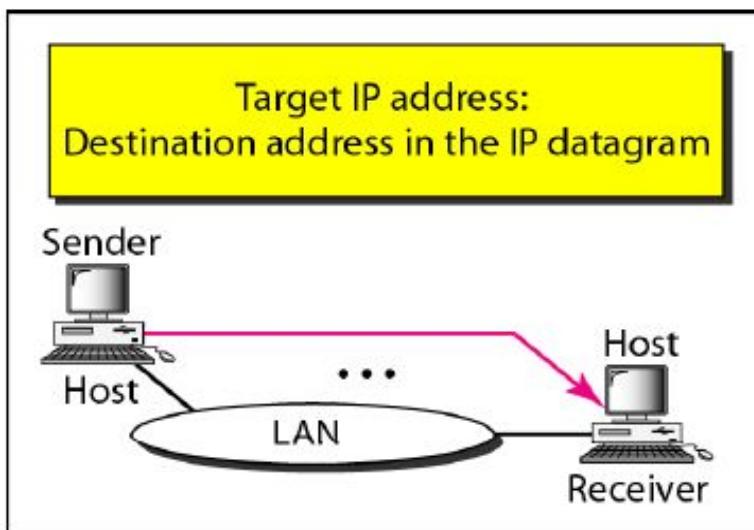
# ARP Operation

1. The sender knows the IP address of the target.
2. IP asks ARP to create an **ARP request message**, filling in the sender physical address, the sender 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 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 target machine.
7. The IP datagram, which carries data for the target machine, is now encapsulated in a frame and is unicast to the destination.

# Different Cases of ARP Operation

**Case 1:** *The sender is a host and wants to send a packet to another host on the same network.*

In this case, the logical address that must be mapped to a physical address is the **destination IP address** in the datagram header.

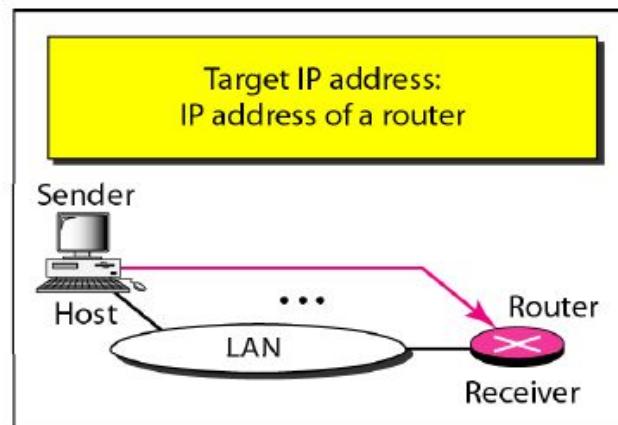


Case 1. A host has a packet to send to another host on the same network.

# Different Cases of ARP Operation

**Case 2:** *The sender is a host and wants to send a packet to another host on another network.*

In this case, the host looks at its **routing table** and finds the IP address of the **next hop (router) for this destination**. If it does not have a routing table, it looks for the IP address of the **default router**. The **IP address of the router becomes the logical address that must be mapped to a physical address**.

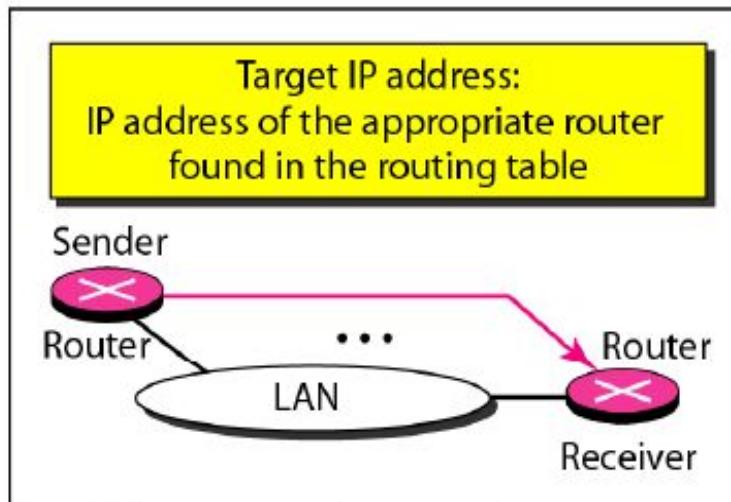


Case 2. A host wants to send a packet to another host on another network.  
It must first be delivered to a router.

# Different Cases of ARP Operation

**Case 3:** *The sender is a router that has received a datagram destined for a host on another network.*

It checks its routing table and finds the IP address of the next router. **The IP address of the next router becomes the logical address that must be mapped to a physical address.**

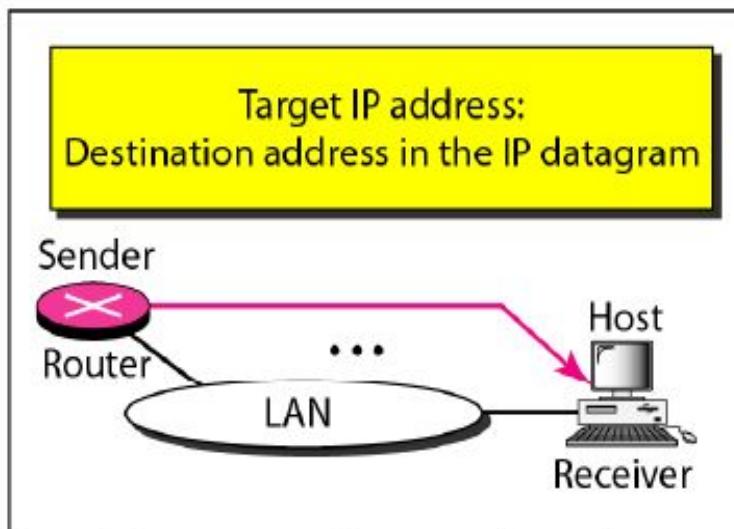


Case 3. A router receives a packet to be sent to a host on another network. It must first be delivered to the appropriate router.

# Different Cases of ARP Operation

**Case 4:** *The sender is a router that has received a datagram destined for a host on the same network.*

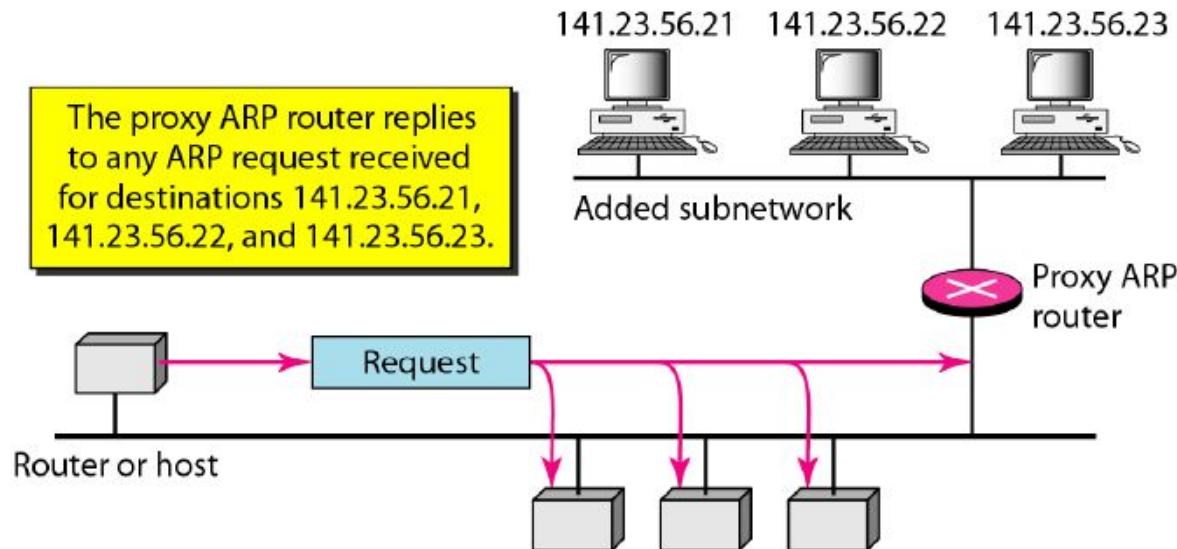
The destination IP address of the datagram becomes the logical address that must be mapped to a physical address.



Case 4. A router receives a packet to be sent to a host on the same network.

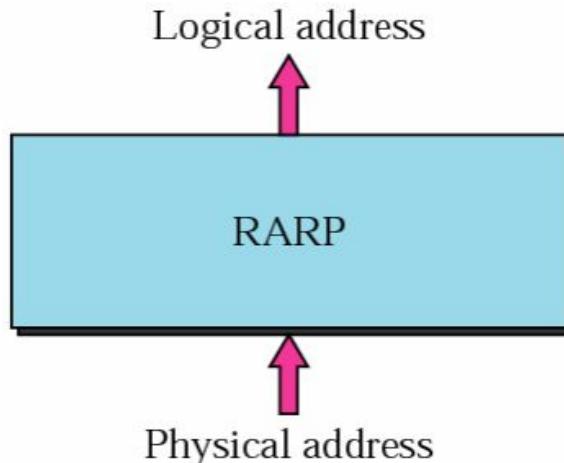
# Proxy ARP

- 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.



# Mapping Physical Address to Logical Address

- There are occasions in which a **host knows its physical address, but needs to know its logical address**. 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.



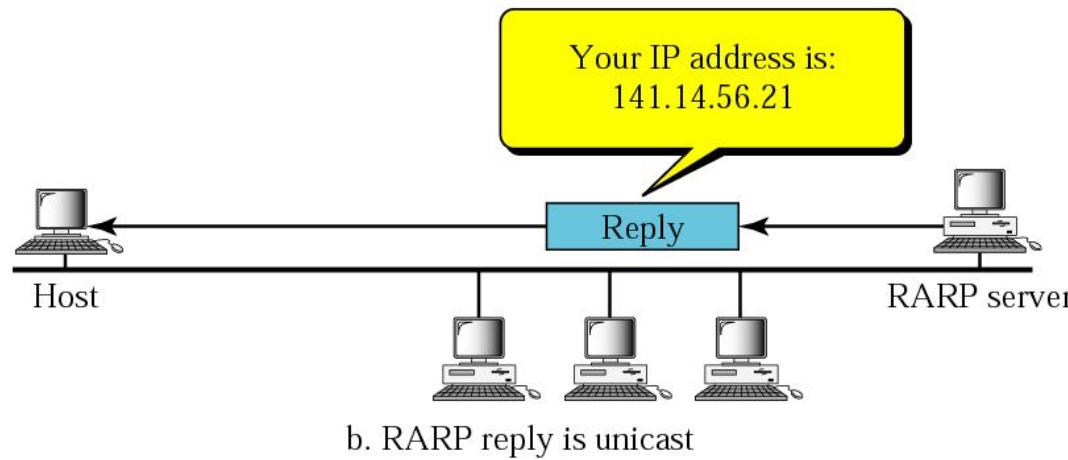
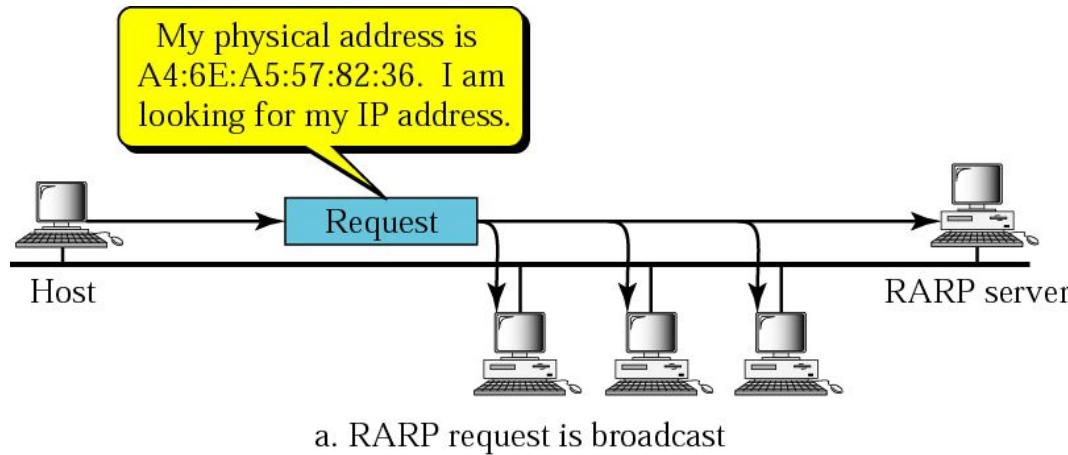
# Reverse Address Resolution Protocol (RARP)

- Reverse Address Resolution Protocol (RARP) finds the logical address for a machine that knows only its physical address.
- To create an IP datagram, a host or a router needs to know its own IP address .
- The IP address of a machine is usually read from its configuration file stored on a disk file.
- However, a diskless machine is usually booted from ROM, which has minimum booting information.
- The ROM is installed by the manufacturer. It cannot include the IP address because the IP addresses on a network are assigned by the network administrator.
- The machine can get its physical address (by reading its NIC, for example), which is unique locally. It can then use the physical address to get the logical address by using the RARP protocol.

# RARP Operation

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

# RARP Operation



# RARP Packet Format & Encapsulation

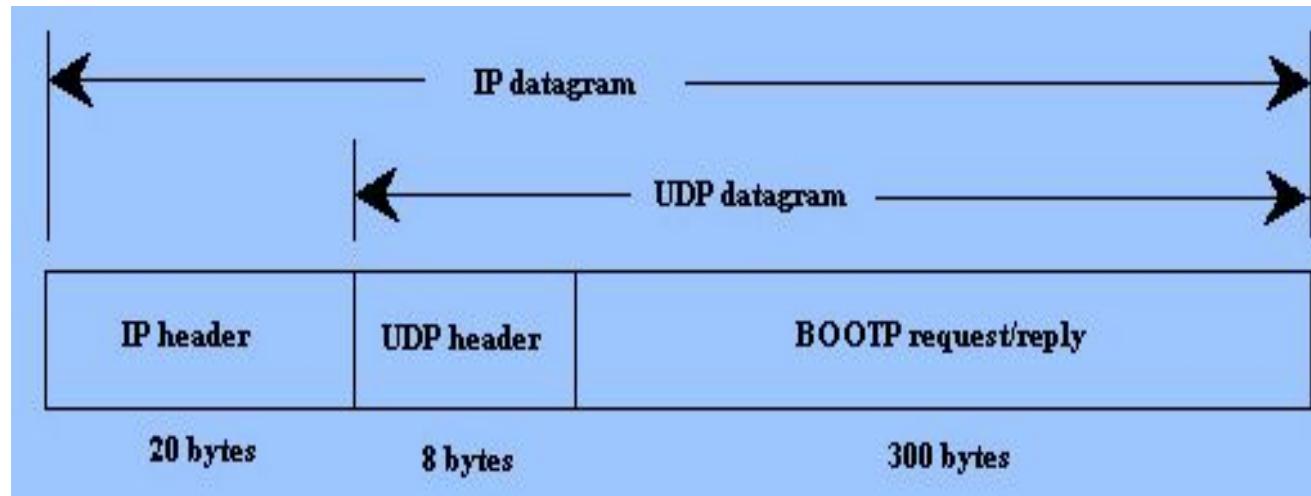
- The format of the **RARP packet** is the **same as the ARP packet** format, except that the Operation field.
- **Its value is 3 for RARP request message and 4 for RARP reply message.**
- An **RARP packet** is also **encapsulated** directly into a **data link frame** just like ARP packet.

# Limitations of RARP

- As **broadcasting is done at the data link layer**. The physical broadcast address, all 1's in the 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 to **assign 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**.

# Bootstrap Protocol (BOOTP)

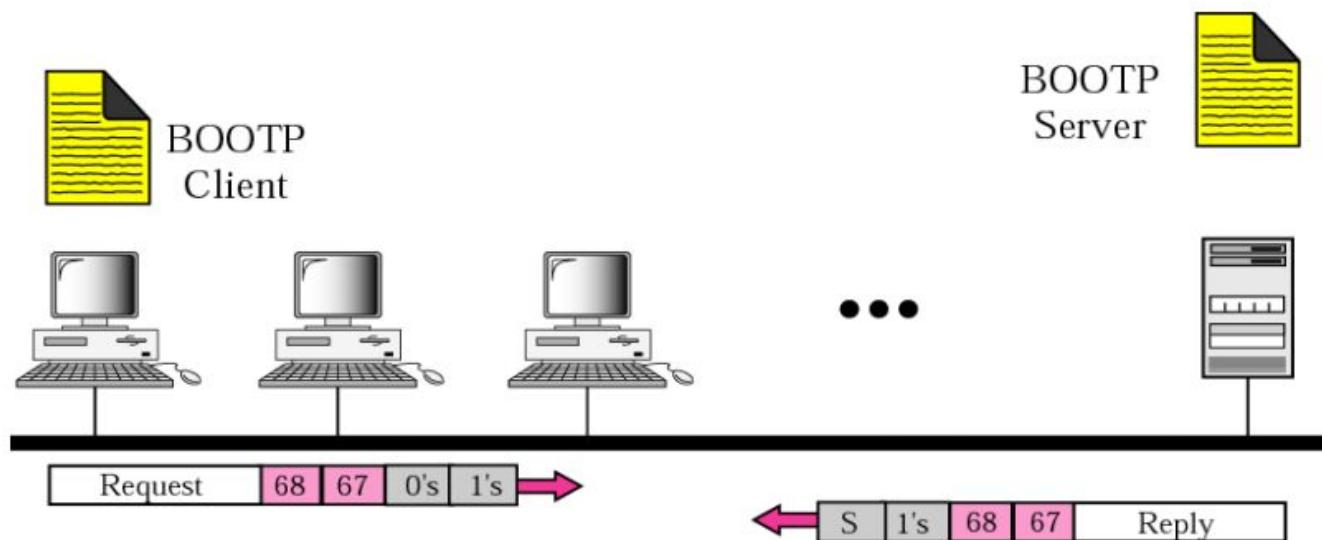
- The **Bootstrap Protocol (BOOTP)** is a **Client/Server based protocol** at **Application layer**, designed to **provide physical address to logical address mapping**.
- The administrator may put the client and the server on the same network or on different networks.
- BOOTP** messages are **encapsulated** in a **UDP packet**, and the UDP packet itself is encapsulated in an **IP packet**.



# ***Case 1: Client and server on same network***

- 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:
  1. The **client’s IP address**, **subnet mask**, and **default gateway address**.
  2. The **IP address** and **host name** of the **BOOTP server**.
- 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.

# BOOTP client and server on the same network



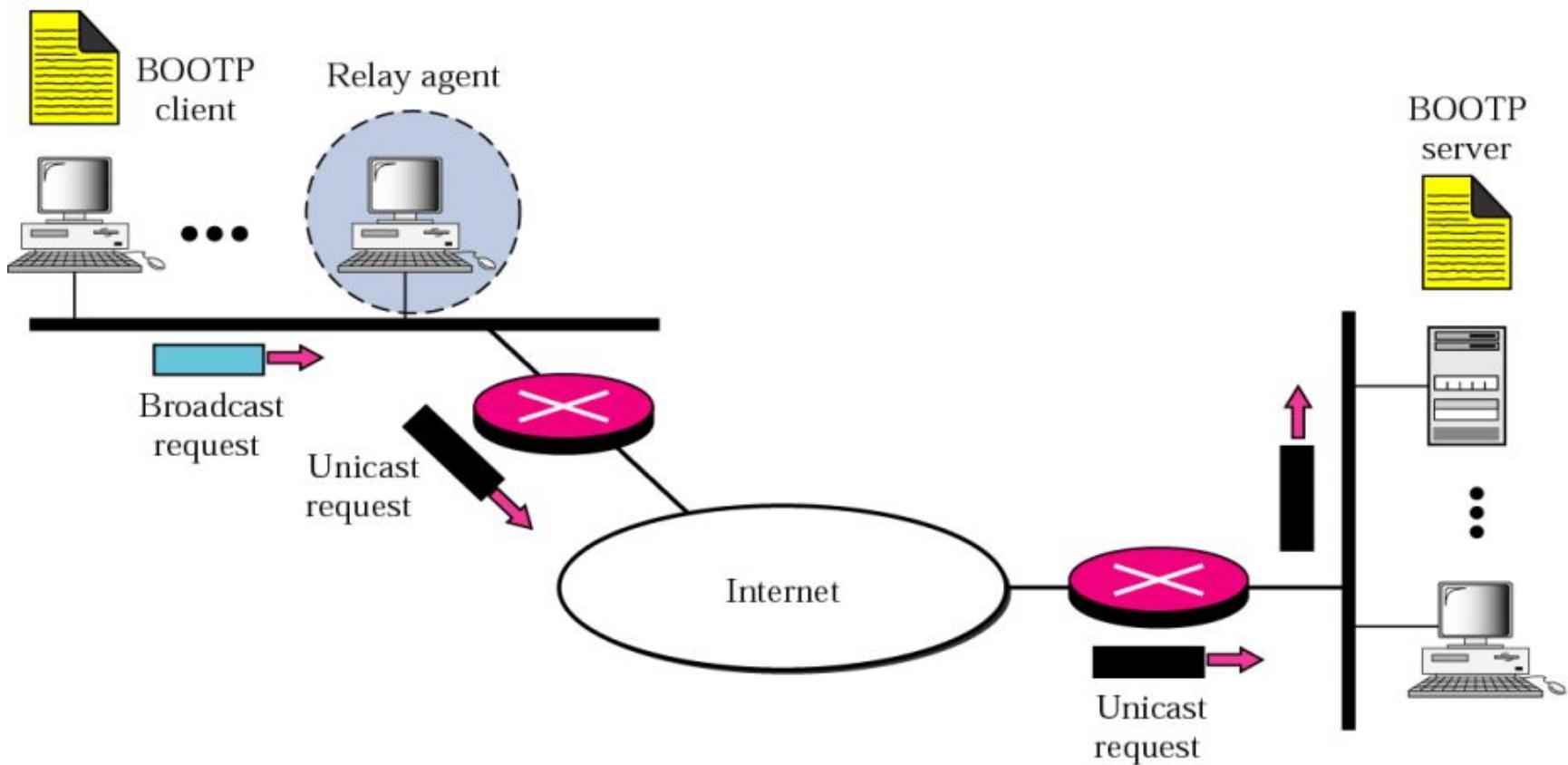
# Case 2 : Client and server on different networks

- If the **server exists** on some **distant network** the **BOOTP request** is **broadcasted** because the **client does not know the IP address of the server**.
- The client simply uses **all 0's as the source address** and **all 1's as the destination address**.
- But a **broadcast IP datagram** cannot pass through any **router**. To solve the problem, there is a need for an **intermediary**.
- **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**.
- 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**.

# Case 2 : Client and server on different networks

- The packet, carrying a **unicast destination address**, is routed by any router and reaches the **BOOTP server**.
- 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.
- **BOOTP server sends a BOOTP reply message to the relay agent.**
- The **Relay Agent**, after receiving the reply, **sends it to the BOOTP client.**

# BOOTP client and server on different network



# Limitations of BOOTP

- **BOOTP is not a dynamic configuration protocol.**
- When a client requests its IP address, the BOOTP server consults a table that matches the physical address of the client with its IP address.
- This implies that the binding between the physical address and the IP address of the client **already exists**. The **binding is predetermined** i.e. **static**.
- However, what if a host moves from one physical network to another? What if a host wants a temporary IP address?
- **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.
- **BOOTP is a static configuration protocol.**

# Dynamic Host Configuration Protocol (DHCP)

The **Dynamic Host Configuration Protocol** (DHCP) has been devised to provide **static** and **dynamic address allocation** that can be manual or automatic as required.

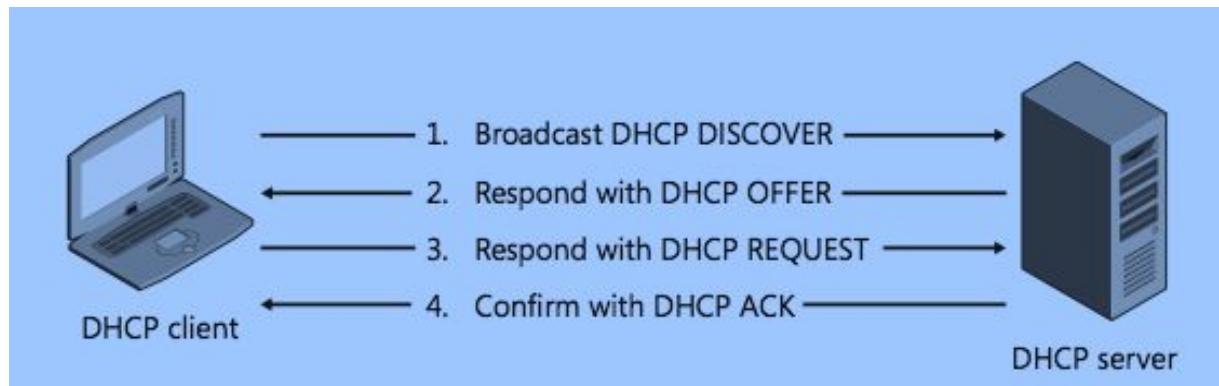
- **Static Address Allocation**
- In this capacity **DHCP acts as BOOTP** does. It is **backward compatible with BOOTP**, which means a host running the BOOTP client can request a static address from a DHCP server. A **DHCP server** has a **database** that **statically binds physical addresses to IP addresses**.
- **Dynamic Address Allocation**
- DHCP has a **second database with a pool of available IP addresses**. 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.

# Dynamic Host Configuration Protocol (DHCP)

- 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** or is connected and disconnected from a network (as is a subscriber to a service 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 client must 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 below.



# DHCP Operation

DHCP client goes through the **four 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.

# **Lecture 4.2**

## **Network Layer: Error Reporting Protocol**

### **ICMP**

**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Error Reporting

- IP provides **unreliable and connectionless datagram delivery**.
- The IP protocol is a **best-effort delivery service** that delivers a 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 the final 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 datagram because it has not received all fragments within a predetermined time limit?

# Error Reporting

- These are 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 deficiencies.
- It is a **companion to the IP protocol**.

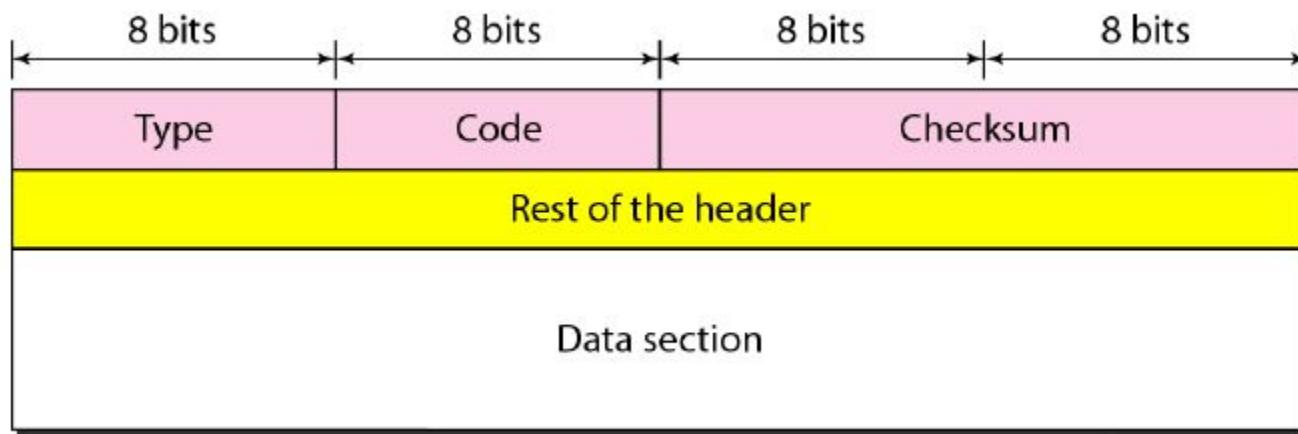
# Types of Messages in ICMP

- ICMP messages are divided into **two broad categories**:
  1. Error-reporting messages
  2. 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**, which **occur in pairs**, help a **host or a network manager** get specific information from a router or another host.

# ICMP Message Format

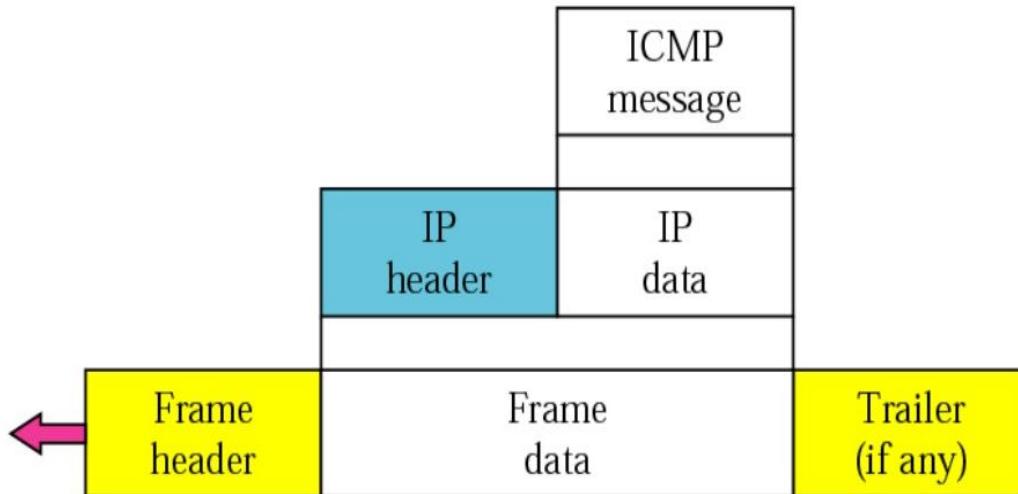
- An ICMP message has an **8-byte header** and a **variable-size data section**. Although the general format of the header is different for each message type, the **first 4 bytes are common to all**.
- The first field, **ICMP type**, defines the type of the message.
- The **code field** specifies the 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 packet that had the error**.
- In **ICMP query messages**, the **data section carries extra information based on the type of the query**.

# ICMP Message Format



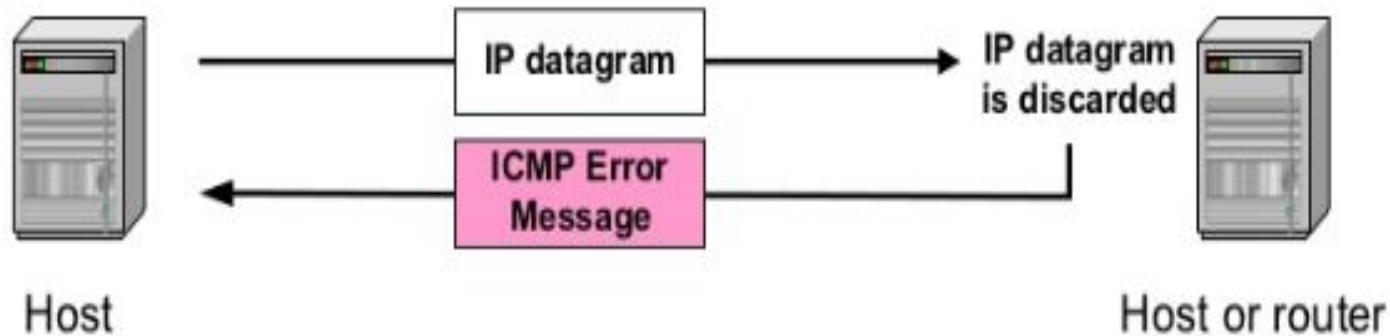
# ICMP Encapsulation

- ICMP itself is a **network layer protocol**.
- However its messages are **not passed directly to data link layer**. Instead the messages are **first encapsulated inside IP datagrams** before going to the lower layer.



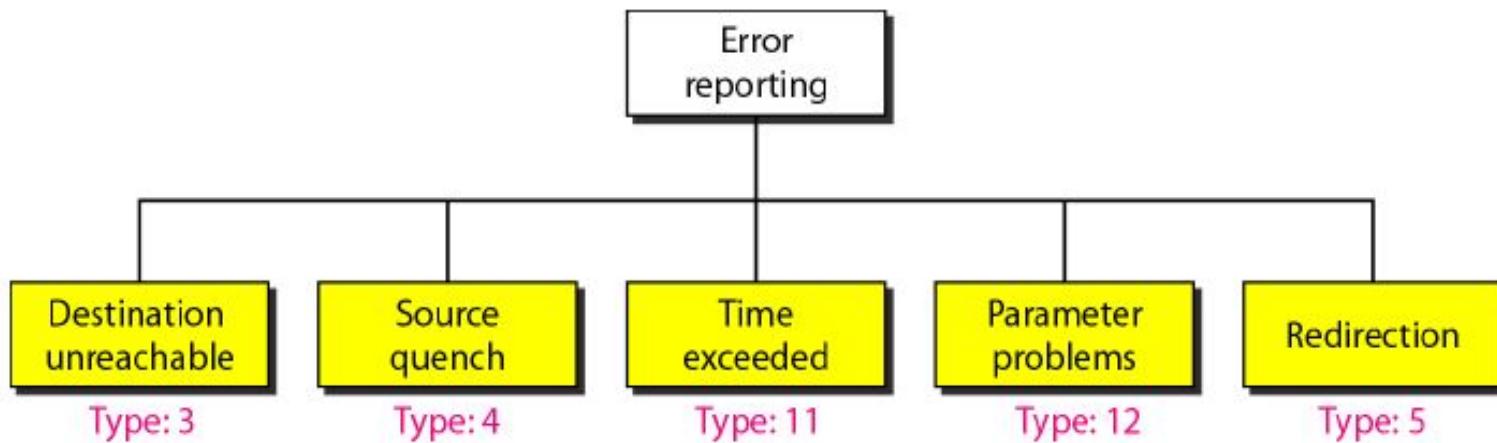
# Error Reporting Messages

- One of the main responsibilities of ICMP is to **report errors**. Although technology has produced increasingly reliable transmission media, errors still exist and must be handled.
- IP is an **unreliable protocol**. This means that error checking and error control are not a concern of IP. ICMP was designed, in part, to compensate for this **shortcoming**.
- However, **ICMP does not correct errors-it simply reports them**. Error correction is left to the higher-level protocols.



# Error Reporting Messages

- Error messages are **typically sent** when a **datagram is discarded due to some error**.
- 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, time exceeded, parameter problems, and redirection*.



# Destination Unreachable

- When a router cannot route a datagram or a host cannot deliver a datagram, the datagram is **discarded**.
- Some of the reasons for a datagram not to be delivered are: Net/Host Unreachable, Fragmentation Needed and DF flag was set, Communication with destination network is administratively prohibited etc.
- And the router or the host sends a **destination-unreachable** message back to the **source host** that initiated the datagram.
- Note that destination-unreachable messages can be created by either a router or the destination host.
- The Type field is set to one, which is the value for the Destination Unreachable message.
- The Code field supplies more information about the reason why the datagram was not delivered.

# Source Quench

- The IP protocol is a connectionless protocol. IP does not have a **flow control** mechanism embedded in the protocol.
- The lack of flow control can create a major problem in the operation of IP.
- The source host never knows if the routers or the destination host has been overwhelmed with datagrams.
- The source host never knows if it is producing datagrams faster than can be forwarded by routers or processed by the destination host.
- The lack of flow control can create congestion in routers or the destination host.
- In this case, the router or the host has no choice but to discard some of the datagrams.
- The **source-quench** message in ICMP was designed to add a kind of **flow control** to the IP.

# Source Quench

- 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**.

# 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 loop or 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 up a message arrive at the destination host **within a certain time limit**.

# 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**.
- ICMP Parameter Problem message also has an option for a special **pointer** to inform the sender **where** in the original IPv4 header the error had occurred.

# Redirection

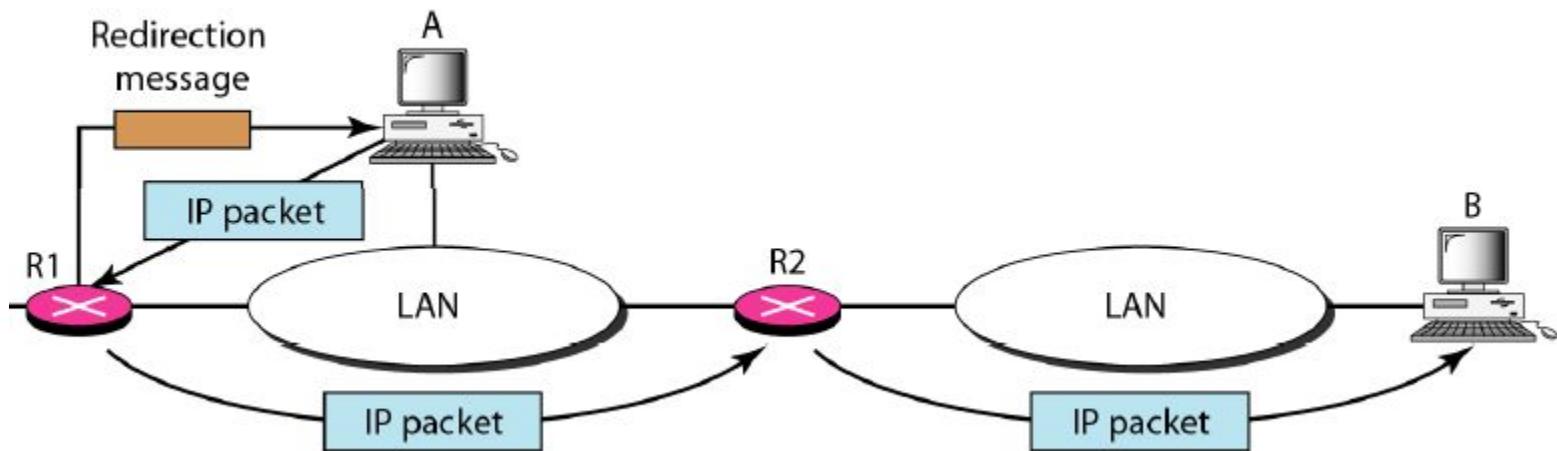
- When a **router** needs to send a packet destined for another network, it must know the **IP address** of the **next appropriate router**.
- The same is true if the **sender is a host**.
- Both **routers and hosts**, then, must have a **routing table** to find the address of the router or the next router.
- **Routers take part in the routing update process**, and are supposed to be updated constantly.
- Routing is **dynamic**. However, for efficiency, **hosts do not take part in the routing update process** because there are many more hosts in an internet than routers.
- **Updating the routing tables** of hosts dynamically produces **unacceptable traffic**.
- **The hosts** usually use **static routing**. When a host comes up, its routing table has a limited number of entries. It usually knows the IP address of only one router, the default router.

# Redirection

- For this reason, the **host may send** a datagram, which is destined for another network, to the **wrong router**.
- In this case, the **router** that receives the datagram will **forward the datagram to the correct router**.
- However, to **update the routing table of the host**, it sends a **redirection message** to the **host**.
- This concept of redirection is shown in Figure on next slide.

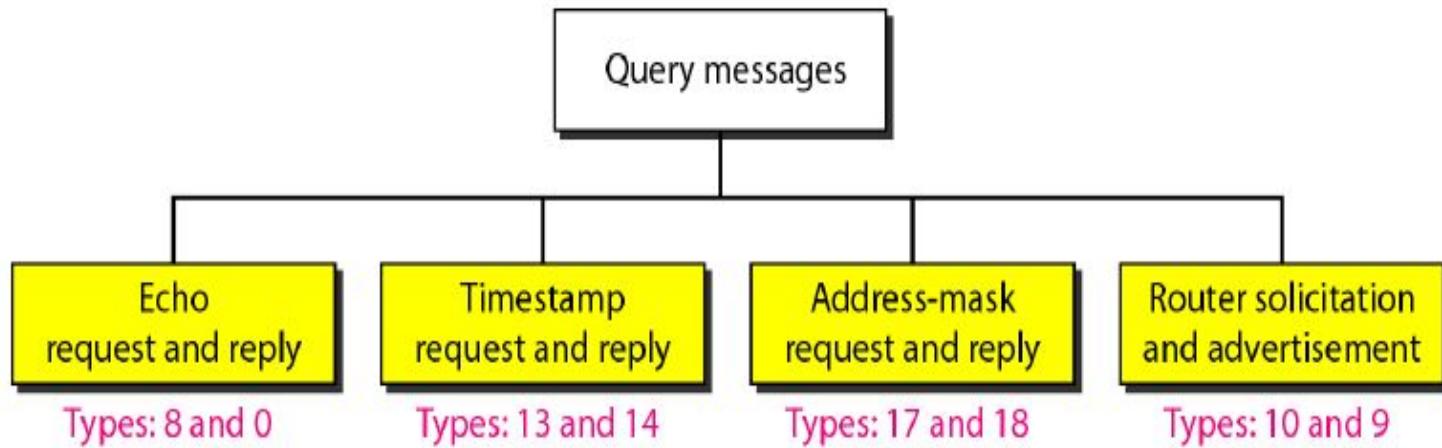
# Redirection

- Host A wants to send a datagram to host B.
- Router R2 is obviously the most efficient routing choice, but host A did not choose router R2. The datagram goes to R1 instead.
- Router R1, after consulting its table, finds that the packet should have gone to R2.
- It sends the packet to R2 and, at the same time, sends a redirection message to host A.
- Host A's routing table can now be updated.



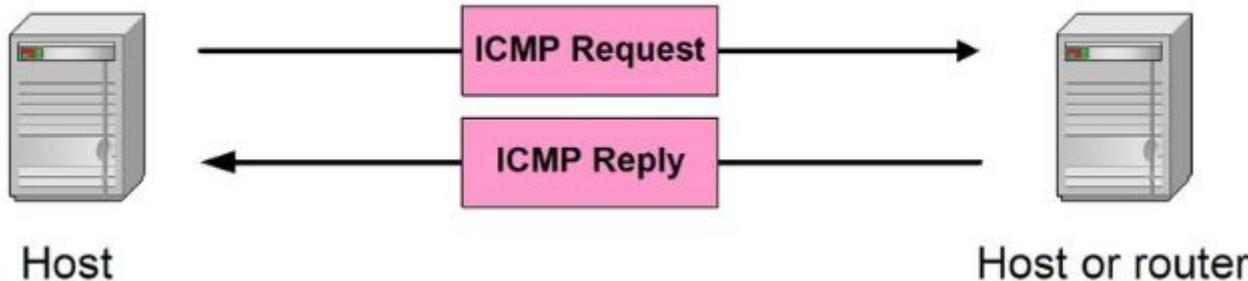
# ICMP Query Messages

- 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**.



- In this type of **ICMP message**, a node sends a **ICMP request** message that is answered in a specific format as **ICMP reply** by the destination node.

# ICMP Query Messages

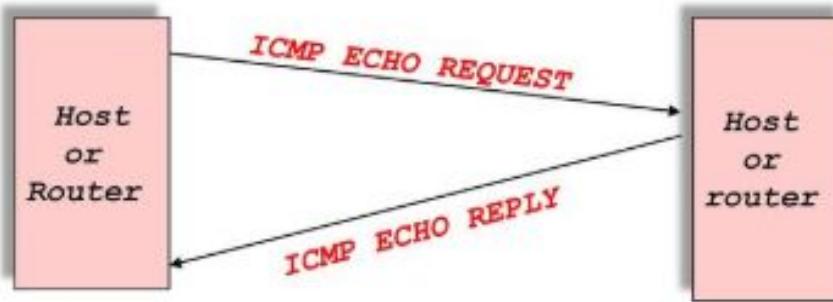


- A **query message** is **encapsulated in an IP packet**, which in turn is **encapsulated in a data link layer frame**.
- However, in this case, no bytes of the original IP are included in the message, as shown in Figure below.



# 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.
- It also confirms that the **intermediate routers** are **receiving, processing, and forwarding IP datagrams**.



- 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.

# 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**.

# 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 IP address to get its subnet address.

# 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**.
- Note that when a router sends out an advertisement, it announces not only its own presence but also the presence of all routers on the network of which it is aware.

# **Lecture 5.1**

## **Network Layer: Delivery, Forwarding, and Routing**

**Dr. Vandana Kushwaha**

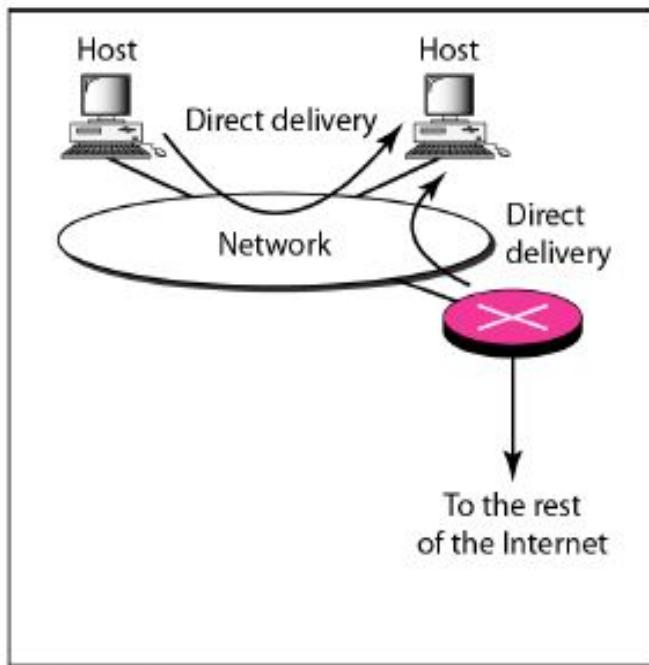
Department of Computer Science  
Institute of Science, BHU, Varanasi

# INTRODUCTION

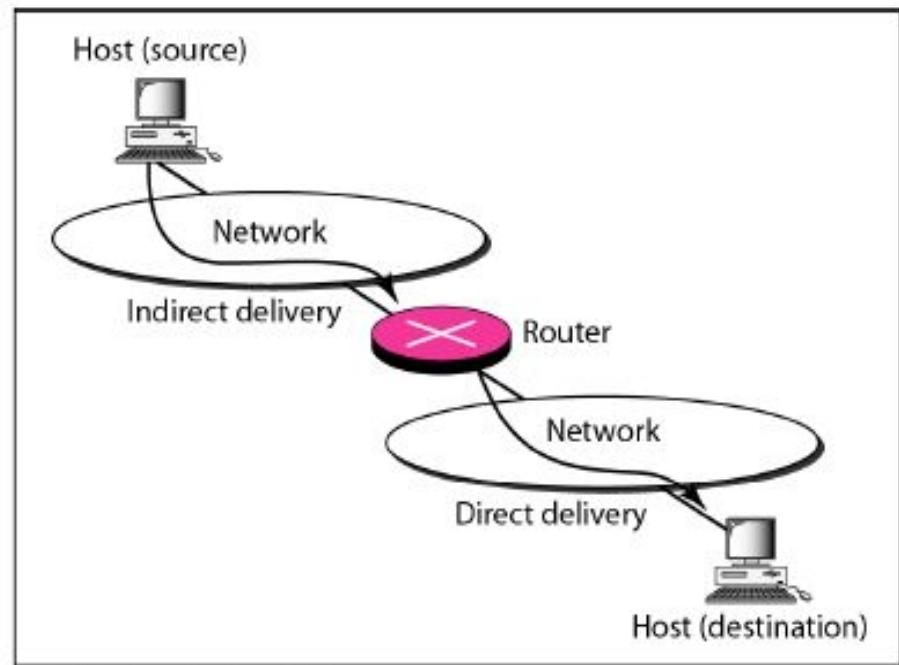
- There are **three interrelated terms** used at **Network layer** while data communication takes place:
  1. **Delivery:** refers to the way a packet is handled by the underlying networks under the control of the **network layer**.
  2. **Forwarding:** refers to the way a packet is ***delivered*** to the next station.
  3. **Routing:** refers to the way **routing tables** are created to help in ***forwarding***.

# DELIVERY

The delivery of a packet to its final destination is accomplished by using **two different methods** of delivery, **direct and indirect**, as shown in Figure below.



a. Direct delivery



b. Indirect and direct delivery

# Direct Delivery

- In a **direct delivery**, the **final destination** of the packet is a host connected to the **same physical network** as the deliverer.
- **Direct delivery occurs when the source and destination of the packet are located on the same physical network or when the delivery is between the last router and the destination host.**
- The **sender** can easily determine if the delivery is **direct**.
- It can extract the **network address of the destination** (using the mask) and compare this address with the addresses of the networks to which it is connected.
- If a match is found, the delivery is direct.

# Indirect Delivery

- If the **destination host** is **not** on the **same network** as the deliverer, the packet is delivered **indirectly**.
- In an indirect delivery, the packet goes from **router to router** until it reaches the one connected to the same physical network as its final destination.
- A delivery always involves **one direct delivery** but **zero or more indirect deliveries**.
- **The last delivery** is always a **direct delivery**.

# FORWARDING

- Forwarding means to ***place the packet in its route to its destination.***
- Forwarding requires a host or a router to have a **routing 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 route to the final destination.
- However, this simple solution is difficult today in an internetwork such as the Internet because the **number of entries** needed in the **routing table** would make **table lookups** inefficient.

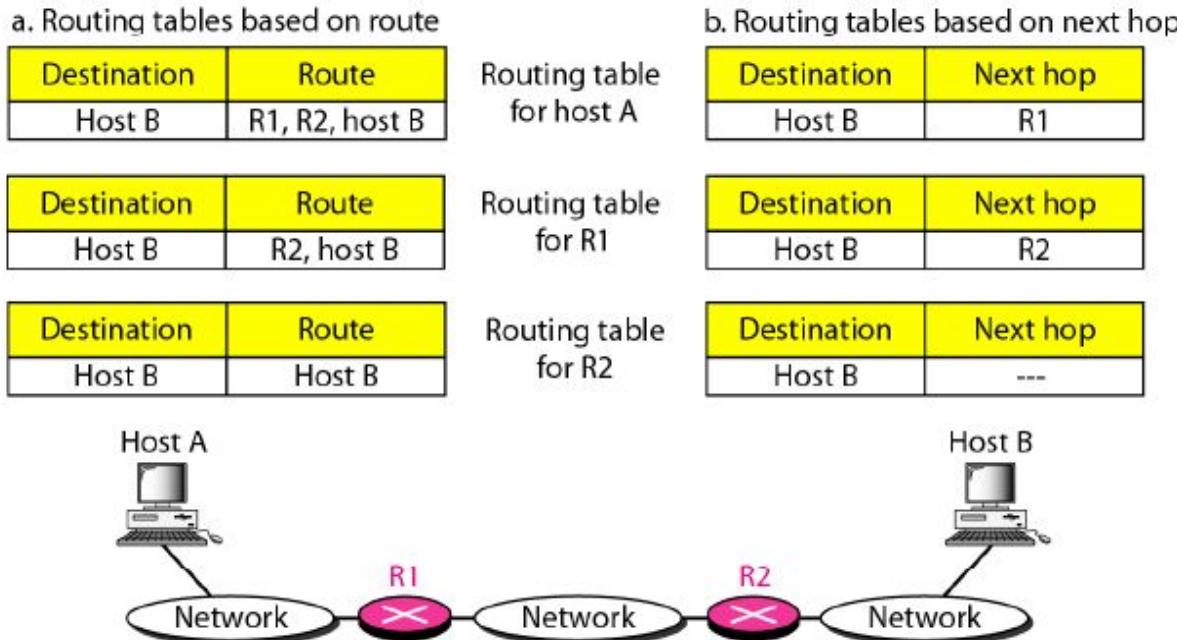
# Forwarding Techniques

Several techniques can make the **size** of the **routing table manageable** and also handle issues such as **security**. Following are some forwarding techniques:

1. *Next-Hop Method versus Route Method*
2. *Network-Specific Method versus Host-Specific Method*
3. *Default Method*

# Next-Hop Method versus Route Method

- One technique to reduce the contents of a routing table is called the **next-hop method**.
- In this technique, the routing table holds only the address of the next hop instead of information about the complete route (route method).
- The entries of a routing table must be consistent with one another.



# Network-Specific Method versus Host-Specific Method

- A **second technique** to reduce the routing table and simplify the searching process is called the **network-specific method**.
- In this method, instead of having an entry for every destination host connected to the same physical network (host-specific method), we have only **one entry** that defines the **address of the destination network** itself.
- In other words, we treat all hosts connected to the same network as one single entity.
- For **example**, if 1000 hosts are attached to the same network, only one entry exists in the routing table instead of 1000.

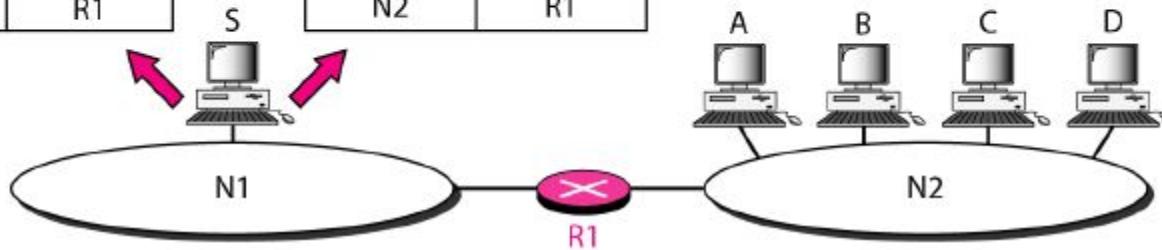
# Network-Specific Method versus Host-Specific Method

Routing table for host S based  
on host-specific method

Destination	Next hop
A	R1
B	R1
C	R1
D	R1

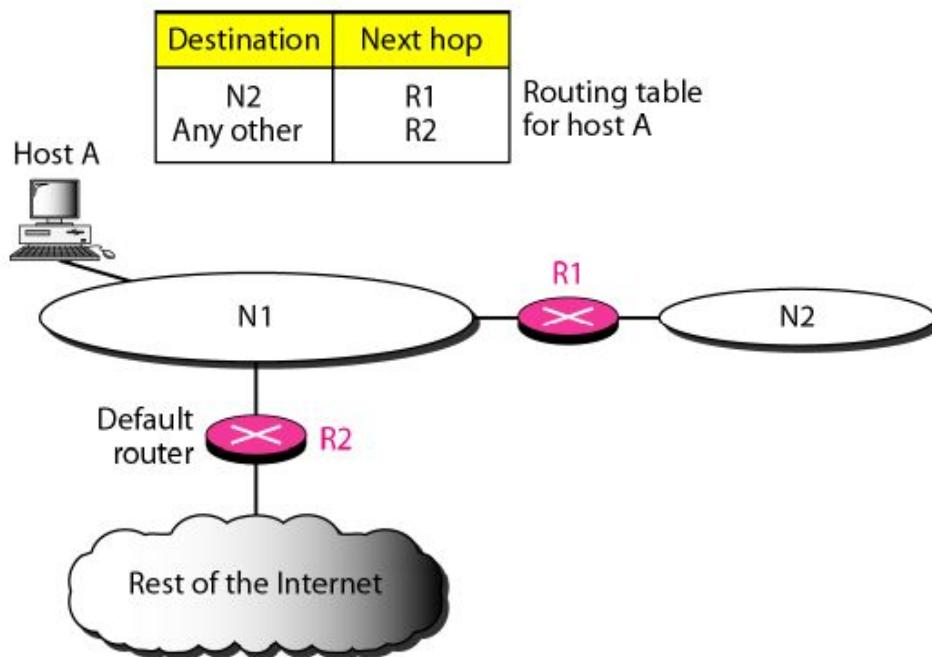
Routing table for host S based  
on network-specific method

Destination	Next hop
N2	R1



# Default Method

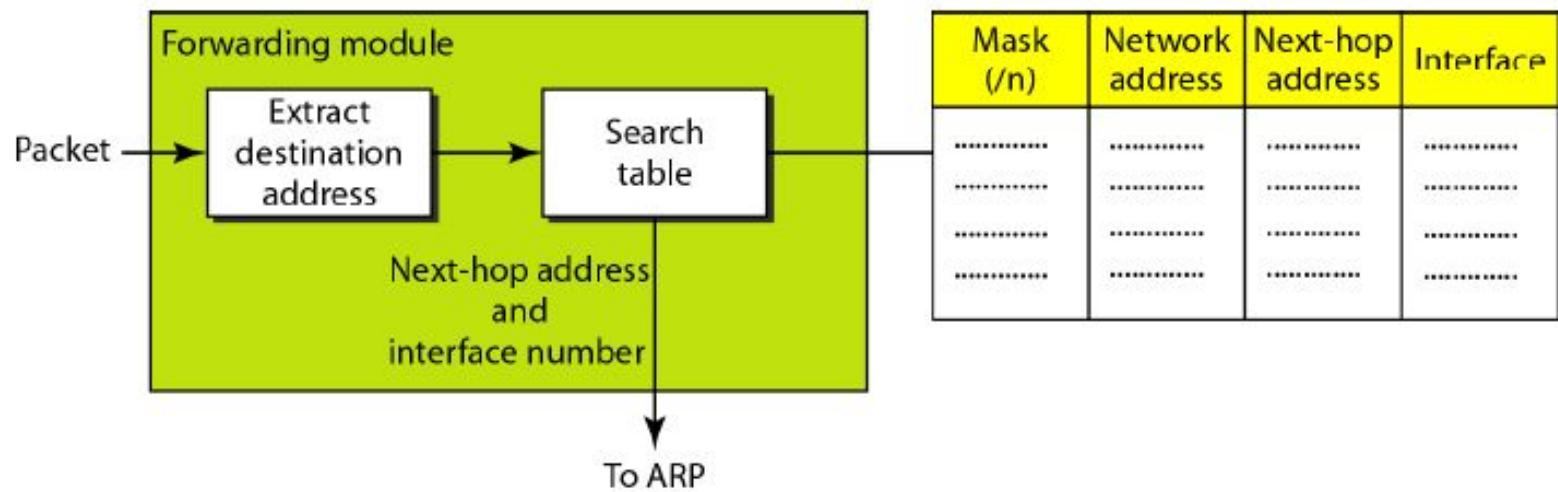
- Another technique to simplify routing is called the default method. In Figure below host A is connected to a network with two routers. Router R1 routes the packets to hosts connected to network N2. However, for the rest of the Internet, router R2 is used. So instead of listing all networks in the entire Internet, host A can just have one entry called the ***default(any other)***.



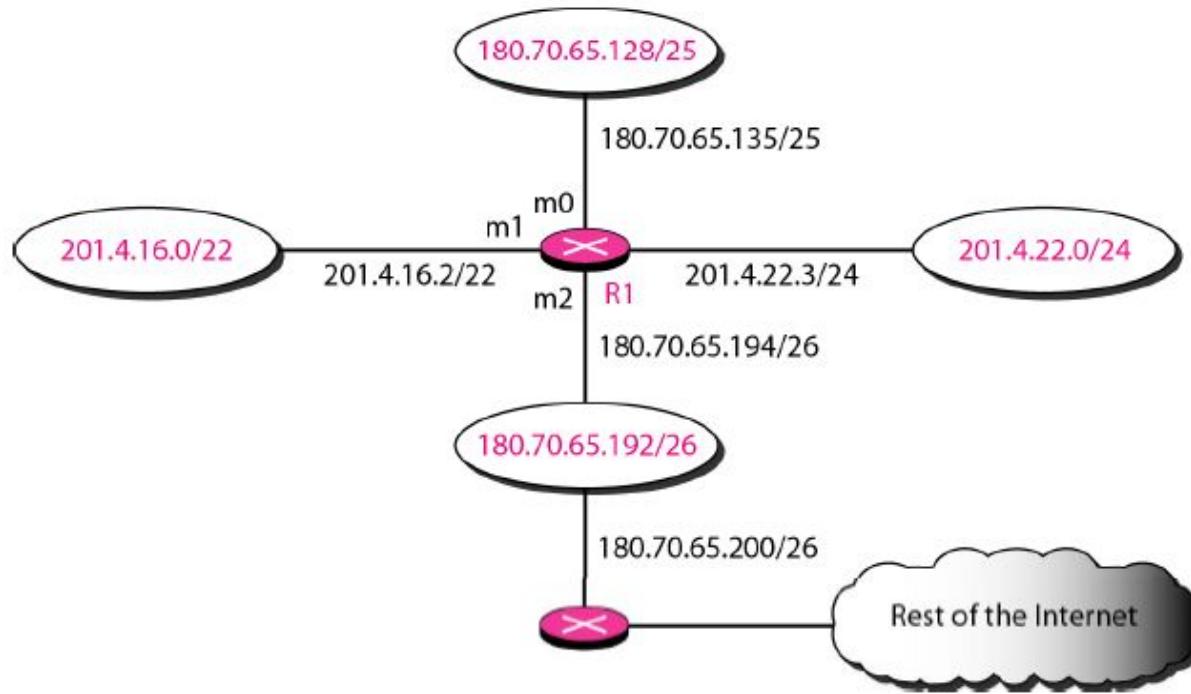
# Forwarding Process

- In **classless addressing**, the routing table needs to have one row of information for each **block** involved.
- The **table** needs to be searched based on the **network address (first address in the block)**.
- Unfortunately, the **destination address** only in the packet gives **no clue about the network address**.
- To solve the problem, we need to include the **mask (/n) in the table**; we need to have an **extra column** that includes the mask for the corresponding block.
- We need **at least four columns in our routing table**; usually there are more.

# Forwarding Process



# Forwarding Process Example



<i>Mask</i>	<i>Network Address</i>	<i>Next Hop</i>	<i>Interface</i>
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	....	m1
Any	Any	180.70.65.200	m2

# *Example 1*

Show the forwarding process if a packet arrives at **R1** in previous Figure with the **destination address 180.70.65.140**.

## **Solution**

- The **router** performs the **following steps**:
  1. The **first mask (/26)** is applied to the **destination address**. The result is **180.70.65.128**, which **does not match the corresponding network address**.
  2. The **second mask (/25)** is applied to the **destination address**. The result is **180.70.65.128**, which **matches the corresponding network address**. The **next-hop address** (the destination address of the packet in this case) and the **interface number m0** are used for further processing.

# *Example 2*

Show the forwarding process if a packet arrives at **R1** with the **destination address 18.24.32.78**.

## **Solution**

- This time all masks are applied, one by one, to the destination address, but no matching network address is found.
- When it reaches the end of the table, the module gives the next-hop address **180.70.65.200** and interface number **m2**.
- This is probably an **outgoing packet** that needs to be sent, via the **default router**, to someplace else in the **Internet**.

# Routing Table

A **host or a router** has a **routing table** which can be either **Static** or **Dynamic**.

## Static Routing Table

- A **Static routing table** contains information entered **manually**.
- The **administrator** enters the **route for each destination** into the table.
- When a table is created, it **cannot update automatically** when there is a change in the Internet.
- The table must be **manually altered by the administrator**.
- A **static routing table** can be used in a **small internet** that **does not change very often**, or in an experimental internet for troubleshooting.
- It is **poor strategy** to use a **static routing table** in a **big network** such as the **Internet**.

# Dynamic Routing Table

- A **Dynamic routing table** is **updated periodically** by using one of the **Dynamic routing protocols** such as RIP, OSPF, or BGP.
- Whenever there is a **change in the Internet**, such as:
  - shutdown of a router
  - breaking of a link etc.
- The **Dynamic routing protocols** **update all the tables** in the routers (and eventually in the host) **automatically**.
- The routers in a big internet such as the Internet need to be **updated dynamically** for **efficient delivery** of the IP packets.

# Format of a Routing Table

- A **routing table** for **classless addressing** has a minimum of **four columns**. However, some of today's routers have even more columns.
- The **number of columns is vendor-dependent**, and not all columns can be found in all routers.

Mask	Network address	Next-hop address	Interface	Flags	Reference count	Use
.....	.....	.....	.....	.....	.....	.....

- **Mask.** This field defines the mask applied for the entry.
- **Network address.** This field defines the network address to which the packet is finally delivered. In the case of host-specific routing, this field defines the address of the destination host.
- **Next-hop address.** This field defines the address of the next-hop router to which the packet is delivered.
- **Interface.** This field shows the interface id.

# Format of a Routing Table

- **Flags.** This field defines up to **five flags**:
  - **U (up).** The U flag indicates the **router is up and running**. If this flag is not present, it means that the router is down. The packet cannot be forwarded and is discarded.
  - **G (gateway).** The G flag means that the **destination is in another network**. The packet is delivered to the next-hop router for delivery (**indirect delivery**). When this flag is missing, it means the destination is in this network (**direct delivery**).
  - **H (host-specific).** The H flag indicates that the entry in the network address field is a **host-specific address**. When it is missing, it means that the address is only the **network address** of the destination.
  - **D (added by redirection).** The D flag indicates that routing information for this destination has been added to the host routing table by a **redirection message from ICMP**.
  - **M (modified by redirection).** The M flag indicates that the routing information for this destination has been **modified by** a redirection message from **ICMP**.

# Format of a Routing Table

- **Reference count.** This field gives the **number of users of this route** at the moment. For **example**, if five people at the same time are connecting to the same host(destination) from this router, the value of this column is 5.
- **Use.** This field shows the **number of packets transmitted through this router** for the corresponding destination.

# UNICAST ROUTING PROTOCOLS

- A **routing protocol** is a **combination of rules and procedures** that let routers in the internet **inform each other of changes**.
- It allows **routers** to **share** whatever they know about the internet or their **neighborhood**.
- The sharing of information allows a router in New Delhi to know about the failure of a network in Lucknow.
- The **routing protocols** also include **procedures for combining information received from other routers**.

# Optimization

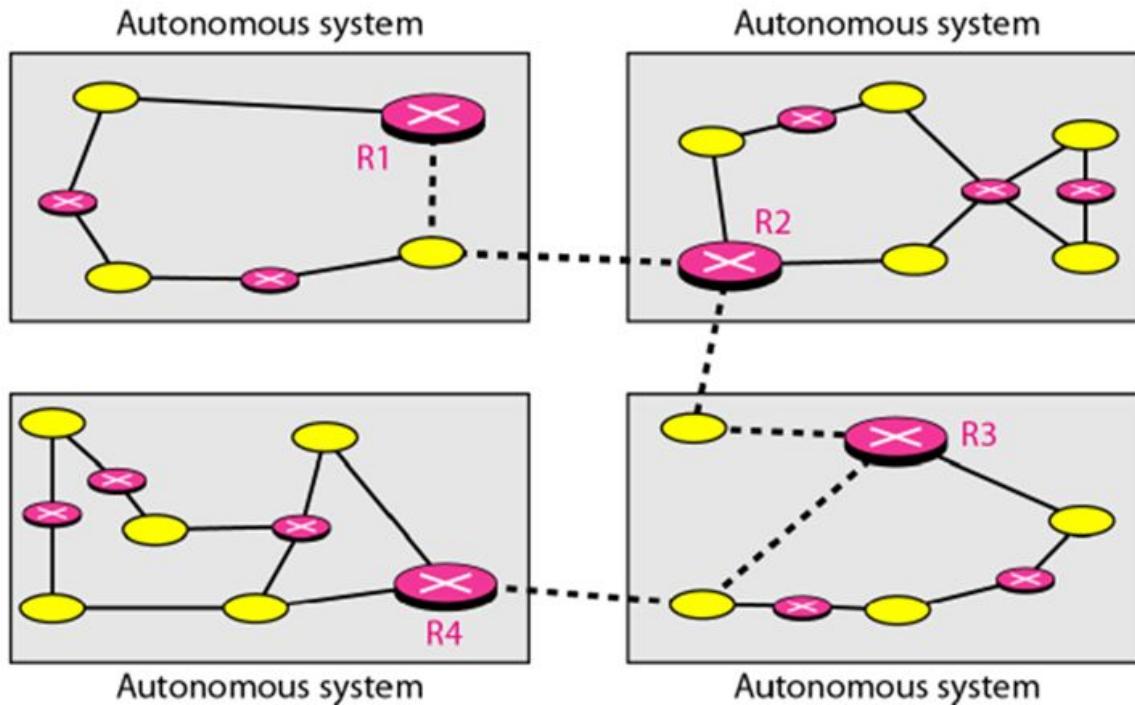
- A router receives a packet from a network and passes it to another network.
- A router is usually attached to several networks.
- When it receives a packet, to which network should it pass the packet?
- The decision is based on optimization: Which of the available pathways is the optimum pathway?
- What is the definition of the term *optimum*?
- One approach is to assign a cost for passing through a network. We call this cost metric.
- The metric assigned to each network depends on the type of protocol.

# Optimization

- Some simple protocols, such as the **Routing Information Protocol (RIP)**, treat all networks as equals. The cost of passing through a network is the same; it is **one hop count**. So if a packet passes through 10 networks to reach the destination, the total cost is 10 hop counts.
- Other protocols, such as **Open Shortest Path First (OSPF)**, allow the **administrator to assign a cost for passing through a network** based on the **type of service required**. A route through a network can have different costs (metrics). For **example**,
  - If **maximum throughput** is the **desired type of service**, a **satellite link** has a **lower metric** than a fiber-optic line.
  - If **minimum delay** is the **desired type of service**, a fiber-optic line has a lower metric than a satellite link.
- **OSPF protocol** allows each router to have **several routing tables** based on the required type of service.

# Autonomous System

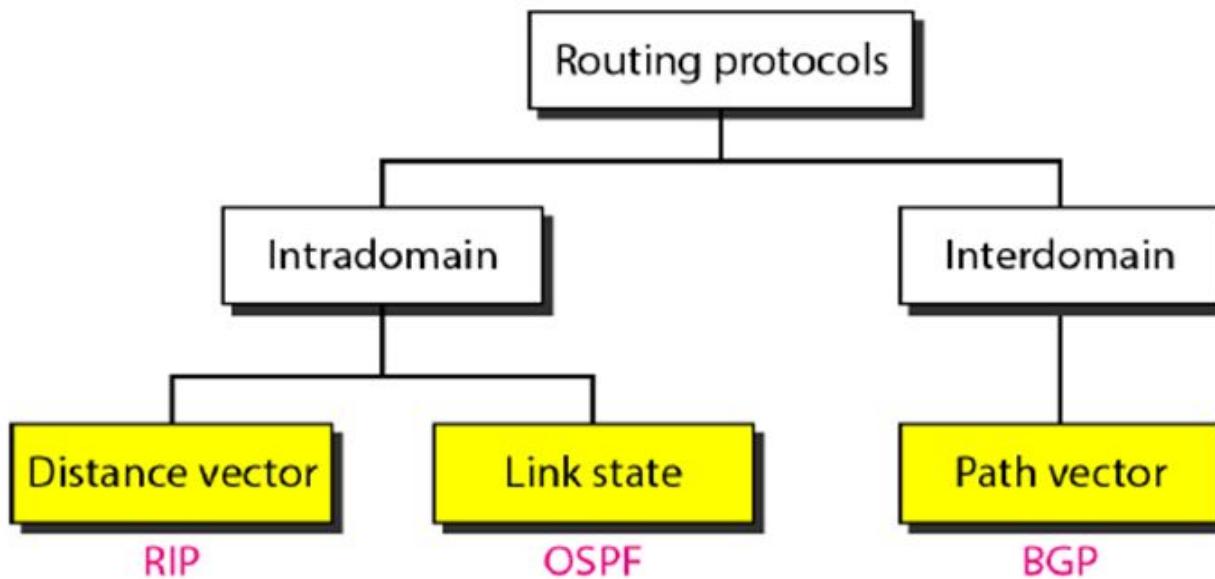
- Today, an internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers.
- For this reason, an internet is divided into autonomous systems.
- “*An autonomous system (AS) is a group of networks and routers under the authority of a single administration*”.



# Intra- and Interdomain Routing

- Routing **inside an autonomous system** is referred to as **intradomain routing**. Two intradomain routing algorithms are **distance vector routing** and **link state routing**.
- Routing **between autonomous systems** is referred to as **interdomain routing**. One interdomain routing algorithm is **path vector routing**.
- Each autonomous system can choose one or more intradomain routing protocols to handle routing inside the autonomous system.
- However, only one interdomain routing protocol handles routing between autonomous systems.
- **Routing Information Protocol (RIP)** is an implementation of the **Distance Vector routing algorithm**.
- **Open Shortest Path First (OSPF)** is an implementation of the **Link state routing algorithm**.
- **Border Gateway Protocol (BGP)** is an implementation of the **Path vector algorithm**.

# Popular routing protocols

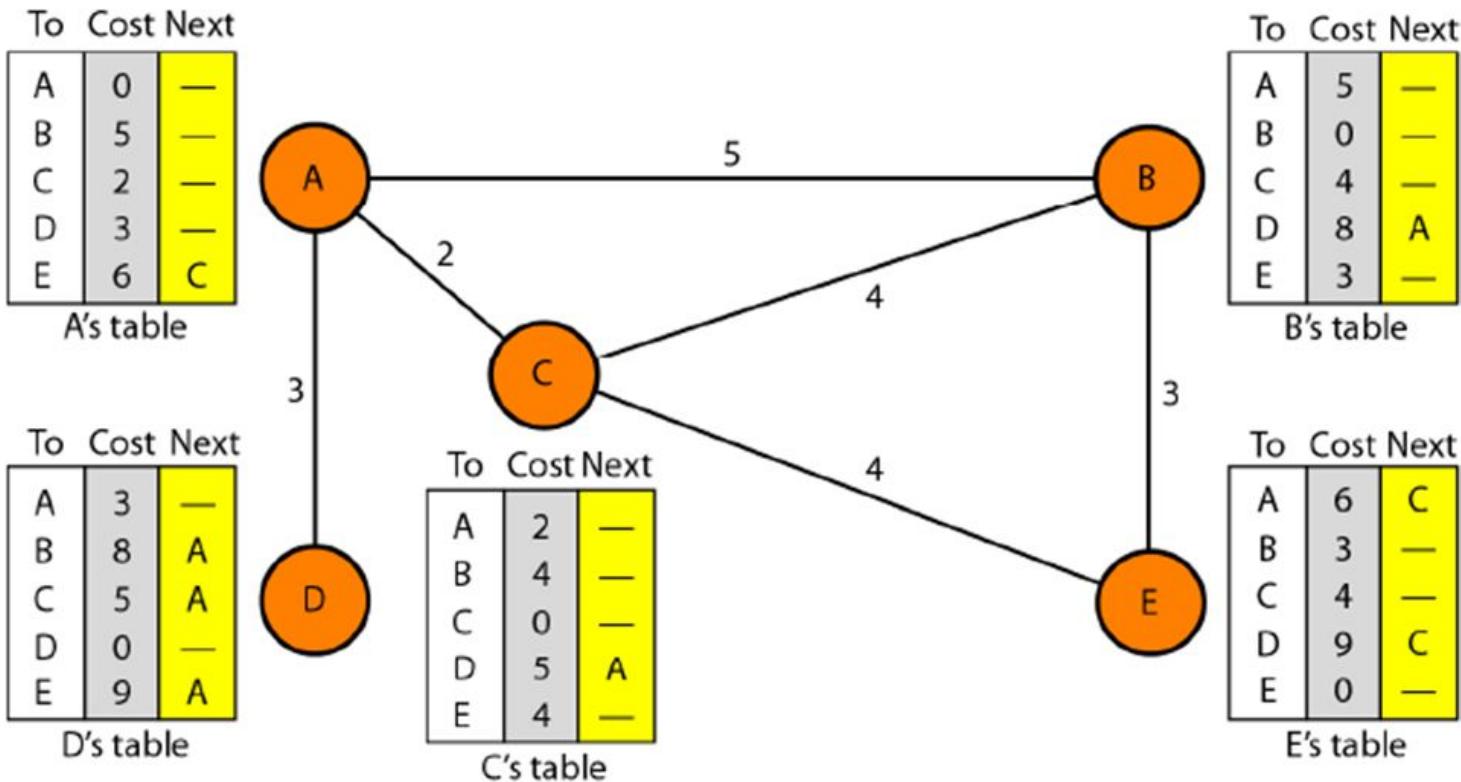


# Distance Vector Routing

- In distance vector routing, the least-cost route between any two nodes is the route with **minimum distance**.
- In this protocol, as the name implies, **each node maintains a vector** (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 (next-hop routing).
- We can think of nodes as the cities in an area and the lines as the roads connecting them.
- A table can show a tourist the minimum distance between cities.

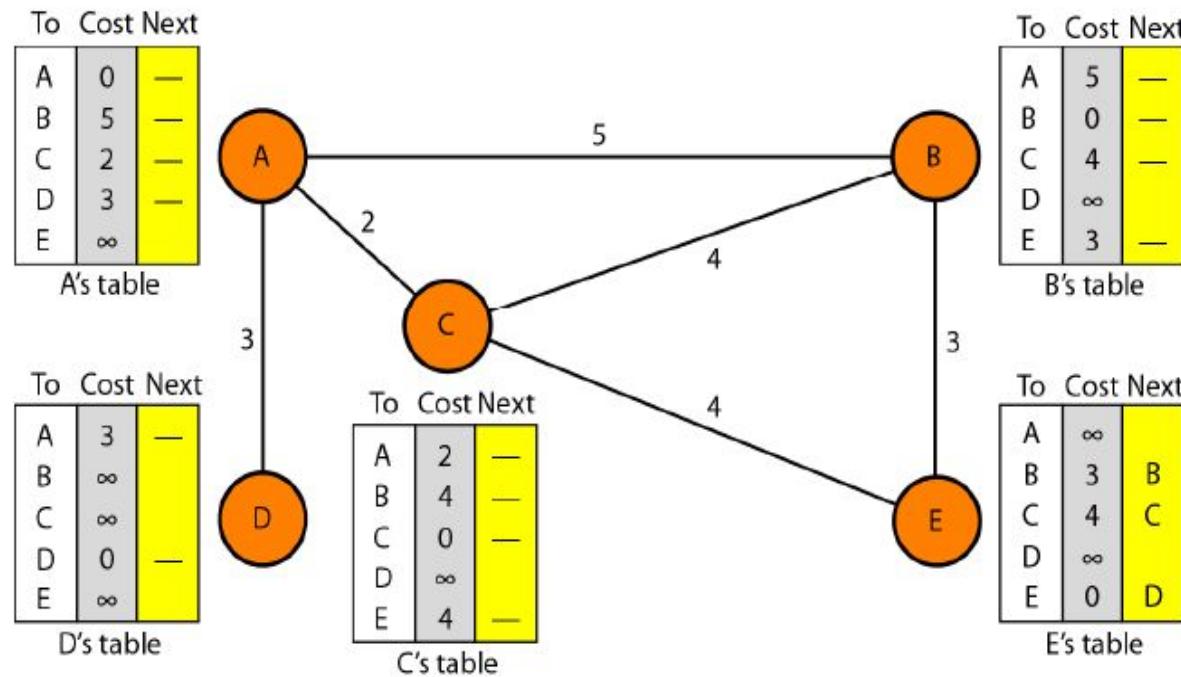
# Distance Vector Routing Tables

- The table for node A shows how we can reach any node from this node.
- For example, our least cost to reach node E is 6. The route passes through C.



# Initialization

- At the beginning, each node can know only the distance between itself and its immediate neighbors, those directly connected to it.
- So for the moment, we assume that each node can send a message to the immediate neighbors and find the distance between itself and these neighbors.
- Figure below shows the initial tables for each node. The distance for any entry that is not a neighbor is marked as infinite (unreachable).



# Sharing

- The whole idea of distance vector routing is the sharing of information between neighbors.
- Although node A does not know about node E, node C does.
- So if node C shares its routing table with A, node A can also know how to reach node E.
- On the other hand, node C does not know how to reach node D, but node A does.
- If node A shares its routing table with node C, node C also knows how to reach node D.
- In other words, nodes A and C, as immediate neighbors, can improve their routing tables if they help each other.

# Sharing

- There is only one problem. **How much of the table must be shared with each neighbor?**
- The best solution for each node is to send its entire table to the neighbor and let the neighbor decide what part to use and what part to discard.
- However, the third column of a table (next hop) is not useful for the neighbor.
- When the neighbor receives a table, this column needs to be replaced with the sender's name.
- A node therefore can send only the **first two columns** of its table to any neighbor.
- In other words, sharing here means sharing only the first two columns.

# Updating

When a node receives a two-column table from a neighbor, it needs to update its routing table.

**Updating takes three steps:**

**1.** The receiving node **adds the cost between itself and the sending node** to each value in the **second column of received table**. If node C claims that its distance to a destination is  $x$  meter, and the distance between A and C is  $y$  meter, then the distance between A and that destination, via C, is  $x + y$  meter.

**2.** The receiving node add the **name of the sending node** to each row as **the third column**.

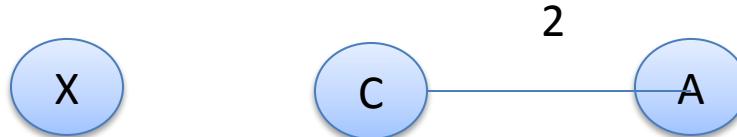
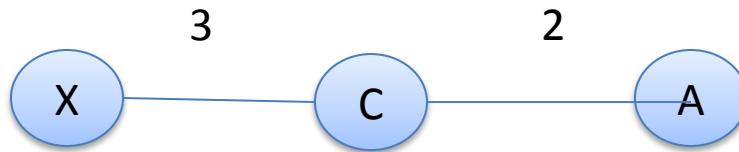
Now this table is referred as **modified version of received table**.

**3.** The receiving node **compares each row of its old table with the corresponding row of the modified version of the received table**.

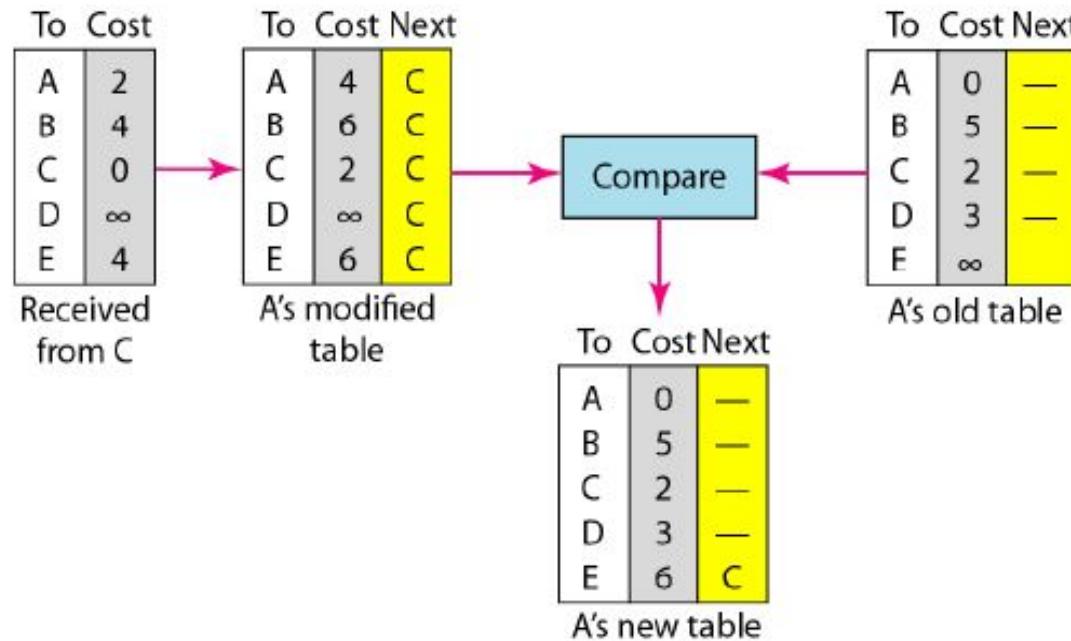
- a. If the **next-node entry is different** in the above two tables, the receiving node **chooses the row with the smaller cost**. If there is a **tie(in cost)**, the old one is kept.
- b. If the **next-node entry is the same**, the receiving node **chooses the new row**.

# Updating

**Example**, suppose node C has previously advertised a route to node X with distance 3. Suppose that now there is no path between C and X; node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist anymore. The new route has a distance of infinity.



node A updates its routing table after receiving the partial table from node C.



- Each node can update its table by using the tables received from other nodes.
- In a short time, if there is no change in the network itself, such as a failure in a link, each node reaches a **stable condition** in which the contents of its table remains the same.

# When to Share

- The question now is, **when does a node send its partial routing table** (only two columns) to all its **immediate neighbors**? There are **two types of update**:
- **Periodic Update** A node sends its routing table, normally **every 30 sec**, in a periodic update.
- **Triggered Update** A node sends its two-column routing table to its neighbors **anytime there is a change in its routing table**. This is called a triggered update. The change can result from the following:
  1. A node receives a table from a neighbor, resulting in changes in its own table after updating.
  2. A node detects some **failure** in the neighboring links which results in a distance change to infinity.

# Two-Node Loop Instability

- A problem with distance vector routing is **instability**, which means that a network using this protocol can become **unstable**. To understand the problem, let us look at the scenario depicted in Figure below.

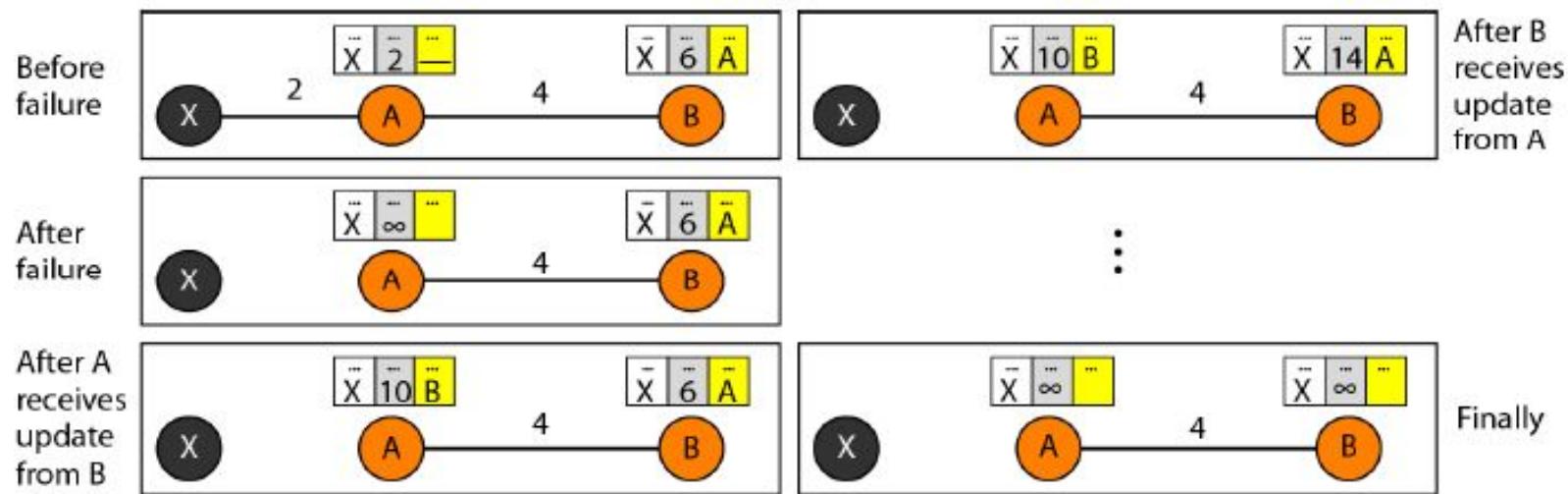


Figure shows a system with **three nodes**. We have shown only the portions of the routing table needed for our discussion.

# Two-Node Loop Instability

- At the beginning, both nodes A and B know how to reach node X.
- But suddenly, the link between A and X fails and Node A changes its table.
- If A can send its table to B immediately, everything is fine.
- However, the system becomes unstable if B sends its routing table to A before receiving A's routing table.
- In such case Node A receives the update and, assuming that B has found a way to reach X, 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.
- However, during this time 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. Similarly, 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.

# Solutions to two node loop instability problem

A few solutions have been proposed for instability of this kind:

## 1. Defining Infinity

- The first obvious solution is to **redefine infinity** to a smaller number, such as **100**.
- For our previous scenario, the system will be stable in less than **20 updates**.
- As a matter of fact, most implementations of the **distance vector protocol** define the **distance between each node to be 1** and **define 16 as infinity**.
- However, this means that the ***distance vector routing cannot be used in large systems.***
- The **size of the network**, in each direction, **cannot exceed 15 hops**.

# Solutions to two node loop instability problem

## 2. Split Horizon

- In this strategy, instead of flooding the table through each interface, **each node sends only part of its table** through each interface.
- If, according to its table, *node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has come from A (A already knows).*
- *Taking information from node A, modifying it, and sending it back to node A creates the confusion.*
- In our scenario, node B eliminates the last line of its routing table before it sends it to A.
- In this case, node A keeps the value of infinity as the distance to X.
- *Later when node A sends its routing table to B, node B also corrects its routing table.*
- The **system becomes stable** after the first update: both node A and B know that X is not reachable.

# Solutions to two node loop instability problem

## 3. Split Horizon and Poison Reverse

- Using the **split horizon** strategy has one **drawback**.
- Normally, the distance vector protocol uses a timer, and if there is no news about a route, the node deletes the route from its table.
- When node B in the previous scenario eliminates the route to X from its advertisement to A, *node A cannot guess that this is due to the split horizon strategy (the source of information was A) or because B has not received any news about X recently.*
- The **split horizon strategy can be combined with the poison reverse strategy**.
- *Node B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."*

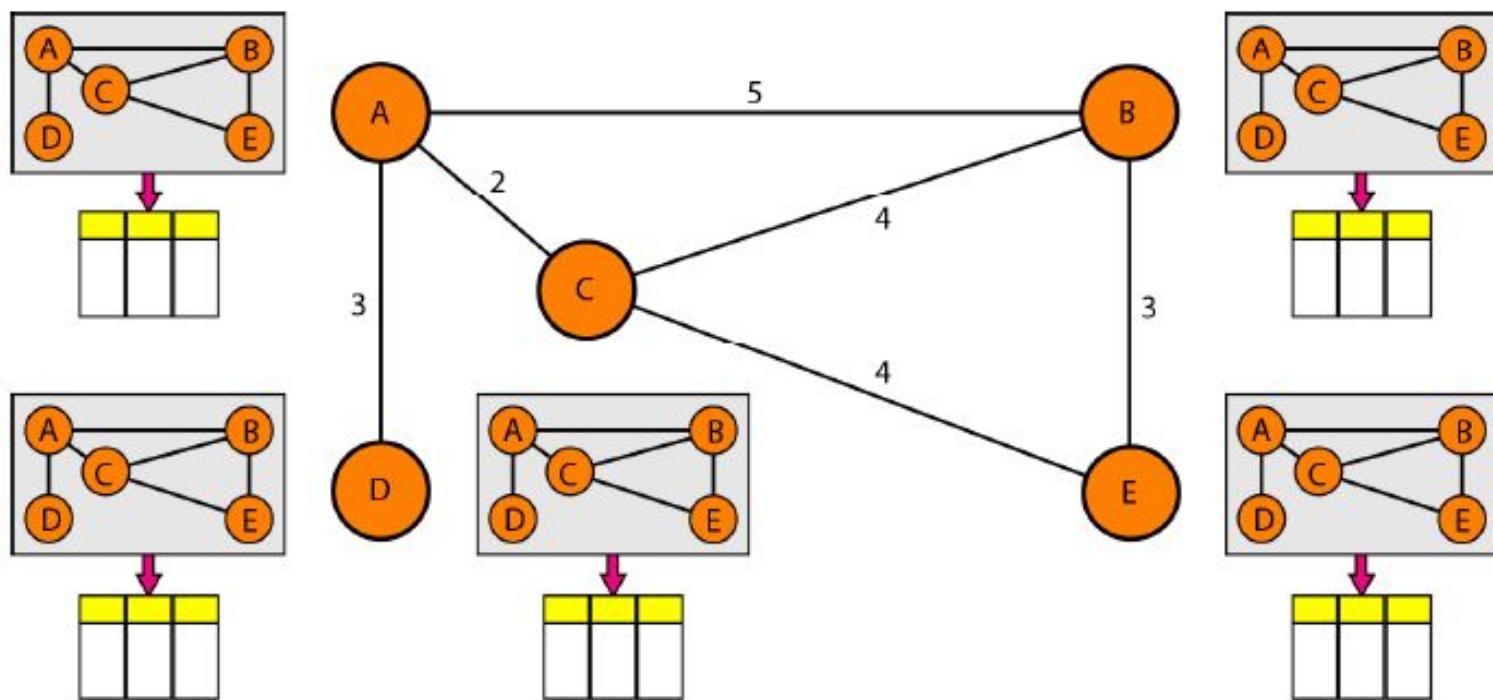
# RIP(Routing Information Protocol)

The **Routing Information Protocol (RIP)** is an **intradomain routing protocol** used **inside an autonomous system**. It is a very simple protocol based on **distance vector routing**. RIP implements distance vector routing directly with some considerations:

1. In an autonomous system, we are dealing with **routers and networks** (links). The **routers have routing tables**; networks do not.
2. The destination in a routing table is a network, which means the **first column defines a network address**.
3. The **metric** used by RIP is very simple; the **distance is defined as the number of links (networks) to reach the destination**. For this reason, the **metric in RIP** is called a **hop count**.
4. **Infinity is defined as 16**, which means that any route in an autonomous system using RIP cannot have more than 15 hops.
5. The next-node column defines the address of the router to which the packet is to be sent to reach its destination.

# Link State Routing Algorithm

- Link state routing has a different philosophy from that of distance vector routing.
- In link state routing, if each node in the domain has the entire topology of the domain the list of nodes and links, cost (metric), and condition of the links (up or down)
- The node can use **Dijkstra's algorithm** to build a **routing table**. Figure below shows the concept.



# Link State Routing Algorithm

- The figure on previous slide shows a simple domain with five nodes.
- Each node uses the same topology to create a routing table, but the routing table for each node is unique because the calculations are based on different interpretations of the topology.
- This is analogous to a city map. While each person may have the same map, each needs to take a different route to reach her specific destination.
- Link state routing is based on the assumption that, although the global knowledge about the topology is not clear, each node has **partial knowledge** of its links.
- In other words, the whole topology can be compiled from the partial knowledge of each node.

# 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 of LSPs to every other router, called **flooding**, 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**.

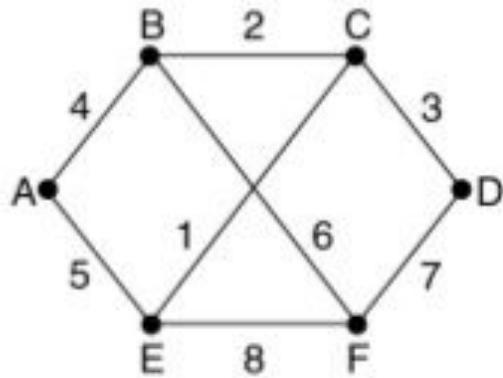
# 1. Creation of Link State Packet (LSP)

- A link state packet can carry a large amount of information.
- For the moment, however, we assume that it carries a minimum amount of data:
  - the **node identity**,
  - the **list of links**,
  - a **sequence number**, and
  - **age**.

A	
Seq.	
Age	
B	4
E	5

- The first two, **node identity** and the **list of links**, are needed to make the topology.
- The third, **sequence number**, facilitates flooding and distinguishes new LSPs from old ones.
- The fourth, **age**, prevents old LSPs from remaining in the domain for a long time.

# Building Link State Packets



(a)

Link	State	Packets
A	B	E
Seq.	Seq.	Seq.
Age	Age	Age
B 4	B 2	C 5
E 5	D 3	B 6
	F 7	D 7
	E 1	C 1
		F 8
		E 8

(b)

(a) A subnet. (b) The link state packets for this subnet.

# 1. Creation of Link State Packet (LSP)

- LSPs are generated on ***two occasions***:
  - ***When there is a change in the topology of the domain.***
    - Triggering of LSP dissemination is the main way of quickly informing any node in the domain to update its topology.
  - ***On a periodic basis.***
    - The period in this case is much longer compared to distance vector routing.
    - ***As a matter of fact, there is no actual need for this type of LSP dissemination.***
    - It is done to ensure that old information is removed from the domain.
    - ***The timer set for periodic dissemination is normally in the range of 60 min or 2 hours based on the implementation.***
    - A longer period ensures that flooding does not create too much traffic on the network.

## 2. Flooding of LSPs

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** and based on the following:

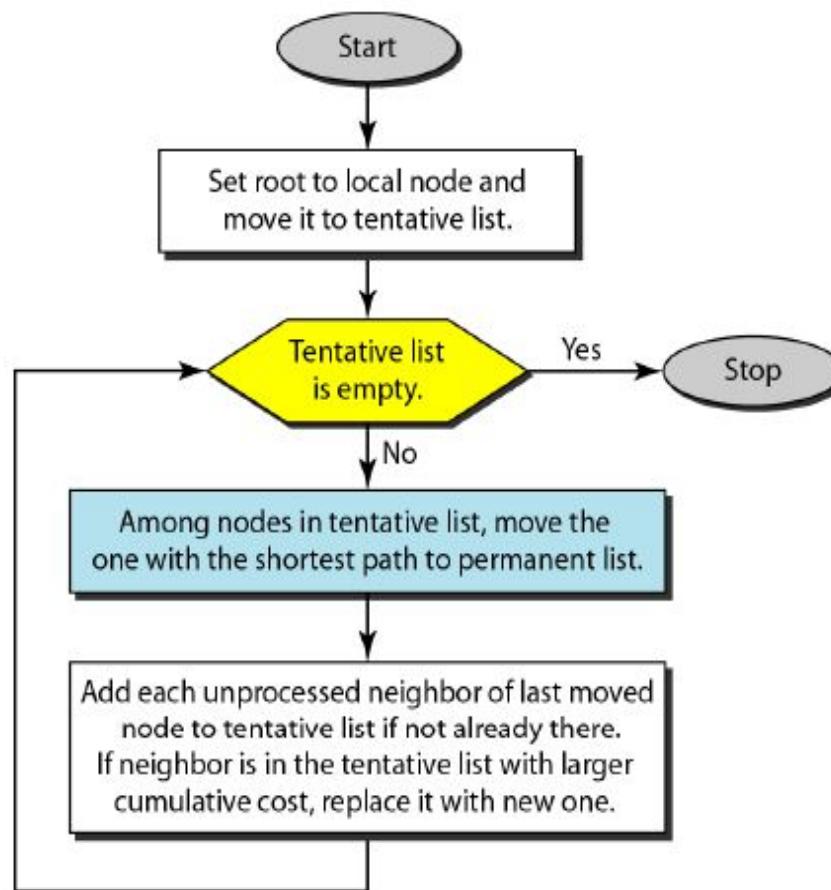
1. The creating node **sends a copy** of the **LSP** out of each interface.
2. A node that receives an LSP **compares** it with the copy it may **already have**.
3. If the newly arrived LSP is **older** than the one it has (found by checking the sequence number), it **discards the LSP**. If it is **newer**, the node does the following:
  - a. It **discards the old LSP** and keeps the new one.
  - b. It sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the domain (where a node has only one interface).

### **3. Formation of Shortest Path Tree using *Dijkstra Algorithm*:**

- After **receiving all LSPs**, each node will have a **copy of the whole topology**.
- However, the **topology is not sufficient** to find the shortest path to every other node;
- A **Shortest Path Tree** is needed.
- A tree is a **acyclic graph having nodes and links**; one node is called the **root**.
- All other nodes can be reached from the root through only one single route.
- A **shortest path tree** is a tree in which the **path between the root and every other node is the shortest**.
- What we need for **each node** is a shortest path tree with that node as the root.

# Dijkstra's Algorithm

- Dijkstra Algorithm creates a **Shortest Path tree** from a **graph**.
- The algorithm **divides the nodes into two sets**: **tentative** and **permanent**.
- It **finds the neighbors** of a **current node**, **makes them tentative**, examines them, and if they pass the criteria, makes them **permanent**.

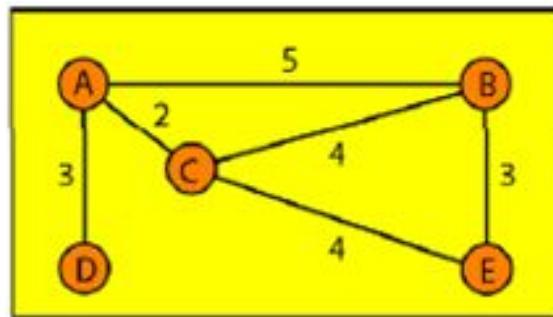


# Steps of Applying Dijkstra's Algorithm

- The following shows the steps. At the end of each step, we show the **permanent (filled circles)** and the **tentative (open circles)** nodes and lists with the **cumulative costs**.

i. **Permanent list:** empty

**Tentative list:** A(0)

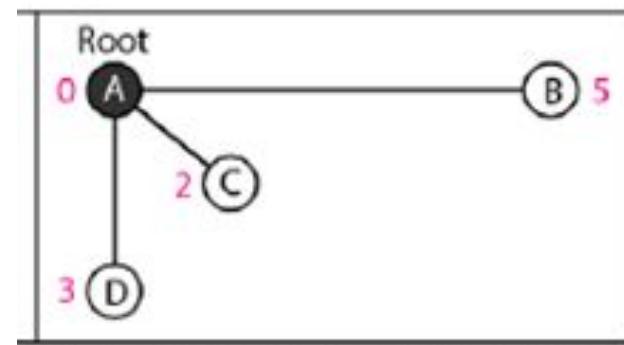


ii. **Permanent list:** A(0)

**Tentative list:** B(5), C(2), D(3)



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

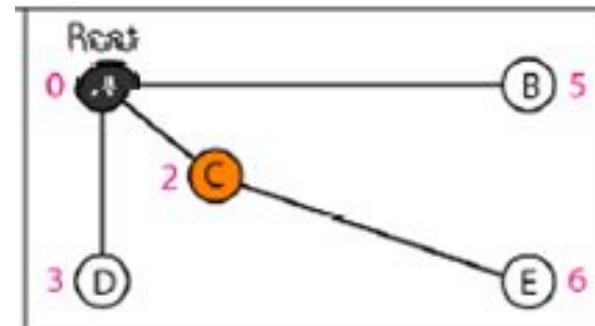
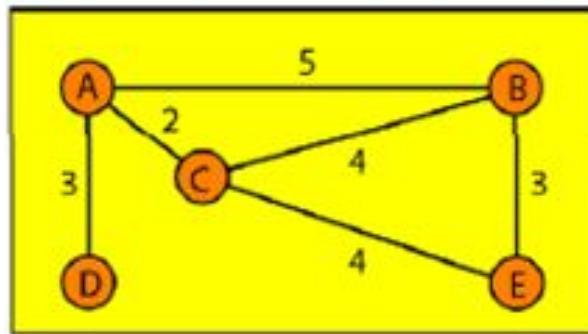


2. Move A to permanent list and add B, C, and D to tentative list.

# Steps of Applying Dijkstra's Algorithm

iii. Permanent list: A(0), C(2)

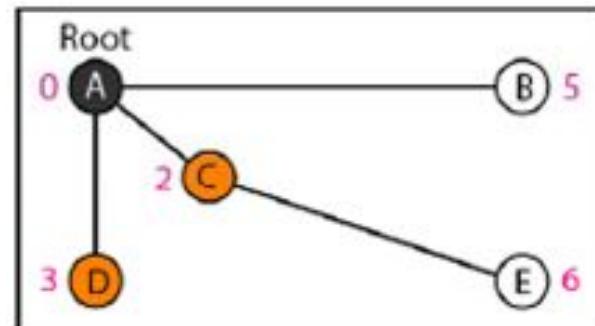
Tentative list: B(5), D(3), E(6)



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

iv. Permanent list: A(0), C(2), D(3)

Tentative list: B(5), E(6)

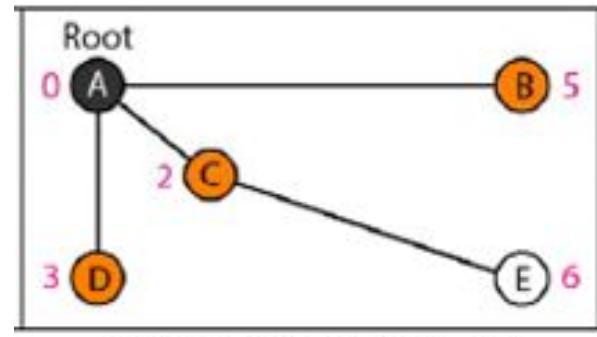
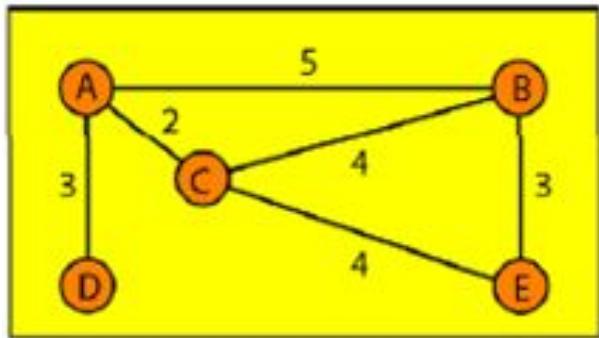


4. Move D to permanent list.

# Steps of Applying Dijkstra's Algorithm

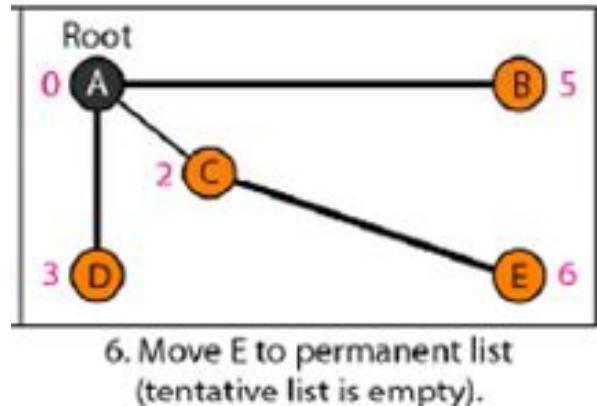
v. Permanent list:  $A(0)$ ,  $B(5)$ ,  $C(2)$ ,  $D(3)$

Tentative list:  $E(6)$



vi. Permanent list:  $A(0)$ ,  $B(5)$ ,  $C(2)$ ,  $D(3)$ ,  $E(6)$

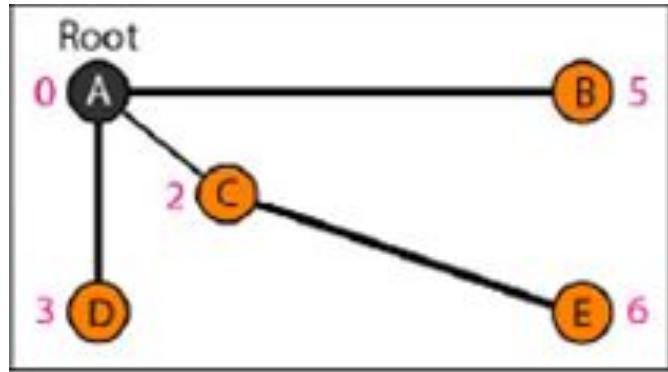
Tentative list: empty



# 4. Calculation of Routing Table from Shortest Path Tree

- Each node uses the **shortest path tree algorithm** to construct its **routing table**.
- The **routing table** shows the cost of reaching each node from the root.
- Table below shows the routing table for **node A**.

**Table. Routing table for node A**



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

- Both distance vector routing and link state routing end up with the **same routing table for node A**.

# OSPF(Open Shortest Path First )

- The Open Shortest Path First or OSPF protocol is an **intradomain routing protocol** based on **link state routing algorithm**.
- The OSPF protocol allows the administrator to assign a **cost**, called the **metric**, to **each route**.
- The metric can be based on a **type of service** (minimum delay, maximum throughput, and so on).
- As a matter of fact, a router can have **multiple routing tables**, each based on a different type of service.

# Path Vector Routing

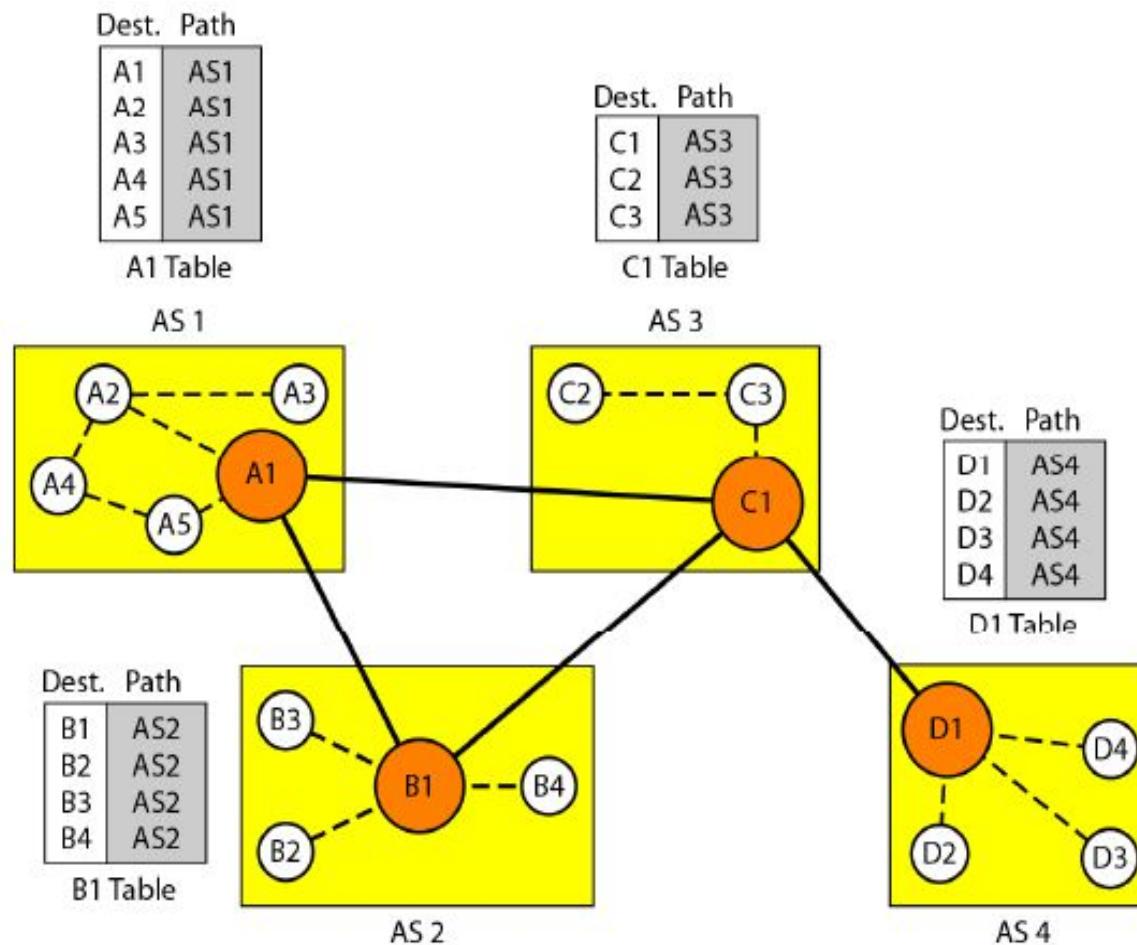
- Distance vector and Link state routing are both **intradomain routing protocols**.
- They can be used inside an autonomous system (AS), but **not between autonomous systems**.
- These two protocols are **not suitable for interdomain routing** mostly because of **scalability**.
- Distance vector routing is subject to **instability** if there are more than a few hops in the domain of operation.
- Link state routing needs a **huge amount of resources** to calculate routing tables. It also creates **heavy traffic** because of **flooding**.
- There is a need for a **third routing protocol** which we call **path vector routing**.
- Path vector routing proved to be useful for **interdomain routing**.

# Path Vector Routing

- The principle of path vector routing is similar to that of distance vector routing.
- In path vector routing, we assume that there is one node in each autonomous system that acts on behalf of the entire autonomous system. Let us call it the speaker node.
- The speaker node in an AS creates a routing table and advertises it to speaker nodes in the neighboring ASs.
- The idea is the same as for distance vector routing except that only speaker nodes in each AS can communicate with each other.
- However, what is advertised is different.
- A speaker node advertises the path, not the metric of the nodes, in its autonomous system or other autonomous systems.

# Path Vector Routing

- At the beginning, each **speaker node** can know only the reachability of nodes inside its autonomous system. Figure below shows the initial tables for each speaker node in a system made of four ASs.



# Path Vector Routing

## Step 1. Initialization

- Node A1 is the speaker node for AS1, B1 for AS2, C1 for AS3, and D1 for AS4.
- Node A1 creates an **initial table** that shows A1 to A5 are located in AS1 and can be reached through it.
- Node B1 advertises that B1 to B4 are located in AS2 and can be reached through B1.  
And so on.

## Step 2. Sharing

- Just as in **distance vector routing**, in path vector routing, a **speaker in an autonomous system shares its table with immediate neighbors**.
- Node A1 shares its table with nodes B1 and C1.
- Node C1 shares its table with nodes D1, B1, and A1.
- Node B1 shares its table with C1 and A1.
- Node D1 shares its table with C1.

# Path Vector Routing

## Step 3. Updating

- When a **speaker node receives a two-column table from a neighbor, it updates its own table** by adding the nodes that are not in its routing table and adding its own autonomous system and the autonomous system that sent the table.
- After a while each speaker has a table and knows how to reach each node in other ASs. Figure on next slide shows the tables for each speaker node after the system is stabilized.
- According to the figure, if router A1 receives a packet for nodes A3, it knows that the path is in AS1 (the packet is at home); but if it receives a packet for D1, it knows that the packet should go from AS1, to AS2, and then to AS3.
- The routing table shows the path completely. On the other hand, if node D1 in AS4 receives a packet for node A2, it knows it should go through AS4, AS3, and AS1.

# Stabilized tables for three Autonomous Systems

Dest.	Path
A1	AS1
...	
A5	AS1
B1	AS1-AS2
...	...
B4	AS1-AS2
C1	AS1-AS3
...	...
C3	AS1-AS3
D1	AS1-AS2-AS4
...	...
D4	AS1-AS2-AS4

A1 Table

Dest.	Path
A1	AS2-AS1
...	
A5	AS2-AS1
B1	AS2
...	...
B4	AS2
C1	AS2-AS3
...	...
C3	AS2-AS3
D1	AS2-AS3-AS4
...	...
D4	AS2-AS3-AS4

B1 Table

Dest.	Path
A1	AS3-AS1
...	
A5	AS3-AS1
B1	AS3-AS2
...	...
B4	AS3-AS2
C1	AS3
...	...
C3	AS3
D1	AS3-AS4
...	...
D4	AS3-AS4

C1 Table

Dest.	Path
A1	AS4-AS3-AS1
...	
A5	AS4-AS3-AS1
B1	AS4-AS3-AS2
...	...
B4	AS4-AS3-AS2
C1	AS4-AS3
...	...
C3	AS4-AS3
D1	AS4
...	...
D4	AS4

D1 Table

Dest. Path

A1	AS1
A2	AS1
A3	AS1
A4	AS1
A5	AS1

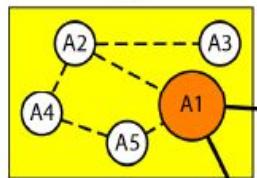
A1 Table

Dest. Path

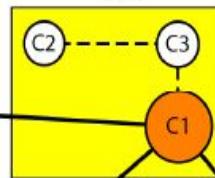
C1	AS3
C2	AS3
C3	AS3

C1 Table

AS 1



AS 3



Dest. Path

D1	AS4
D2	AS4
D3	AS4
D4	AS4

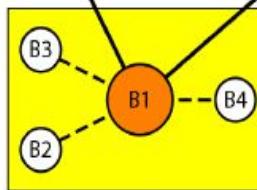
D1 Table

Dest. Path

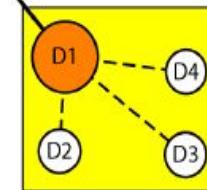
B1	AS2
B2	AS2
B3	AS2
B4	AS2

B1 Table

AS 2



AS 4



# **Lecture 6.1**

## **Transport Layer: Process-to-Process Delivery: UDP and TCP**

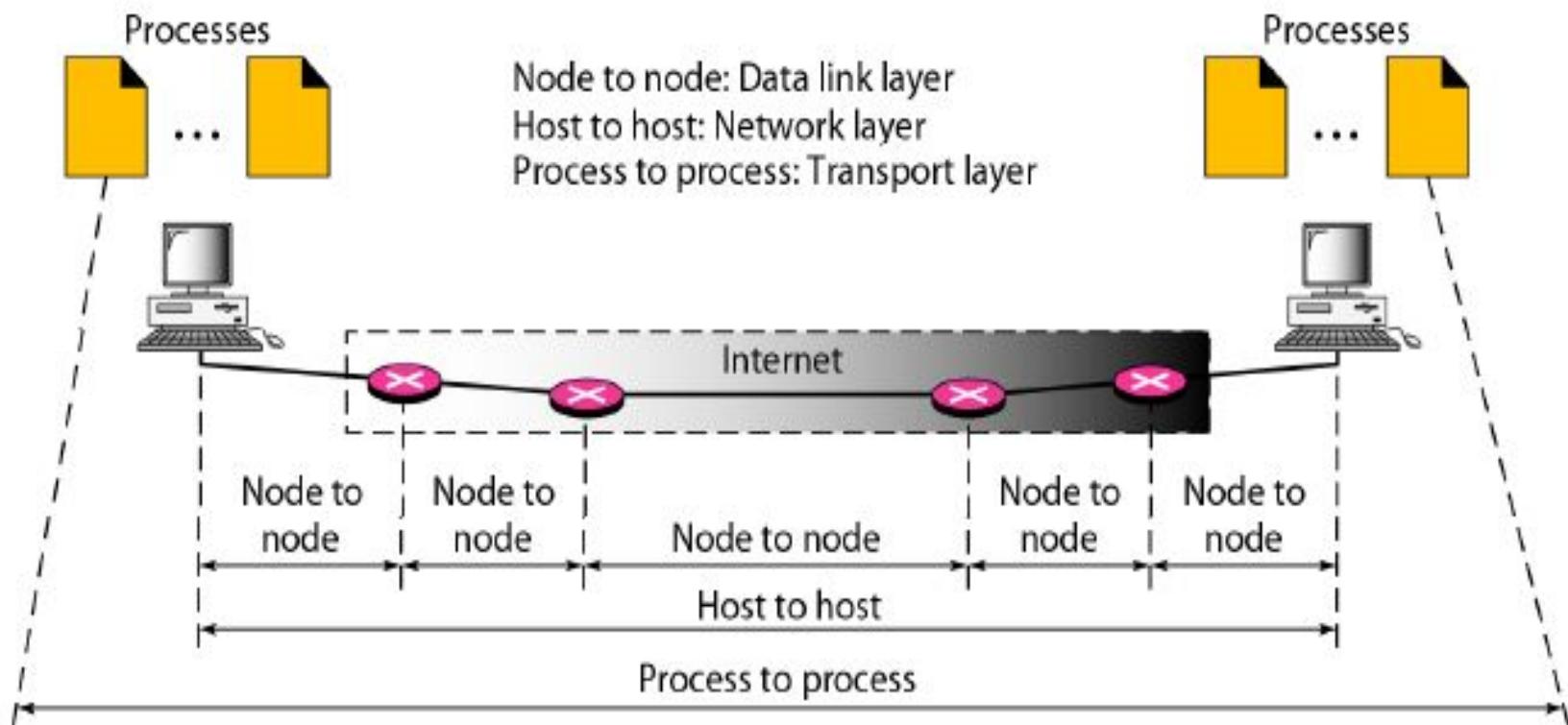
**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# PROCESS-TO-PROCESS DELIVERY

- Communication on the Internet is not defined as the exchange of data between two nodes or between two hosts.
- But real communication takes place between two **processes (application programs)**.
- We need **process-to-process delivery**.
- However, at any moment, several processes may be running on the source host and several on the destination host.
- To complete the delivery, we need a mechanism to deliver data from one of these processes running on the source host to the corresponding process running on the destination host.
- The **Transport layer** is responsible for **process-to-process delivery**-the delivery of a packet, part of a message, from one process to another.

# PROCESS-TO-PROCESS DELIVERY



# Client/Server Paradigm

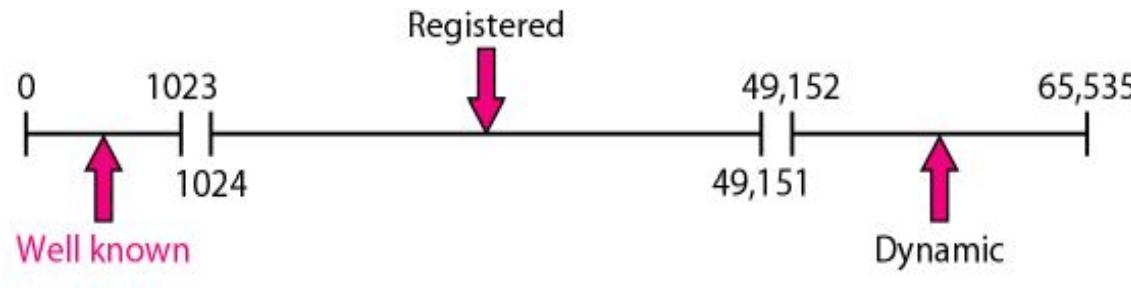
- Although there are several ways to achieve process-to-process communication, the most common one is through the **client/server paradigm**.
- A process on the **local host**, called a **client**, needs services from a process usually on the **remote host**, called a **server**.
- Both processes (client and server) have the **same name**.
- For example, to get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.
- A remote computer can run several server programs at the same time, just as local computers can run one or more client programs at the same time.

# Port Address

- In the Internet model, the **port numbers** are **16-bit integers** between **0** and **65,535**.
- The **client program** defines itself with a **port number**, chosen **randomly** by the **transport layer software** running on the client host. This is the **ephemeral port number**.
- The **server process** must also define itself with a **port number**. This port number, however, **cannot be chosen randomly**.
- The Internet has decided to use **universal port numbers for servers**; these are called **well-known port numbers**.

# Types of Port Numbers

- The IANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: *well known*, *registered*, and *dynamic* (or *private*).
- i. **Well-known ports.** The ports ranging from **0 to 1023** are assigned and controlled by IANA. These are the well-known ports.
- ii. **Registered ports.** The ports ranging from **1024 to 49,151** are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication. These are used by vendors for their own server applications.
- iii. **Dynamic ports.** The ports ranging from **49,152 to 65,535** are neither controlled nor registered. They can be used by any process. These are the **ephemeral ports**.



# Socket Addresses

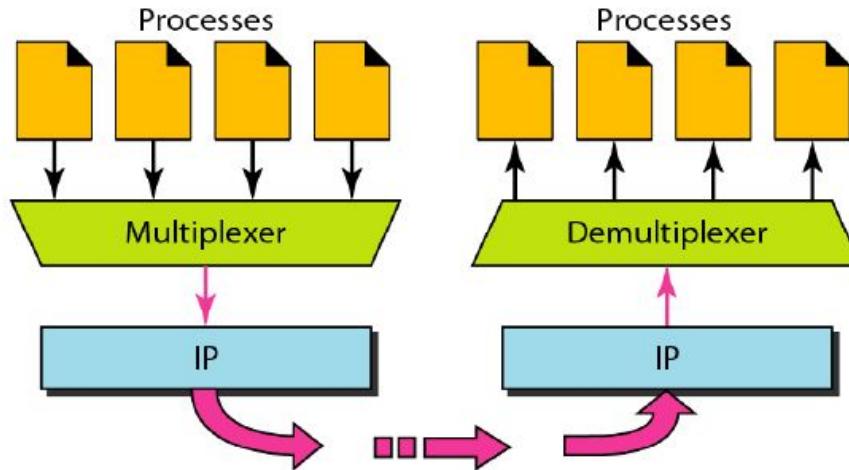
- Process-to-process delivery needs **two identifiers, IP address and the port number**, at each end to make a connection.
- **The combination of an IP address and a port number is called a socket address.**
- The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely.



- A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address.
- These four pieces of information are part of the IP header and the transport layer protocol header.
- The IP header contains the IP addresses; the UDP or TCP header contains the port numbers.

# Multiplexing and Demultiplexing

- The addressing mechanism allows multiplexing and demultiplexing by the transport layer, as shown in Figure below



## Multiplexing

- At the sender site, there may be several processes that need to send packets.
- However, there is only one transport layer protocol at any time.
- This is a many-to-one relationship and requires multiplexing.
- The protocol accepts messages from different processes, differentiated by their assigned port numbers.
- After adding the header, the transport layer passes the packet to the network layer.

# Multiplexing and Demultiplexing

## *Demultiplexing*

- At the **receiver site**, the **relationship is one-to-many** and requires **demultiplexing**.
- The **transport layer** receives datagrams from the **network layer**.
- After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the **port number**.

# Connectionless Versus Connection-Oriented Service

- A transport layer protocol can either be connectionless or connection-oriented.
- ***Connectionless Service***
- In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release.
- The packets are not numbered; they may be delayed or lost or may arrive out of sequence.
- There is no acknowledgment either. We will see shortly that one of the transport layer protocols in the Internet model, UDP, is connectionless.
- ***Connection Oriented Service***
- In a connection-oriented service, a connection is first established between the sender and the receiver, data are transferred.
- At the end, the connection is released.

# Reliable Versus Unreliable

- The **Transport layer** service can be **reliable** or **unreliable**.
- If the **application layer program** needs **reliability**, we use a reliable transport layer protocol by implementing flow and error control at the transport layer.
- This means a **slower** and more **complex service**.
- **TCP and SCTP** are connection oriented and reliable delivery protocol.
- On the other hand, if the **application program does not need reliability** because it uses its own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control (real-time applications), then an **unreliable protocol** can be used.
- **UDP is connectionless and unreliable delivery protocol.**

# Protocols at Transport Layer

The original **TCP/IP protocol suite** specifies **two protocols** for the transport layer: **UDP and TCP**.

## USER DATAGRAM PROTOCOL (UDP)

- The **User Datagram Protocol (UDP)** is called a **connectionless, unreliable transport protocol**.
- It does not add anything to the services of IP except to provide process-to process communication instead of host-to-host communication.
- Also, it performs very limited error checking.
- **UDP** is a **very simple protocol** using a **minimum of overhead**.
- If a process wants to send a small message and does not care much about reliability, it can use UDP.
- Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

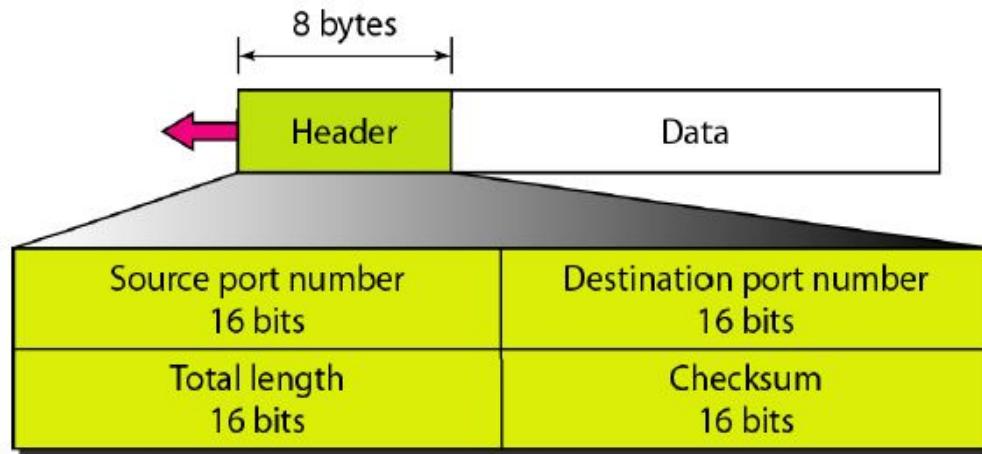
# Format of User Datagram

UDP packets, called user datagrams, have **a fixed-size header of 8 bytes**. The fields are as follows:

## i. Source port number.

- This is the port number used by the process running on the source host.
- It is **16 bits long**, which means that the port number can **range from 0 to 65,535**.
- If the source host is the **client** (a client sending a request), the **port number**, in most cases, is an **ephemeral port number** requested by the process and chosen by the **UDP software** running on the source host.
- If the source host is the **server** (a server sending a response), the **port number**, in most cases, is a **well-known port number**.

# Format of User Datagram



## ii. Destination port number.

- This is the **16 bits** long port number used by the process running on the **destination host**.
- If the destination host is the **server** (a client sending a request), the port number, in most cases, is a **well-known port number**.
- If the **destination host** is the client (a server sending a response), the port number, in most cases, is an **ephemeral port number**. In this case, the server copies the ephemeral port number it has received in the request packet.

# Format of User Datagram

## iii. Length.

- This is a 16-bit field that defines the **total length of the user datagram, header plus data**.
- The 16 bits can define a total length of 0 to 65,535 bytes.
- However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes.
- The length field in a UDP user datagram is actually not necessary. A user datagram is encapsulated in an IP datagram.
- There is a field in the IP datagram that defines the total length.
- There is another field in the IP datagram that defines the length of the header.
- So if we subtract the value of the second field from the first, we can deduce the length of a UDP datagram that is encapsulated in an IP datagram.
- **UDP length = IP length - IP header's length**
- **Checksum**. This field is used to detect errors over the entire user datagram (header plus data).

# UDP Operation

## Connectionless Services

- As mentioned previously, **UDP** provides a connectionless service.
- This means that each user datagram sent by UDP is an **independent** datagram.
- There is **no relationship** between the different user datagrams even if they are coming from the same source process and going to the same destination program.
- The user datagrams are **not numbered**.
- Also, there is **no connection establishment** and **no connection termination**, as is the case for TCP.
- This means that each user datagram can **travel on a different path**.

# UDP Operation

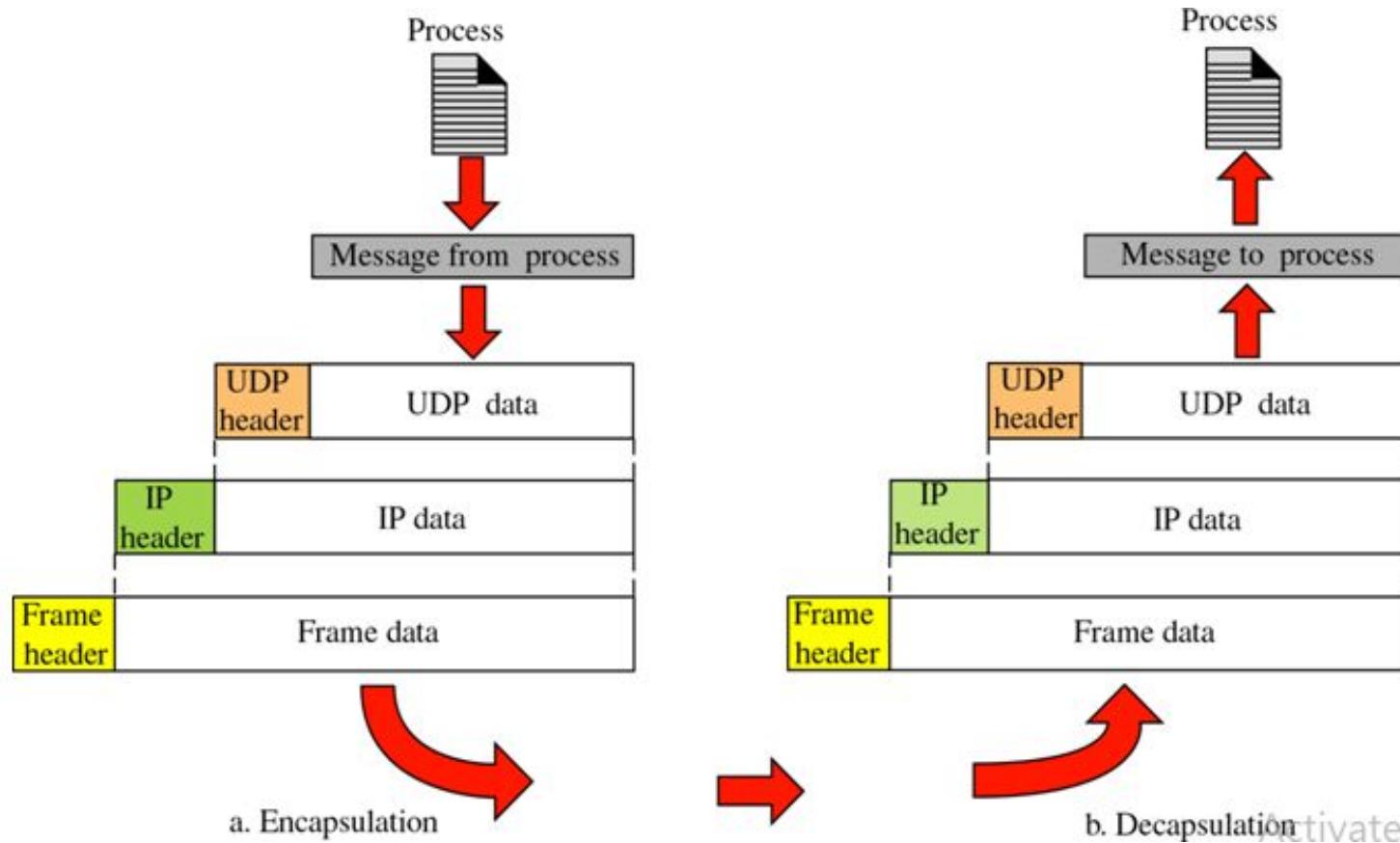
## Flow and Error Control

- UDP is a very simple, unreliable transport protocol.
- There is no flow control and hence no window mechanism.
- The receiver may overflow with incoming messages.
- There is no error control mechanism in UDP except for the checksum.
- This means that the sender does not know if a message has been lost or duplicated.
- When the receiver detects an error through the checksum, the user datagram is silently discarded.

## Encapsulation and Decapsulation

- To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.

# UDP Encapsulation and Decapsulation Process



# Use of UDP

- UDP is suitable for a process that requires **simple request-response communication with little concern for flow and error control.**
- It is not usually used for a process such as FTP (File Transfer Protocol) that needs to send bulk data.
- UDP is suitable for a **process with internal flow and error control mechanisms.** For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
- UDP is a suitable transport protocol for **multicasting.** Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for **management processes** such as SNMP (Simple Network Management Protocol).
- UDP is used for some **route updating protocols** such as Routing Information Protocol (RIP).

# TCP (Transmission Control Protocol)

- TCP (Transmission Control Protocol) is a *connection-oriented, reliable* transport protocol.
- It adds **connection-oriented** and **reliability** features to the services of IP.
- In addition, TCP uses **flow and error control** mechanisms at the transport level.
- Following section explains the **services offered by TCP** to the processes at the application layer.

## i. Process-to-Process Communication

- Like UDP, TCP provides **process-to-process communication using port numbers.**

# Well-known ports used by TCP

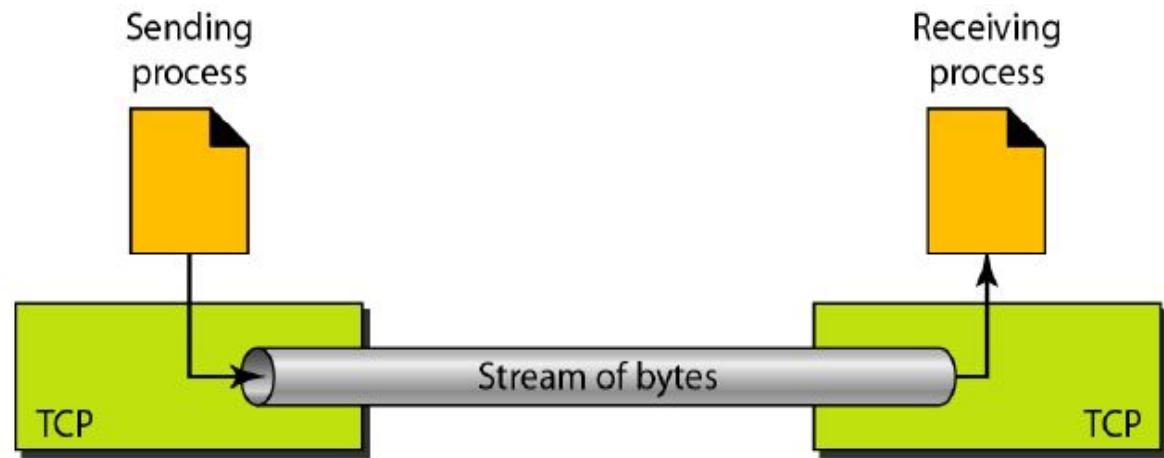
<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FIP, Data	File Transfer Protocol (data connection)
21	FIP, Control	File Transfer Protocol (control connection)
23	TELNET	Tenninal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

# TCP Services

## ii. Stream Delivery Service

- TCP, allows the sending process to deliver data as a **stream of bytes** and allows the receiving process to obtain data as a stream of bytes.
- TCP creates an environment in which the two processes seem to be connected by an **imaginary "tube"** that carries their data across the Internet.
- The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.
- The sending and the receiving processes may not write or read data at the same speed.
- There are two buffers, the **sending buffer** and the **receiving buffer**, one for each direction.

# TCP Services: Stream Delivery Service



# TCP Services

## iii. Segments

- TCP groups a number of bytes together into a **packet called a segment**.
- TCP adds a **header** to each **segment** (for control purposes) and delivers the segment to the IP layer for transmission.
- The segments are encapsulated in IP datagrams and transmitted. The segments are **not necessarily the same size**.

## iv. Full-Duplex Communication

- TCP offers **full-duplex service**, in which data can flow in both directions at the same time.
- Each TCP then has a **sending and receiving buffer**, and segments move in both directions.

# Connection-Oriented & Reliable Service

- TCP, unlike UDP, is a connection-oriented protocol.
- When a process at site A wants to send and receive data from another process at site B, the following three events occur:
  1. The two TCPs establish a connection between them.
  2. Data are exchanged in both directions.
  3. The connection is terminated.
- The TCP segment is encapsulated in an IP datagram.
- TCP is a **reliable** transport protocol. It uses an **acknowledgment mechanism** to check the safe and sound arrival of data.

# TCP Features

## A. Numbering System

TCP software keeps track of the segments being transmitted or received using two fields called the **sequence number** and the **acknowledgment number**.

### i. Byte Number

- TCP numbers all data bytes that are transmitted in a connection.
- Numbering is **independent** in each direction.
- When TCP receives bytes of data from a process, it stores them in the **sending buffer** and numbers them.
- The numbering does not necessarily start from 0.
- Instead, TCP generates a random number between 0 and  $2^{32} - 1$  for the number of the first byte.
- For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056.

# TCP Features

## ii. Sequence Number

- After the bytes have been numbered, TCP assigns a **sequence number** to each segment that is being sent.
- *The sequence number for each segment is the number of the first byte carried in that segment.*

**Example :** Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

## Solution

The following shows the sequence number for each segment:

- Segment 1 Sequence Number: 10,001 (range: 10,001 to 11,000)
- Segment 2 Sequence Number: 11,001 (range: 11,001 to 12,000)
- Segment 3 Sequence Number: 12,001 (range: 12,001 to 13,000)
- Segment 4 Sequence Number: 13,001 (range: 13,001 to 14,000)
- Segment 5 Sequence Number: 14,001 (range: 14,001 to 15,000)

# TCP Features

## iii. Acknowledgment Number

- The **acknowledgment number** defines the **number of the next byte that the receiver expects to receive**.
- In addition, the acknowledgment number is **cumulative**, which means that the receiver takes the number of the last byte that it has received, safe and sound, adds 1 to it, and announces this sum as the acknowledgment number.
- The term **cumulative** here means that if a receiver uses 5643 as an acknowledgment number, it has received all bytes from the beginning up to 5642.
- Note that this does not mean that the receiver has received 5642 bytes because the first byte number does not have to start from 0.

# TCP Features

## B. Flow Control

- TCP, unlike UDP, provides **flow control**.
- The receiver of the data controls the amount of data that are to be sent by the sender.
- This is done to prevent the receiver from being **overwhelmed with data**.
- The numbering system allows TCP to use a **byte-oriented flow control**.

## C. Error Control

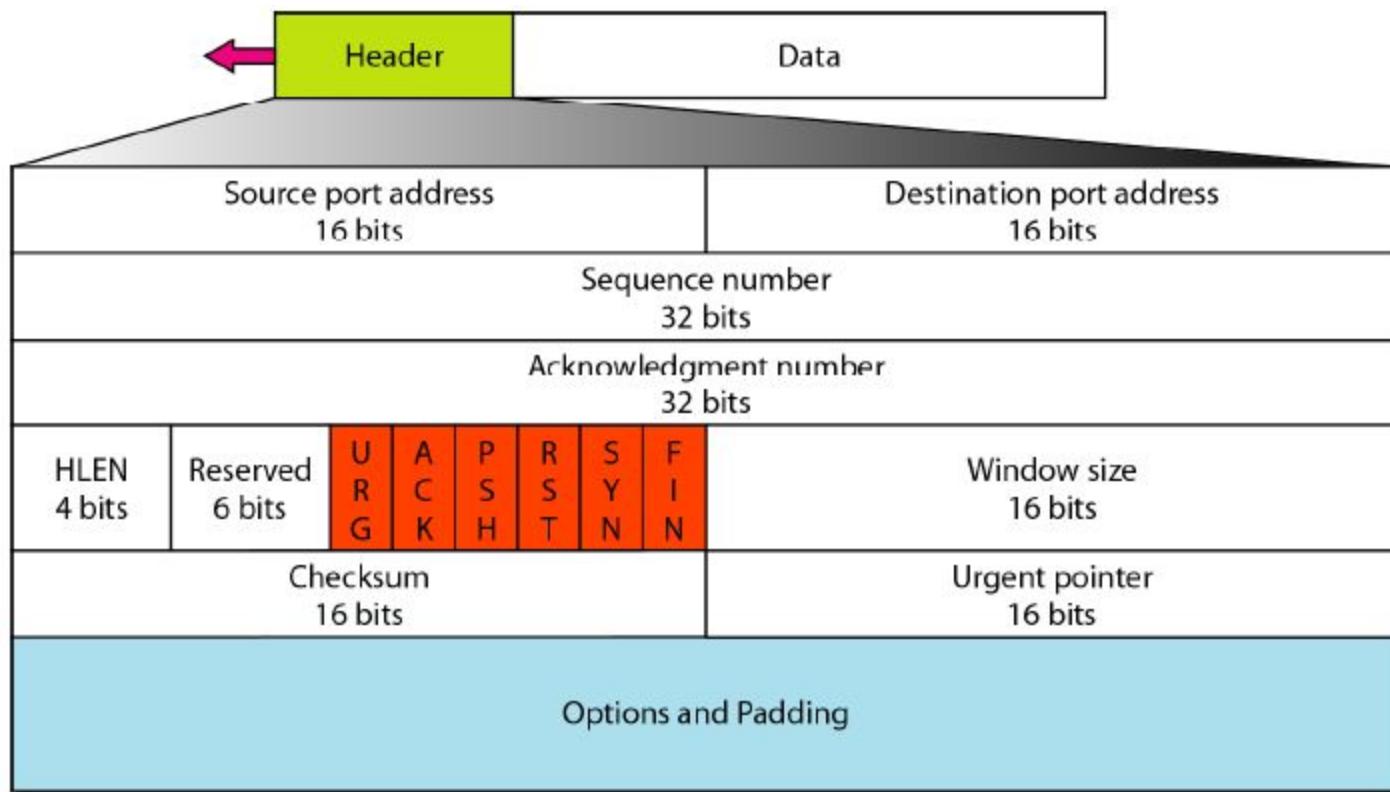
- To provide reliable service, TCP implements an error control mechanism.
- Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is **byte-oriented**.

## D. Congestion Control

- TCP, unlike UDP, takes into account congestion in the network.
- The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the **level of congestion in the network**.

# TCP Segment Format

- The TCP segment consists of a **20 to 60 byte header**, followed by **data** from the application program.
- The header is **20 bytes** if there are no options and up to 60 bytes if it contains options.



# TCP Segment Format

## Source port address.

- This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.
- This serves the same purpose as the source port address in the UDP header.

## Destination port address.

- This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.
- This serves the same purpose as the destination port address in the UDP header.

## Sequence number.

- This **32-bit field** defines the number assigned to the **first byte** of data contained in this segment.
- During connection establishment, each party uses a **random number generator** to create an **initial sequence number** (ISN), which is usually different in each direction.

# TCP Segment Format

## Acknowledgment number.

- This **32-bit field** defines the byte number that the receiver of the segment is expecting to receive from the other party.
- If the receiver of the segment has successfully received **byte number x** from the other party, it defines **x + 1** as the **acknowledgment number**.
- Acknowledgment and data can be **piggybacked together**.

## Header length.

- This 4-bit field indicates the number of 4-byte words in the TCP header.
- The length of the header can be between 20 and 60 bytes.
- Therefore, the value of this field can be between **5** ( $5 \times 4 = 20$ ) and **15** ( $15 \times 4 = 60$ ).

## Reserved.

- This is a **6-bit field** reserved for future use.

# TCP Segment Format

## Control Flags

- This field defines **6 different control bits** or flags as shown in below. One or more of these bits can be set at a time.

<i>Flag</i>	<i>Description</i>
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

## Window size.

- This field defines the size of the window, in bytes, that the **other party must maintain**.
- Note that the length of this field is **16 bits**, which means that the maximum size of the window is 65,535 bytes.
- This value is normally referred to as the **receiving window (rwnd)** and is **determined by the receiver**.
- The sender must obey the dictation of the receiver in this case.

# TCP Segment Format

## Checksum.

- This **16-bit field** contains the checksum for **header security**.

## Urgent pointer.

- This **16-bit field**, which is valid only if the **urgent flag** is set, is used when the **segment contains urgent data**.
- It **defines the number** that must be **added to the sequence number** to obtain the number of the **last urgent byte** in the data section of the segment.

## Options.

- There can be up to **40 bytes** of optional information in the TCP header.

# A TCP Connection

- TCP is **connection-oriented** protocol.
- A connection-oriented transport protocol establishes a **virtual path between the source and destination**.
- All the segments belonging to a message are then sent over this **virtual path**.
- Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.
- In TCP, **connection-oriented transmission** requires **three phases**:
  1. *Connection establishment,*
  2. *Data transfer, and*
  3. *Connection termination.*

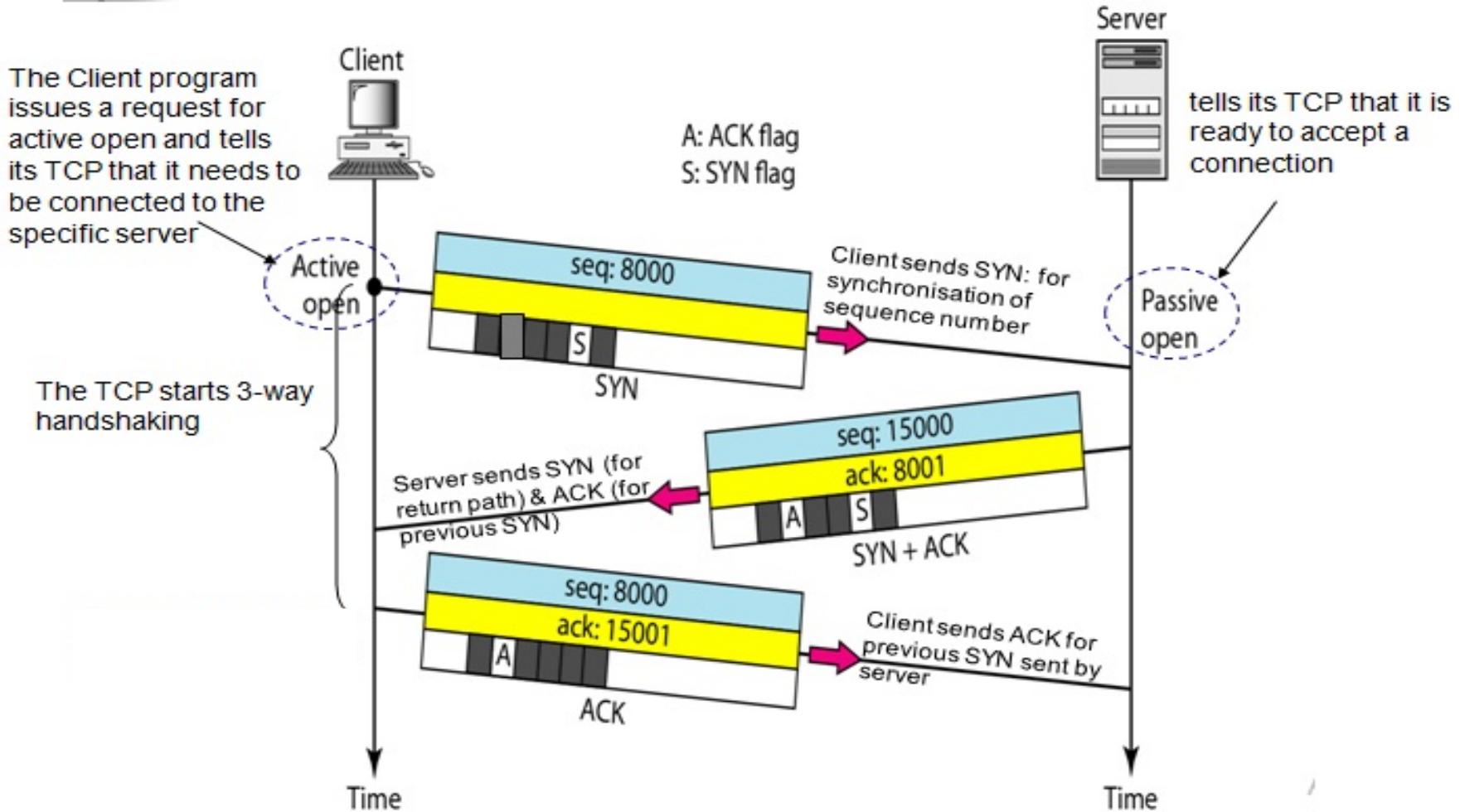
# 1. Connection Establishment

- TCP transmits data in **full-duplex mode**.
- When two TCPs in two machines are connected, they are able to send segments to each other **simultaneously**.
- This implies that each party must initialize communication and get approval from the other party before any data are transferred.
- The **connection establishment** in TCP is called **three-way handshaking**.
- For **example**, an application program, called the **client**, wants to make a connection with another application program, called the **server**, using TCP as the transport layer protocol.
- The process starts with the **server**. The server program tells its TCP that it is **ready to accept a connection**.

# Connection Establishment

- This is called a request for a ***passive open***.
- Although the server TCP is ready to accept any connection from any machine in the world, it cannot make the connection itself.
- The client program issues a request for an ***active open***.
- A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server.
- TCP can now start the **three-way handshaking process** as shown in Figure on next slide.
- To show the process, we use **two time lines**: one at each site.
- Each segment has value for all its header fields and perhaps for some of its option fields, too.
- We show the sequence number, the acknowledgment number, the control flags (only those that are set), and the window size, if not empty.

# Connection Establishment using three-way handshaking



# Connection establishment using three-way handshaking

- The **three steps** in this phase are as follows:

## Step 1:

- The **client** sends the **first segment**, a **SYN segment**, in which only the **SYN flag** is **set**.
- This segment is for synchronization of sequence numbers.
- A **SYN segment consumes one sequence number**.
- When the **data transfer** starts, the **sequence number** is **incremented by 1**.
- We can say that the SYN segment carries no real data, but we can think of it as containing 1 imaginary byte.
- Thus a SYN segment cannot carry data, but it consumes one sequence number.

# Connection establishment using three-way handshaking

## Step 2:

- The **Server** sends the **second segment**, a **SYN +ACK** segment, with **2 flag bits set: SYN and ACK**.
- This segment has a **dual purpose**.
- It is a **SYN** segment for communication in the other direction and serves as the **acknowledgment** for the **SYN** segment.
- It **consumes one sequence number**.
- A **SYN +ACK segment cannot carry data, but does consume one sequence number**.

# Connection establishment using three-way handshaking

## Step 3:

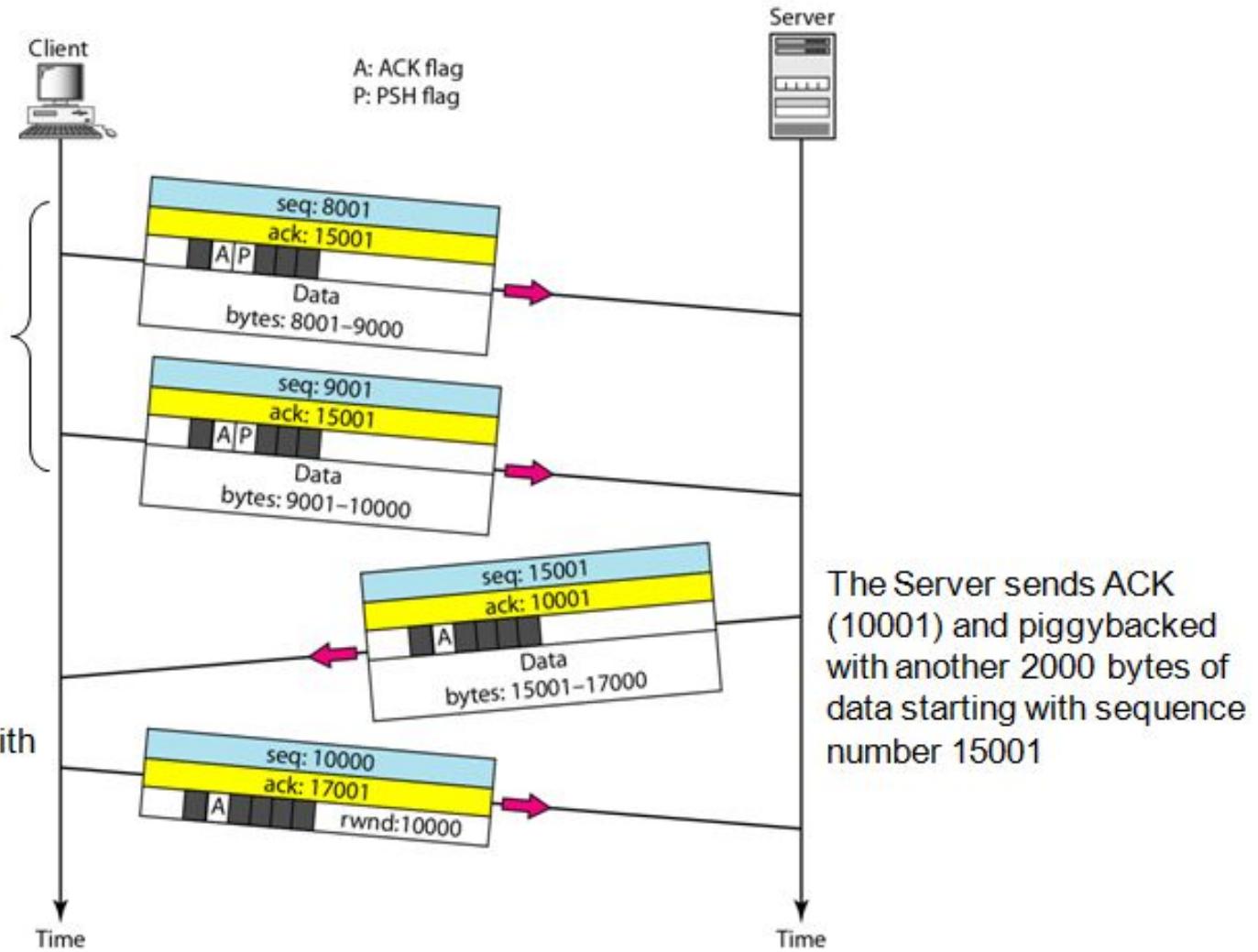
- The **client sends the third segment.**
- This is just an **ACK segment.**
- It **acknowledges** the receipt of the second segment with the ACK flag and acknowledgment number field.
- Note that the sequence number in this segment is the same as the one in the SYN segment; **the ACK segment does not consume any sequence numbers.**
- An **ACK segment, if carrying no data, consumes no sequence number.**

# Data Transfer

- After **connection is established**, **bidirectional data transfer** can take place.
- The client and server can both send data and acknowledgments.
- The acknowledgment is **piggybacked** with the data.
- Figure on next slide shows an example. In this example, after connection is established (not shown in the figure), the **client** sends **2000 bytes** of data in **two segments**.
- The **server** then sends **2000 bytes** in **one segment**.
- The client sends one more segment.
- The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.
- Note the values of the sequence and acknowledgment numbers.
- The data segments sent by the client have the **PSH** (push) **flag set** so that the server TCP knows to deliver data to the server process as soon as they are received.
- The segment from the server, on the other hand, does not set the push flag.

# Data Transfer

The Client sends two 2000 bytes of data in 2 segments at the same time acknowledged previous sending by server: 15001



# Pushing Data

- The **sending TCP uses a buffer** to store the stream of data coming from the sending application program.
- The **receiving TCP also buffers the data** when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP.
- This type of **flexibility** increases the **efficiency of TCP**.
- However, on occasion the application program has no need for this flexibility.
- For **example**, consider an application program that communicates interactively with another application program on the other end.
- The application program on one site wants to send a keystroke to the application at the other site and receive an immediate response.
- Delayed transmission and delayed delivery of data may not be acceptable by the application program.

# Pushing Data

- TCP can handle such a situation.
- The application program at the **sending site** can request a ***push*** operation.
- This means that the **sending TCP** must **not wait for the window to be filled**.
- It must **create a segment and send it immediately**.
- The sending TCP must also set the **push bit (PSH)** to let the **receiving TCP** know that the segment includes data that must be delivered to the receiving application program **as soon as possible** and not to wait for more data to come.

# Urgent Data

- At some occasion an application program needs to send ***urgent bytes***.
- This means that the sending application program wants a piece of data to be read out of order by the receiving application program.
- As an example, suppose that the sending application program is sending data to be processed by the receiving application program.
- When the result of processing comes back, the sending application program finds that everything is wrong.
- It wants to abort the process, but it has already sent a huge amount of data.
- If it issues an abort command (control +C), these two characters will be stored at the end of the receiving TCP buffer.
- It will be delivered to the receiving application program after all the data have been processed.

# Urgent Data

- The solution is to send a segment with the **URG bit set**.
- The sending application program tells the sending TCP that the piece of data is **urgent**.
- The sending TCP creates a segment and **inserts the urgent data at the beginning of the segment**.
- The rest of the segment can contain normal data from the buffer.
- **The urgent pointer field in the header defines the end of the urgent data and the start of normal data**.
- When the **receiving TCP** receives a segment with the **URG bit set**, it extracts the urgent data from the segment, using the value of the urgent pointer, and delivers them, **out of order**, to the receiving application program.

# Connection Termination in TCP

- Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client.
- Most implementations today allow two options for connection termination:
  - i. *Three-way handshaking* for full-close
  - ii. *Four-way handshaking* for half-close
- Most implementations today allow *three-way handshaking* for connection termination.

## Step 1:

- In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set.
- Note that a FIN segment can include the last chunk of data sent by the client, or it can be just a control segment.
- If it is only a control segment, it consumes only one sequence number.

# Connection Termination

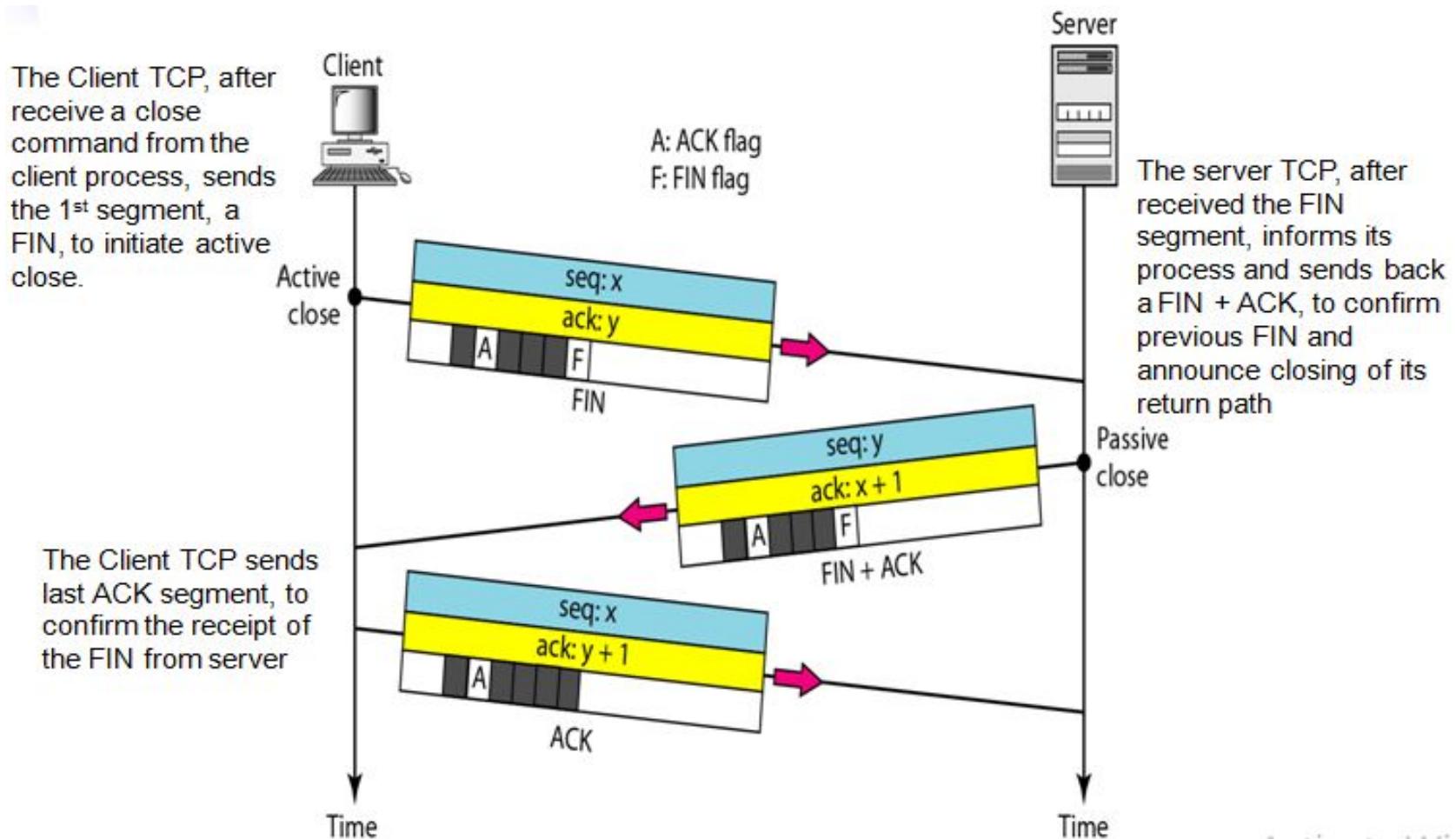
## Step 2.

- The **server TCP**, after **receiving the FIN segment**, informs its process of the situation and sends the **second segment**, a **FIN +ACK** segment, to **confirm the receipt of the FIN segment** from the client and at the same time to **announce the closing of the connection** in the other direction.
- This **segment can also contain the last chunk of data** from the server. If it does not carry data, it **consumes only one sequence number**.

## Step 3.

- The **client TCP sends the last segment**, an **ACK segment**, to confirm the receipt of the **FIN segment** from the **TCP server**.
- This segment contains the acknowledgment number, which is 1 plus the sequence number received in the **FIN segment** from the **server**.
- This segment cannot carry data and **consumes no sequence numbers**.

# Connection termination using three-way handshaking



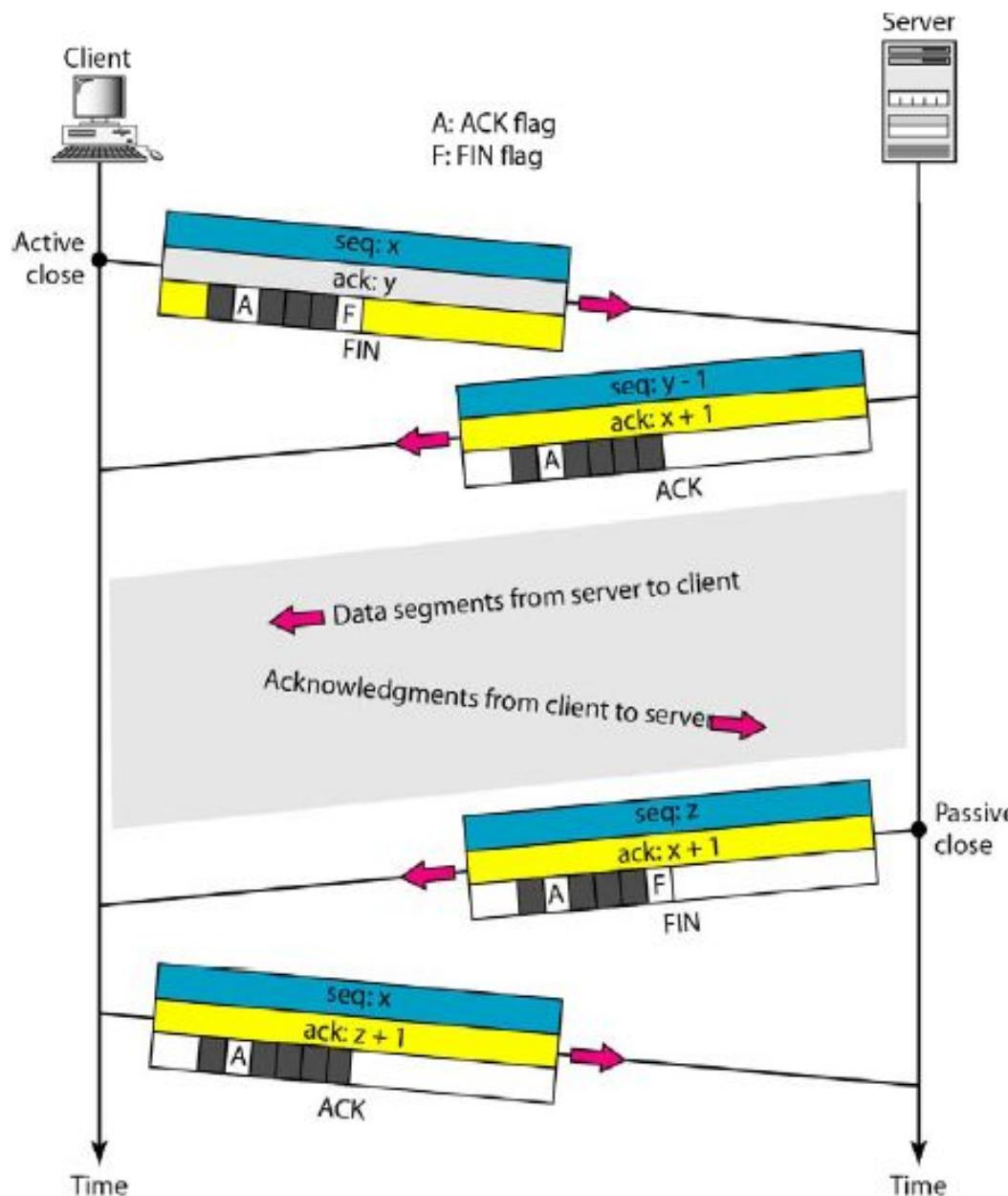
# Half-Close

- In TCP, ***one end can stop sending data while still receiving data.*** This is called a **half-close**.
- Although either end can issue a half-close, it is **normally initiated by the client**.
- It can occur when the server needs all the data before processing can begin. A good **example** is sorting.
- When the client sends data to the server to be sorted, the server needs to receive all the data before sorting can start.
- This means the **client**, after sending all the data, **can close the connection in the outbound direction**.
- However, **the inbound direction must remain open to receive the sorted data**.
- The server, after receiving the data, still needs time for sorting; its outbound direction must remain open.

# Half-Close

- The client half-closes the connection by sending a **FIN segment**.
- The server accepts the **half-close** by sending the **ACK segment**.
- The data transfer from the client to the server stops.
- The server, however, can still send data.
- When the server has sent all the processed data, it sends a **FIN segment**, which is acknowledged by an **ACK** from the client.
- After **half-closing** of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server.
- The client cannot send any more data to the server.
- Although the client has received sequence number  $y - 1$  and is expecting  $y$ , the server sequence number is still  $y - 1$ .
- When the connection finally closes, the sequence number of the last ACK segment is still  $x$ , because no sequence numbers are consumed during data transfer in that direction.

# Half-Close



# Flow Control

- TCP uses a **sliding window mechanism**, to handle **flow control**.
- The **sliding window protocol** used by TCP, however, is something **between** the **Go-Back-N** and **Selective Repeat** sliding window.
- The sliding window protocol in TCP looks like the **Go-Back-N protocol** because it **does not use NAKs**; it looks like **Selective Repeat** because the **receiver holds the out-of-order segments** until the missing ones arrive.
- There are **two big differences** between this sliding window and the one we used at the data link layer.
- First, the **sliding window of TCP is byte-oriented**; the one we discussed in the **data link layer is frame-oriented**.
- Second, the **TCP's sliding window is of variable size**; the one we discussed in the **data link layer was of fixed size**.

# Flow Control

- The **window spans a portion of the buffer** containing bytes received from the process.
- The **bytes inside the window** are the bytes that **can be in transit**; they **can be sent** without worrying about acknowledgment.
- The **imaginary window** has two walls: one **left** and one **right**.
- The **window is *opened*, *closed*, or *shrunk***. These **three activities**, as we will see, are in the **control of the receiver** (and **depend on congestion** in the network), not the sender.



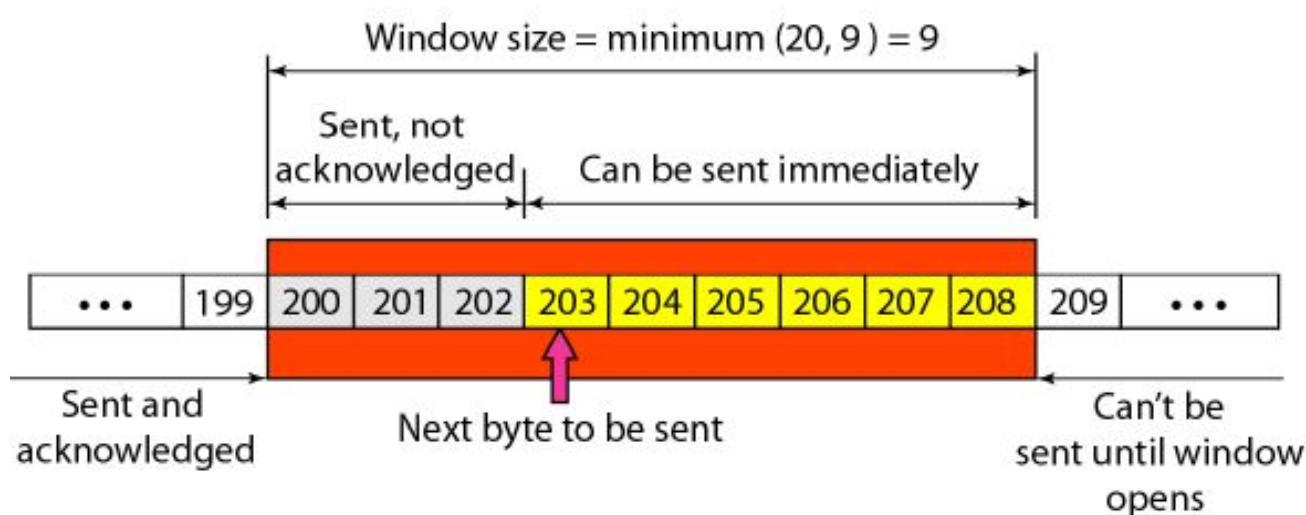
# Flow Control

- **Opening a window** means **moving the right wall to the right**. This allows more new bytes in the buffer that are eligible for sending.
- **Closing the window** means **moving the left wall to the right**. This means that some bytes have been acknowledged and the sender need not worry about them anymore.
- **Shrinking the window** means **moving the right wall to the left**. It means revoking the eligibility of some bytes for sending.
- The size of the window at one end is determined by the **lesser of two values: *receiver window (rwnd)* or *congestion window (cwnd)***.
- The **receiver window** is the value advertised by the opposite end in a segment containing acknowledgment. It is the number of bytes the other end can accept before its buffer overflows and data are discarded.
- The **congestion window** is a value **determined by the network** to avoid congestion.

# Flow Control

## Example of a Sliding window.

- The **sender** has sent bytes up to **202**. Let **cwnd** is 20.
- The **receiver** has sent an acknowledgment number of 200 with an **rwnd** of 9 bytes.
- The size of the **sender window** is the **minimum of rwnd and cwnd**, or 9 bytes.
- Bytes **200** to **202** are sent, but not acknowledged.
- Bytes **203** to **208** can be sent without worrying about acknowledgment.
- Bytes **209** and above cannot be sent.



# Error Control

- TCP is a **reliable** transport layer protocol.
- This means that an **application program** that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order, **without error**, and **without any part lost or duplicated**.
- TCP provides **reliability** using **error control**.
- Error control includes mechanisms for **detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments**.
- Error control also includes a **mechanism for correcting errors** after they are detected.
- Error detection and correction in TCP is achieved through the use of **three simple tools**:
  - ***Checksum, Acknowledgment and Time-out.***

# Error Control

## Checksum

- Each **segment** includes a **checksum field** which is used to check for a **corrupted segment**.
- If the **segment** is **corrupted**, it is **discarded** by the destination TCP and is considered as lost.
- TCP uses a **16-bit checksum** that is mandatory in every segment.

## Acknowledgment

- TCP uses **acknowledgments** to confirm the receipt of **data segments**.
- **Control segments** that carry **no data** but **consume a sequence number** are also **acknowledged**.
- **ACK segments** are **never acknowledged**.
- **ACK segments** do not consume sequence numbers and are **not acknowledged**.

# Retransmission

- The **heart** of the **error control mechanism** is the **retransmission of segments**.
- When a **segment is corrupted, lost, or delayed**, it is **retransmitted**.
- In modern implementations, a **segment** is **retransmitted** on **two occasions**:
  - when a **retransmission timer expires** or
  - when the **sender receives three duplicate ACKs**.
- Note that **no retransmission occurs for segments that do not consume sequence numbers**.
- In particular, there is **no transmission** for an **ACK segment**.

# Retransmission After RTO

- A recent implementation of TCP maintains one **retransmission time-out (RTO) timer** for all **outstanding** (sent, but not acknowledged) segments.
- When the **timer matures, the earliest outstanding segment is retransmitted** even though lack of a received ACK can be due to a delayed segment, a delayed ACK, or a lost acknowledgment.
- Note that no time-out timer is set for a segment that carries only an acknowledgment, which means that no such segment is resent.
- The value of **RTO is dynamic** in TCP and is **updated based on the round-trip time (RTT) of segments.**
- An **RTT** is the time needed for a segment to reach a destination and for an acknowledgment to be received.

# Retransmission After Three Duplicate ACK Segments

- The previous rule about retransmission of a segment is **sufficient** if the **value of RTO is not very large**.
- Sometimes, however, **one segment is lost** and the receiver receives so many **out-of-order segments** that they **cannot be saved (limited buffer size)**.
- To **alleviate** this situation, most implementations today follow the **three-duplicate-ACKs rule** and retransmit the missing segment immediately.
- This feature is referred to as **fast retransmission**, which we will see in an example shortly.

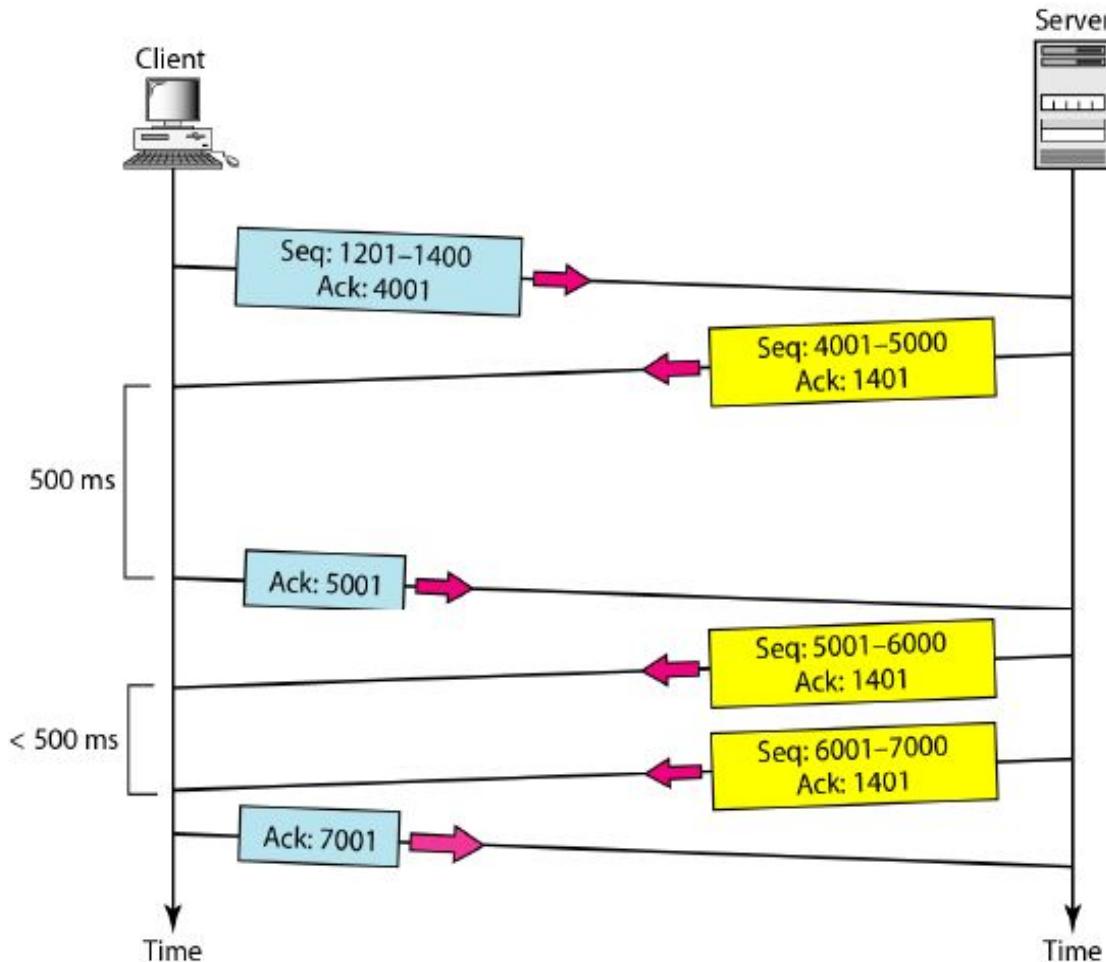
# Out-of-Order Segments

- When a **segment** is **delayed**, **lost**, or **discarded**, the segments following that segment arrive **out of order**.
- Originally, TCP was designed to discard all out-of-order segments, resulting in the retransmission of the missing segment and the following segments.
- Most implementations today do not discard the out-of-order segments.
- They **store them temporarily** and **flag them as out-of-order segments** until the missing segment arrives.
- Note, however, that the out-of-order segments are not delivered to the process.
- **TCP guarantees** that data are delivered to the process **in order**.

# Some Scenarios of TCP Operation:Normal Operation

- The **client TCP** sends one segment; the **server TCP** sends three.
- There are data to be sent, so the segment displays the next byte expected.
- When the client receives the first segment from the server, it does not have any more data to send; it sends only an ACK segment.
- However, the acknowledgment needs to be delayed for 500 ms to see if any more segments arrive.
- When the **timer matures**, it **triggers an acknowledgment**.
- When the next segment arrives, another acknowledgment timer is set.
- However, before it matures, the third segment arrives.
- The arrival of the third segment triggers another acknowledgment(**cumulative ACK**).

# Normal Operation



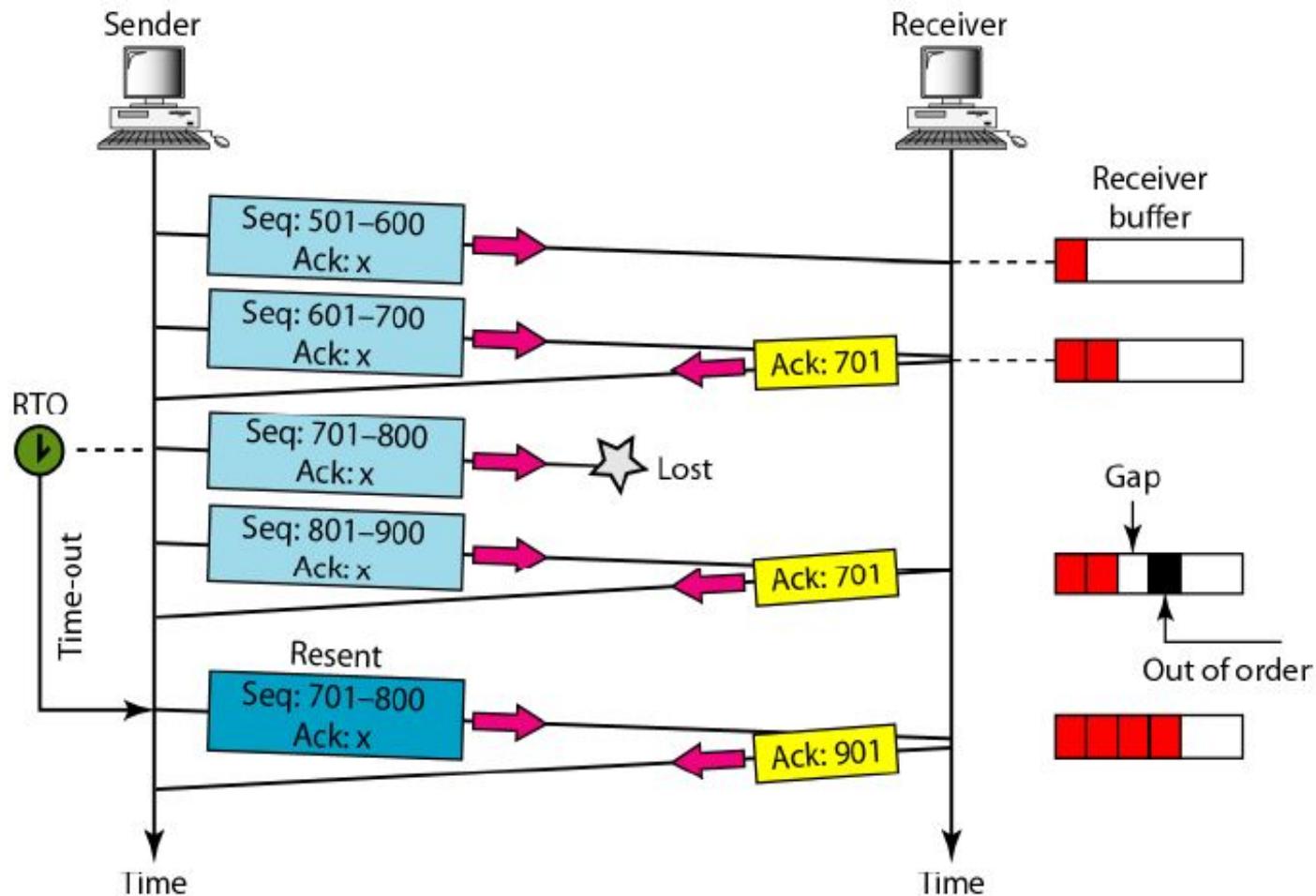
# Lost Segment

- A lost segment and a corrupted segment are treated the same way by the receiver.
- A lost segment is discarded somewhere in the network; a corrupted segment is discarded by the receiver itself. Both are considered lost.
- We are assuming that data transfer is unidirectional: one site is sending, the other is receiving.
- In our scenario, the sender sends segments 1 and 2, which are acknowledged immediately by an ACK.
- Segment 3, however, is lost.
- The receiver receives segment 4, which is out of order.
- The receiver stores the data in the segment in its buffer but leaves a gap to indicate that there is no continuity in the data.

# Lost Segment

- The receiver immediately sends an acknowledgment to the sender, displaying the next byte it expects.
- Note that the receiver stores bytes 801 to 900, but never delivers these bytes to the application until the gap is filled.
- We have shown the timer for the earliest outstanding segment.
- The timer for this definitely runs out because the receiver never sends an acknowledgment for lost or out of-order segments.
- When the timer matures, the sending TCP resends segment 3, which arrives this time and is acknowledged properly.
- Note that the value in the second and third acknowledgments differs according to the corresponding rule.

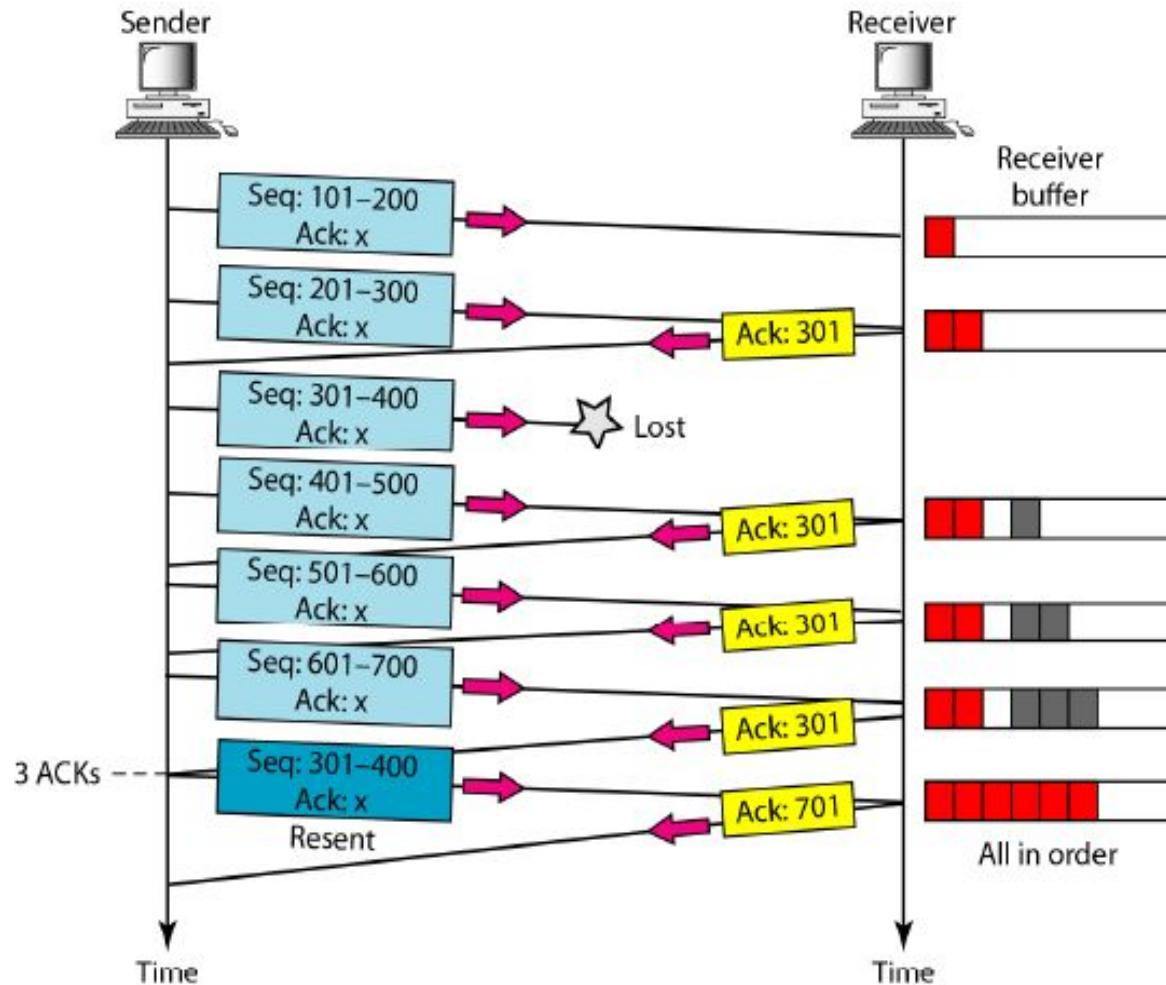
# Lost Segment



# Fast Retransmission

- This scenario is the same as the second except that the RTO has a higher value.
- When the receiver receives the fourth, fifth, and sixth segments, it triggers an acknowledgment.
- The sender receives four acknowledgments with the same value (three duplicates).
- Although the timer for segment 3 has not matured yet, the fast transmission requires that segment 3, the segment that is expected by all these acknowledgments, be resent immediately.
- Note that only one segment is retransmitted although four segments are not acknowledged.
- When the sender receives the retransmitted ACK, it knows that the four segments are safe and sound because acknowledgment is cumulative.

# Fast Retransmission



# Lecture 6.2

## Transport Layer: Congestion Control

Dr. Vandana Kushwaha

Department of Computer Science  
Institute of Science, BHU, Varanasi

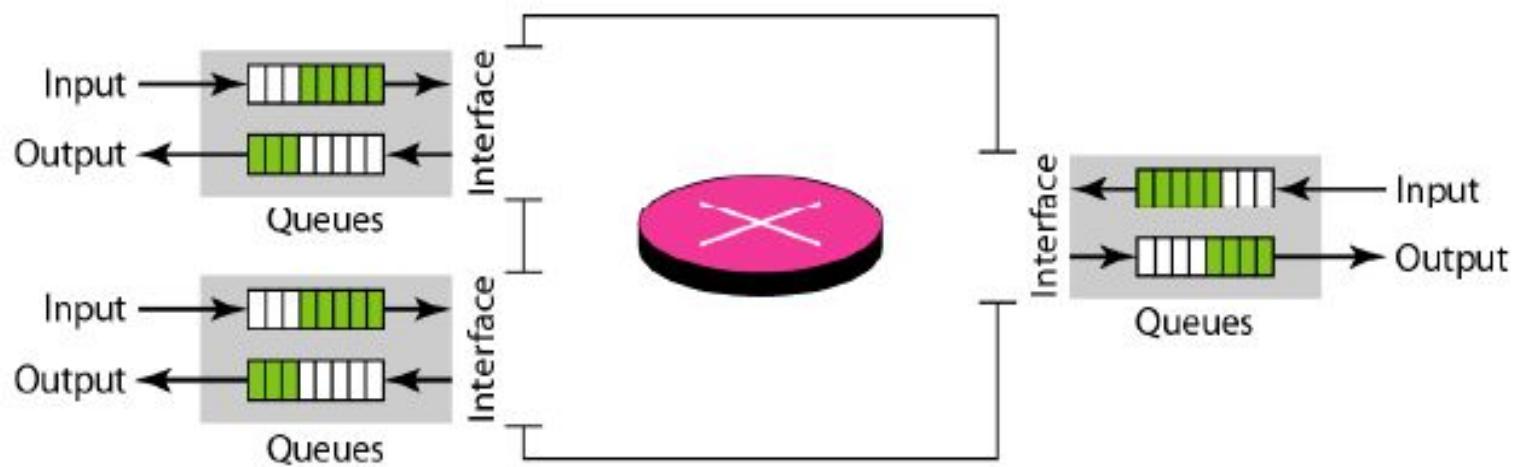
# Network Congestion

- Network congestion is one of the serious issues in communication network.
- There are several side effects of network congestion which severely affects the network performance and quality of service.
- Congestion is an important issue in a packet-switched network.
- Congestion in a network may occur if the load on the network (*the number of packets sent to the network*) is greater than the capacity of the network (*the number of packets a network can handle*).
- Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.
- Congestion in a network or internetwork occurs because routers and switches have queues- buffers that hold the packets before and after processing.

# Network Congestion

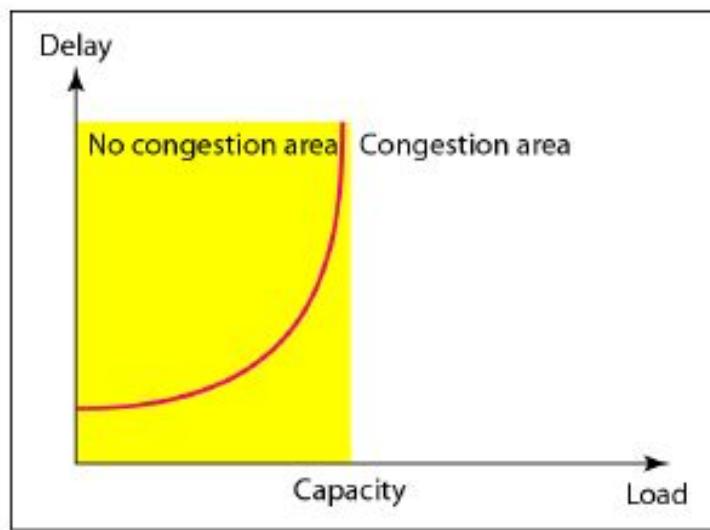
- A **router**, for **example**, has an **input queue** and an **output queue** for each interface. When a packet arrives at the incoming interface, it undergoes **three steps** before departing.
  1. The **packet** is put at the end of the **input queue** while waiting to be checked.
  2. The processing module of the router removes the packet from the input queue once it reaches the front of the queue and uses its **routing table** and the **destination address** to find the **route**.
  3. The **packet** is put in the appropriate **output queue** and **waits its turn to be sent**.
- If the **rate of packet arrival** is **higher than the packet processing rate**, the **input queues** become **longer and longer**.
- If the **packet departure rate** is **less than the packet processing rate**, the **output queues** become **longer and longer**.

# Queues in a Router

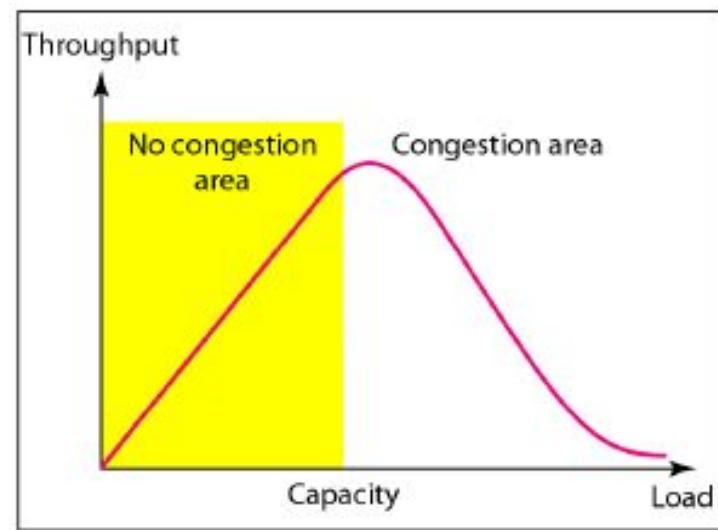


# Network Performance

- Congestion control involves two factors that measure the **performance** of a network: **delay** and **throughput**.
- Figure below shows these two performance measures as function of **load**.



a. Delay as a function of load



b. Throughput as a function of load

# Network Performance: Delay versus Load

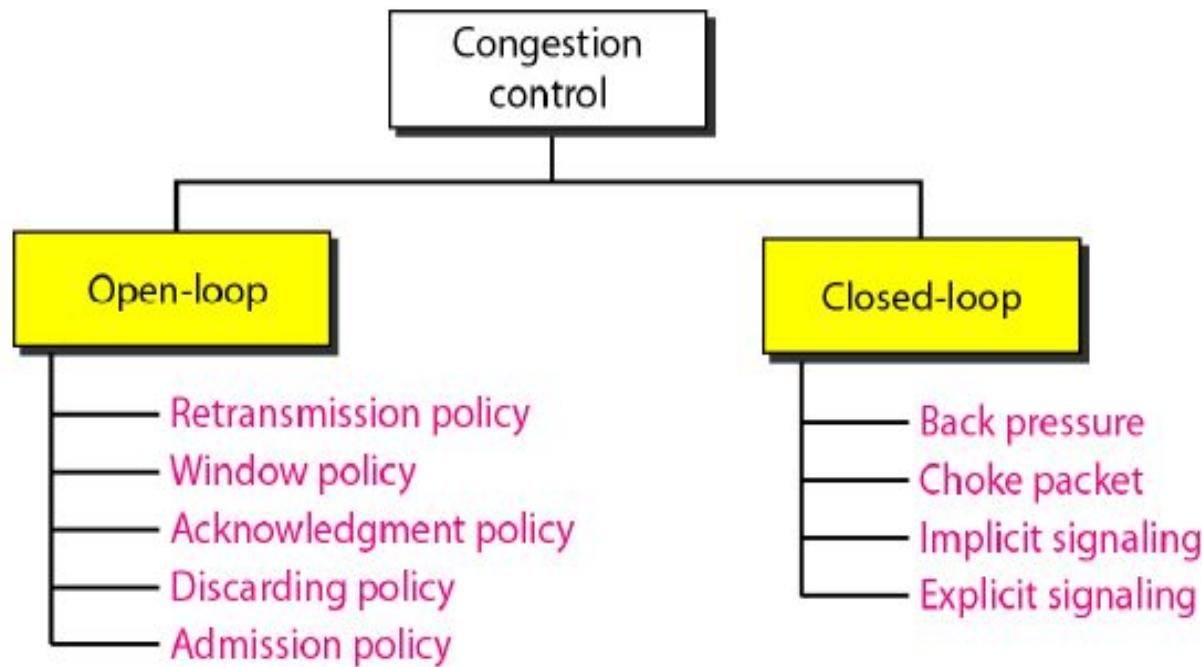
- When the **load** is much **less than** the **capacity** of the **network**, the **delay** is at a **minimum**.
- This **minimum delay** is composed of **propagation delay** and **processing delay**, both of which are **negligible**.
- However, when the **load reaches** the **network capacity**, the **delay increases sharply** because we now need to add the **waiting time in the queues** (for all routers in the path) to the **total delay**.
- The **delay becomes infinite** when the **load is greater than the capacity**.
- Due to **infinite delay**; the **queues become longer and longer**.
- **Delay has a negative effect on the load** and consequently the **congestion occurs**.
- When a **packet is delayed**, the **source, not receiving the acknowledgment, retransmits the packet**, which makes the **delay, and the congestion, worse**.

# Network Performance: Throughput versus Load

- Throughput in a network is defined as the number of packets passing through the network in a unit of time.
- When the load is below the capacity of the network, the throughput increases proportionally with the load .
- We expect the throughput to remain constant after the load reaches the capacity, but instead the throughput declines sharply. The reason is the discarding of packets by the routers.
- When the load exceeds the capacity, the queues become full and the routers have to discard some packets.
- Discarding packets does not reduce the number of packets in the network because the sources retransmit the packets, using time-out mechanisms, when the packets do not reach the destinations.

# Congestion Control

- Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.
- In general, we can divide **congestion control mechanisms** into two broad categories:
  - *Open-loop congestion control* (prevention)
  - *Closed-loop congestion control* (removal)



# Open-Loop Congestion Control

- In **Open-loop congestion control**, policies are applied to *prevent congestion before it happens*.
- In these mechanisms, **congestion control is handled by either the source or the destination**.

## 1. Retransmission Policy

- If the **sender** feels that a sent packet is **lost** or **corrupted**, the packet needs to be **retransmitted**.
- Retransmission in general **may increase congestion** in the network as it **increases the network load**.
- The **retransmission policy** and the **retransmission timers** must be designed to **optimize efficiency (minimize load)** and at the same time **prevent congestion**.
- For example, the **retransmission policy** used by **TCP** is designed to prevent or alleviate congestion.

# Open-Loop Congestion Control

## 2. Window Policy

- The **type of window** at the **sender** may also **affect congestion**.
- The **Selective Repeat window** is **better** than the **Go-Back-N window** for **congestion control**.
- In the **Go-Back-N window**, when the timer for a packet times out, **several packets may be resent**, although some may have arrived safe and sound at the receiver.
- This **duplication** may make the **congestion worse**.
- The **Selective Repeat window**, on the other hand, tries to **send the specific packets** that have been **lost or corrupted**.

# Open-Loop Congestion Control

## 3. Acknowledgment Policy

- The **acknowledgment policy** imposed by the receiver may also **affect congestion**.
- If the **receiver** does **not acknowledge every packet** it receives, it may slow down the sender and help prevent congestion.
- A receiver may decide to acknowledge only  $N$  packets at a time which is called **cumulative acknowledgement policy**.
- We need to know that the **acknowledgments** are also **part of the load** in a **network**.
- **Sending fewer acknowledgments** means imposing **fewer loads on the network**.

# Open-Loop Congestion Control

## 4. Discarding Policy

- A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission.
- For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.

## 5. Admission Policy

- An admission policy can also prevent congestion in virtual-circuit networks.
- Routers first check the resource requirement of a flow before admitting it to the network.
- A router can deny establishing a virtual circuit connection if there is congestion in the network or if there is a possibility of future congestion.

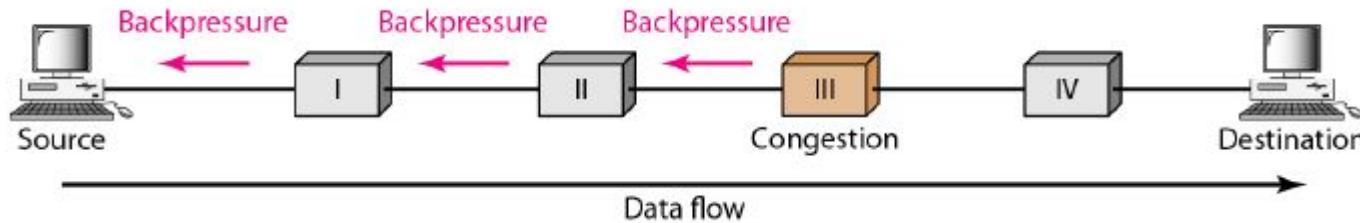
# Closed-Loop Congestion Control

**Closed-loop congestion control** mechanisms try to **alleviate congestion** after it happens.

## 1. Backpressure

- The technique of ***backpressure*** refers to a **congestion control** mechanism in which a **congested node stops receiving data** from the **immediate upstream node** or nodes.
- This may cause the **upstream node or nodes to become congested**, and they, in turn, **reject data from their upstream nodes or nodes**. And so on.
- **Backpressure is a node-to-node congestion control** that starts with a node and **propagates, in the opposite direction of data flow, to the source**.
- The **backpressure technique** can be **applied** only to **virtual circuit networks**, in which **each node knows the upstream node** from which a flow of data is coming.

# Closed-Loop Congestion Control



- Node III has more input data than it can handle.
- It drops some packets in its input buffer and informs node II to slow down.
- Node II, in turn, may be congested because it is slowing down the output flow of data.
- If node II is congested, it informs node I to slow down, which in turn may create congestion.
- If so, node I informs the source of data to slow down.
- This, in time, alleviates the congestion.
- Note that the pressure on node III is moved backward to the source to remove the congestion.

# Closed-Loop Congestion Control

- Backpressure technique was implemented in the first **virtual-circuit network**, **X.25**.
- The technique **cannot be implemented in a datagram network** because in this type of network, a node (router) does not have the slightest knowledge of the **upstream router**.

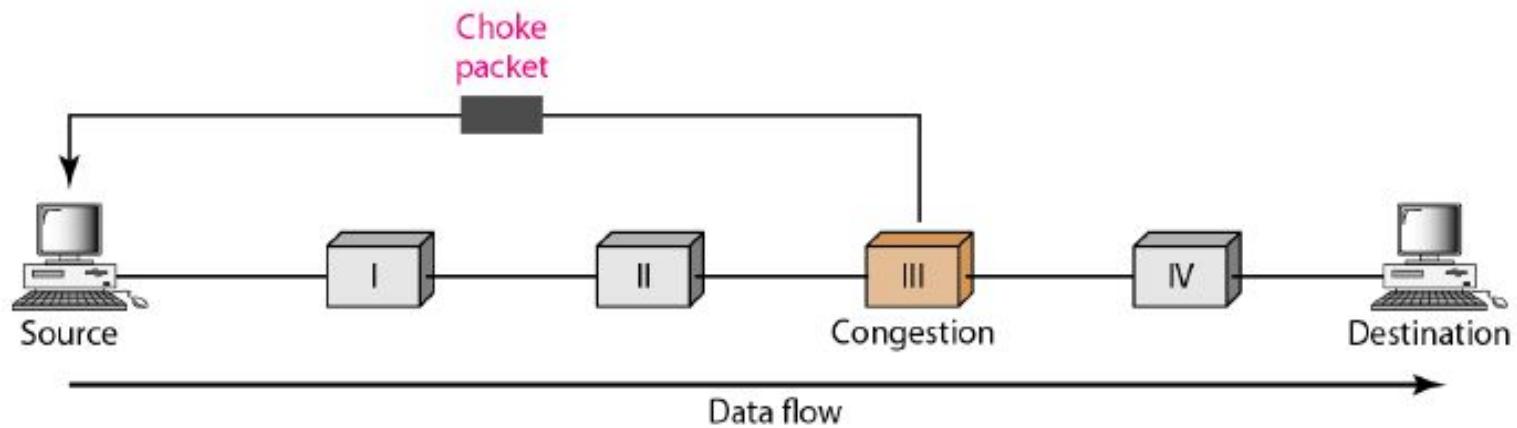
# Closed-Loop Congestion Control

## 2. Choke Packet

- A **choke packet** is a packet sent by a node to the source to inform it of congestion.
- Note the difference between the backpressure and choke packet methods.
- *In backpressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station.*
- In the **choke packet** method, the warning is from the router, which has encountered congestion, to the source station directly.
- The intermediate nodes through which the packet has traveled are not warned.
- We have seen an example of this type of control in **ICMP**.
- When a router in the Internet is overwhelmed with IP datagrams, it may discard some of them; but it informs the source host, using a **source quench ICMP message**.

# Choke packet

- The warning message goes directly to the source station; the intermediate routers, and does not take any action.



# Closed-Loop Congestion Control

## 3. Implicit Signaling

- In **implicit signaling**, there is **no communication between the congested node or nodes and the source**.
- The **source guesses that there is a congestion somewhere** in the network **from other symptoms**.
- For **example**, when a source sends several packets and there is **no acknowledgment for a while**, one assumption is that the network is **congested**.
- The **delay in receiving an acknowledgment** is interpreted as **congestion** in the network; the source should slow down.
- **TCP uses such signaling for congestion control.**

# Closed-Loop Congestion Control

## 4. Explicit Signaling

- The **node that experiences congestion can explicitly send a signal to the source or destination.**
- The **explicit signaling** method, however, is **different** from the **choke packet method**.
- In the **choke packet** method, a **separate packet is used for this purpose**; in the **explicit signaling method**, the **signal is included in the packets that carry data**.
- **Explicit signaling**, as used in **Frame Relay congestion control**, can occur in either the forward or the backward direction.

# Closed-Loop Congestion Control

## 4.1. Backward Signaling

- A bit can be set in a packet moving in the direction opposite to the congestion.
- This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

## 4.2. Forward Signaling

- A bit can be set in a packet moving in the direction of the congestion.
- This bit can warn the destination that there is congestion.
- The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

# Congestion Control in TCP

- TCP uses **congestion control** to **avoid congestion** or **alleviate congestion** in the network.

## Congestion Window

- The **sender window size** is determined by the available buffer space in the receiver (*rwnd*).
- In addition to the receiver, the **network** is a second entity that determines the size of the sender's window.
- Thus, *the sender's window size is determined not only by the receiver but also by congestion in the network*.
- The **sender has two pieces of information**: the **receiver-advertised window size** and the **congestion window size**.
- The actual size of the window is the **minimum of these two**.
- i.e. **Sender's window size= minimum (rwnd, cwnd)**

# TCP Congestion Policy

- TCP's general policy for handling **congestion** is based on **three phases**:
  1. Slow start(SS)
  2. **Congestion avoidance(CA)**
  3. Congestion detection(CD)
- In the **slow-start phase**, the **sender starts with a very slow rate of transmission**, but **increases the rate rapidly to reach a threshold**.
- When the **threshold is reached**, the **data rate is reduced to avoid congestion**.
- Finally if **congestion is detected**, the **sender goes back to the slow-start or congestion avoidance phase** based on **how the congestion is detected**.

# 1. Slow Start: Exponential Increase

- One of the algorithms used in **TCP congestion control** is called ***slow start***.
- This algorithm is based on the idea that the size of the **congestion window (*cwnd*) starts with one maximum segment size (MSS)**.
- The **MSS** is **determined** during **connection establishment** phase.
- The **size of the window increases one MSS each time an acknowledgment is received**.
- As the name implies, the **window starts slowly, but grows exponentially**.
- We have assumed that ***rwnd* is much higher than *cwnd*, so that the sender window size always equals *cwnd***.
- We have assumed that each segment is acknowledged individually.

# 1. Slow Start: Exponential Increase

- The **Sender starts** with **cwnd = 1 MSS**.
- This means that the **sender** can send only **one segment**.
- After receipt of the acknowledgment for segment 1, the size of the congestion window is increased by 1, which means that **cwnd is now 2**.
- Now two more segments can be sent.
- When each **acknowledgment is received**, the size of the **window** is increased by **1 MSS**.
- When all seven segments are acknowledged, **cwnd = 8**.

# Slow Start: Exponential Increase

- If we look at the **size of *cwnd*** in terms of rounds (acknowledgment of the whole window of segments), we find that the rate is exponential as shown below:

Start ..... *cwnd*=1

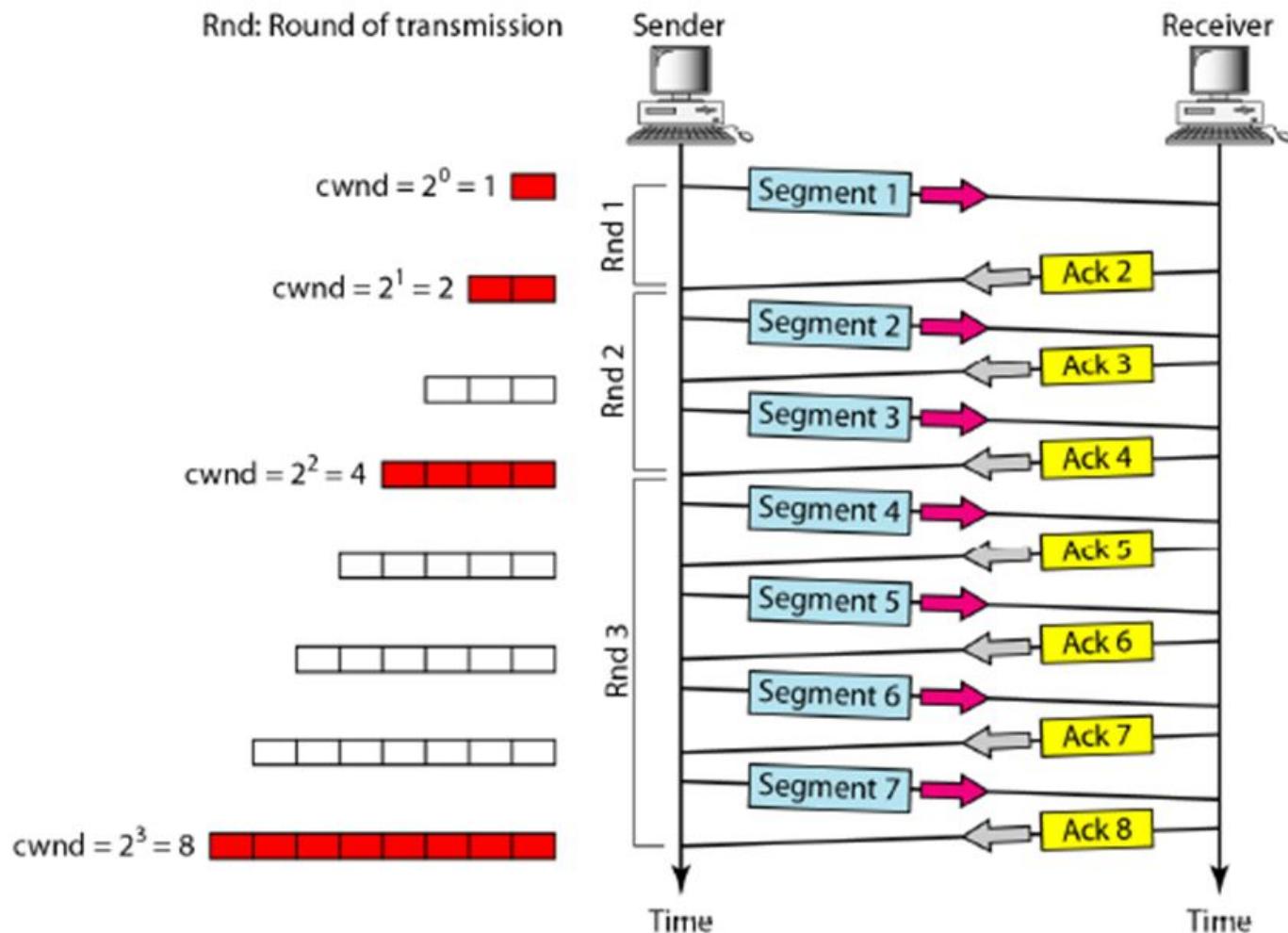
After round 1 ..... *cwnd*= $2^1$  =2

After round 2 ..... *cwnd*= $2^2$  =4

After round 3 ..... *cwnd*= $2^3$  =8

- We need to mention that **if there is delayed ACKs, the increase in the size of the window is less than power of 2.**
- **Slow start cannot continue indefinitely.** There must be a **threshold to stop this phase.**
- The **sender keeps track of a variable named *ssthresh*** (slow-start threshold).
- When the **size of window in bytes reaches this threshold, slow start stops and the next phase starts.**
- In most implementations the value of ***ssthresh* is 65,535 bytes.**

# Slow Start: Exponential Increase



# Congestion Avoidance: Additive Increase

- Congestion avoidance undergoes an **additive increase** instead of an exponential one.
- When the size of the **congestion window reaches the slow-start threshold**, the **slow-start phase stops and the additive phase begins**.
- In this **algorithm**, each time the whole window of segments is acknowledged (one round), the **size of the congestion window is increased by 1**.
- **Congestion avoidance algorithm usually starts when the size of the window is much greater than 1**.
- In this case, after the sender has received acknowledgments for a complete window size of segments, the size of the window is increased by one segment. If we look at the size of *cwnd* in terms of rounds, we find that the rate is additive as shown below:

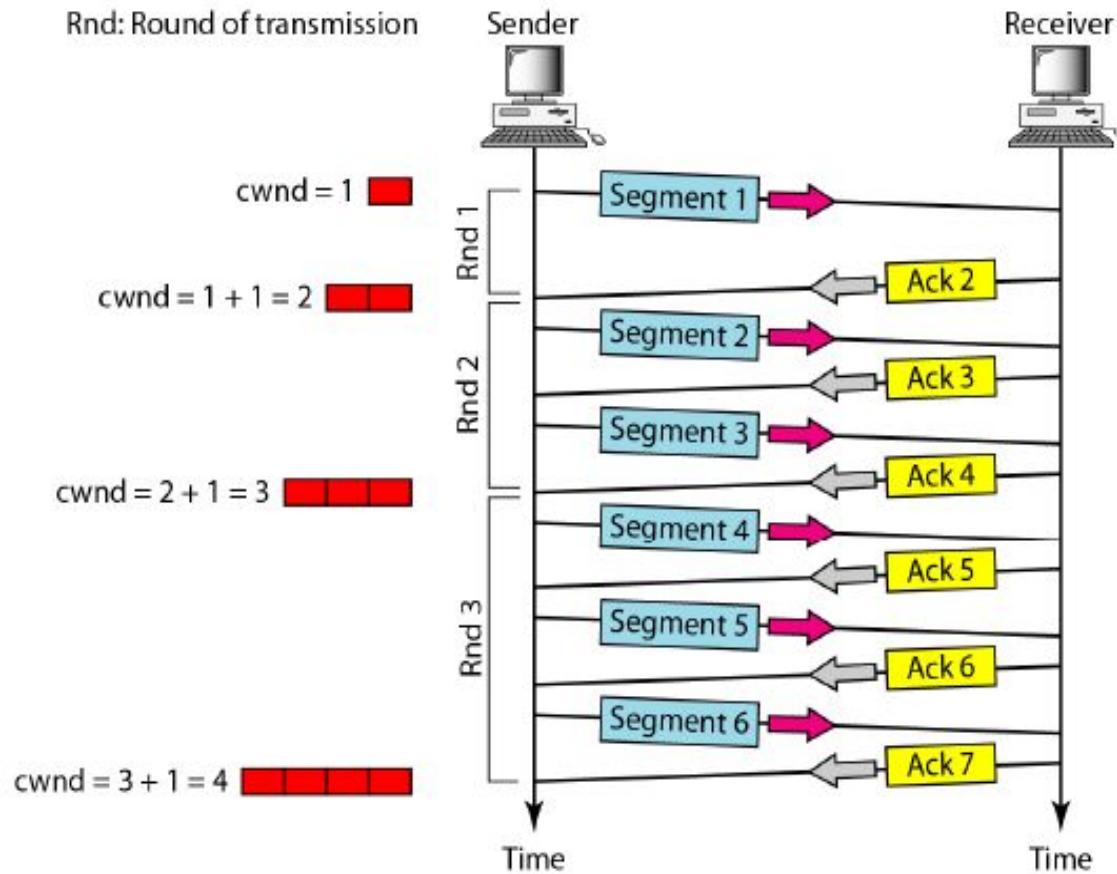
**Start 1 *cwnd*=1**

**After round 1 *cwnd*= 1+ 1 =2**

**After round 2 *cwnd*=2+ 1 =3**

**After round 3 *cwnd*=3+ 1 =4**

# Congestion avoidance, additive increase



# Congestion Detection: Multiplicative Decrease

- If congestion occurs, the **congestion window size must be decreased**.
- The only way the sender can guess that **congestion has occurred** is by the need to **retransmit a segment**.
- However, **retransmission can occur in one of two cases**: when a **timer times out** or when **three duplicate ACKs are received**.
- In both cases, the size of the **threshold is dropped to one-half**, a **multiplicative decrease**.
- Most **TCP implementations** have **two cases**:

# Congestion Detection: Multiplicative Decrease

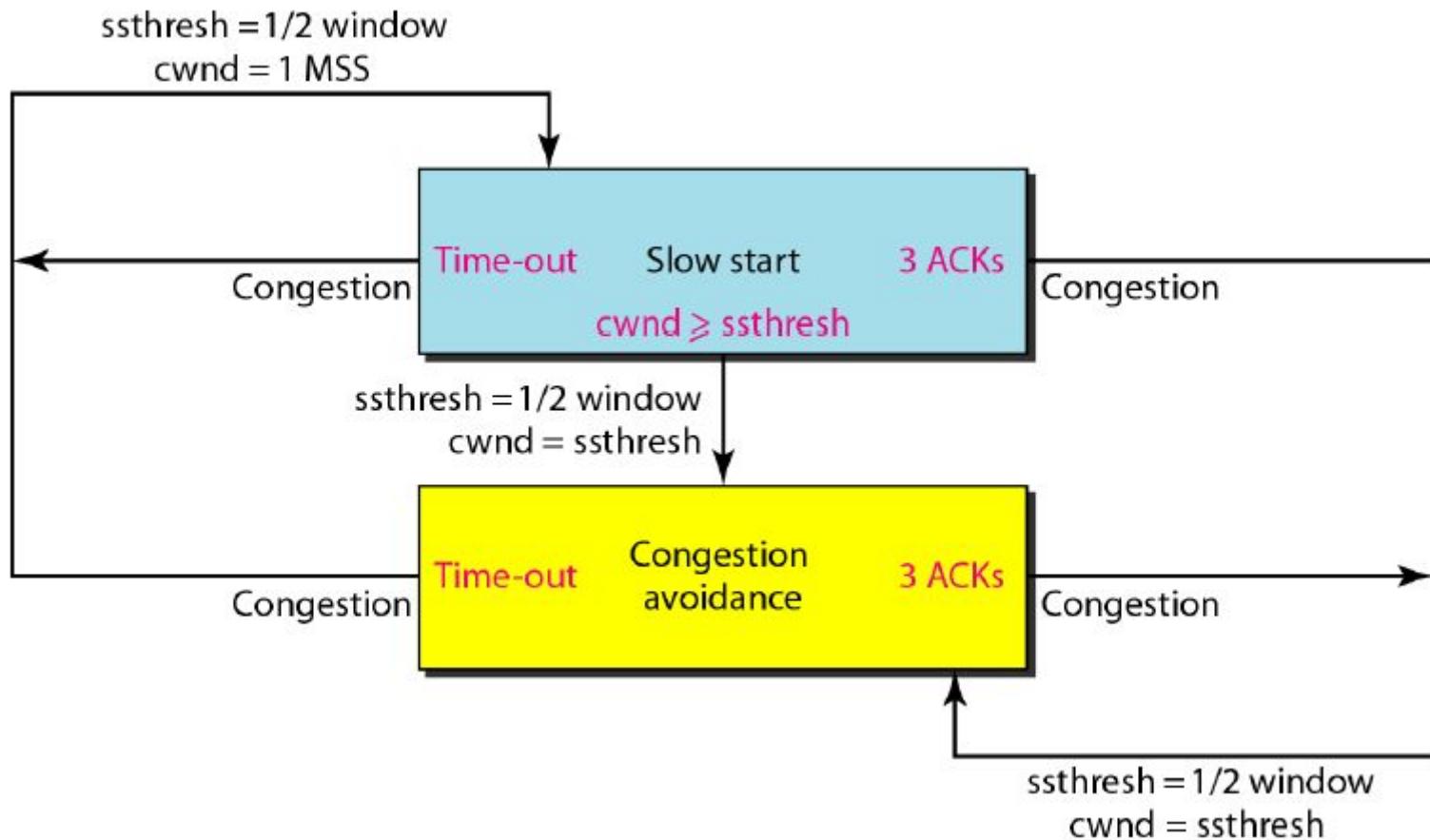
1. If a **time-out** occurs, there is a **stronger possibility of congestion**; a segment has probably been **dropped in the network**, and there is **no news about the sent segments**.

- In this case **TCP reacts strongly**:
  - a. It sets the value of the **threshold to one-half of the current window size**.
  - b. It sets **cwnd** to the **size of one segment**.
  - c. It starts the slow-start phase again.

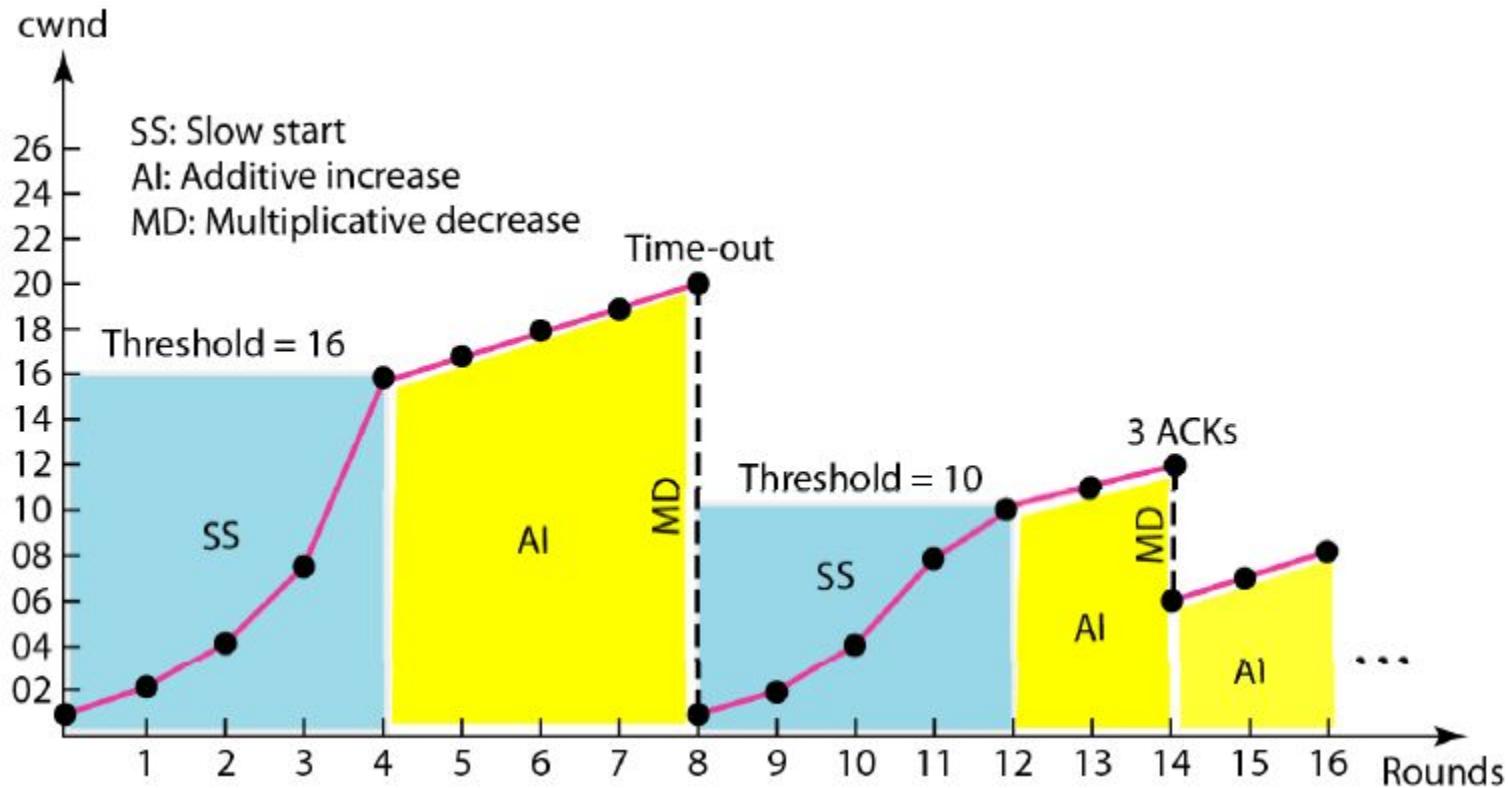
2. If three ACKs are received, there is a **weaker possibility of congestion**; a segment **may have been dropped, but some segments after that may have arrived safely since three ACKs are received**. This is called **fast transmission** and **fast recovery**. In this case, TCP has a **weaker reaction**:

- a. It sets the value of the **threshold to one-half of the current window size**.
- b. It sets **cwnd** to the **value of the threshold**
- c. It starts the **congestion avoidance phase**.

# TCP Congestion Policy Summary



# TCP Congestion Control Example



- We assume that the **maximum window size is 32 segments**. The **threshold** is set to **16 segments** (one-half of the maximum window size).
- In the ***slow-start* phase** the window size **starts from 1** and grows **exponentially** until it reaches the threshold.

# TCP Congestion Control Example

- After it reaches the threshold, the *congestion avoidance (additive increase)* procedure allows the window size to increase linearly until a timeout occurs or the maximum window size is reached. In Figure 24.9, the time-out occurs when the window size is 20.
- At this moment, the *multiplicative decrease* procedure takes over and reduces the threshold to one-half of the previous window size. The previous window size was 20 when the time-out happened so the new threshold is now 10.
- TCP moves to slow start again and starts with a window size of 1, and TCP moves to additive increase when the new threshold is reached.
- When the window size is 12, a **three-DUPACKs** event happens. The multiplicative decrease procedure takes over again.
- The threshold is set to 6 and TCP goes to the additive increase phase this time.
- It remains in this phase until another time-out or another three ACKs happen.

# **Lecture 6.3**

## **Transport Layer: Quality of Service**

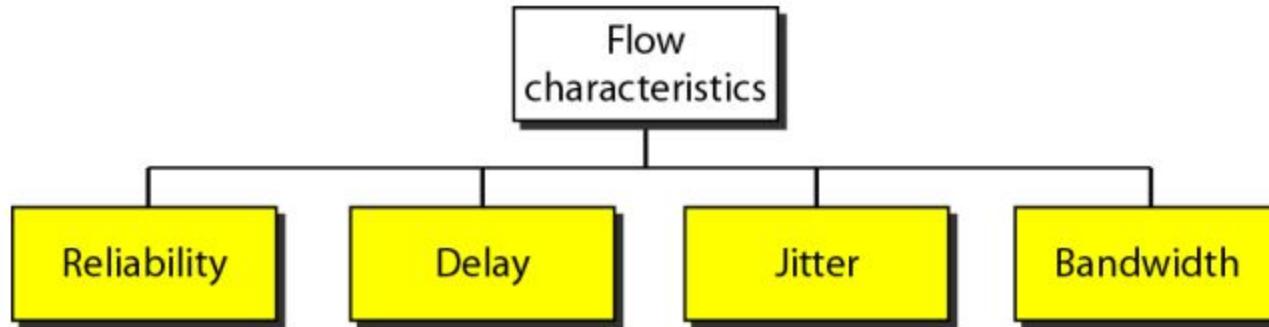
**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Quality of Service

- **Quality of Service (QoS)** refers to the **capability of a network** to provide **better service** to **selected network traffic** over various technologies.
- The **primary goal** of QoS is to provide **priority** including dedicated **bandwidth**, controlled **jitter** and **latency** (required by some real-time and interactive traffic), and improved **loss characteristics**.
- One important issue is making sure that providing priority for one or more flows does not make other flows fail.
- A flow can be defined in a number of ways.
- One common way refers to a combination of source and destination addresses, source and destination socket numbers etc.
- It can also be defined more broadly as any packet from a certain application or from an incoming interface.

# Flow Characteristics



## 1. Reliability

- Reliability is a characteristic that a flow needs.
- Lack of reliability means losing a packet or acknowledgment, which entails retransmission.
- However, the sensitivity of application programs to reliability is not the same.
- For example, it is more important that electronic mail, file transfer, and Internet access have reliable transmissions than telephony or audio conferencing.

# Flow Characteristics

## 2. Delay

- Different applications can tolerate **delay** in different degrees.
- In this case, telephony, audio conferencing, video conferencing, and remote log-in need minimum delay, while delay in file transfer or e-mail is less important.

## 3. Jitter

- **Jitter** is the **variation in delay** for **packets** belonging to the **same flow**.
- For example, if **four packets** depart at times 0, 1, 2, 3 and arrive at 20, 21, 22, 23, all have the **same delay, 20 units** of time.
- On the other hand, if the above **four packets** arrive at 21, 23, 21, and 28, they will have **different delays**: 21, 22, 19, and 24.

# Flow Characteristics

- For **applications** such as **audio** and **video**, the first case is completely acceptable; the second case is not.
- For **audio** and **video applications**, it does not matter if the packets arrive with a short or long delay as long as the delay is the same for all packets.
- **Jitter** is defined as the **variation in the packet delay**.
- **High jitter** means the difference between delays is large; **low jitter** means the variation is small.

## 4. Bandwidth

- Different applications need different bandwidths.
- In **video conferencing** we need to send **millions of bits per second** to refresh a colour screen while the total number of bits in an **e-mail** may not reach even a **million**.

# Flow Classes

- Based on the **flow characteristics**, we can classify flows into groups, with each group having similar levels of characteristics.
- This categorization is not formal or universal; some protocols such as ATM have defined classes.

# TECHNIQUES TO IMPROVE QoS

Some common methods to improve QoS are:

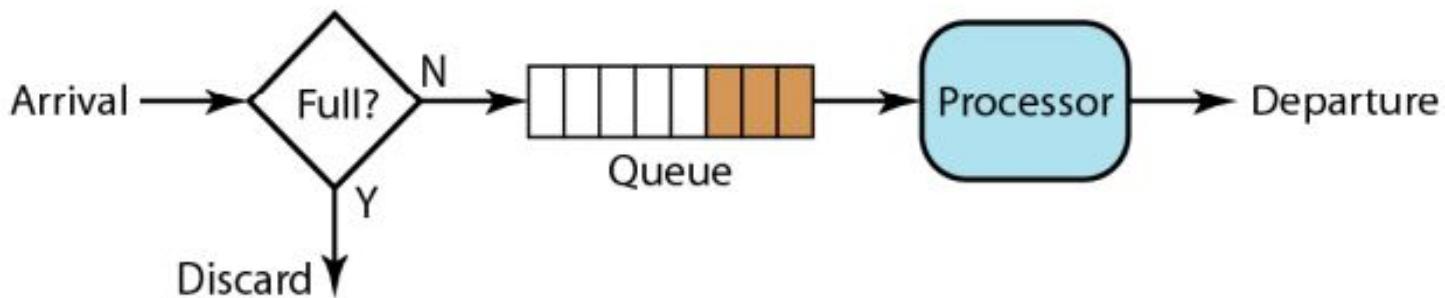
1. **Scheduling,**
2. **Traffic shaping,**
3. **Admission control,**
4. **Resource Reservation.**

## 1. Scheduling

- Packets from different flows arrive at a **switch** or **router** for processing.
- A good scheduling technique treats the different flows in a fair and appropriate manner.
- Several **scheduling techniques** are designed to **improve** the **quality of service**.
- The three of them here: **FIFO queuing**, **Priority queuing**, and **Weighted fair queuing(WFQ)**.

# Scheduling: FIFO Queuing

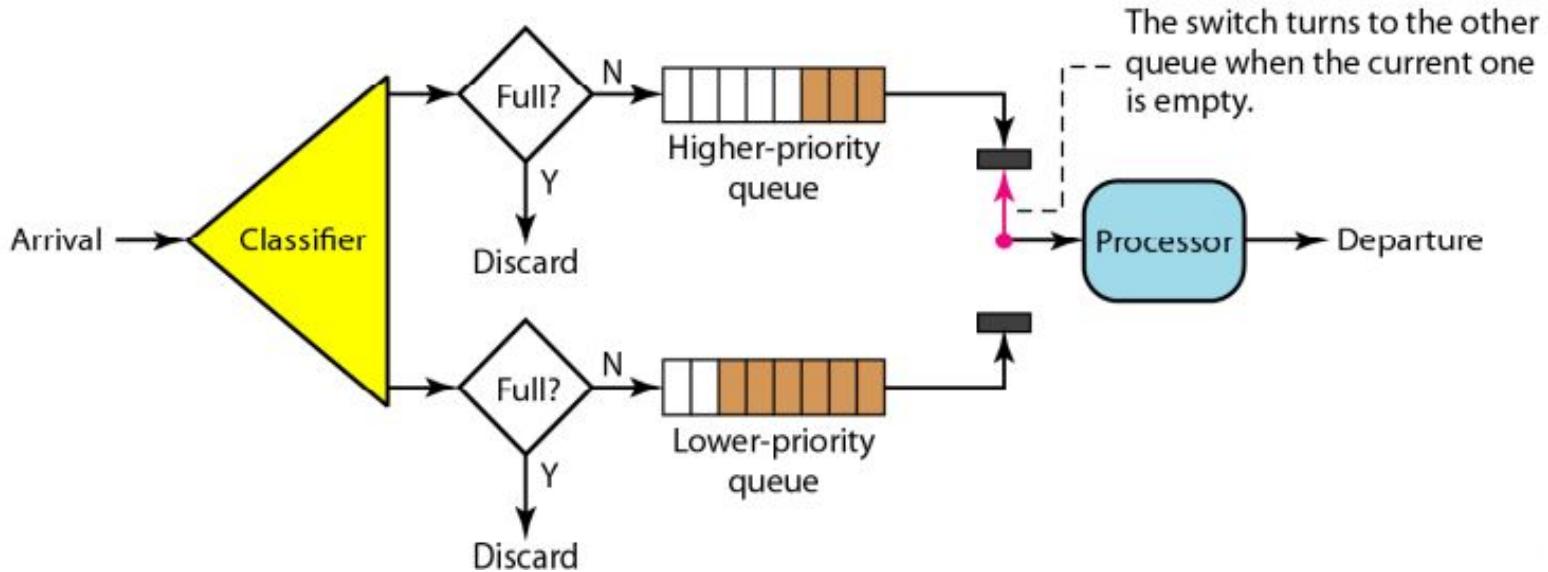
- In **first-in, first-out (FIFO) queuing**, packets wait in a **buffer (queue)** until the node (router or switch) is ready to process them.
- If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be **discarded**.
- A **FIFO queue** is familiar to those who have had to wait for a bus at a bus stop.



# Priority Queuing

- In priority queuing, packets are first assigned to a priority class.
- Each priority class has its own queue.
- The packets in the highest-priority queue are processed first.
- Packets in the lowest-priority queue are processed last.
- Note that the system does not stop serving a queue until it is empty.
- A priority queue can provide better QoS than the FIFO queue because higher-priority traffic, such as multimedia, can reach the destination with less delay.
- However, there is a potential drawback. If there is a continuous flow in a high-priority queue, the packets in the lower-priority queues will never have a chance to be processed.
- This is a condition called *starvation*.

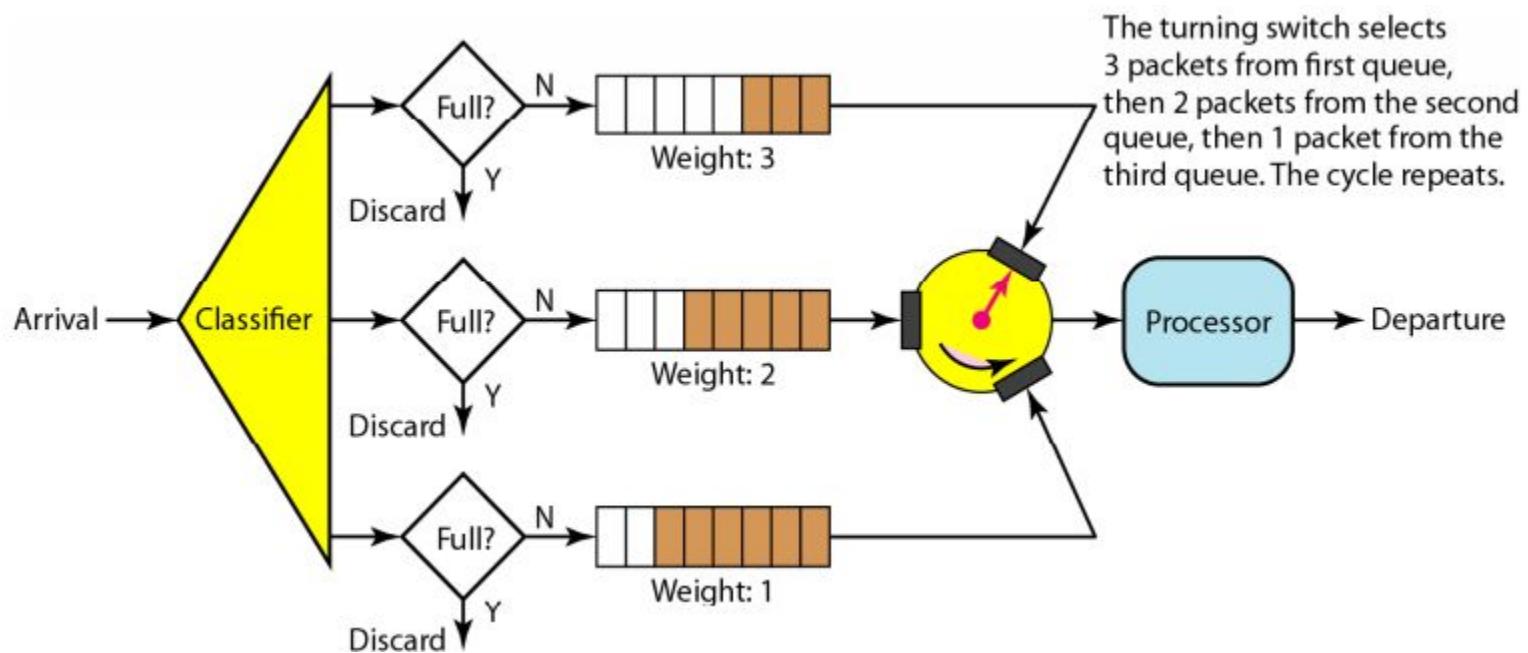
# Priority Queuing



# Weighted Fair Queuing

- In this technique, the **packets** are still assigned to different **classes** and admitted to different **queues**.
- The **queues**, how-ever, are **weighted** based on the **priority of the queues**; higher priority means a **higher weight**.
- The **system processes** packets in each **queue** in a **round-robin fashion** with the
- **Number of packets selected from each queue** based on the **corresponding weight**.
- For **example**, if the **weights** are **3, 2, and 1**, **three packets** are **processed** from the **first queue**, **two** from the **second queue**, and **one** from the **third queue**.
- If the system does not impose priority on the classes, all weights can be equal In this way, we have fair queuing without priority.

# Weighted Fair Queuing



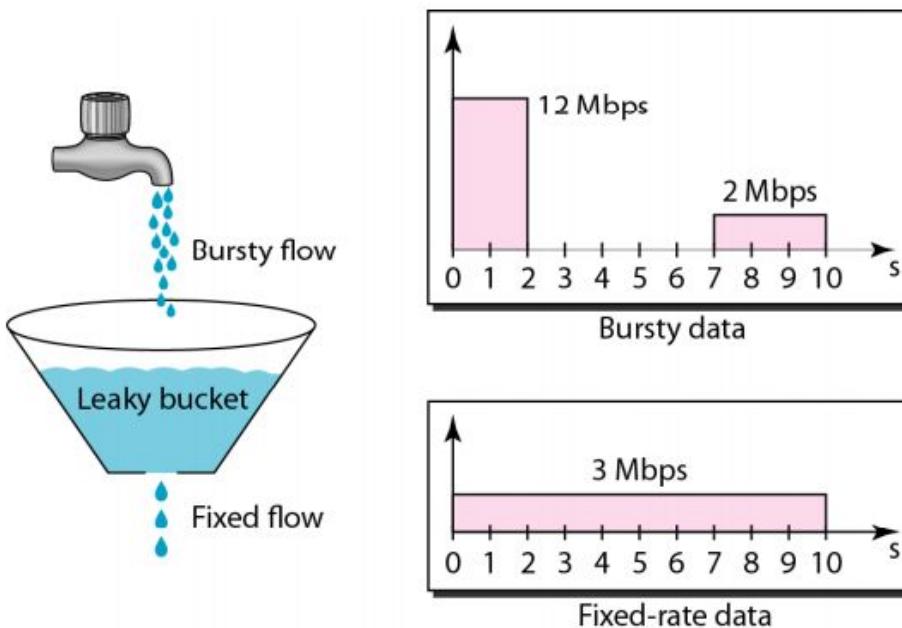
## 2.Traffic Shaping

- Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network.
- Two techniques can shape traffic: Leaky bucket and Token bucket.

### Leaky Bucket

- If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket.
- The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty.
- The input rate can vary, but the output rate remains constant.
- Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic.
- Bursty chunks are stored in the bucket and sent out at an average rate.

# Leaky bucket



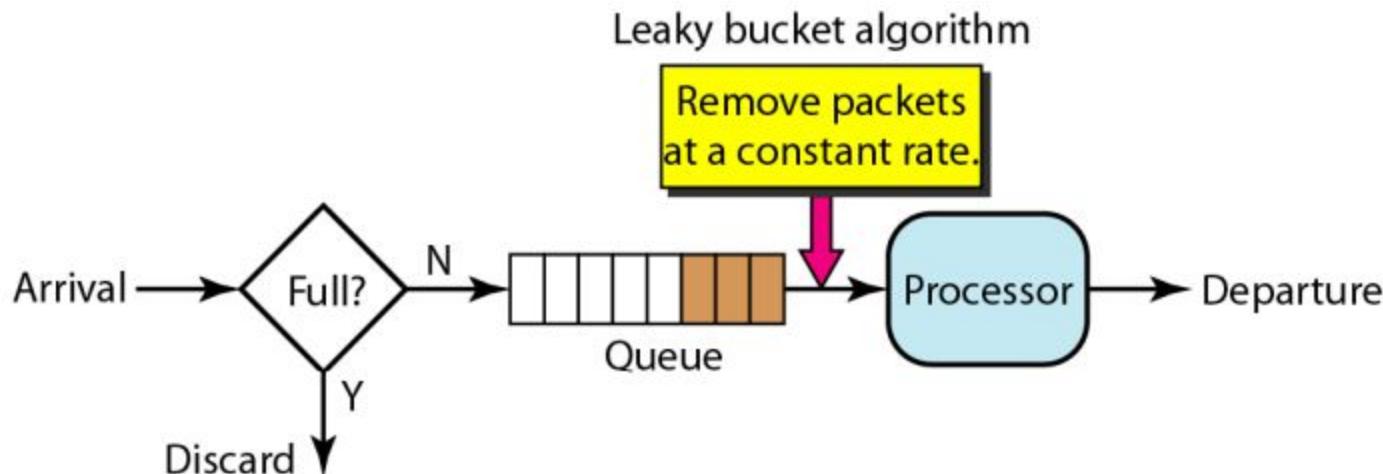
- Assume that the network has committed a bandwidth of 3 Mbps for a host.
- The use of the **leaky bucket** shapes the **input traffic** to make it conform to this commitment.
- The host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data.
- The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data.

# Leaky bucket

- In all, the **host** has **sent 30 Mbits** of **data** in all.
- The **leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps** during the same **10s**.
- **Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth** than is set aside for this host.
- We can also see that the **leaky bucket may prevent congestion**.

# Leaky bucket implementation

- A simple **leaky bucket implementation** is shown in Figure below.
- A **FIFO queue holds the packets**.
- If the **traffic consists of fixed-size packets** (e.g., cells in ATM networks), the process **removes a fixed number of packets** from the **queue** at each **tick of the clock**.
- If the **traffic consists of variable-length packets**, the **fixed output rate** must be **based on the number of bytes or bits**.



# Leaky bucket implementation

The following is an **Algorithm** for variable-length packets:

1. **Initialize a counter to  $n$  at the tick of the clock.**
2. **If  $n > \text{size of the packet}$ , send the packet and decrement the counter by the packet size. Repeat this step until  $n < \text{packet size}$ .**
3. **Reset the counter and go to step 1.**

# Leaky bucket Example

- Let  $n=1000$

Packets =

200	700	500	450	400	200
-----	-----	-----	-----	-----	-----

F



- Since  $n >$  front of Queue i.e.  $n > 200$

Therefore,  $n=1000-200=800$

Packet size of 200 is sent to the network.

F

200	700	500	450	400
-----	-----	-----	-----	-----

- Now Again  $n >$  front of the queue i.e.  $n > 400$

Therefore,  $n=800-400=400$

Packet size of 400 is sent to the network.

F

200	700	500	450
-----	-----	-----	-----

- Since  $n <$  front of queue

Therefore, the procedure is stop.

- Initialize  $n=1000$  on another tick of clock.

- This procedure is repeated until all the packets are sent to the network.

# Numerical: Leaky bucket

**Question 1:** Consider a host using leaky bucket strategy for traffic shaping. The host send a burst data at a rate of 15 Mbps for first 3 sec and remains silent for 2 sec. Then again a burst data at a rate of 6 Mbps is sent for next 2 sec and then the host remains silent for next 2 sec. Now again the host send data at a rate of 5 Mbps for next 3 sec.

What will be the output data rate of the leaky bucket?

**Solution:**

Data sent is  $15 \times 3 + 6 \times 2 + 5 \times 3 = 72$  Mb

During  $3+2+2+2+3=12$  sec

So, the output data rate is  $72/12 = 6$  Mbps

# Numerical: Leaky bucket

**Question 2:** Consider a frame relay network having bucket capacity of 1Mb and data is input at the rate of 25mbps.Calculate

- i. What is the time needed to fill the bucket.
- ii. If the output rate is 2 mbps , the time needed to empty the bucket.

**Solution:**

Here , C is Capacity of bucket = 1Mb , Data input rate = 25 mbps , Output rate = 2mbps.

- i.  $T = C/\text{input rate} = 1/25 = 40 \text{ msec}$
- ii.  $T = C/\text{output rate} = 1/2 = 500 \text{ msec}$

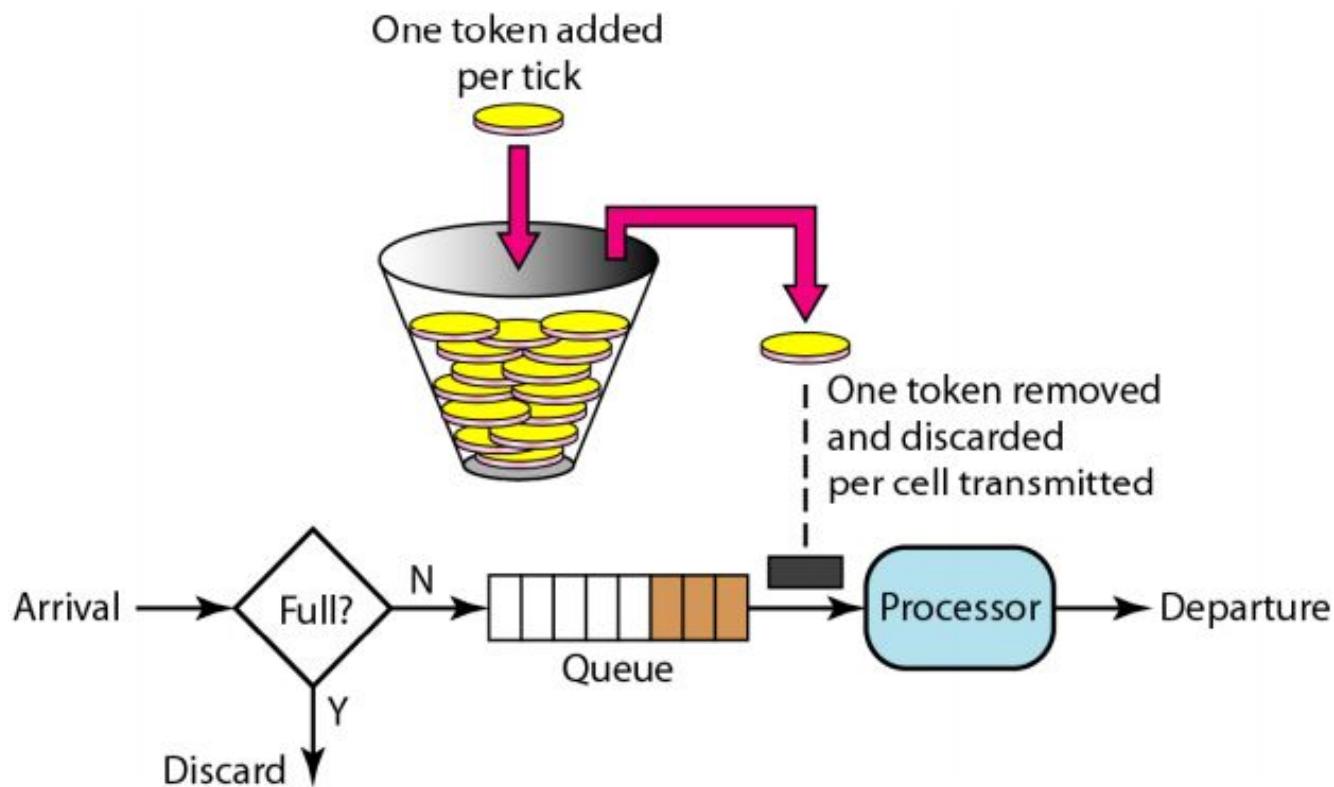
# Token Bucket

- The **leaky bucket** is **very restrictive**.
- It does not **credit an idle host**.
- For example, if a **host is not sending for a while**, its bucket becomes **empty**.
- Now if the host has **bursty data**, the **leaky bucket allows only an average rate**.
- The time when the host was idle is not taken into account.
- On the other hand, the **token bucket algorithm** allows **idle hosts to accumulate credit for the future in the form of tokens**.
- For each tick of the clock, the system sends  $n$  tokens **to the bucket**.
- The system removes one token for every byte of data sent.
- For example, if  $n$  is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens.

# Token Bucket

- Now the host can consume all these tokens in one tick with **10,000 bytes**, or the host takes **1000 ticks** with **10 bytes per tick**.
- In other words, the host can send bursty data as long as the bucket is not empty.
- The token bucket can easily be implemented with a **counter**.
- The token is initial-ized to zero.
- Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1.
- When the counter is zero, the host cannot send data.

# Token Bucket



# Token Bucket: Algorithm

1. A **token** is added to the bucket every  $1/r$  seconds, i.e. token generation rate is  $r$  tokens/second.
2. The **bucket** can hold at the most **b** tokens. If a token arrives when the bucket is full, it is discarded.
3. When a packet of **n bytes** arrives,
  - if at least **n tokens** are in the bucket, **n tokens are removed** from the bucket, and the **packet is sent** to the network.
  - if fewer than **n** tokens are available, no tokens are removed from the bucket, and the packet is considered to be **non-conformant**.
  - The non-conformant packets may either be dropped or queued for subsequent transmission when sufficient tokens have accumulated in the bucket.

# Numerical: Token Bucket

**Note:** If C= Bucket Capacity, r=token arrival rate, M= maximum output rate and burst length=S then  $C + rS = MS$

**Question:** Consider a frame relay network where 1Mb of data is arriving at the rate of 25mbps for 40msec. The Token arrival rate is 2mbps and the capacity of bucket is 500 Kb with maximum output rate 25mbps. Calculate:

- i. The burst Length.
- ii. Total output time.

**Solution:** Here , C is Capacity of bucket = 500kb, M= 25 mbps, r= 2mbps.

- $S = 500 / ((25 - 2) * 1000) = 21.73\text{msec} \approx 22\text{msec}$
- It means for 22msec the output rate is 25mbps after that the output rate becomes 2mbps i.e. token arrival rate.
- Therefore, for another 500 kb the time taken will be.  $500 / (2000) = 250\text{ msec}$   
Therefore, **total output time** =  $22 + 250 = 272\text{ msec}$ .

# TECHNIQUES TO IMPROVE QoS

## 3. Resource Reservation

- A flow of data needs resources such as a **buffer**, **bandwidth**, **CPU time**, and so on.
- The quality of service is improved if these resources are reserved beforehand.
- There are some QoS model called Integrated Services, which depends heavily on resource reservation to improve the quality of service.

## 4. Admission Control

- Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called **flow specifications**.
- Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

# Quality of service in the Internet

- Two models have been designed to provide quality of service in the Internet:
  - 1. Integrated Services**
  - 2. Differentiated Services.**
- Both models emphasize the use of quality of service at the network layer (IP), although the model can also be used in other layers such as the data link.

# Integrated Services

- IP was originally designed for ***best-effort delivery***.
- This means that every user receives the same level of services.
- This type of delivery does not guarantee the minimum of a service, such as bandwidth, to applications such as real-time audio and video.
- If such an application accidentally gets extra bandwidth, it may be detrimental to other applications, resulting in congestion.
- Integrated Services, sometimes called **IntServ**, is a ***flow-based QoS model***, which means that a user needs to create a flow, a kind of virtual circuit, from the source to the destination and inform all routers of the resource requirement.

# Integrated Services

## 1. Signaling

- IP is a connectionless, datagram, packet-switching protocol.
- How can we implement a flow-based model over a connectionless protocol?
- The solution is a signaling protocol to run over IP that provides the signaling mechanism for making a reservation.
- This protocol is called **Resource Reservation Protocol (RSVP)**.

## 2. Flow Specification

- When a source makes a reservation, it needs to define a **flow specification**.
- A **flow specification has two parts: Rspec (resource specification) and Tspec (traffic specification)**.
- **Rspec** defines the resource that the flow needs to reserve (buffer, bandwidth, etc.).
- **Tspec** defines the traffic characterization of the flow.

# Integrated Services

## 3. Admission

- After a router receives the flow specification from an application, it decides to admit or deny the service.
- The decision is based on the previous commitments of the router and the current availability of the resource.

## 4. Service Classes

- Two classes of services have been defined for **Integrated Services**:
  - Guaranteed service.
  - Controlled-load service.

# Integrated Services

## Guaranteed Service Class

- This type of service is designed for real-time traffic that needs a guaranteed minimum end-to-end delay.
- The end-to-end delay is the sum of the delays in the routers, the propagation delay in the media, and the setup mechanism.
- Only the sum of the delays in the routers, can be guaranteed by the router.
- This type of service guarantees that the packets will arrive within a certain delivery time and are not discarded if flow traffic stays within the boundary of Tspec.
- We can say that guaranteed services are quantitative services, in which the amount of end-to-end delay and the data rate must be defined by the application.

# Integrated Services

## Controlled-Load Service Class

- This type of service is designed for applications that can accept some delays, but are sensitive to an overloaded network and to the danger of losing packets.
- Good examples of these types of applications are file transfer, e-mail, and Internet access.
- The controlled- load service is a qualitative type of service in that the application requests the possibility of low-loss or no-loss packets.

# Resource Reservation Protocol (RSVP)

- In the **Integrated Services** model, an application program needs **resource reservation** for a **flow**.
- It is required to **create a flow**, a kind of **virtual-circuit network**.
- A **virtual-circuit network** needs a **signaling system** to set up the **virtual circuit** before data traffic can start.
- The **Resource Reservation Protocol (RSVP)** is a **signaling protocol** to help IP create a **flow** and consequently make a **resource reservation**.
- RSVP is an **independent protocol** separate from the Integrated Services model.

# Resource Reservation Protocol (RSVP)

## Multicast Trees

- RSVP is a **signaling system** designed for **multicasting**.
- However, **RSVP** can be also used for **uni-casting** because unicasting is just a special case of multicasting with only one member in the multicast group.
- The reason for this design is to enable RSVP to provide resource reservations for all kinds of traffic including multimedia which often uses multicasting.

## Receiver-Based Reservation

- In **RSVP**, the **receivers**, not the sender, **make the reservation**.

## RSVP Messages

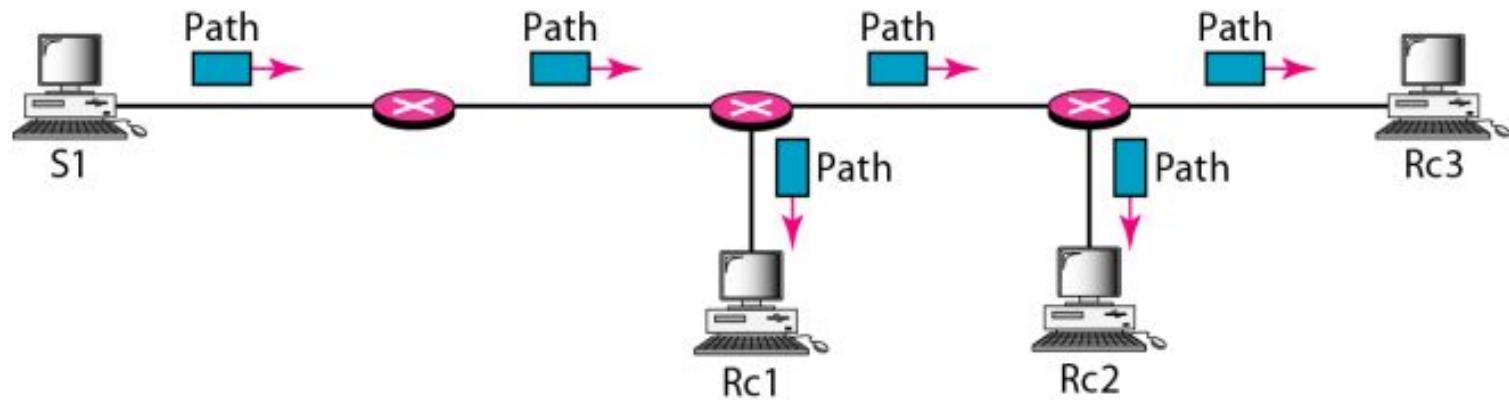
- **RSVP** has several types of messages.
- **Two** of them are **Path and Resv**.

# RSVP Messages

## Path Messages

- The receivers in a flow make the reservation in RSVP.
- However, the receivers do not know the path travelled by packets before the reservation is made.
- The **path** is needed for the **reservation**.
- To solve the problem, RSVP uses ***Path messages***.
- A **Path message** travels from the sender and reaches all receivers in the **multi-cast path**.
- A **Path message** is sent in a **multicast environment**; a new message is created when the **path diverges**.

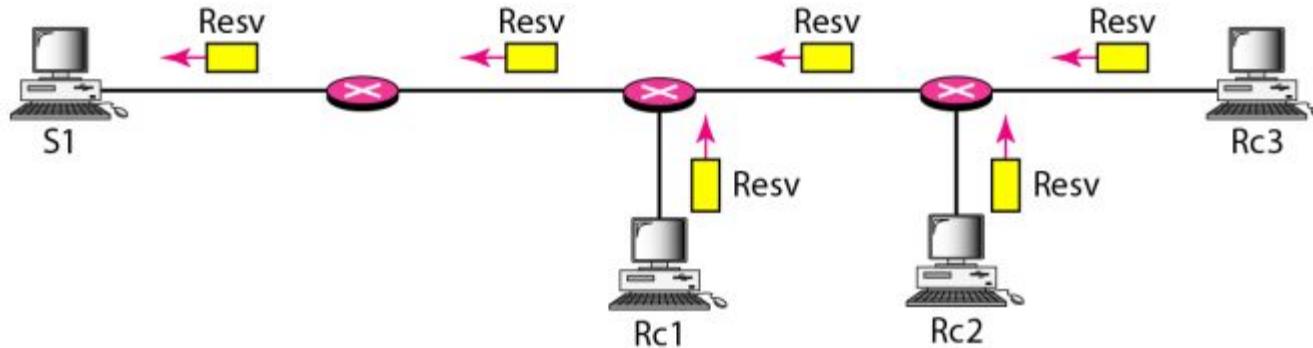
# RSVP: Path Messages



# RSVP Messages

## Resv Messages

- After a **receiver** has received a Path message, it sends a ***Resv message***.
- The **Resv message travels toward the sender (upstream) and makes a resource reservation on the routers that support RSVP.**
- If a router does not support RSVP on the path, it routes the packet based on the best-effort delivery methods we discussed before.

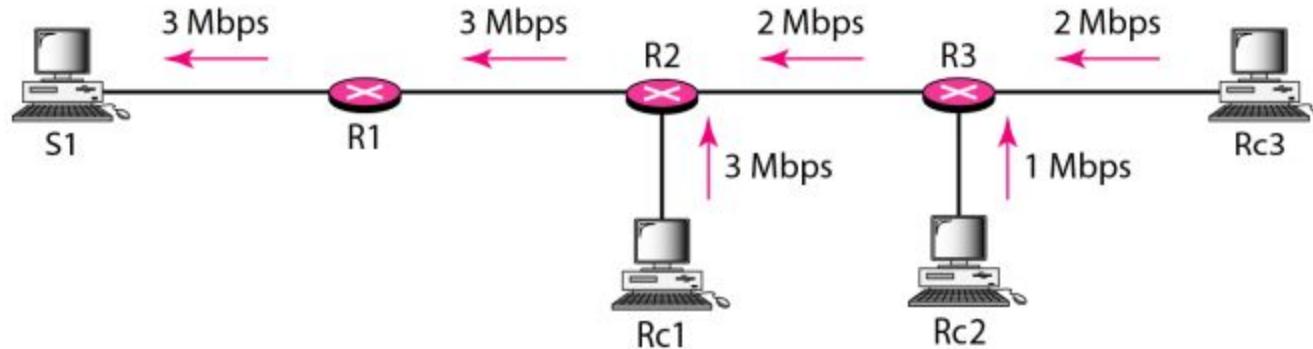


# Reservation Merging

- In RSVP, the **resources are not reserved** for each receiver in a **flow**; the reservation is **merged**.
- For example **Rc3** requests a **2-Mbps bandwidth** while **Rc2** requests a **1-Mbps bandwidth**.
- Router **R3**, which needs to make a bandwidth reservation, **merges the two requests**.
- The reservation is made for 2 Mbps, the larger of the two, because a 2-Mbps input reservation can handle both requests.
- The same situation is true for R2.
- Why Rc2 and Rc3, both belonging to one single flow, request different amounts of bandwidth?

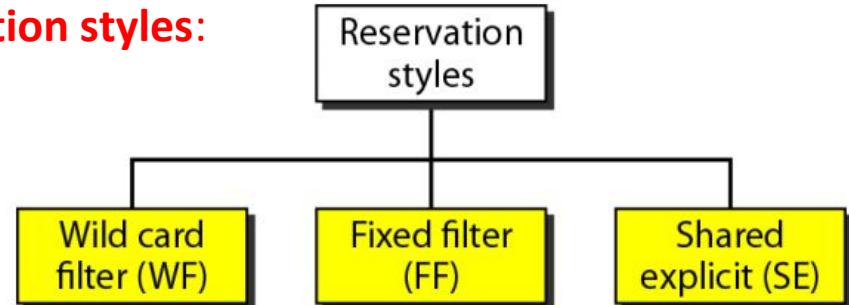
# Reservation Merging

- Because in a **multimedia environment**, different receivers may handle **different grades of quality**.
- For example,  $Rc_2$  may be able to receive video only at 1 Mbps (lower quality), while  $Rc_3$  may be able to receive video at 2 Mbps (higher quality).



# Reservation Styles

- When there is **more than one flow**, the router needs to make a reservation to accommodate all of them.
- **RSVP defines three types of reservation styles:**



## i. Wild Card Filter Style

- In this style, the **router creates a single reservation for all senders**.
- **The reservation is based on the largest request**.
- This type of style is used when the flows from different senders **do not occur at the same time**.

# Reservation Styles

## ii. Fixed Filter Style

- In this style, the router creates a **distinct reservation for each flow**.
- This means that if there are *n flows*, *n different reservations are made*.
- *This type of style* is used when there is a high probability that flows from different senders will **occur at the same time**.

## iii. Shared Explicit Style

- In this style, the router creates a **single reservation** which can be **shared by a set of flows**.

# Reservation Styles

## Soft State

- The **reservation information (state)** stored in every node for a **flow** needs to be **refreshed periodically**.
- This is referred to as a *soft state* as compared to the *hard state* used in other virtual-circuit protocols such as **ATM or Frame Relay**, where the **information about the flow is maintained until it is erased**.
- The **default interval** for refreshing is currently **30 s**.

# Problems with Integrated Services

- There are **two problems** with **Integrated Services** that may prevent its full implementation in the Internet: *scalability* and *service-type limitation*.

## Scalability

- The **Integrated Services model** requires that **each router keep information for each flow**.
- As the Internet is growing every day, this is a **serious problem**.
- **IntServ** would be suitable for **smaller private networks** whereas **DiffServ** is much suitable for **the Internet**.

## Service-Type Limitation

- The Integrated Services model provides **only two types of services**, guaranteed and control-load.
- Those opposing this model argue that applications may need more than these two types of services.

# DIFFERENTIATED SERVICES

- Differentiated Services (DS or Diffserv) was introduced by the IETF (Internet Engineering Task Force) to handle the shortcomings of Integrated Services.
- Two fundamental changes were made:
  1. The routers do not have to store information about flows. The applications, or hosts, define the type of service they need each time they send a packet.
  2. The per-flow service is changed to per-class service. The router routes the packet based on the class of service defined in the packet, not the flow. This solves the service-type limitation problem.
- We can define different types of classes based on the needs of applications.
- Packets are marked by the DiffServ devices at the boarders of the network with information about the level of service required by them.
- Other nodes in the network read this information and respond accordingly to provide the requested level of service.

# DS Field

- In **Diffserv**, each **packet contains a field called the DS field**. The value of this field is set at the boundary of the network by the host or the first router designated as the boundary router.
- IETF proposes to **replace the existing TOS (type of service) field in IPv4 by the DS field**, as shown in Figure below.



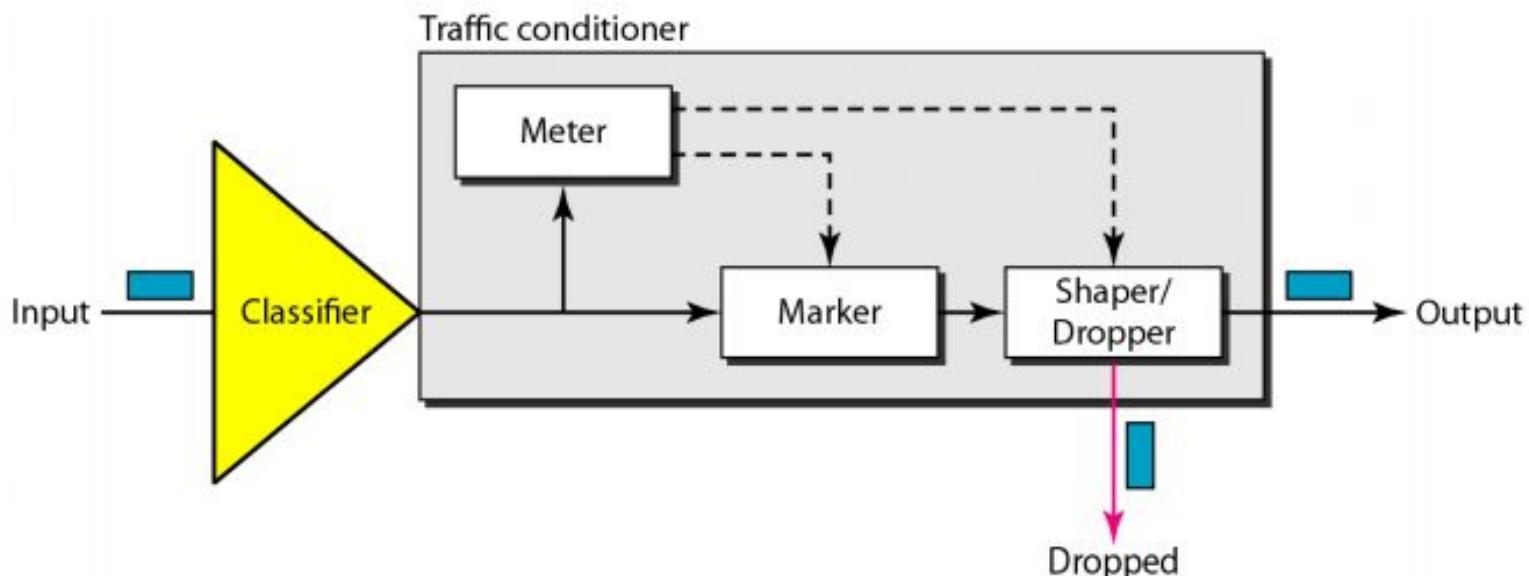
- The **DS field** contains **two subfields: DSCP and CU**. The DSCP (Differentiated Services Code Point) is a **6-bit subfield** that defines the **per-hop behavior (PHB)**.
- The **2-bit CU (currently unused) subfield** is not currently used.
- The **Diffserv capable node (router)** uses the **DSCP 6 bits** as an **index to a table** defining the **packet-handling mechanism** for the current packet being processed.

# Per-Hop Behaviour

- The **Diffserv model** defines **per-hop behaviors (PHBs)** for **each node** that receives a packet.
- So far **three PHBs** are defined: **DE PHB, EF PHB, and AF PHB**.
  - i. **DE PHB** The DE PHB (**default PHB**) is the same as **best-effort delivery**, which is compatible with **TOS**.
  - ii. **EF PHB** The EF PHB (**expedited forwarding PHB**) provides the following services: **Low loss, Low latency** and **Ensured bandwidth**.
  - iii. **AF PHB** The AF PHB (**assured forwarding PHB**) delivers the packet with a **high assurance** as long as the class traffic does not exceed the traffic profile of the node.

# Traffic Conditioner

- To implement **Diffserv**, the **DS node(router)** uses **traffic conditioners** such as **meters, markers, shapers, and droppers**, as shown in Figure below.



# Traffic Conditioner

- **Meters** The meter checks to see if the incoming flow matches the negotiated traffic profile.
- The meter also sends this result to other components.
- The **meter** can use several tools such as a **token bucket** to check the profile.
- **Marker** A marker can **mark a packet** that is **using best-effort delivery** (DSCP: 000000) or **down-mark a packet** based on information received from the **meter**.
- **Down- marking** (lowering the class of the flow) occurs if the **flow does not match the profile**.
- A **marker** does not **up-mark** (promote the class) a packet.
- **Shaper** A shaper uses the information received from the meter to **reshape the traffic if it is not compliant with the negotiated profile**.
- **Dropper** A dropper, which works as a shaper with **no buffer**, **discards packets if the flow severely violates the negotiated profile**.

# **Lecture 7.1**

## **Application Layer: Domain Name System(DNS)**

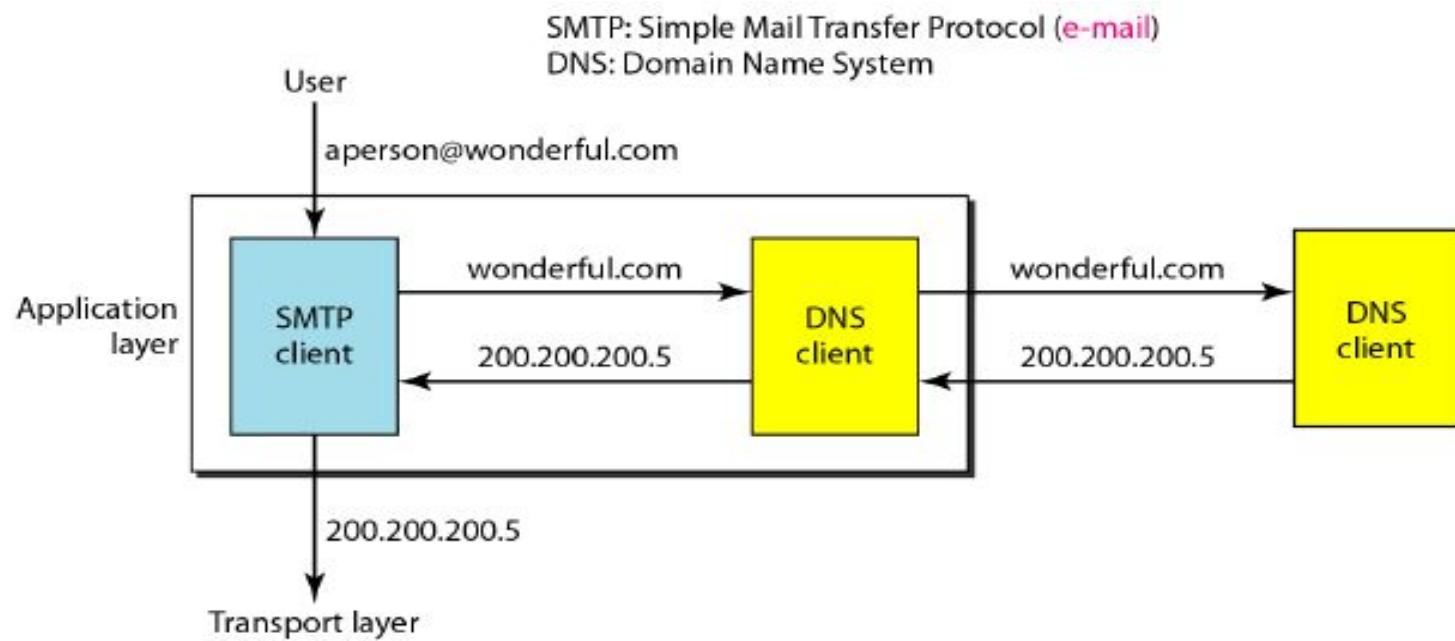
**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Introduction

- There are several applications in the application layer of the Internet model that follow the **client/server paradigm**.
- The **client/server programs** can be divided into **two categories**: those that can be directly used by the user, such as e-mail, and those that support other application programs.
- The **Domain Name System (DNS)** is a **supporting program** that is used by other programs such as **e-mail**.
- A user of an **e-mail** program may know the e-mail address of the recipient; however, the IP protocol needs the IP address.
- The **DNS client** program sends a request to a **DNS server** to **map the e-mail address to the corresponding IP address**.
- Figure on next slide shows an **example** of how a **DNS client/server** program can support an **e-mail program** to find the IP address of an e-mail recipient.

# Example of using the DNS service



# Domain Name System

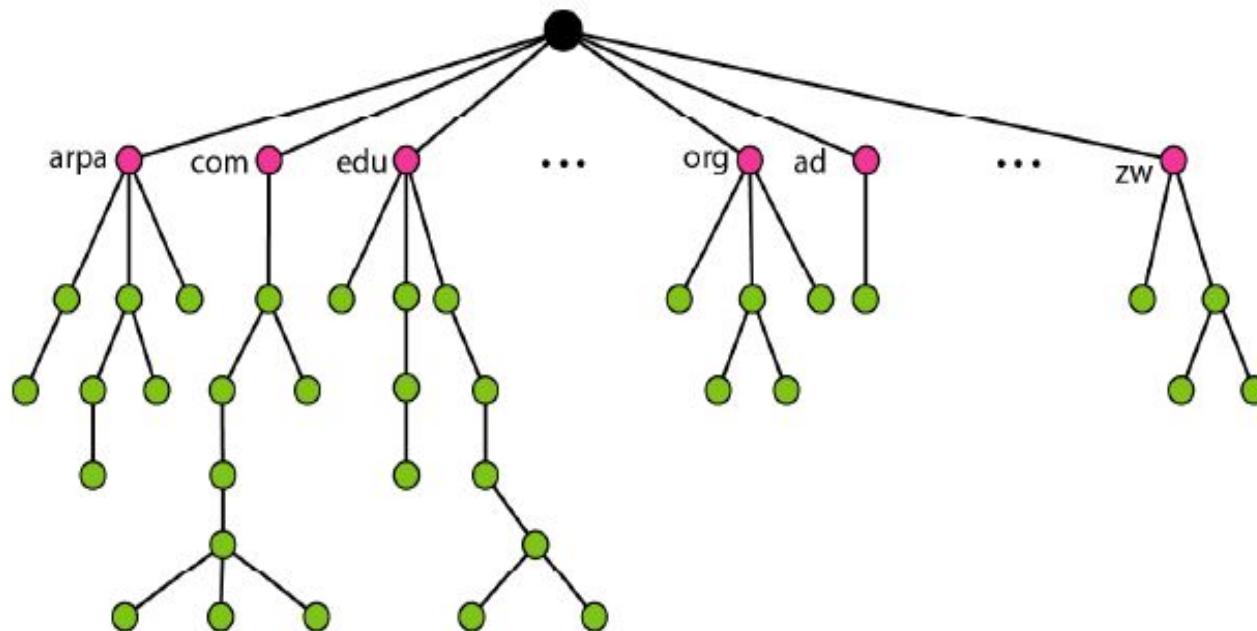
- To identify an entity, **TCP/IP protocols** use the **IP address**, which **uniquely identifies** the connection of a host to the Internet.
- However, people prefer to use names instead of numeric addresses.
- Therefore, we need a system that can **map a name to an address or an address to a name**.
- When the Internet was small, mapping was done by using a host file.
- The host file had only two columns: **name and address**.
- Every host could store the host file on its disk and update it periodically from a master host file.
- When a program or a user wanted to map a name to an address, the host consulted the host file and found the mapping.
- Today, however, it is **impossible to have one single host file to relate every address with a name and vice versa**.

# Domain Name System

- The **host file** would be **too large to store in every host**.
- In addition, it would be **impossible to update all the host files every time there was a change**.
- One **solution** would be to **store the entire host file in a single computer** and allow access to this centralized information to every computer that needs mapping.
- But it would **create a huge amount of traffic on the Internet**.
- Another solution, the one used today, is to **divide this huge amount of information into smaller parts and store each part on a different computer**.
- In this method, the host that needs mapping can contact the **closest computer holding the needed information**.
- This method is used by the **Domain Name System (DNS)**.

# DOMAIN NAME SPACE

- A **domain name space** was designed using **hierarchical name space**.
- In this design the **names are defined in an inverted-tree structure** with the **root at the top**.
- The **tree** can have only **128 levels: level 0 (root) to level 127**.



# DOMAIN NAME SPACE

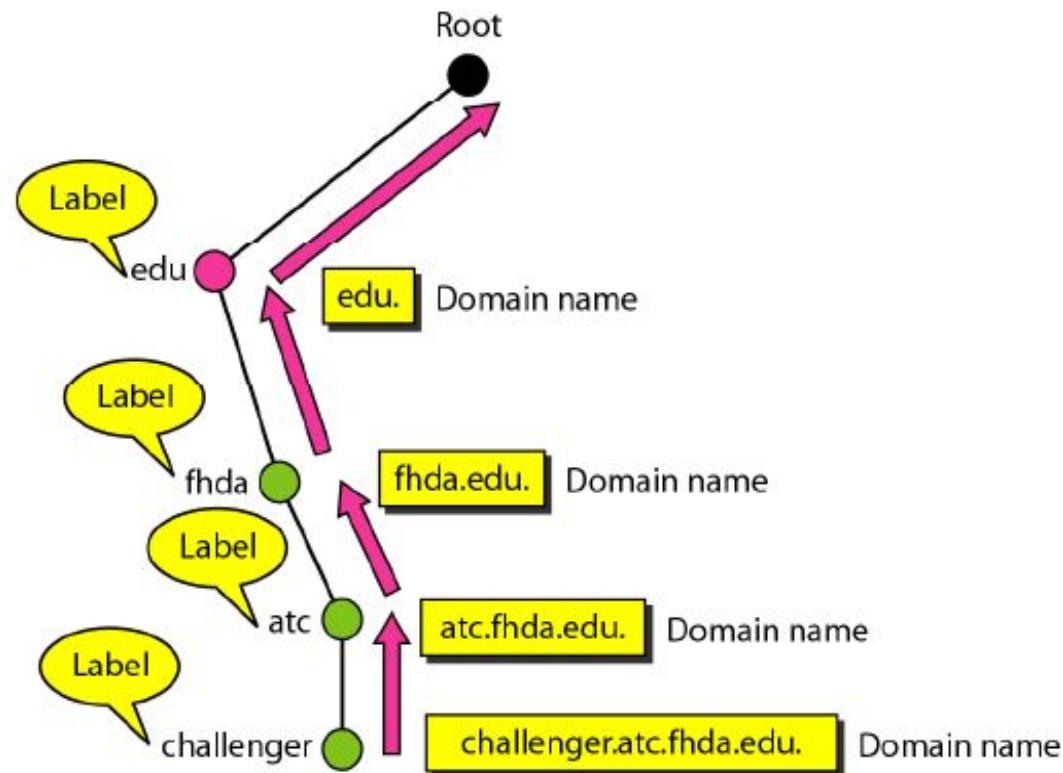
## Label

- Each **node** in the **tree** has a **label**, which is a **string** with a maximum of **63 characters**.
- The **root label** is a **null string** (empty string).
- DNS requires that **children of a node** (nodes that branch from the same node) have **different labels**, which guarantees the **uniqueness of the domain names**.

## Domain Name

- Each **node** in the **tree** has a **domain name**.
- A **full domain name** is a **sequence of labels separated by dots (.)**
- The **domain names** are always **read from the node up to the root**.
- The **last label** is the **label of the root** (null).
- This means that a **full domain name** always **ends in a null label**, which means the last character is a **dot** because the null string is nothing.

# Domain Name



# Fully Qualified Domain Name

- If a **label** is **terminated by a null string**, it is called a **fully qualified domain name (FQDN)**.
- An **FQDN** is a **domain name** that **contains the full name of a host**.
- It **contains all labels**, from the **most specific to the most general**, that uniquely define the name of the host. For example, the domain name:

**challenger.atc.fhda.edu.**

- This is the **FQDN** of a computer named ***challenger*** installed at the **Advanced Technology Center (ATC) at De Anza College**.
- A **DNS server** can only match an **FQDN** to an **address**.
- Note that the **name must end with a null label**, but because null means nothing, the label ends with a dot (.).

# Partially Qualified Domain Name

- If a **label** is not terminated by a null string, it is called a **partially qualified domain name (PQDN)**.
- A **PQDN** starts from a node, but it does not reach the root.
- It is used when the **name** to be resolved **belongs to the same site as the client**.
- Here the **resolver** can supply the **missing part**, called the **suffix**, to create an **FQDN**.
- For example, if a user at the ***fhda.edu***. site wants to get the **IP address** of the **challenger computer**, he or she can define the **partial name *challenger***
- The **DNS client** adds the **suffix *atc.fhda.edu***. before passing the address to the **DNS server**.
- The **DNS client** normally holds a list of **suffixes**.

# Partially Qualified Domain Name

- The following can be the list of suffixes at De Anza College.
- **The null suffix defines nothing.**
- This suffix is added when the user defines an **FQDN**.

atc.fhda.edu

fhda.edu

*null*

FQDN

challenger.atc.fhda.edu.  
cs.hmme.com.  
www.funny.int.

PQDN

challenger.atc.fhda.edu  
cs.hmme  
www

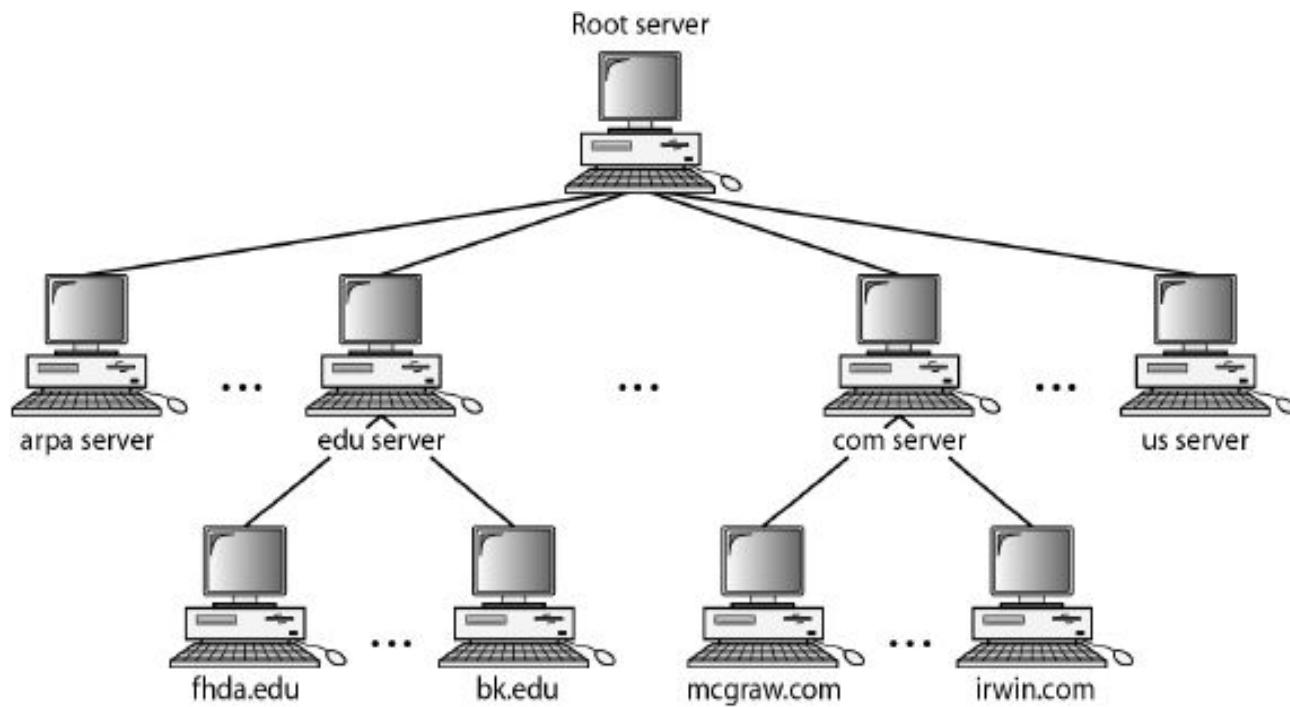
# Distribution of Name Space

- The **information** contained in the **domain name space** must be stored somewhere.
- However, it is very **inefficient** and also **unreliable** to have just **one computer** store such a **huge amount of information**.
- It is **inefficient** because responding to requests from all over the world places a **heavy load** on the system.
- It is **unreliable** because any failure makes the **data inaccessible**.

# Hierarchy of Name Servers

- The **solution** to these problems is to **distribute the information** among **many computers** called **DNS servers**.
- One way to do this is to **divide the whole space** into **many domains** based on the **first level**.
- In other words, we let the root stand alone and create as **many domains (subtrees)** as there are **first-level nodes**.
- Because a domain created in this way could be very large, **DNS** allows domains to be **divided further** into smaller domains (**subdomains**).
- Each server can be responsible (authoritative) for either a large or a small domain.
- In other words, we have a **hierarchy of servers** in the same way that we have a **hierarchy of names**.

# Hierarchy of name servers

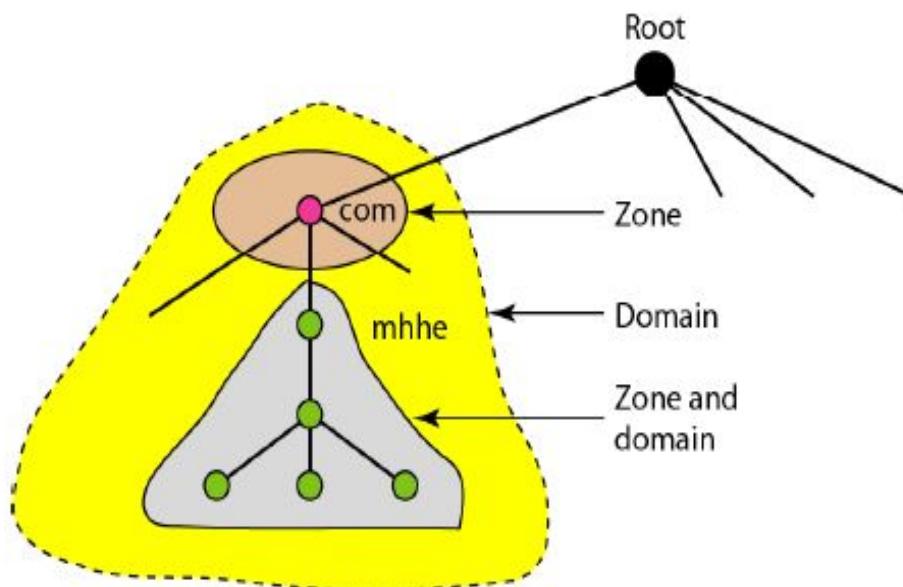


# Zone

- Since the complete domain name hierarchy cannot be stored on a **single server**, it is divided among many servers.
- *What a server is responsible for or has authority over is called a zone.*
- We can **define a zone as a contiguous part of the entire tree.**
- If a server accepts responsibility for a domain and does not divide the domain into smaller domains, the *domain* and the *zone* refer to the same thing.
- The **server** makes a **database** called a **zone file** and keeps all the information for every node under that domain.
- However, if a server divides its domain into **subdomains** and delegates part of its authority to other servers, *domain* and *zone* refer to different things.

# Zone

- The information about the nodes in the subdomains is stored in the servers at the lower levels, with the original server keeping some sort of reference to these lower-level servers.
- The original server does not free itself from responsibility totally: It still has a zone, but the detailed information is kept by the lower-level servers.



# Root Server

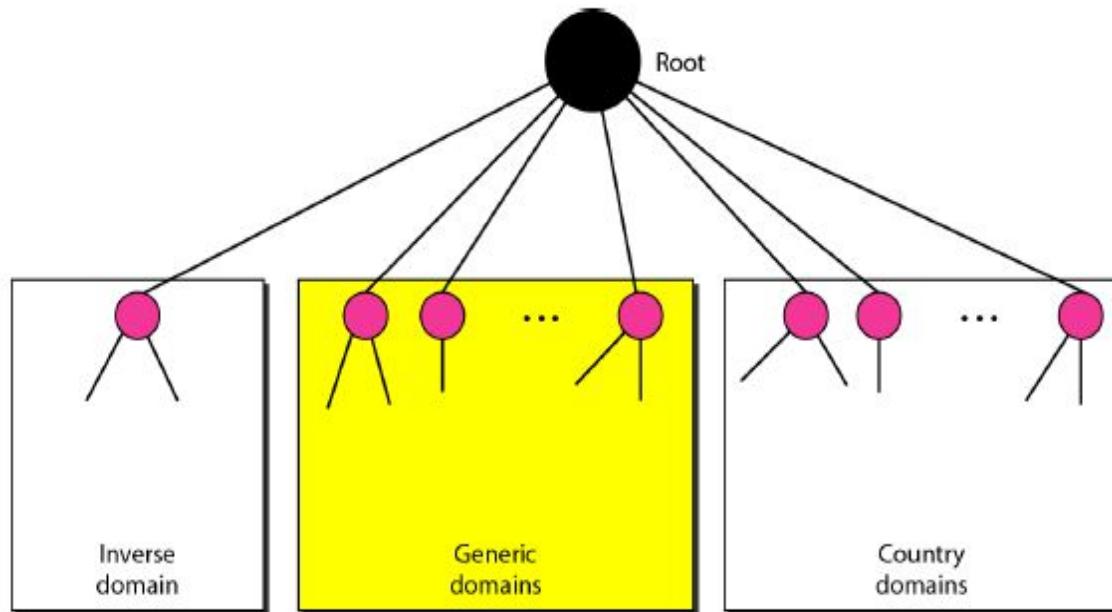
- A **root server** is a server whose **zone** consists of the **whole tree**.
- A **root server** usually does not store any information about domains but **delegates its authority to other servers**, keeping references to those servers.
- There are several root servers, each covering the whole domain name space.
- The servers are distributed all around the world.

# Primary and Secondary Servers

- DNS defines two types of servers: **primary** and **secondary**.
- A **primary server** is a server that **stores a file about the zone** for which it is an **authority**.
- It is responsible for **creating, maintaining, and updating the zone file**.
- It **stores the zone file on a local disk**.
- A **secondary server** is a server that **transfers the complete information** about a **zone from** another server (primary or secondary) and stores the file on its **local disk**.
- The **secondary server** neither creates nor updates the zone files.
- If updating is required, it must be done by the **primary server**, which sends the updated version to the secondary.

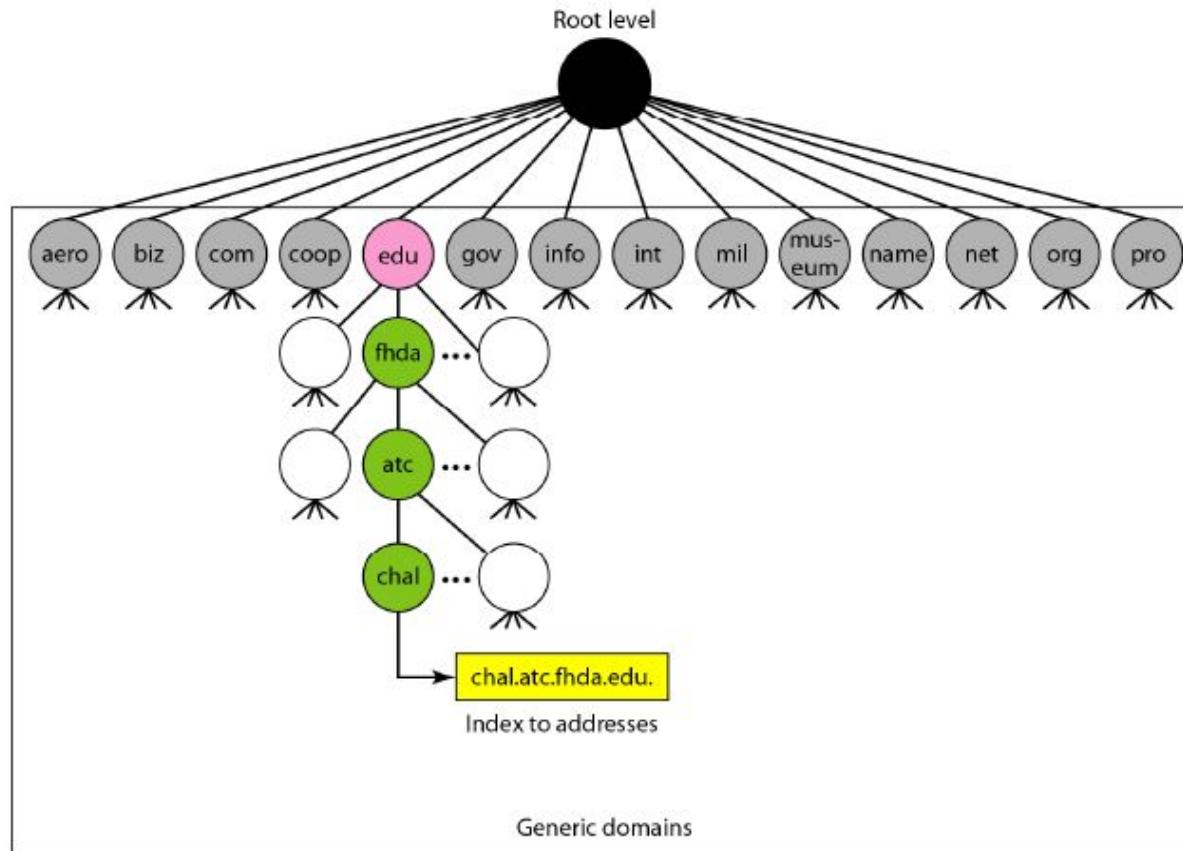
# DNS IN THE INTERNET

- **DNS is a protocol** that can be used in different platforms.
- In the Internet, the domain name space (tree) is divided into **three different sections: Generic domains, Country domains, and the Inverse domain**



# Generic Domains

- The **Generic domains** define **registered hosts** according to their **generic behavior**.
- **Each node in the tree defines a domain, which is an index to the domain name space database.**



# Generic Domains

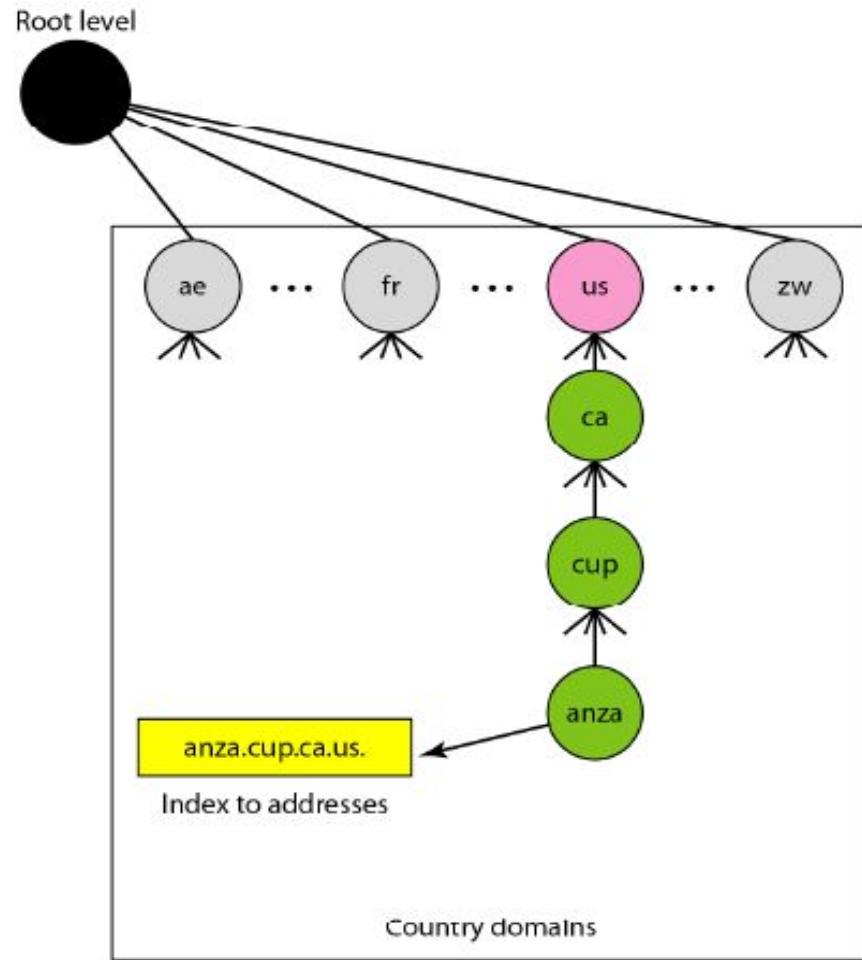
- First level in the generic domains section allows **14 possible labels**.
- These labels describe the **organization types** as listed in Table below.

<i>Label</i>	<i>Description</i>
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to "com")
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military groups
museum	Museums and other nonprofit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations

# Country Domains

- The **country domains** section uses **two-character country abbreviations** (e.g., us for United States).
- **Second labels** can be **organizational**, or they can be more specific, national designations.
- The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.).
- The address ***anza.cup.ca.us*** can be translated to **De Anza College in Cupertino, California, in the United States**.

# Country Domains



# Inverse Domain

- The **inverse domain** is used to **map an address to a name**.
- This may happen, for **example**, when a **server** has received a **request** from a **client** to do a task.
- Although the **server** has a **file** that contains a name **list of authorized clients**.
- The **server** asks its **resolver** to send a **query** to the **DNS server** to **map an address to a name** to determine if the **client** is on the **authorized list**.

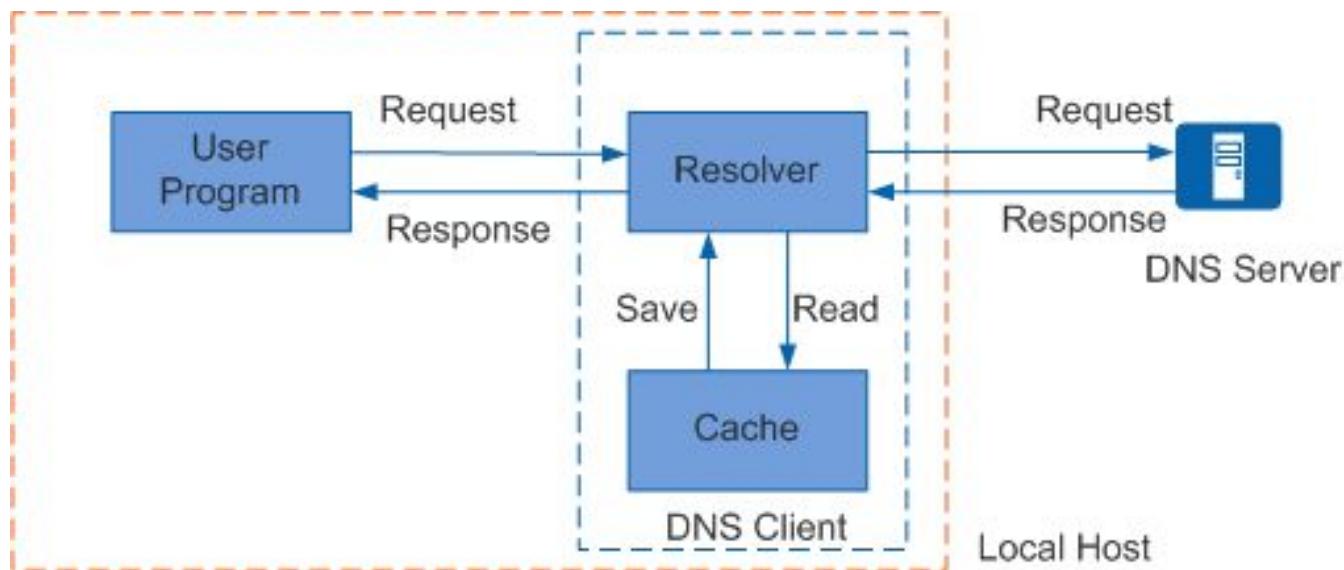
# Address Resolution

- Mapping a name to an address or an address to a name is called *name-address resolution*.

## Resolver

- **DNS is designed as a client/server application.**
- A host that needs to map an address to a name or a name to an address calls a **DNS client called a resolver**.
- The **resolver accesses the closest DNS server with a mapping request**.
- If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the information.
- After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it.

# DNS Resolver

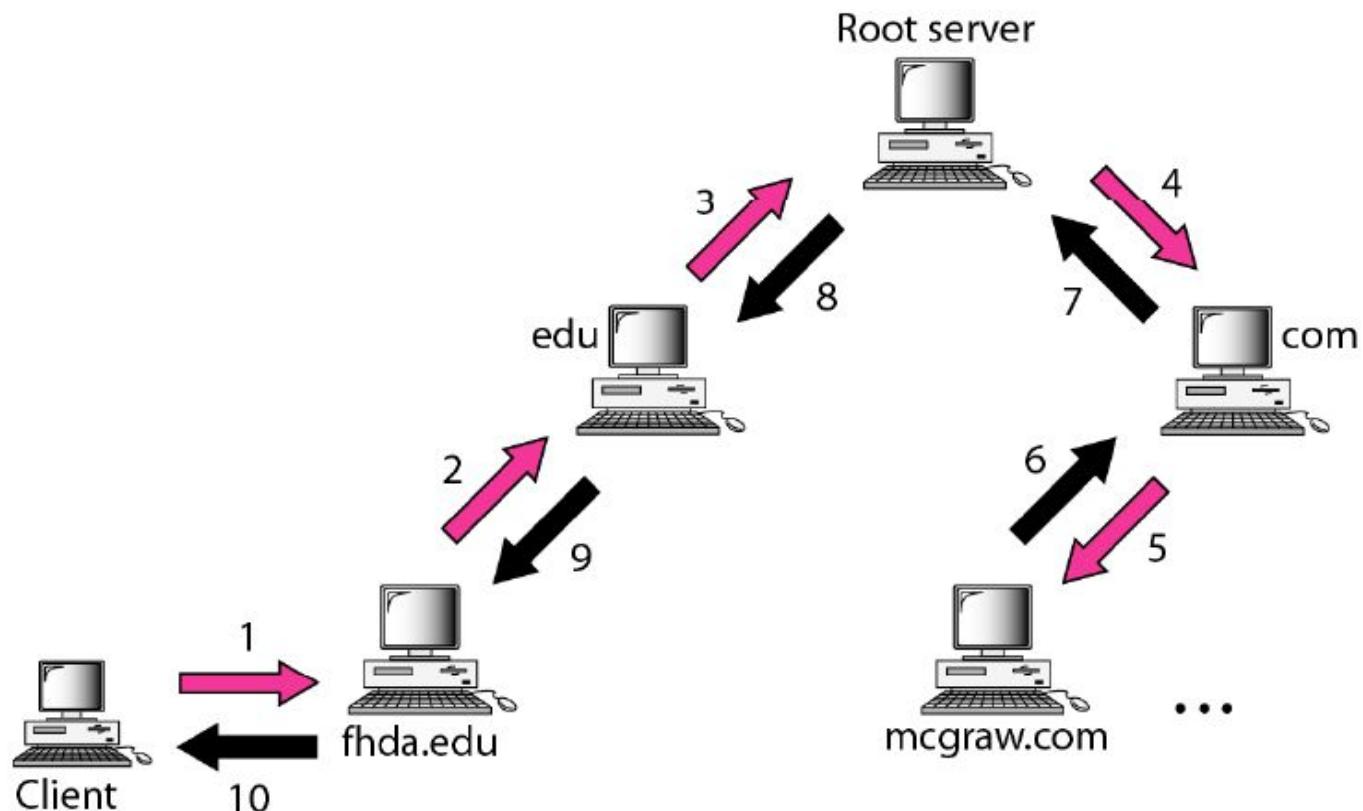


# Address Resolution

## Recursive Resolution

- The **client (resolver)** can ask for a **recursive answer** from a **name server**.
- This means that the resolver expects the server to supply the final answer.
- If the server is the authority for the domain name, it checks its database and responds.
- If the server is not the authority, it sends the request to another server (the parent usually) and waits for the response.
- If the parent is the authority, it responds; otherwise, it sends the query to yet another server.
- When the query is finally resolved, the response travels back until it finally reaches the requesting client.

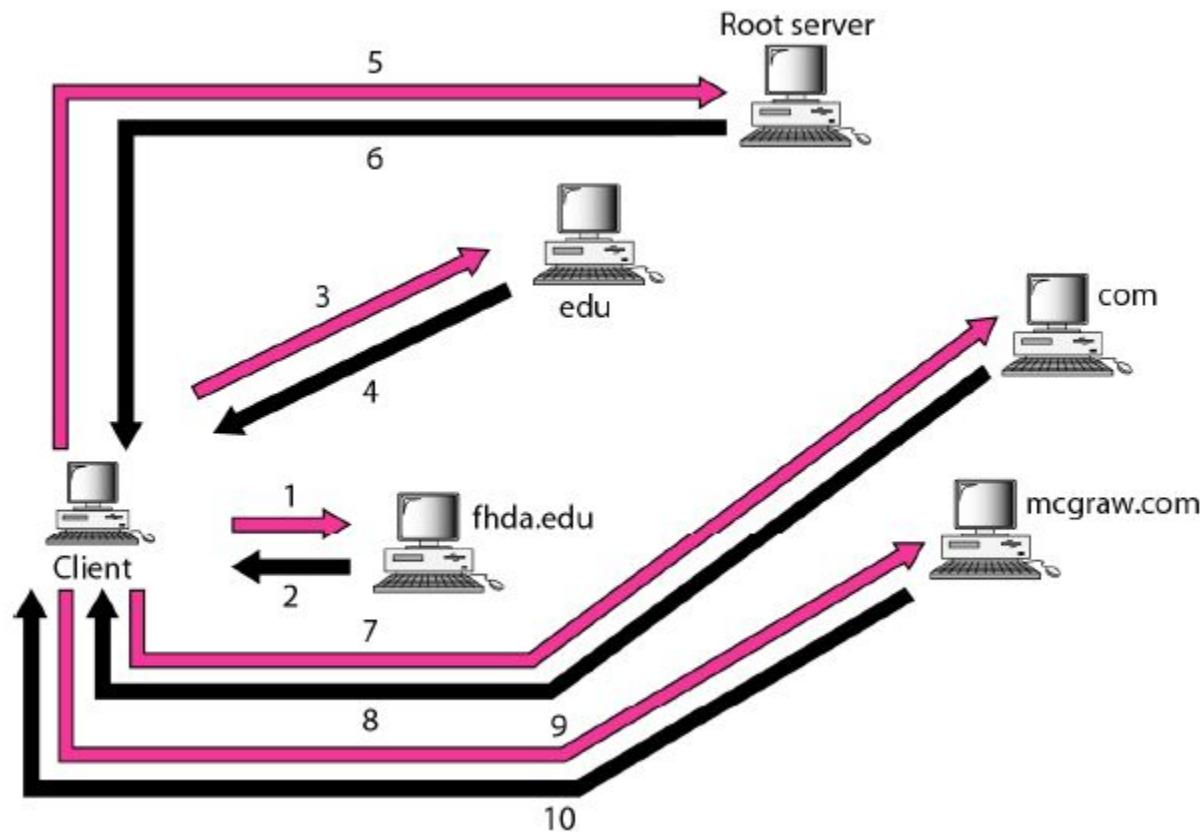
# Recursive Resolution



# Iterative Resolution

- If the client does not ask for a recursive answer, the mapping can be done iteratively.
- If the server is an authority for the name, it sends the answer.
- If it is not, it **returns** (to the client) the **IP address of the server that it thinks can resolve the query**.
- The client is responsible for repeating the query to this second server.
- If the newly addressed server can resolve the problem, it answers the query with the IP address; otherwise, it returns the IP address of a new server to the client.
- Now the client must repeat the query to the third server.
- This process is called **iterative resolution** because the client **repeats the same query to multiple servers**.

# Iterative Resolution



# Caching

- Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address.
- Reduction of this search time would increase efficiency.
- DNS handles this with a mechanism called **caching**.
- When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client.
- If the same or another client asks for the same mapping, it can check its cache memory and solve the problem.
- Caching speeds up resolution, but it can also be problematic.
- If a server caches a mapping for a long time, it may send an **outdated mapping** to the client.
- To counter this, two techniques are used.

# Caching

- **First**, the authoritative server always adds information to the mapping called ***time-to-live (TTL)***.
- It defines the time in seconds that the receiving server can cache the information.
- After that time, the mapping is **invalid** and any query must be sent again to the authoritative server.
- **Second**, DNS requires that each server keep a **TTL counter** for each mapping it caches.
- The cache memory must be searched periodically, and those mappings with an **expired TTL** must be **purged**.

# DYNAMIC DOMAIN NAME SYSTEM

- When the DNS was designed, no one predicted that there would be so many address changes.
- In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file.
- These types of changes involve a lot of manual updating.
- The size of today's Internet does not allow for this kind of manual operation.
- The DNS master file must be updated **dynamically**.
- The Dynamic Domain Name System (DDNS) therefore was devised to respond to this need.
- In DDNS, when a binding between a name and an address is determined, the information is sent, usually by DHCP to a primary DNS server.

# DYNAMIC DOMAIN NAME SYSTEM

- The primary server updates the zone.
- The secondary servers are notified either actively or passively.
- In active notification, the primary server sends a message to the secondary servers about the change in the zone, whereas in passive notification, the secondary servers periodically check for any changes.
- In either case, after being notified about the change, the secondary requests information about the entire zone (zone transfer).
- To provide security and prevent unauthorized changes in the DNS records, DDNS can use an **authentication mechanism**.

# ENCAPSULATION

- DNS can use either **UDP or TCP**. In both cases the well-known port used by the server is **port 53**.
- UDP is used when the size of the response message is less than 512 bytes because most UDP packages have a 512-byte packet size limit.
- If the size of the response message is more than 512 bytes, a **TCP** connection is used.
- In that case, one of two scenarios can occur:

## Case 1:

- If the resolver has prior knowledge that the size of the response message is more than 512 bytes, it uses the TCP connection.
- For example, if a secondary name server (acting as a client) needs a zone transfer from a primary server, it uses the TCP connection because the size of the information being transferred usually exceeds 512 bytes.

# ENCAPSULATION

## Case 2:

- If the resolver does not know the size of the response message, it can use the UDP port.
- However, if the size of the response message is more than 512 bytes, the server truncates the message and turns on the TC bit.
- The resolver now opens a TCP connection and repeats the request to get a full response from the server.
- **Note: TC Truncated** : 1 bit in DNS header, response too large for UDP (1).

# **Lecture 7.2**

## **Application Layer: Remote Login(TELNET)**

**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Introduction

- In the **Internet**, users may want to run **application programs** at a **remote site** and create results that can be transferred to their **local site**.
- For **example**, students may want to connect to their university computer lab from their home to access application programs for doing homework assignments or projects.
- One way to satisfy that demand and others is to create a client/server application program for each desired service.
- Programs such as file transfer programs (FTPs), e-mail (SMTP), and so on are currently available.
- The better solution is a **general-purpose client/server program** that lets a user **access any application program** on a **remote computer**; in other words, allow the user to log on to a remote computer.
- After logging on, a **user can use the services available on the remote computer** and transfer the results back to the **local computer**.

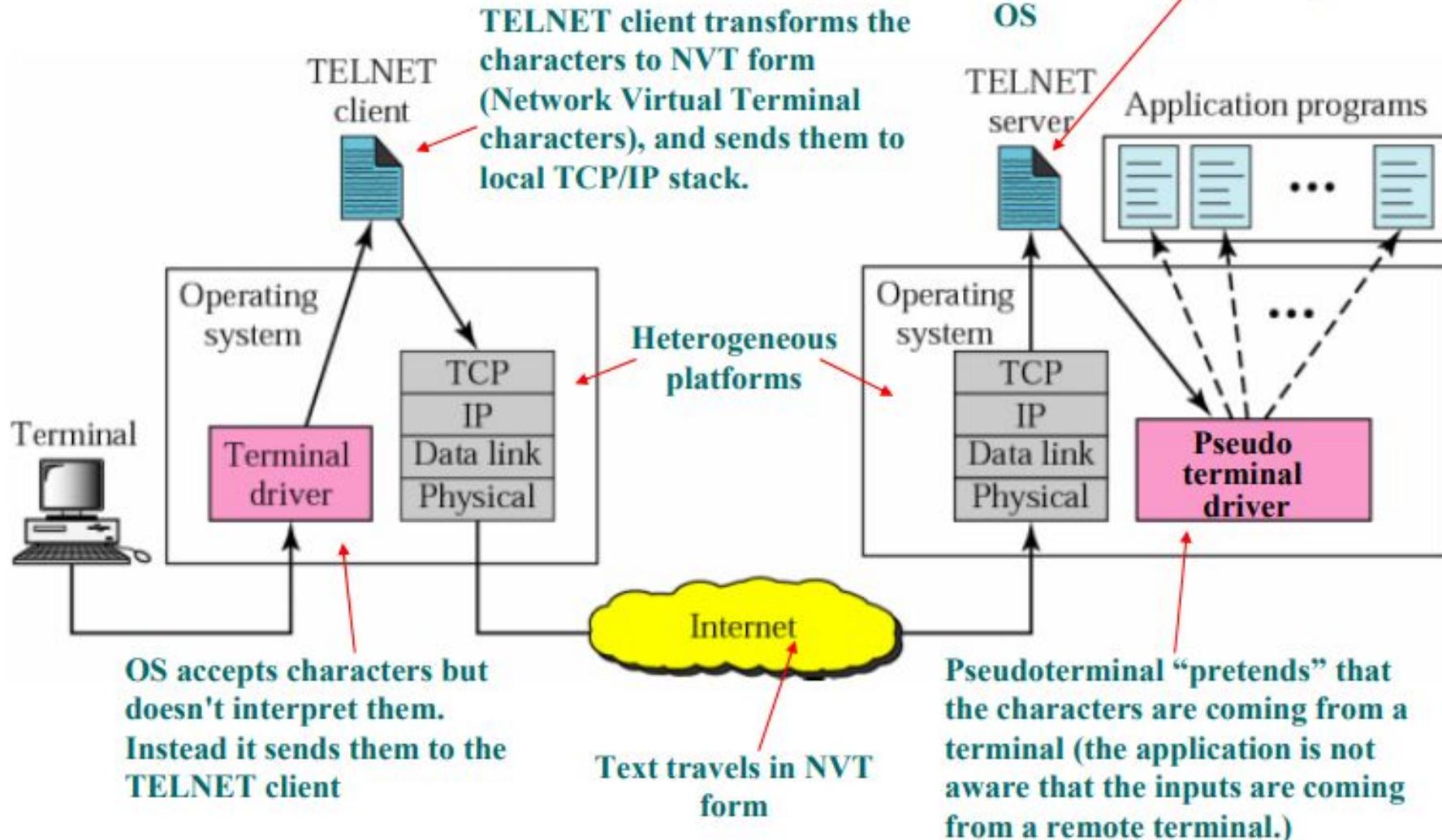
# TELNET

- TELNET is an abbreviation for *TERminal NETwork* is a client/server application program is the standard TCP/IP protocol for virtual terminal service as proposed by the International Organization for Standards (ISO).
- TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.
- When a user wants to access an application program or utility located on a remote machine, he performs remote log-in using a TELNET client and server programs.
- The user sends the keystrokes to the terminal driver, where the local operating system accepts the characters but does not interpret them.
- The characters are sent to the TELNET client, which transforms the characters to a universal character set called *network virtual terminal (NVT) characters* and delivers them to the local TCP/IP protocol stack.

# TELNET

- The **commands or text**, in NVT form, travel through the **Internet** and arrive at the **TCP/IP stack at the remote machine**.
- Here the **characters** are **delivered to the operating system** and passed to the **TELNET server**, which **changes the characters** to the corresponding **characters understandable by the remote computer**.
- However, the **char-acters cannot be passed directly** to the **operating system** because the **remote operating system** is **not designed to receive characters** from a **TELNET server**.
- It is designed to receive characters from a **terminal driver**.
- The solution is to add a piece of software called a **pseudoterminal driver** which **pretends that the characters are coming from a terminal**.
- The **operating system** then **passes the characters** to the **appropriate appli-cation program**.

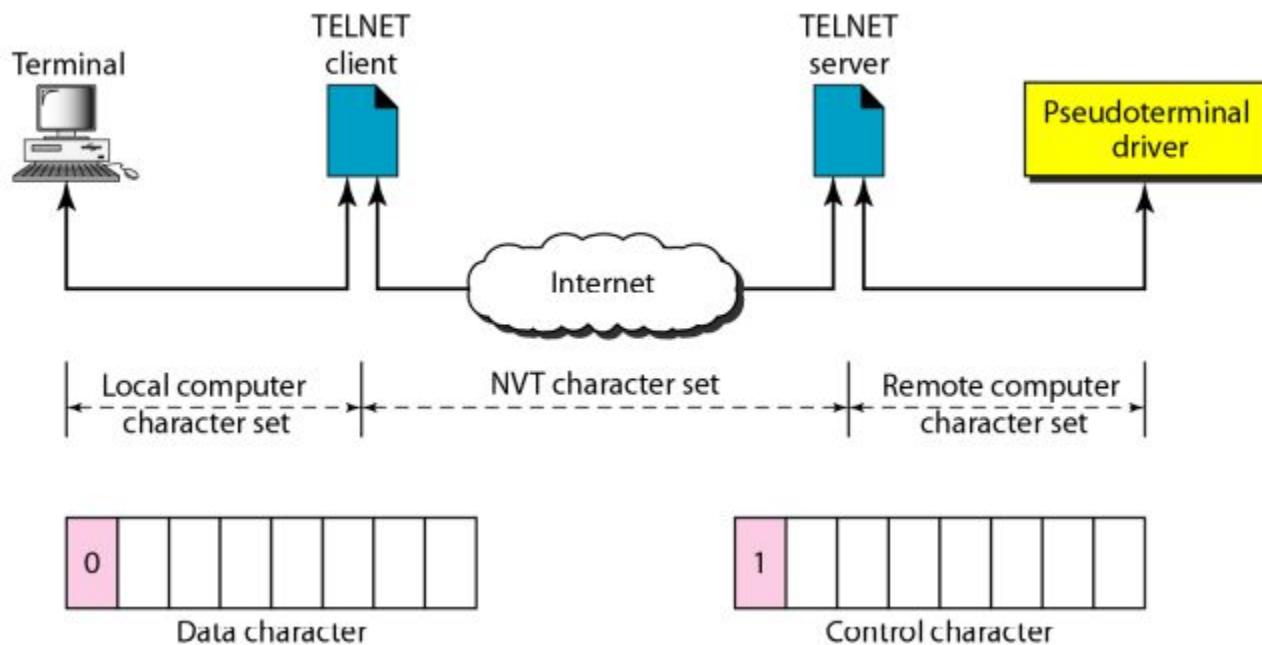
# Remote login



# Network Virtual Terminal(NVT)

- The mechanism to access a remote computer is complex because every computer and its operating system accept a special combination of characters as tokens.
- For example, the **end-of-file** token in a computer running the **DOS operating system** is **Ctrl+z**, while the **UNIX operating system** recognizes **Ctrl+d**.
- We are dealing with **heterogeneous systems**. If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer.
- **TELNET** solves this problem by defining a **universal interface** called the **network virtual terminal (NVT) character set**.
- Via this interface, the **client TELNET translates characters** (data or commands) that come from the local terminal into **NVT form** and delivers them to the network.
- The **server TELNET**, on the other hand, **translates data and commands from NVT form** into the **form acceptable by the remote computer**.

# Network Virtual Terminal



# NVT Character Set

- NVT uses **two sets of characters**, one for **data** and the other for **control**, both are **8-bit bytes**.
- For **data**, NVT is an **8-bit character set** in which the **7 lowest-order bits** are the same as **ASCII** and the **highest-order bit** is **0**.
- To send **control characters** between computers (from client to server or vice versa), NVT uses an **8-bit character set** in which the **highest-order bit is set to 1**.

# Embedding

- TELNET uses only one TCP connection.
- The server uses the well-known port 23, and the client uses an ephemeral port.
- The same connection is used for sending both data and control characters.
- TELNET accomplishes this by embedding the control characters in the data stream.
- However, to distinguish data from control characters, each sequence of control characters is preceded by a special control character called *interpret as control (IAC)*.
- For example, imagine a user wants a server to display a file (*file1*) on a remote server.
- One can type *catfile1*

# Embedding

- However, suppose the name of the file has been mistyped (*filea instead of file*).
- *The user uses the backspace key to correct this situation.*  
*catfilea<backspace>l*
- However, in the default implementation of **TELNET**, the **user cannot edit locally**; the **editing is done at the remote server**.
- **The backspace character is translated into two remote characters (IAC EC), which are embedded in the data and sent to the remote server.**
- What is sent to **the server** is shown below

c	a	t		f	i	l	e	a	IAC	EC	1
---	---	---	--	---	---	---	---	---	-----	----	---

*Typed at the remote terminal*

# Options

- **TELNET** lets the client and server negotiate options before or during the use of the service.
- **Options are extra features available to a user with a more sophisticated terminal.**
- Users with simpler terminals can use default features.

<i>Code</i>	<i>Option</i>	<i>Meaning</i>
0	Binary	Interpret as 8-bit binary transmission.
1	Echo	Echo the data received on one side to the other.
3	Suppress go ahead	Suppress go-ahead signals after data.
5	Status	Request the status of TELNET.
6	Timing mark	Define the timing marks.
24	Terminal type	Set the terminal type.
32	Terminal speed	Set the terminal speed.
34	Line mode	Change to line mode.

# Option Negotiation

- To use any of the options mentioned in the previous section first requires option negotiation between the client and the server.
- Four control characters are used for this purpose; these are shown in Table below.

Character	Decimal	Binary	Meaning
WILL	251	11111011	1. Offering to enable 2. Accepting a request to enable
WONT	252	11111100	1. Rejecting a request to enable 2. Offering to disable 3. Accepting a request to disable
DO	253	11111101	1. Approving an offer to enable 2. Requesting to enable
DONT	254	11111110	1. Disapproving an offer to enable 2. Approving an offer to disable 3. Requesting to disable

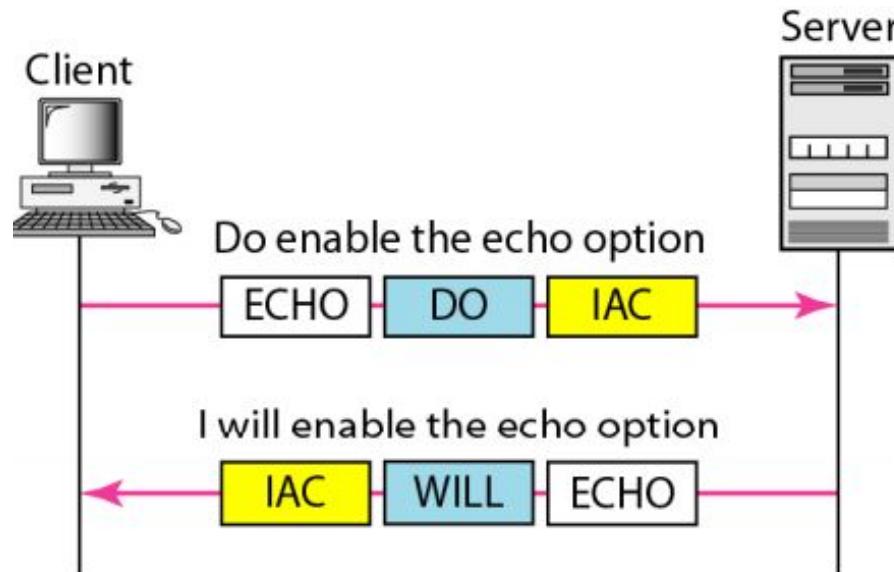
- Format of option negotiation command:

**<IAC> <{WILL|WON'T|DO|DON'T}> <option code>**

# Example of Option Negotiation

- The **client** wants the **server** to **echo each character** sent to the **server**.
- In other words, when a character is typed at the user keyboard terminal, it goes to the server and is sent back to the screen of the user.
- The **echo option** is **enabled by the server** because it is the **server** that sends the characters back to the user terminal.
- Therefore, the **client** should *request from the server the enabling* of the option **using DO**.
- The **request consists of three characters**: IAC, DO, and ECHO.
- The **server accepts the request and enables the option**.
- **Server informs the client** by sending the **three-character approval**: IAC, WILL, and ECHO.

# Option Negotiation



# Mode of Operation

- Most **TELNET** implementations operate in one of **three modes**:
  1. **Default mode**,
  2. **Character mode**,
  3. **Line mode**.

## 1. Default Mode

- The **default mode** is used if no other modes are invoked through option negotiation.
- In this mode, the **echoing is done by the client**.
- The **user types a character**, and the **client echoes the character on the screen** but does **not send it until a whole line is completed**.

# Mode of Operation

## 2. Character Mode

- In the **character mode**, each character typed is **sent by the client to the server**.
- In this mode the **echoing** of the character **can be delayed** if the **transmission time is long** (such as in a satellite connection).
- In telnet **Character mode**, only one character is transmitted at a time.
- The **server** will acknowledge the receipt of character by echoing it back to the **TELNET client**.
- The **TELNET client** will send back a **TCP ACK packet** to the **TELNET server** to inform the receipt of echo from server.
- It also **creates overhead (traffic)** for the network because **three TCP segments must be sent for each character of data**.

# Mode of Operation

## 3. Line Mode

- A new mode has been proposed to compensate for the deficiencies of the default mode and the character mode.
- In this mode, called the **line mode**, **line editing** (echoing, character erasing, line erasing, and so on) is **done by the client**.
- The **client** then **sends the whole line to the server**.

# Applications and Limitations

- **TELNET** protocol is mostly used by **network admin** to access and manage network devices remotely.
- It helps them access the device by telnetting to the IP address or hostname of a remote device.
- It allows users to access any application on a remote computer thus helps them to establish a connection to a remote system.
- **Telnet is suitable for private networks (LAN) only.**
- **Telnet is vulnerable to security attacks** as it **transfers the data in plain text .**

# **Lecture 7.3**

## **Application Layer: Electronic Mail**

**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Introduction

- **Electronic Mail** (e-mail) is one of most widely used services of Internet.
- This service allows an Internet user to send a **message in formatted manner (mail)** to the other Internet user in any part of world.
- Message in mail not only contain text, but it also contains images, audio and videos data.
- The person who is sending mail is called **sender** and person who receives mail is called **recipient**.
- **Components of Email System:**
  - UA (user agent)
  - MTA (message transfer agent)
  - MAA (message access agent)

# Protocols used in Email System

- Email basically uses two types of protocols, namely a **push protocol** and a **pull protocol**, for enabling end computers/users to **send** and **receive mails**.
- While **SMTP** (Simple Mail Transfer Protocol) is the primary **Mail Transfer Agent (MTA) protocol** used for **transferring (pushing)** mails between **end computers** and **mail servers**.
- **Mail Access Agent (MAA) protocols** like **POP3/IMAP4** are used for **retrieving (pulling)** incoming mails from the **local mail servers**.
- In the case of **web-based email**, **HTTP** is used as the **carrier protocol** in the **first and last segments** of an email, by the sender for sending the mail to the local mail server and by the receiver to retrieve the mail from the receiver's mail server.
- Since email has to be **delivered** in a **reliable** manner by the network, all email carrier protocols use **TCP** as the underlying **transport layer protocol**.

# Protocols used in Email System

- **Sending of emails** typically involves at least **four computers**, namely the **sending computer**, the **sending computer's local mail server**, the **receiving computer's mail server** and the **receiving computer**, with the network being the carrier.
- An **email sent** by an **end user** is first **transferred to its local mail server**, using **SMTP (normally) or HTTP** (in the case of web based mail alone) as the carrier protocol.
- The **local mail server** then **transfers the mail to the recipients mail server**, again using **SMTP** as the carrier protocol.
- The **mail** is then **retrieved from the receiver's mail server** by the **receiving computer** through a **pull protocol like POP3 or IMAP4 or HTTP** (web based email alone).

# User Agents (UAs)

- At the **end computers** (sender and receiver), entities known as **UAs** help the end user in **sending** and **retrieving emails**.
- Functions of **UAs** include **providing GUI or command based interfaces to compose, forward, redirect and receive mails**.
- The **UAs interact** with an **MTA client protocol** to **transfer outgoing mails** sent by the user to the **local mail server** and also **interact with an MAA client protocol** to **retrieve the user's incoming mail** from the **local mail server**.
- **Eudora, Outlook Express, elm, pine etc.** are examples of some **UAs**.

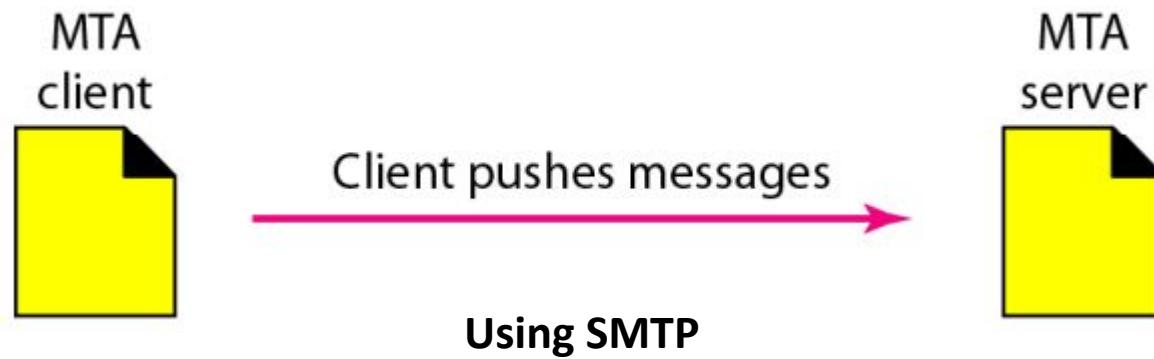
# Mail Servers

- Since the **end user computers cannot be online always**, **dedicated mail servers** do the duty of **sending and receiving mails** for multiple end users, by remaining **always on**.
- **Mail servers** typically **run both the client and the server instances** of an **MTA** protocol like **SMTP**.
- While the **MTA client** instance **protocol is used** to **send mails to remote mail servers**, the **MTA server** instance protocol is used to **receive mails from both end computers and from remote mail servers**.
- Apart from this, **mail servers** run the **server instance** of a **MAA protocol** (like **POP3/IMAP**), to **retrieve mails from the inbox of users** and **send it to the respective user's computer**, when requested through an **MAA client** instance protocol.

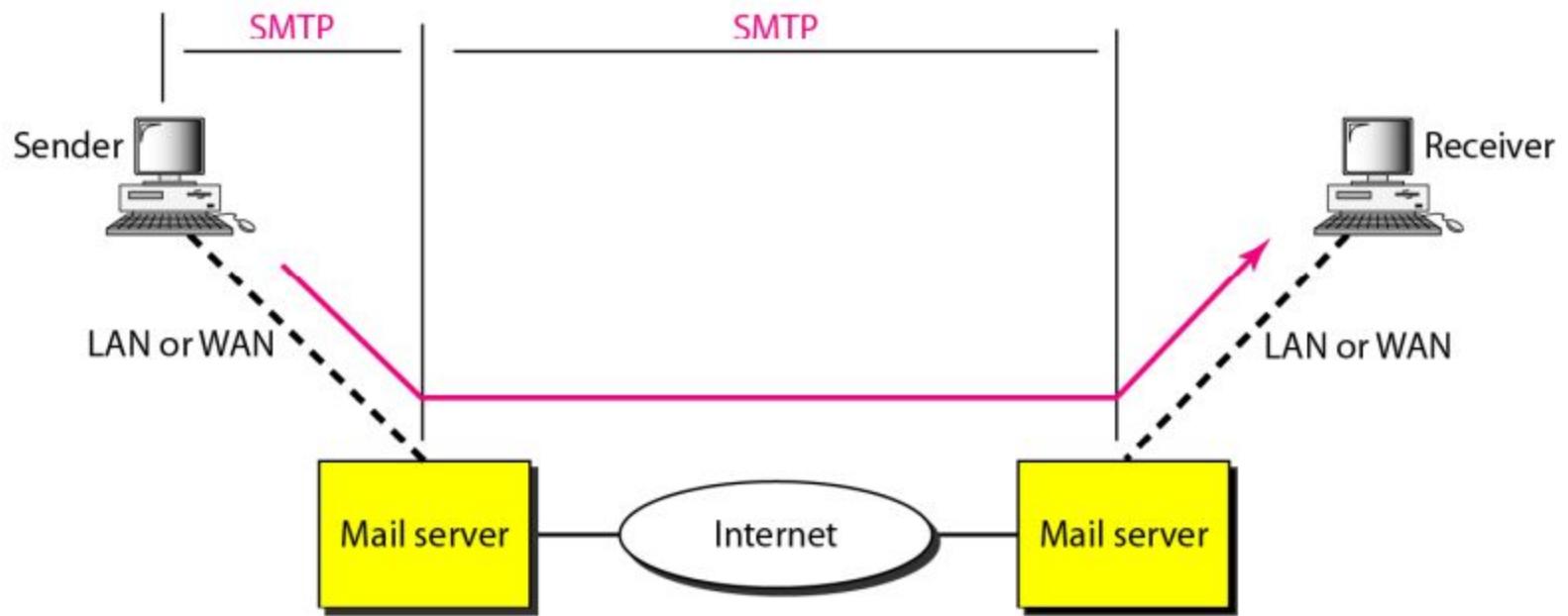
# MTA Protocol-SMTP

- SMTP is the **primary MTA protocol** used **between mail servers** and also **between end clients and mail servers**, for **sending mails**.
- **SMTP is a**
  - Is a **simple text-based protocol** that enables users to **send emails**.
  - It is a **push protocol** because it is used to push an email from a sender to the receiver.
  - Is an **application layer protocol** running on top of **TCP**.
  - **SMTP server typically waits on TCP port number 25**.
  - Though it **supports sending mails only in ASCII format**, it allows extensions in the form of **MIME** (Multipurpose Internet Mail Extensions) for carrying a wide variety of data formats including **binary, images, audio, video** etc.

# Push operation in SMTP



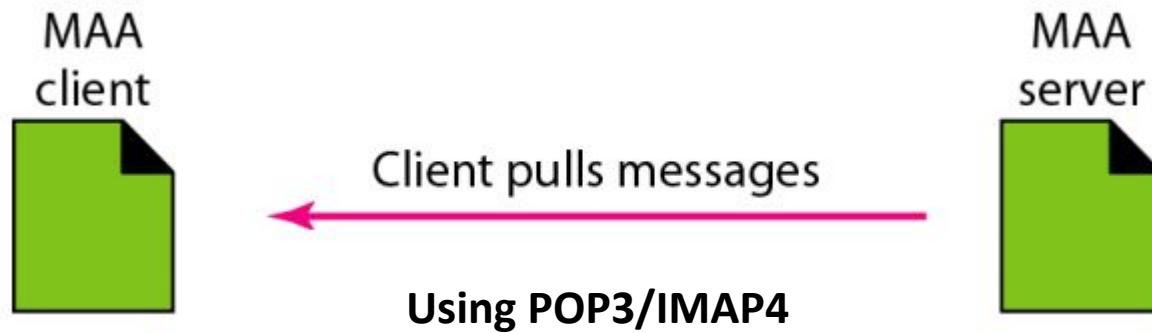
# SMTP Range



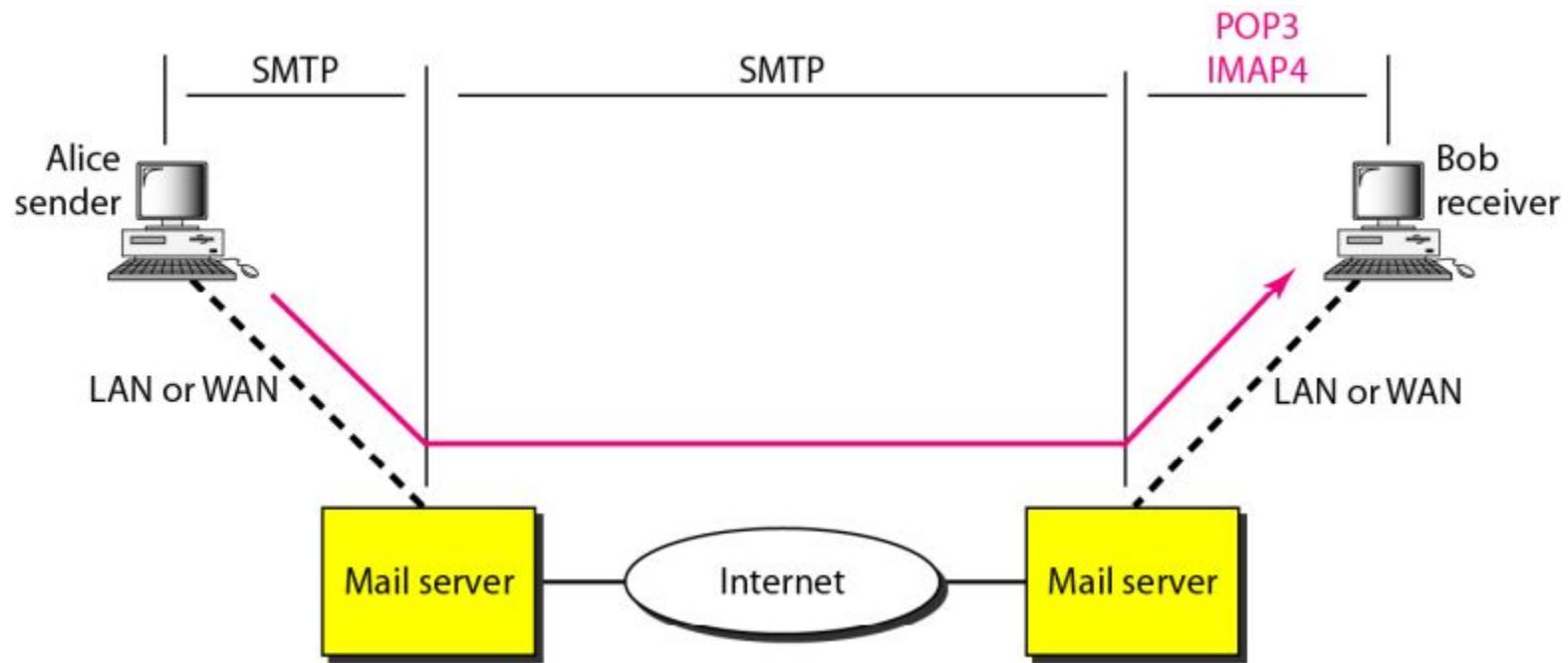
# MAA Protocols (POP3/IMAP)

- The **MAA protocols** like **POP3(Post Office Protocol, version 3)/IMAP4(Internet Mail Access Protocol, version 4)** are used in the **last hop** of an **email** for retrieving incoming mails by the end user from his/her local mail server.
- While **both POP3 and IMAP are pull protocols**, used for retrieving mails from the local mail server to the receiving computer's Inbox, **IMAP is a more powerful protocol than POP3.**
- **IMAP** supports **additional features** like creation and management of multiple folders in the mail server, accessing mails from multiple end points etc.

# Pull operation in POP3/IMAP4



# POP3/IMAP4 Range



# POP3

- Post Office Protocol, version 3 (POP3) is simple and limited in functionality.
- The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the recipient mail server.
- Mail access starts with the client when the user needs to download e-mail from the mailbox on the mail server.
- The MAA client opens a connection to the MAA server on TCP port 110.
- It then sends its user name and password to access the mailbox.
- The user can then list and retrieve the mail messages, one by one.
- POP3 has two modes: the delete mode and the keep mode.
- In the delete mode, the mail is deleted from the mailbox after each retrieval.
- In the keep mode, the mail remains in the mailbox after retrieval.

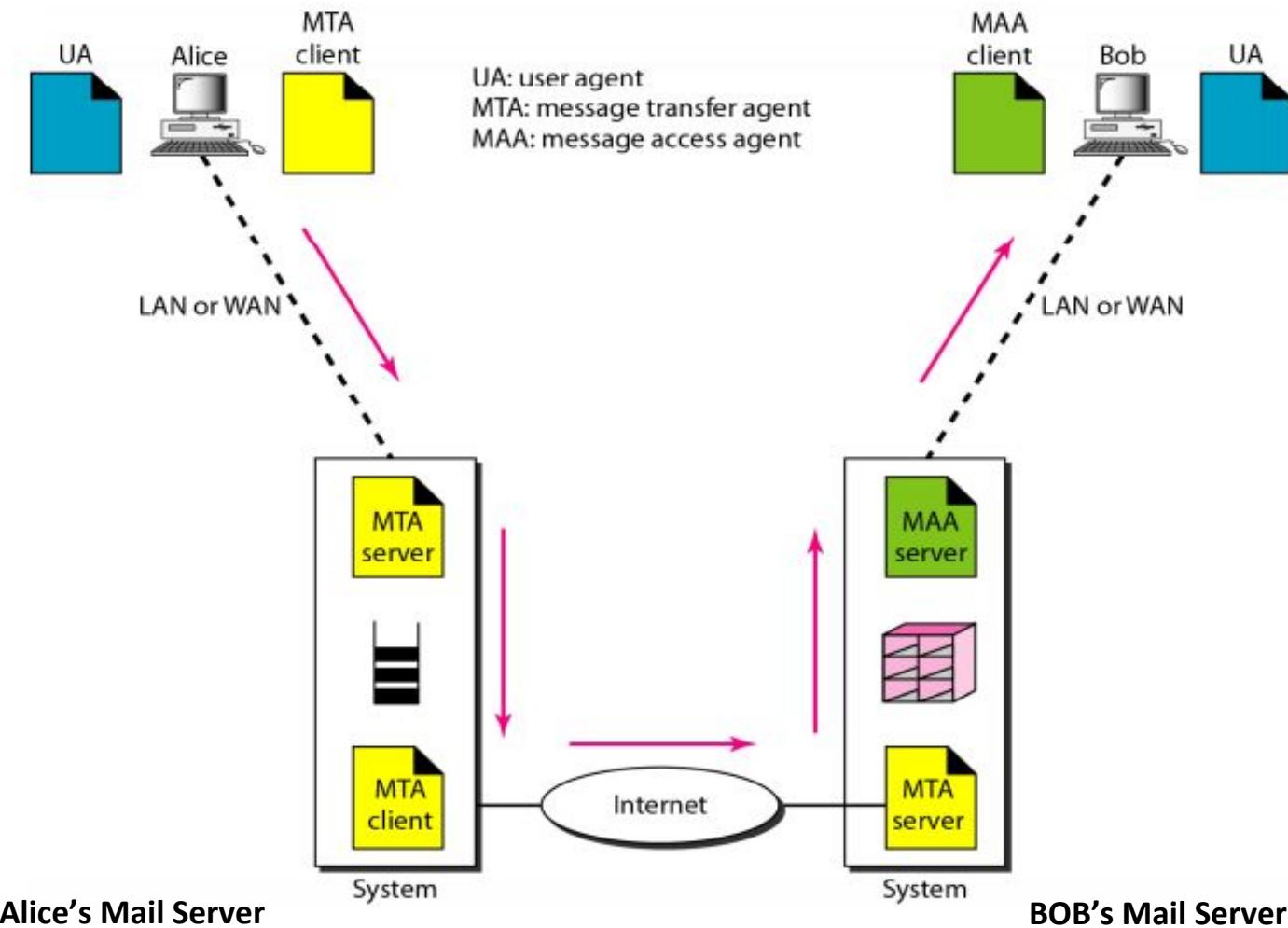
# Limitations of POP3

- It does not allow the user to organize her mail on the server.
- If the emails are downloaded from the server, then all the mails are deleted from the server by default.
- So, mails cannot be accessed from other machines unless they are configured to leave a copy of the mail on the server.
- Since all the attachments are stored on your local machine, there is a high risk of a virus attack if the virus scanner does not scan them. The virus attack can harm the computer.
- The user cannot have different folders on the server.
- In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.

# **IMAP4(Internet Mail Access Protocol, version 4)**

- **IMAP4** provides the following **extra features**:
- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
- A user can create, delete, or rename mailboxes on the mail server.
- A user can create a hierarchy of mailboxes in a folder for e-mail storage.

# Email System Architecture



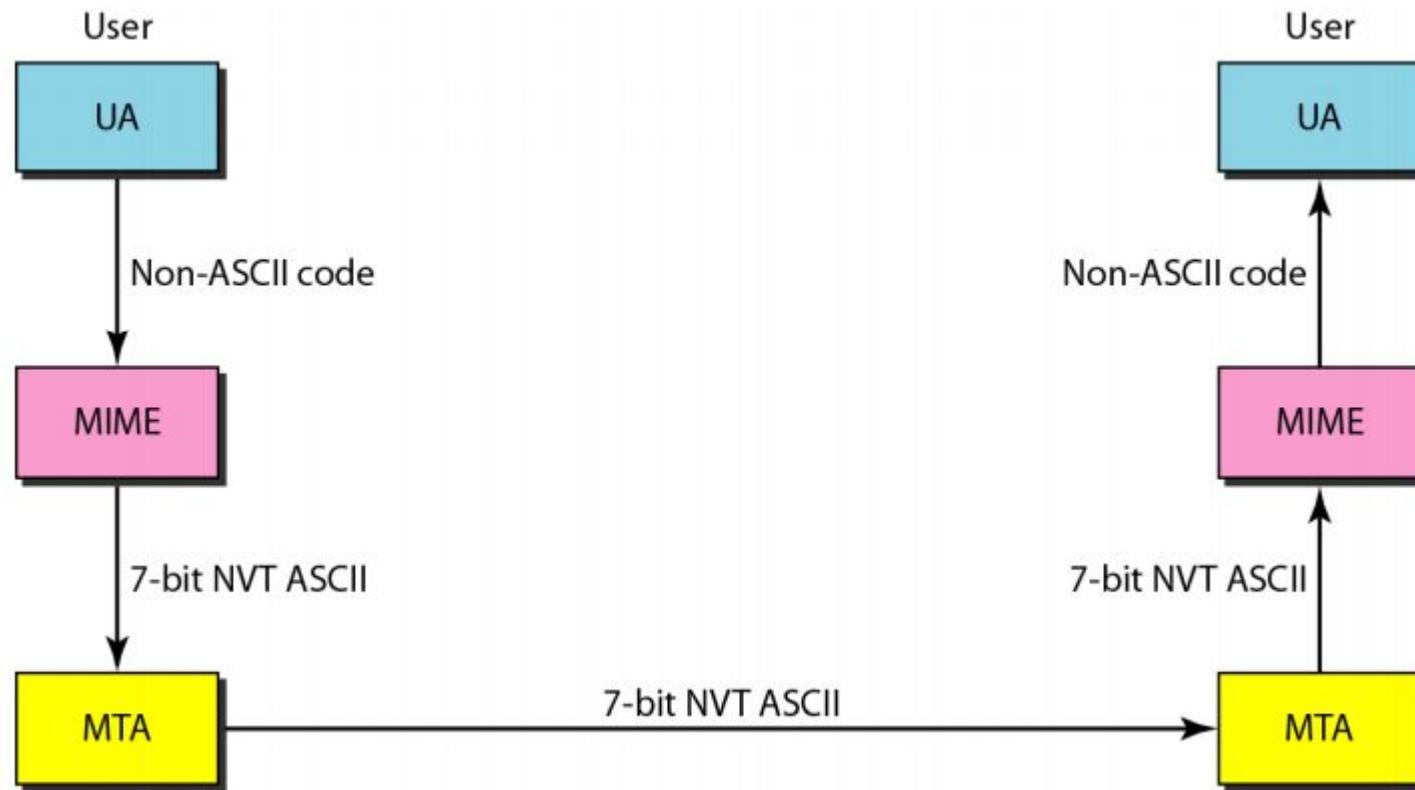
# Web-Based Mail

- E-mail is such a common application that some websites today provide this service to anyone who accesses the site.
- Some common sites are Gmail, Hotmail and Yahoo.
- Mail transfer from sender's browser(MTA) to her mail server is done through **HTTP**.
- The transfer of the message from the sending mail server to the receiving mail server is still through **SMTP**.
- Finally, the message from the receiving server (the Web server) to receiver's browser(MAA) is done through **HTTP**.
- In last phase instead of POP3 or IMAP4, **HTTP** is normally used.
- When receiver needs to retrieve his e-mails, he sends a message to the website (Gmail, for example).
- The website sends a form to be filled in by Bob, which includes the log-in name and the password.
- If the **log-in name and password** match, the **e-mail is transferred** from the Web server to receiver's browser in HTML format.

# Multipurpose Internet Mail Extensions

- Electronic mail send messages **only in NVT 7-bit ASCII format**.
- Thus it **cannot be used for languages that are not supported by 7-bit ASCII characters** (such as French, German, Hebrew, Russian, Chinese, and Japanese).
- Also, it **cannot be used to send binary files or video or audio data**.
- **Multipurpose Internet Mail Extensions (MIME)** is a **supplementary protocol** that **allows non-ASCII data to be sent through e-mail**.
- **MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client MTA to be sent through the Internet.**
- The **message at the receiving side is transformed back to the original data**.
- We can think of **MIME** as a **set of software functions that transforms non-ASCII data (stream of bits) to ASCII data and vice versa**.

# Multipurpose Internet Mail Extensions (MIME)



# **Lecture 7.4**

## **Application Layer: SNMP**

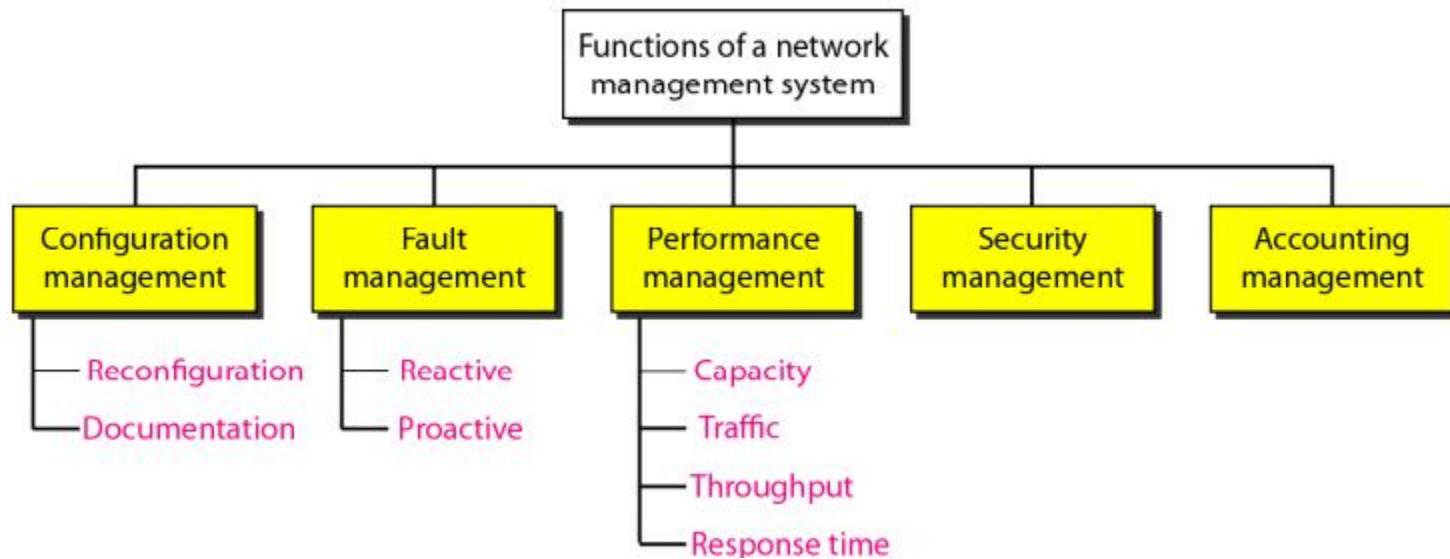
**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# Network Management

- We can define **network management** as **monitoring, testing, configuring, and troubleshooting network components** to meet a set of requirements defined by an organization.
- These requirements include the **smooth, efficient operation of the network** that provides the predefined quality of service for users.
- To accomplish this task, a network management system uses hardware, software, and humans.
- The **functions performed by a network management system** can be divided into **five** broad categories:
  1. Configuration management,
  2. Fault management,
  3. Performance management,
  4. Security management,
  5. Accounting management,

# Functions of a network management system



# 1.Configuration Management

- A large network is usually made up of hundreds of entities that are physically or logically connected to one another.
- These entities have an initial configuration when the network is set up, but can change with time.
- Desktop computers may be replaced by others; application software may be updated to a newer version; and users may move from one group to another.
- The **configuration management system must know, at any time, the status** of each entity and its relation to other entities.
- Configuration management can be divided into two subsystems:
  - **Reconfiguration:** Hardware reconfiguration, Software reconfiguration and User-account reconfiguration
  - **Documentation:** original network configuration and each subsequent change must be recorded meticulously.

## 2. Fault Management

- Complex networks today are made up of hundreds and sometimes thousands of components.
- Proper operation of the network depends on the proper operation of each component individually and in relation to each other.
- **Fault management is the area of network management that handles this issue.**
- An effective fault management system has two subsystems:
  - reactive fault management
  - proactive fault management.
- A reactive fault management system is responsible for detecting, isolating, correcting, and recording faults.
- It handles short-term solutions to faults.
- Proactive fault management tries to prevent faults from occurring.

# Functions of a Network Management System

## 3. Performance Management

- Performance management, which is closely related to fault management, tries to **monitor and control the network** to ensure that it is running as **efficiently** as possible.
- Performance management tries to quantify performance by using some measurable quantity such as **capacity, traffic, throughput, or response time**.

## 4. Security Management

- Security management is responsible for **controlling access to the network** based on the **predefined policy**.

# Functions of a network management system

## 5. Accounting Management

- Accounting management is the **control of users' access to network resources through charges.**
- Under accounting management, individual users, departments, divisions, or even projects are charged for the services they receive from the network.
- Today, organizations use an accounting management system for the following reasons:
  - It prevents users from monopolizing limited network resources.
  - It prevents users from using the system inefficiently.
  - Network managers can do short- and long-term planning based on the demand for network use.

# **SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)**

- Simple Network Management Protocol (SNMP) is a widely used protocol designed to facilitate the management of networked devices from a central location.
- The Simple Network Management Protocol (SNMP) is a framework for managing devices in an internet using the **TCP/IP protocol suite**.
- It provides a set of fundamental operations for **monitoring and maintaining** an internet.
- **SNMP uses the services of UDP on two well-known ports, 161 and 162.**
- The well-known port 161 is used by the server (agent), and the well-known port 162 is used by the client (manager).
- Devices that typically support SNMP include **routers, switches, servers, workstations, printers, modem racks, and more.**
- SNMP is an application-level protocol in which a few manager stations control a set of agents.

# **SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)**

- The **SNMP protocol** is designed at the **application level** so that it can **monitor devices made by different manufacturers** and installed on **different physical networks**.
- In other words, SNMP frees management tasks from both the physical characteristics of the managed devices and the underlying networking technology.
- It can be used in a **heterogeneous internet** made of different LANs and WANs connected by routers made by different manufacturers.

# SNMP Architecture

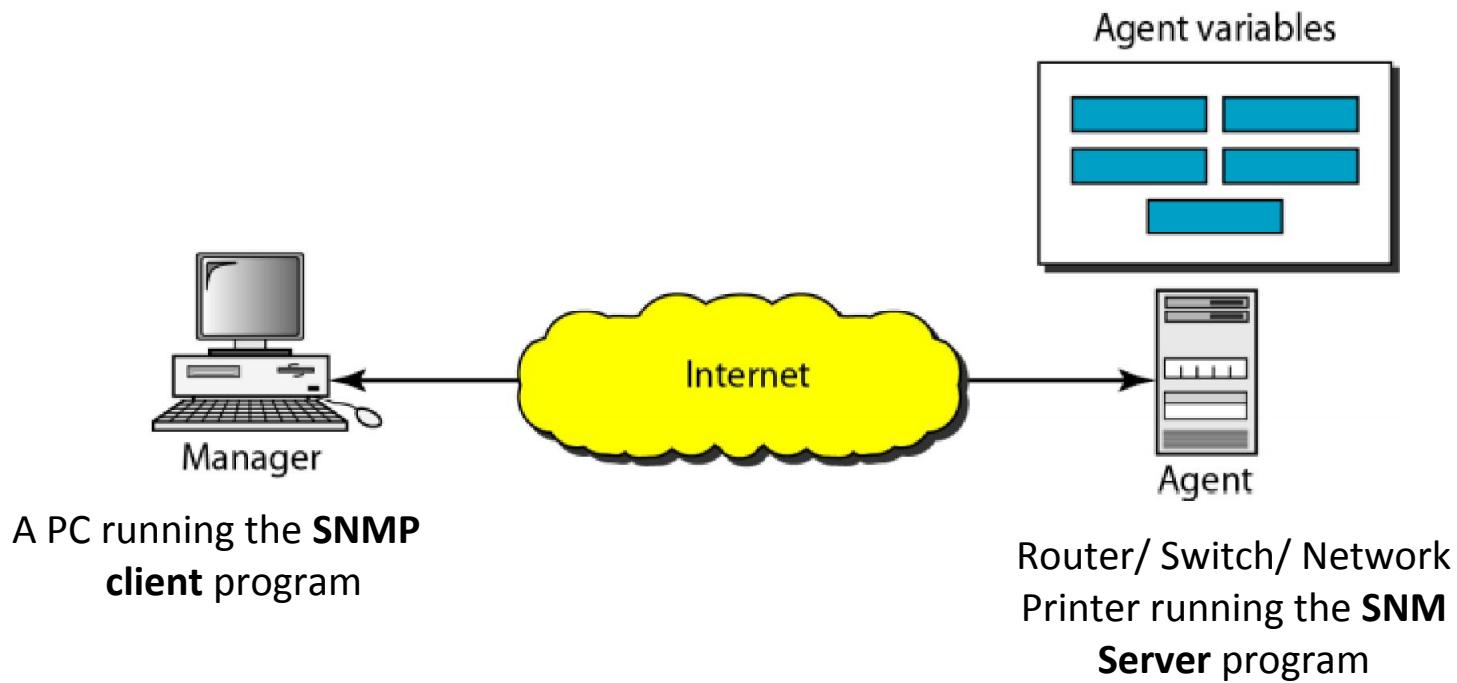
The **SNMP architecture** is composed of three major elements:

1. Managers
  2. Agents
  3. MIBs (Management Information Base)
- A **management station**, called a **manager**, is a host that **runs the SNMP client program**.
  - A **managed station**, called an **agent**, is a router (or a host) that **runs the SNMP server program**.
  - Management is achieved through simple **interaction between a manager and an agent**.
  - The **agent** keeps **performance information in a database**.
  - The **manager** has access to the values in the database.

# SNMP Architecture

- For example, a **router(Agent)** can store in appropriate **variables** the **number of packets received and forwarded**.
- The **manager** can **fetch and compare the values of these two variables** to see if the **router is congested or not**.
- The **manager** can also **make the router perform certain actions**.
- For example, a router periodically checks the value of a reboot counter to see when it should reboot itself.
- It **reboots itself**, for example, if the value of the counter is 0, the manager can use this feature to reboot the agent remotely at any time.
- It simply sends a packet to force a 0 value in the counter.

# SNMP Architecture

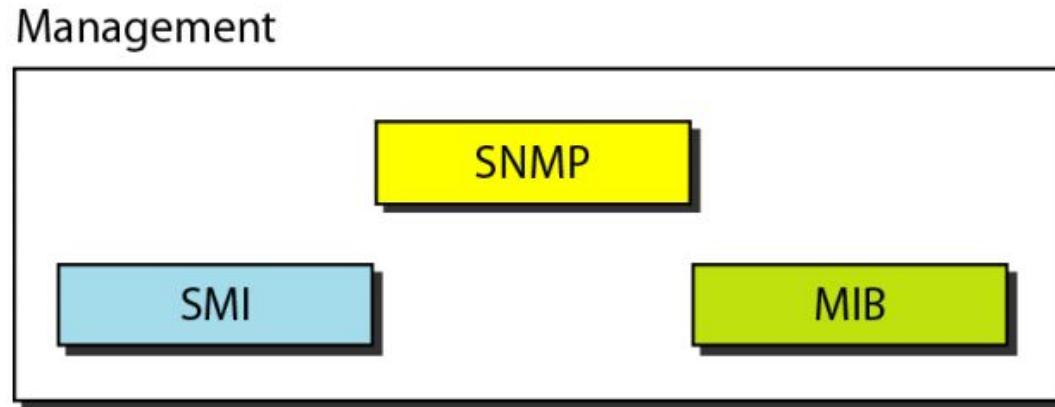


# SNMP Architecture

- Agents can also contribute to the management process.
- The server program running on the agent can check the environment, and if it notices something unusual, it can send a warning message, called a trap, to the manager.
- In other words, management with SNMP is based on three basic ideas:
  1. A manager checks an agent by requesting information that reflects the behaviour of the agent.
  2. A manager forces an agent to perform a task by resetting values in the agent database.
  3. An agent contributes to the management process by warning the manager of an unusual situation.

# Management Components

- To do management tasks, **SNMP** uses **two other protocols**:
  1. **Structure of Management Information (SMI)**
  2. **Management Information Base (MIB).**
- In other words, management on the Internet is done through the **cooperation of the three protocols SNMP, SMI, and MIB**, as shown in Figure below.



# Role of SNMP

- SNMP has some **very specific roles** in **network management**.
- It **defines the format** of the packet to be **sent** from a **manager** to an **agent** and vice versa.
- It also **interprets the result** and **creates statistics** (often with the help of other management software).
- The **packets exchanged** between a **manager** and an **agent** contain the **object (variable) names** and their **status (values)**.
- SNMP is responsible for **reading and changing** these values.

# Roles of SMI

- SMI defines the **general rules** for **naming objects**, **defining object types** (including range and length), and showing how to encode objects and values.
- SMI does not define the number of objects an entity should manage or name the objects to be managed or define the association between the objects and their values.
- SMI functions are:
  1. To **name objects**
  2. To **define the type of data** that can be stored in an object
  3. To show **how to encode data** for transmission over the network
- SMI is a **guideline** for SNMP. It emphasizes three attributes to handle an object: name, data type, and encoding method

# Roles of MIB

- For each entity to be managed, this protocol must **define the number of objects, name them according to the rules defined by SMI, and associate a type to each named object .**
- **MIB creates a collection of named objects, their types, and their relationships to each other** in an entity to be managed.
- Each **agent** has its **own MIB**, which is a **collection of all the objects that the manager can manage.**
- The objects in MIB are categorized under 10 different groups: system, interface, address translation, ip, icmp, tcp, udp, egp, transmission, and snmp.
- **SNMP** stores, changes, and interprets the values of objects already declared by **MIB** according to the rules defined by **SMI**.

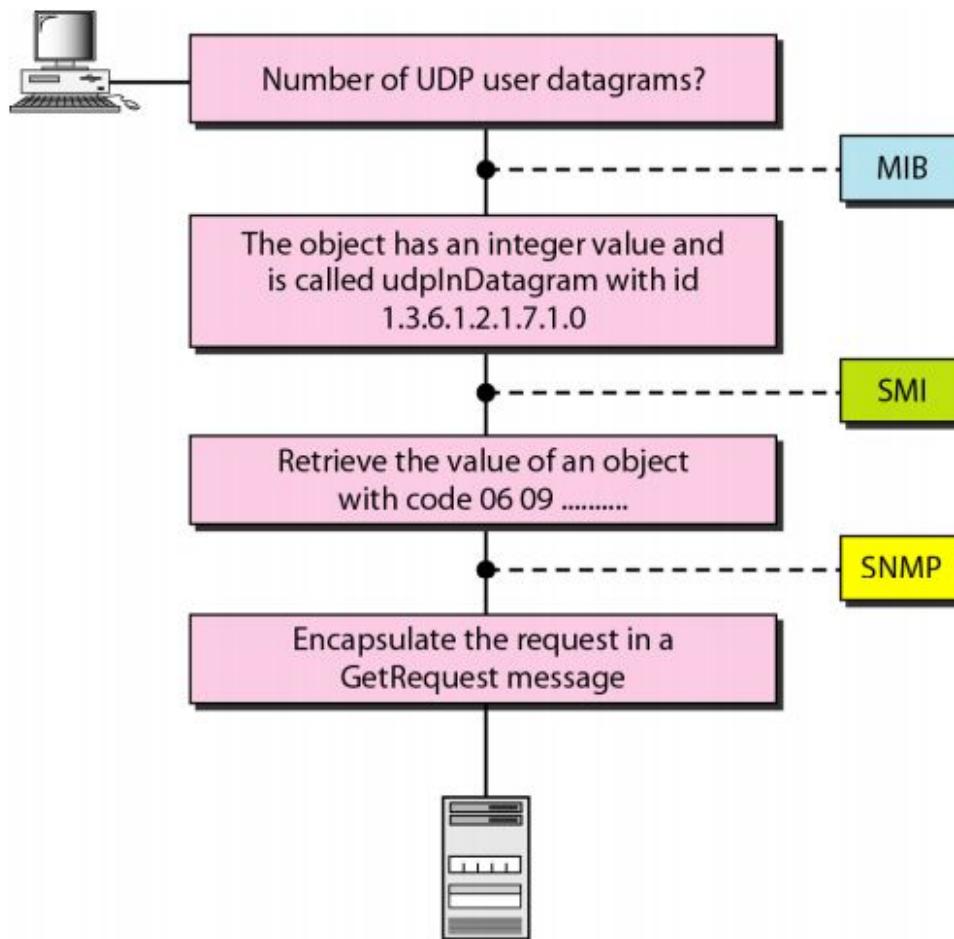
# Network Management Analogy

- We can compare the task of network management to the task of writing a program.
- Both tasks need rules. In network management this is handled by SMI.
- Both tasks need variable declarations. In network management this is handled by MIB.
- Both tasks have actions performed by statements. In network management this is handled by SNMP.

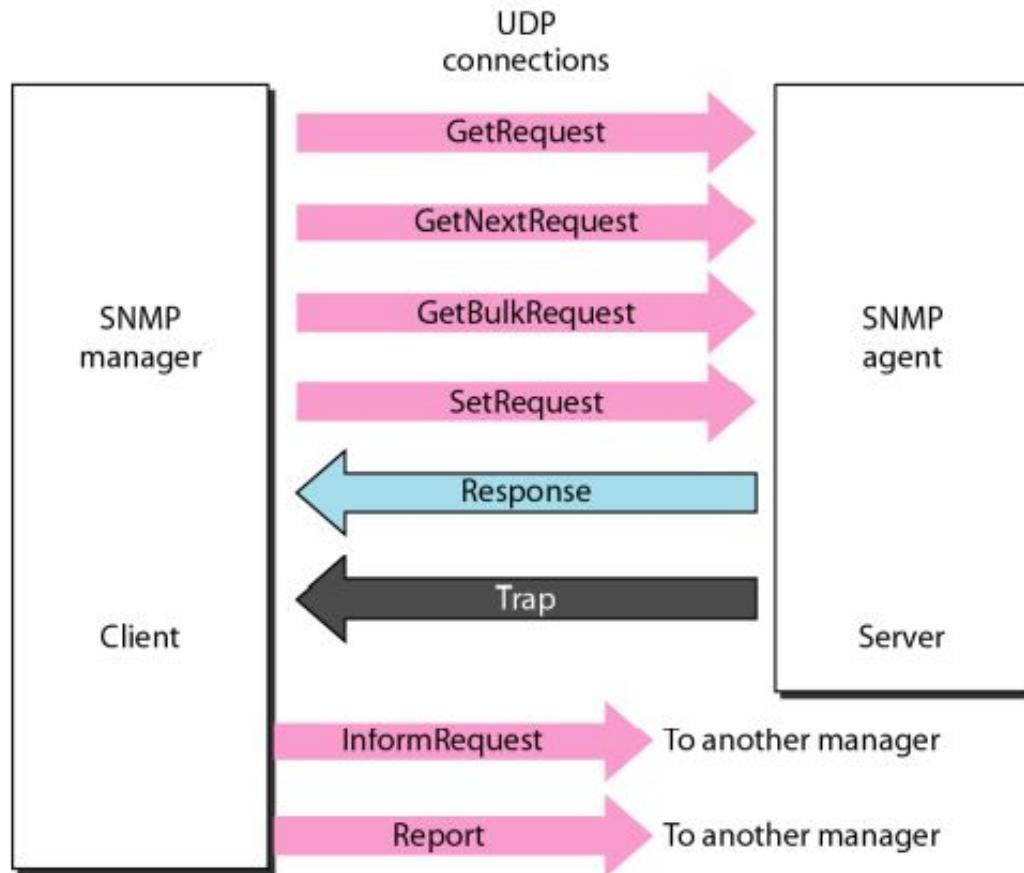
# Example

- A manager station (SNMP client) wants to send a message to an agent station (SNMP server) to find the number of UDP user datagrams received by the agent.
- **MIB** is responsible for **finding the object that holds the number of the UDP user datagrams received**.
- **SMI**, with the help of another embedded protocol, is responsible for **encoding the name of the object**.
- SNMP is responsible for creating a message, called a **GetRequest message**, and encapsulating the encoded message.

# Example



# SNMP Message Exchange



# **Lecture 8.1**

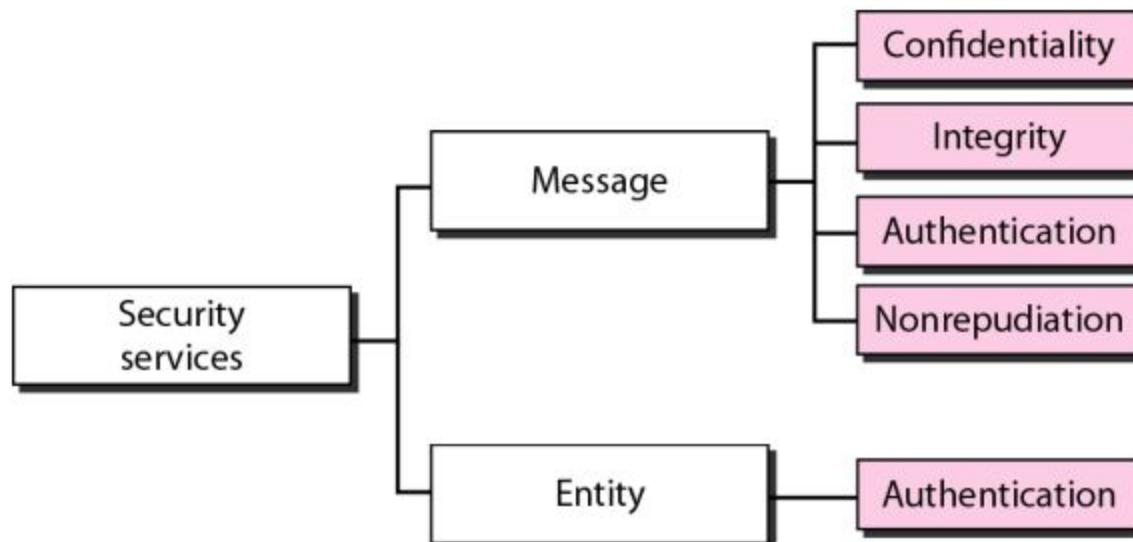
## **Network Security**

**Dr. Vandana Kushwaha**

Department of Computer Science  
Institute of Science, BHU, Varanasi

# SECURITY SERVICES

- Network security can provide one of the five services as shown in Figure below.
- Four of these services are related to the message exchanged using the network: message confidentiality, integrity, authentication, and nonrepudiation.
- The fifth service provides entity authentication or identification.



# **SECURITY SERVICES**

## **1. Message Confidentiality**

- Message confidentiality or privacy means that the sender and the receiver expect confidentiality.
- The transmitted message must make sense to only the intended receiver.
- To all others, the message must be garbage.
- When a customer communicates with her bank, she expects that the communication is totally confidential.

# **SECURITY SERVICES**

## **2. Message Integrity**

- Message integrity means that the data must arrive at the receiver exactly as they were sent.
- There must be no changes during the transmission, neither accidentally nor maliciously.
- As more and more monetary exchanges occur over the Internet, integrity is crucial.
- For example, it would be disastrous if a request for transferring \$100 changed to a request for \$10,000 or \$100,000.
- The integrity of the message must be preserved in a secure communication.

# SECURITY SERVICES

## 3. Message Authentication

- Message authentication is a service beyond message integrity.
- In message authentication the receiver needs to be sure of the **sender's identity** and that an imposter has not sent the message.

## 4. Message Nonrepudiation

- Message nonrepudiation means that a sender must not be able to deny sending a message that he or she, in fact, did send.
- The burden of proof falls on the receiver.
- For example, when a customer sends a message to transfer money from one account to another, the bank must have proof that the customer actually requested this transaction.

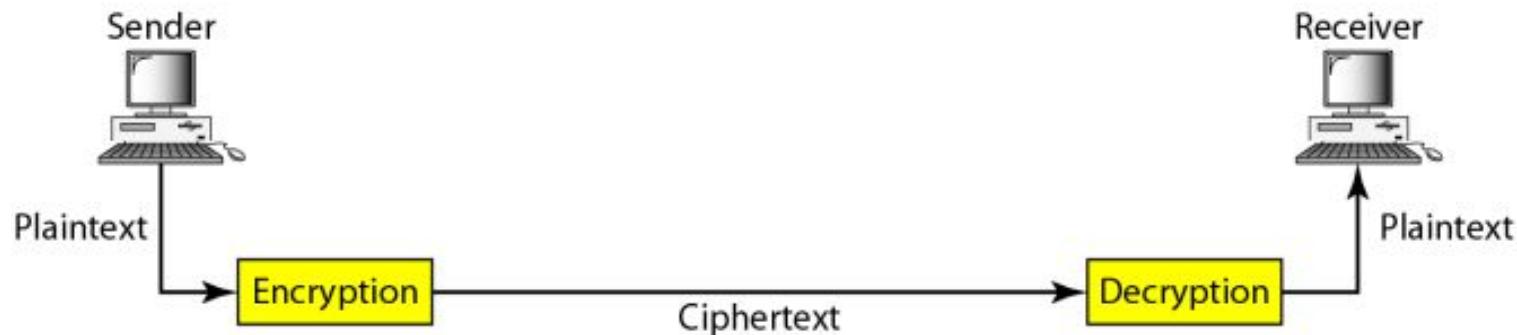
# **SECURITY SERVICES**

## **5. Entity Authentication**

- In entity authentication (or user identification) the entity or user is verified prior to access to the system resources (files, for example).
- For example, a student who needs to access her university resources needs to be authenticated during the logging process.
- This is to protect the interests of the university and the student.

# Cryptography

- Cryptography, a word with Greek origins, means "secret writing."
- However, we use the term to refer to the science and art of transforming messages to make them secure and immune to attacks.
- Figure below shows the components involved in cryptography:



# Cryptography

## Plaintext and Ciphertext

- The **original message**, before being transformed, is called **plaintext**.
- After the **message is transformed**, it is called **ciphertext**.
- An **encryption algorithm transforms the plain-text into ciphertext**;
- A **decryption algorithm transforms the ciphertext back into plain-text**.
- The **sender uses an encryption algorithm**.
- The **receiver uses a decryption algorithm**.

## Cipher

- We refer to **encryption and decryption algorithms as ciphers**.
- The term ***cipher*** is also used to refer to different categories of algorithms in cryptography.

# Cryptography

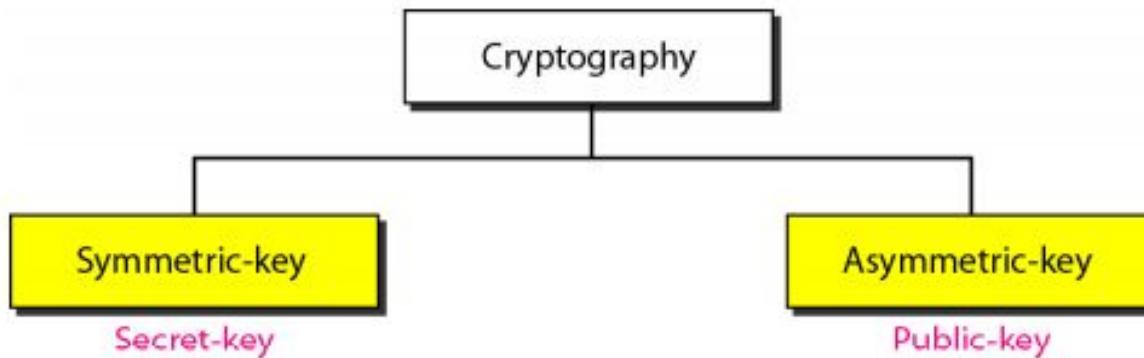
## Key

- A **key** is a **number (or a set of numbers)** that the cipher, as an algorithm, operates on.
- To **encrypt** a message, we need an **encryption algorithm**, an **encryption key**, and the **plaintext**.
- These create the **ciphertext**.
- To **decrypt** a message, we need a **decryption algorithm**, a **decryption key**, and the **ciphertext**.
- These reveal the original **plaintext**.

# Categories of Cryptography Algorithms

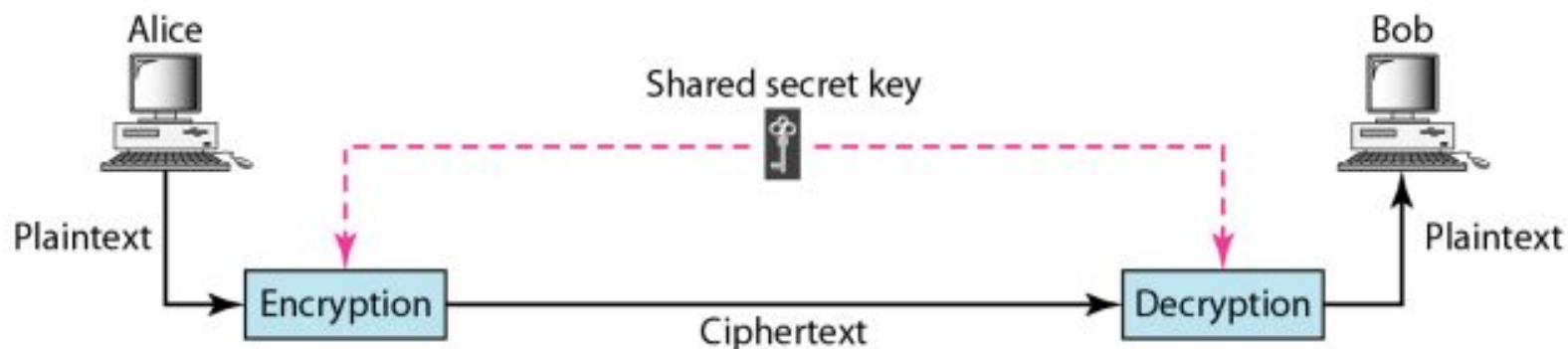
We can divide all the **cryptography algorithms (ciphers)** into **two groups**:

1. **Symmetric- key (also called secret-key) cryptography** algorithms
2. **Asymmetric (also called public-key) cryptography** algorithms.



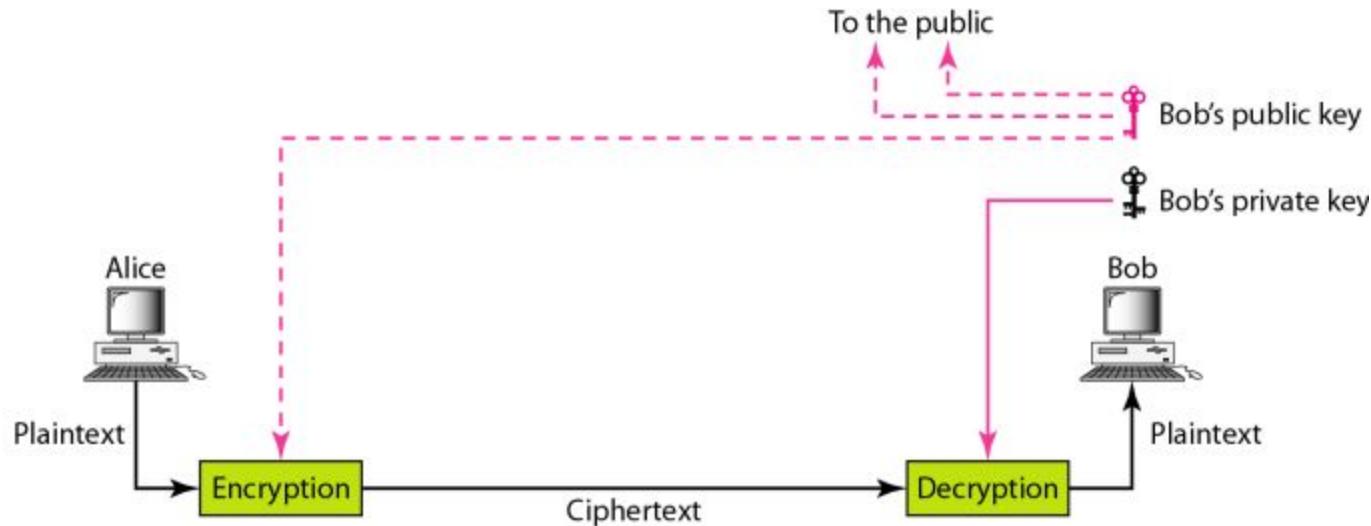
# Symmetric Key Cryptography

- In **Symmetric-key cryptography**, the **same key** is used by **both parties**.
- The **sender** uses this key and an encryption algorithm to encrypt data.
- The **receiver** uses the **same key** and the corresponding decryption algorithm to decrypt the data.



# Asymmetric-Key Cryptography

- In **Asymmetric or Public-key cryptography**, there are **two keys**: a **private key** and a **public key**.
- The **private key** is **kept by the receiver**.
- The **public key** is **announced to the public** by the receiver.
- Imagine **Alice** wants to **send a message** to **Bob**.
- **Alice uses the Bob's public key to encrypt the message.**
- When the message is received by **Bob**, the **private key** is used to **decrypt** the message.



# Asymmetric-Key Cryptography

- In **Asymmetric or public-key encryption/decryption**, the **public key** that is used for **encryption** is **different** from the **private key** that is used for **decryption**.
- The **public key** is available to the public.
- The **private key** is available only to an individual.

# Three Types of Keys

- We are dealing with **three types** of keys in **cryptography**:
  1. **Secret key**,
  2. **Public key**,
  3. **Private key**.
- The first, the **secret key**, is the **shared key** used in **symmetric-key cryptography**.
- The second and the third are the **public and private keys** used in **asymmetric-key cryptography**.



Secret key

Symmetric-key cryptography



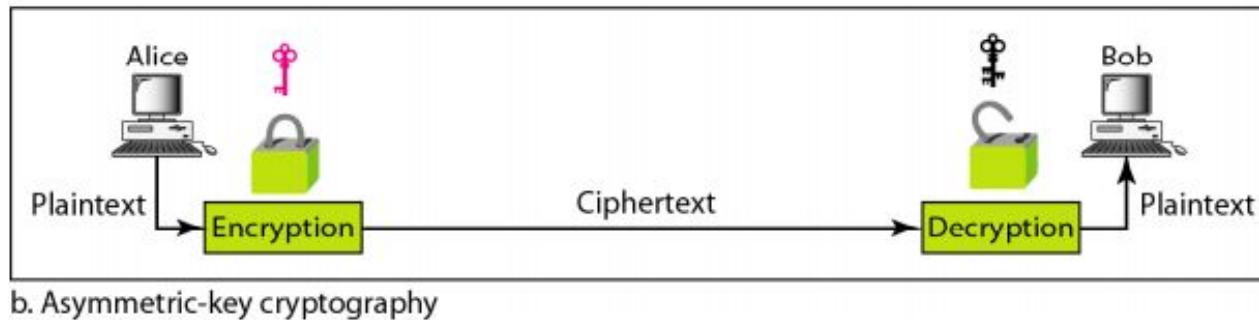
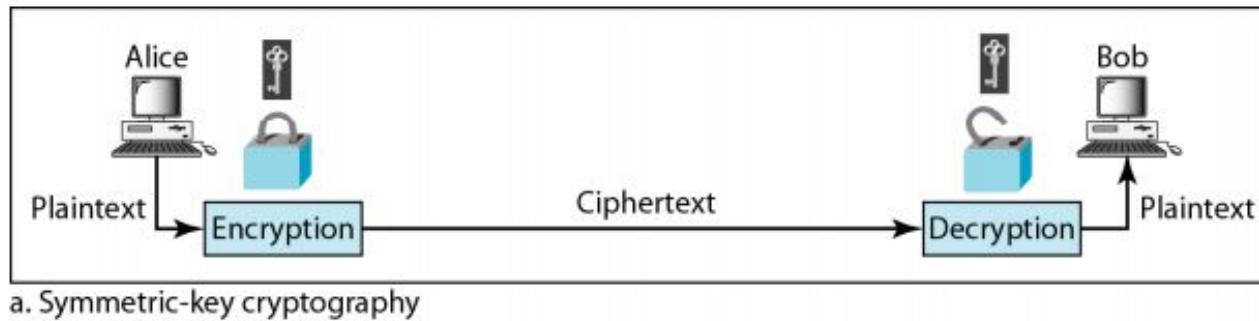
Public key      Private key

Asymmetric-key cryptography

# Symmetric-key vs. Asymmetric-key Cryptography

- **Encryption** can be thought of as **electronic locking** and **Decryption** as **electronic unlocking**.
- The sender puts the message in a box and locks the box by using a key; the receiver unlocks the box with a key and takes out the message.
- The **difference** lies in the **mechanism** of the **locking and unlocking** and the **type of keys used**.
- In **Symmetric-key cryptography**, the **same key locks and unlocks** the box.
- In **Asymmetric-key cryptography**, one key locks the box, but **another key is needed to unlock it**.

# Symmetric-key vs. Asymmetric-key Cryptography



# 1. MESSAGE CONFIDENTIALITY

- The concept of how to achieve message confidentiality or privacy has not changed for thousands of years.
- The message must be encrypted at the sender site and decrypted at the receiver site.
- That is, the message must be rendered unintelligible to unauthorized parties.
- A good privacy technique guarantees to some extent that a potential intruder (eavesdropper) cannot understand the contents of the message.
- This can be done using either **Symmetric-key cryptography** or **Asymmetric-key cryptography**.

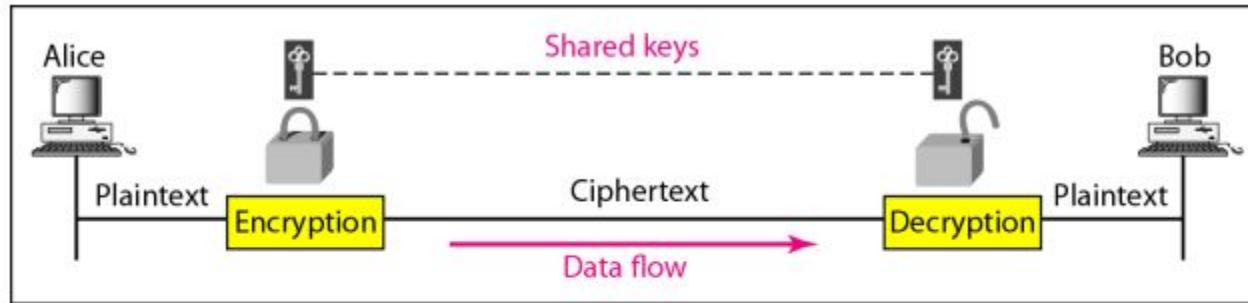
# Confidentiality with Symmetric-Key Cryptography

- Although **modern symmetric-key algorithms** are **more complex** than the ones used through the long history of the secret writing, the principle is the same.
- To provide confidentiality with symmetric-key cryptography, a **sender** and a **receiver need to share a secret key**.
- In the past when data exchange was between two specific persons (for example, two friends or a ruler and her army chief), it was possible to personally exchange the secret keys.
- Today's communication does not often provide this opportunity.
- A person residing in the United States cannot meet and exchange a secret key with a person living in China.
- Furthermore, the communication is between millions of people, not just a few.

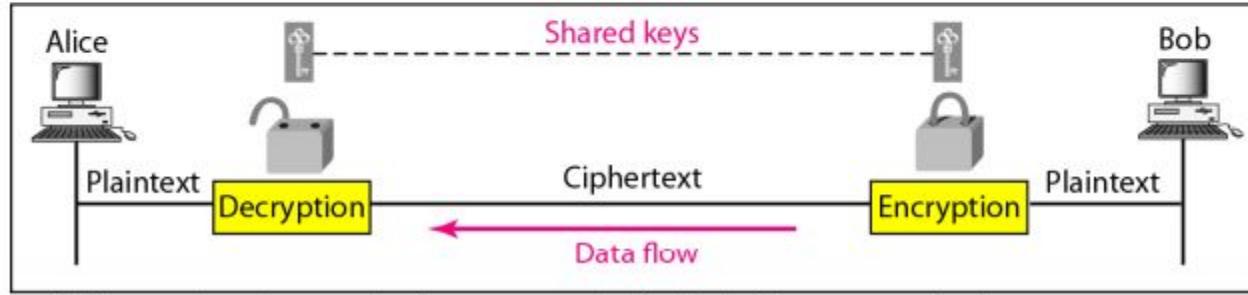
# Confidentiality with Symmetric-Key Cryptography

- To be able to use symmetric-key cryptography, we need to find a solution to the **key sharing**.
- This can be done using a **session key**.
- A **session key** is one that is **used only for the duration of one session**.
- The **session key itself is exchanged using asymmetric- key cryptography** .
- Note that the nature of the symmetric key allows the communication to be carried on in both directions although it is not recommended today.
- Using two different keys is more secure, because if one key is compromised, the communication is still confidential in the other direction.

# Message confidentiality using symmetric keys in two directions



a. A shared secret key can be used in Alice-Bob communication

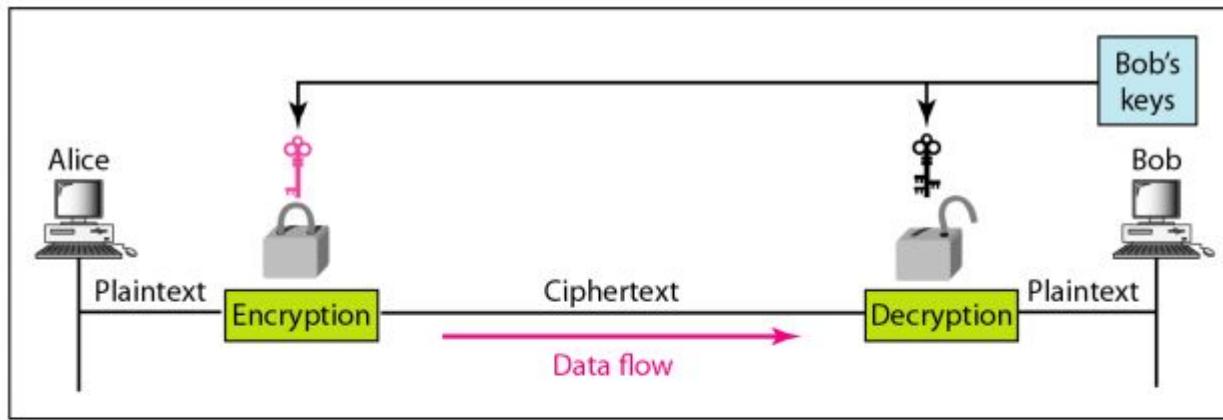


b. A different shared secret key is recommended in Bob-Alice communication

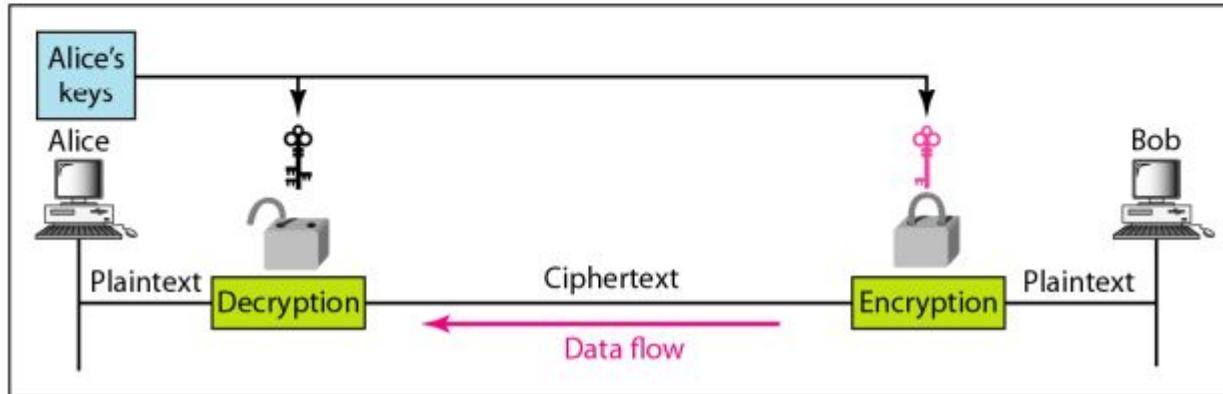
# Confidentiality with Asymmetric-Key Cryptography

- In **Asymmetric-key cryptography**, there is **no key sharing**; there is a **public announcement**.
- **Bob** creates **two keys**: one **private** and one **public**.
- He **keeps the private key for decryption**; he **publicly announces the public key** to the world.
- The **public key** is used **only for encryption**; the **private key** is used **only for decryption**.
- The public key locks the message; the private key unlocks it.
- For a **two-way communication** between **Alice and Bob**, **two pairs of keys** are **needed**.
- When Alice sends a message to Bob, she uses Bob's pair; when Bob sends a message to Alice, he uses Alice's pair.

# Message Confidentiality using Asymmetric keys



a. Bob's keys are used in Alice-Bob communication



b. Alice's keys are used in Bob-Alice communication

# Message Confidentiality using Asymmetric Keys

- **Confidentiality** with **Asymmetric-key** cryptosystem has its own **problems**.
- **First, the method is based on long mathematical calculations** using long keys.
- This means that this system is **very inefficient for long messages**; it should be **applied only to short messages**.
- **Second, the sender of the message still needs to be certain about the public key of the receiver.**
- For **example**, in Alice-Bob communication, Alice needs to be sure that Bob's **public key is genuine**; Eve may have announced her public key in the name of Bob.

## 2. MESSAGE INTEGRITY

- Encryption and Decryption provide Confidentiality, but not Integrity.
- How-ever, on occasion we may not even need secrecy, but instead must have integrity.
- For example, Alice may write a will to distribute her estate upon her death.
- The will does not need to be encrypted.
- After her death, anyone can examine the will.
- The integrity of the will, however, needs to be preserved.
- Alice does not want the contents of the will to be changed.
- As another example, suppose Alice sends a message instructing her banker, Bob, to pay Eve for consulting work.
- The message does not need to be hidden from Eve because she already knows she is to be paid.
- However, the message does need to be safe from any tampering, especially by Eve.

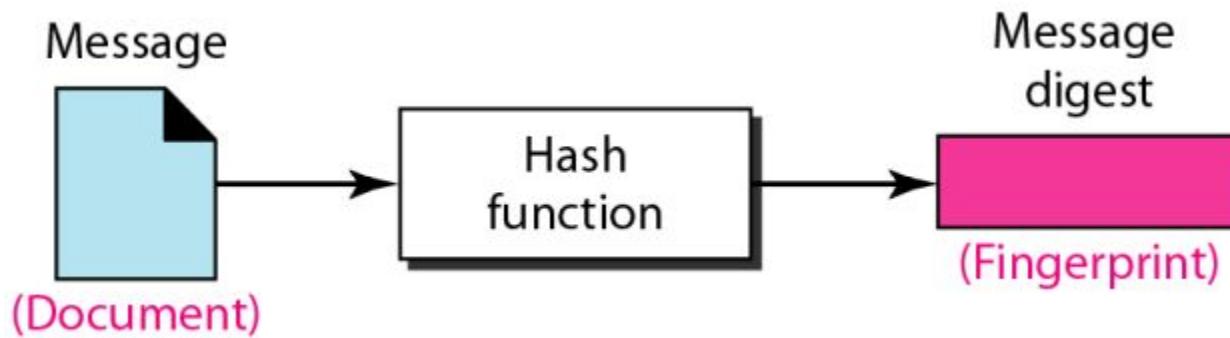
# Document and Fingerprint

- One way to **preserve** the **integrity** of a document is through the use of a **fingerprint**.
- If Alice needs to be sure that the contents of her document will not be illegally changed, she can put her **fingerprint** at the bottom of the document.
- Eve cannot modify the contents of this document or create a false document because she **cannot forge Alice's finger-print**.
- To **preserve the integrity** of a document, **both the document and the fingerprint are needed**.

# Message and Message Digest

- The **electronic equivalent** of the **document** and **fingerprint pair** is the **message** and **message digest pair**.
- To **preserve the integrity** of a message, the **message is passed** through an **algorithm** called a **hash function**.
- The **hash function** creates a **compressed image of the message** that can be used as a **fingerprint**.
- The document and fingerprint are physically linked together; also, neither needs to be kept secret.
- However, the **message and message digest** can be **unlinked** (or sent) **separately** and, most importantly, the **message digest needs to be kept secret**.
- The **message digest** is either **kept secret in a safe place** or **encrypted** if we need to send it through a **communications channel**.

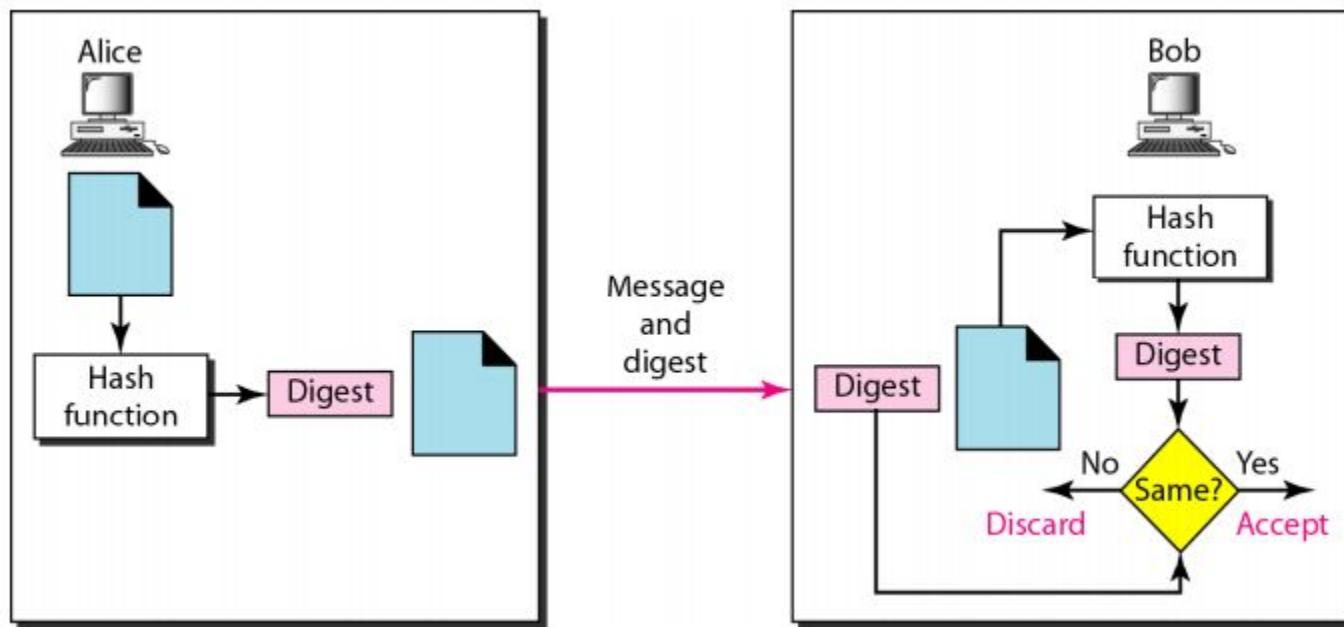
# Message and Message Digest



# Creating and Checking the Digest

- The **message digest** is **created** at the **sender site** and is **sent with the message** to the receiver.
- To check the **integrity** of a message, or document, the **receiver creates the hash function again and compares the new message digest with the one received**.
- If **both are the same**, the receiver is sure that the **original message has not been changed**.
- We are assuming that the **digest** has been sent **secretly**.
- **SHA-1(Secure Hash Algorithm 1)** is used for creating the **Digest**.

# Checking integrity



# 3.MESSAGE AUTHENTICATION

- A **hash function** guarantees the **integrity** of a message.
- It guarantees that the **message has not been changed**.
- A **hash function**, however, **does not authenticate the sender** of the message.
- When **Alice** sends a message to **Bob**, **Bob** needs to know if the message is coming from **Alice or Eve**.
- To provide **message authentication**, **Alice** needs to provide **proof** that it is Alice sending the message and not an **impostor**.
- A **hash function** per se cannot provide such a proof.
- The **digest** created by a **hash function** is normally called a **modification detection code (MDC)**.
- The **MDC can detect any modification** in the **message**.

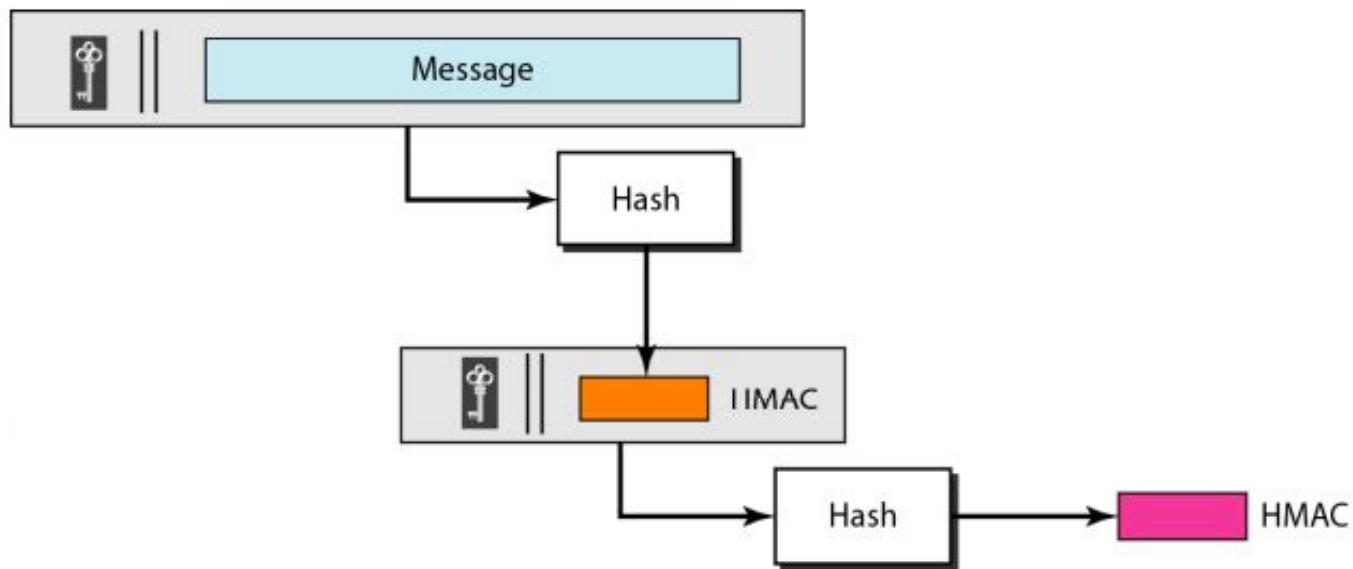
# Message Authentication Code (MAC)

- To provide **message authentication**, we need to change a **Modification Detection Code(MDC)** to a **Message Authentication Code (MAC)**.
- An **MDC** uses a **keyless hash function** ; a **MAC** uses a **keyed hash function**.
- A **keyed hash function** includes the **symmetric key between the sender and receiver when creating the digest**.
- Alice uses a **keyed hash function to authenticate her message** and how Bob can **verify the authenticity** of the message.
- Alice, using the **symmetric key** between herself and Bob ( $K_{AB}$ ) and a **keyed hash function**, generates a **MAC**.
- She then **Concatenates the MAC with the original message** and sends the two to Bob.
- Bob receives the message and the **MAC**.

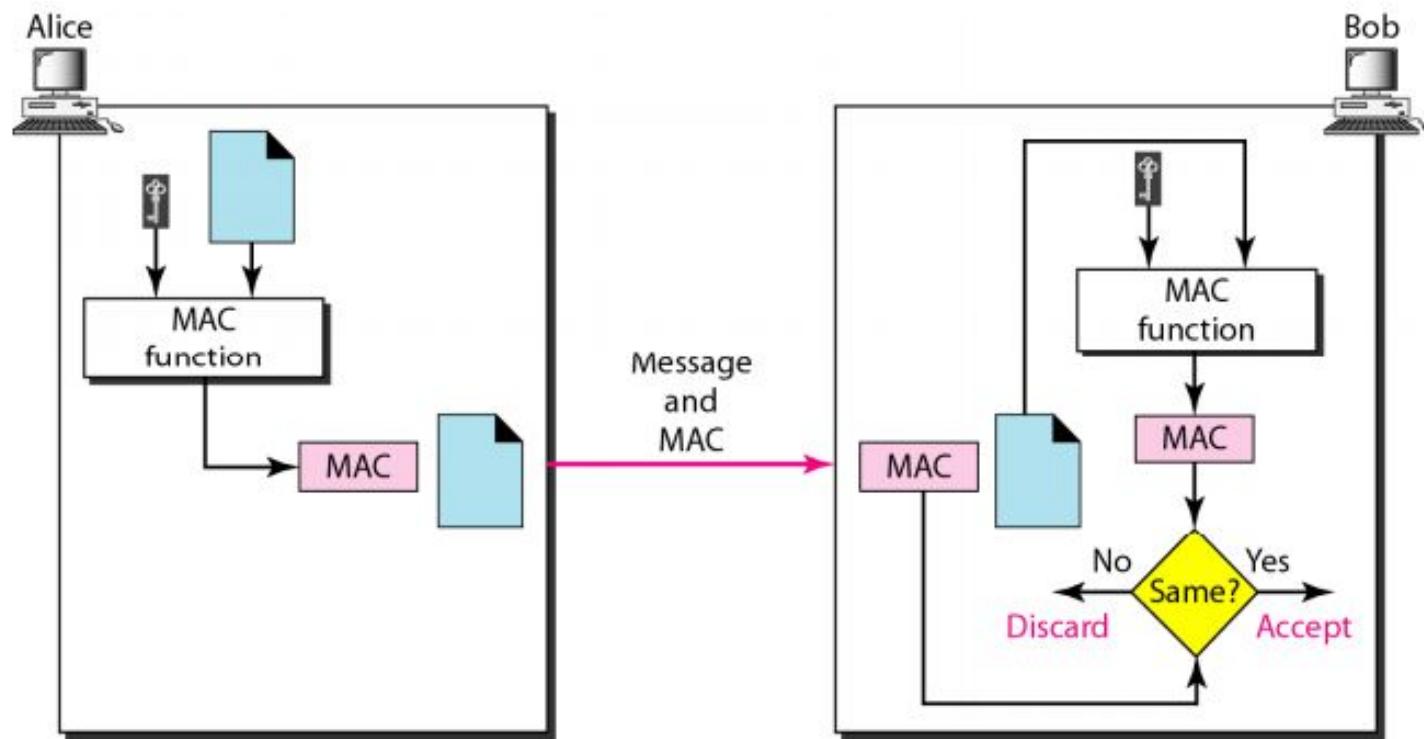
# Message Authentication Code (MAC)

- Bob **separates the message from the MAC**.
- He applies the **same keyed hash function to the message using the symmetric key  $K_{AB}$**  to get a fresh **MAC**.
- He then **compares the MAC** sent by Alice with the newly generated MAC.
- If the **two MACs are identical**, the message has not been modified and the sender of the message is definitely Alice.
- **HMAC(hashed MAC)** is used to create the **MAC**.
- **HMAC** can use any standard **keyless hash function** such as **SHA-1**.
- **HMAC** creates a **nested MAC** by applying a **keyless hash function** to the **concatenation of the message and a symmetric key**.
- **HMAC = hashFunc(secret key + message)**

# HMAC



# MAC, created by Alice and checked by Bob



# DIGITAL SIGNATURE

- Although a **MAC** can provide **message Integrity** and **message Authentication**, it has a **drawback**.
- It **needs a symmetric key** that must be established between the sender and the receiver.
- A **digital signature**, on the other hand, can **use a pair of asymmetric keys** (a public one and a private one).
- We sign a document to show that it originated from us or was approved by us.
- The **signature is proof** to the recipient that the **document comes from the correct entity**.

# DIGITAL SIGNATURE

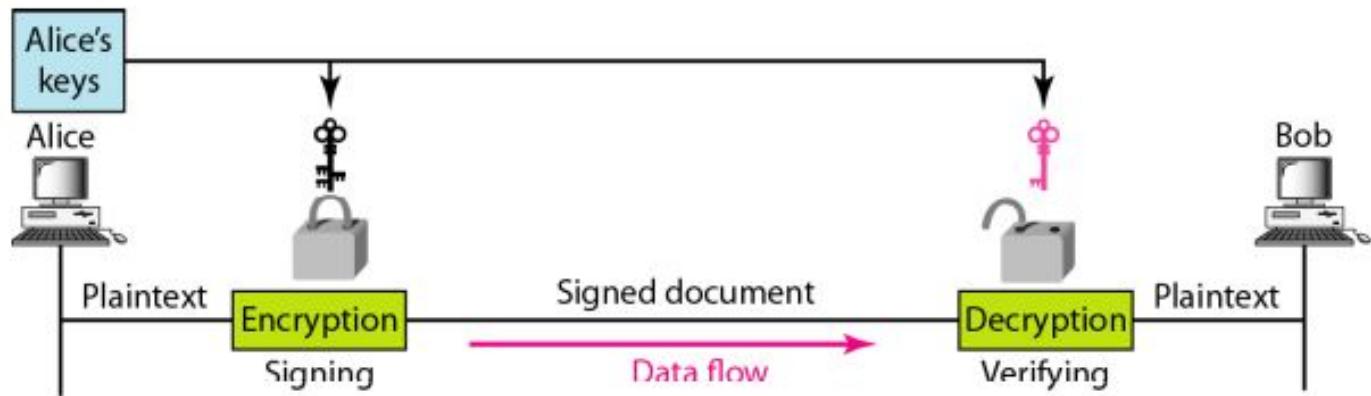
- In other words, a **signature on a document**, when **verified**, is a sign of **authentication**; the document is authentic.
- When Alice sends a message to Bob, Bob needs to check the authenticity of the sender; he needs to be sure that the message comes from Alice and not Eve.
- Bob can ask Alice to sign the message electronically.
- In other words, an **electronic signature can prove the authenticity of Alice as the sender of the message**.
- We refer to this type of sig-nature as a **Digital Signature**.

# Process

Digital Signature can be achieved in **two ways**: **signing the document** or **signing a digest of the document**.

## Signing the Document

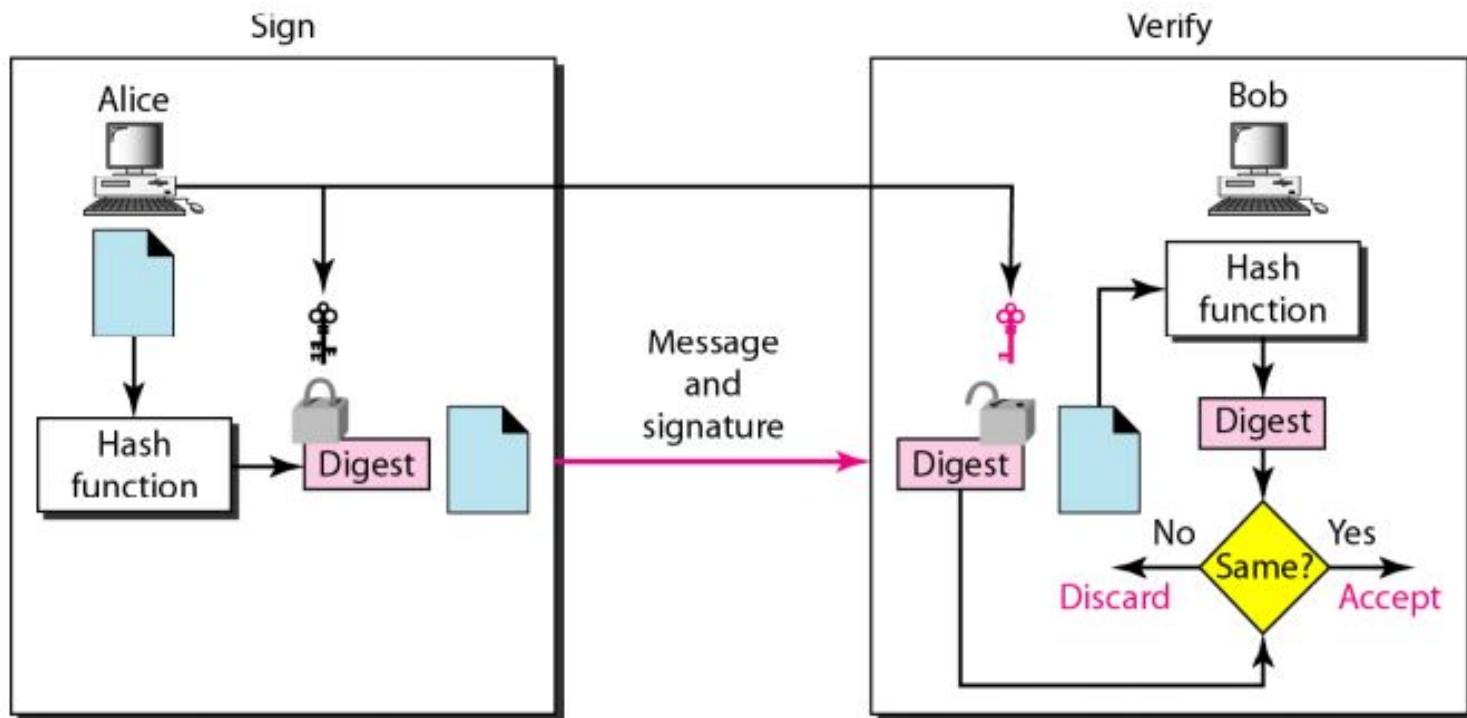
- Signing a document is encrypting it with the **private key of the sender**;
- Verifying the document is decrypting it with the **public key of the sender**.
- In a **cryptosystem**, we use the **private and public keys** of the **receiver**; in **digital signature**, we use the **private and public key** of the **sender**.



# Signing the Digest

- Public key cryptography is very inefficient in a cryptosystem if we are dealing with long messages.
- In a digital signature system, our messages are normally long, but we have to use public keys.
- The solution is not to sign the message itself; instead, we sign a digest of the message.
- The sender can sign the message digest, and the receiver can verify the message digest.
- A digest is made out of the message at Alice's site.
- The digest then goes through the signing process using Alice's private key.
- Alice then sends the message and the sig-nature to Bob.

# Signing the digest in a digital signature



# Services by Digital Signature

- A **Digital signature** can provide **three out of the five services** we mentioned for a security system:
  1. **Message Integrity,**
  2. **Message Authentication,**
  3. **Nonrepudiation.**
- A **digital signature** scheme **does not provide confidential** communication.
- If confidentiality is required, the message and the signature must be encrypted using either a secret-key or public-key cryptosystem.

# 5. ENTITY AUTHENTICATION

- Entity authentication is a technique designed to let one party prove the identity of another party.
- An Entity can be a person, a process, a client, or a server.
- The entity whose identity needs to be proved is called the claimant;
- The party that tries to prove the identity of the claimant is called the verifier.
- When Bob tries to prove the identity of Alice, Alice is the claimant, and Bob is the verifier.
- There are two methods for entity authentication:
  1. Password
  2. Challenge Response

# Passwords

- The simplest and the oldest method of entity authentication is the password, something that the claimant *possesses*.
- *A password is used when a user needs to access a system to use the system's resources (log-in).*
- Each user has a user identification that is public and a password that is private.
- We can divide this authentication scheme into two separate groups:
  - Fixed password
  - One-time password.

# Challenge-Response

- In **password authentication**, the **claimant** proves her identity by demonstrating that she knows a **secret**, the **password**.
- However, since the **claimant reveals this secret**, the secret is **susceptible to interception by the adversary**.
- In **challenge-response authentication**, the **claimant proves that she *knows a secret without revealing it***.
- *In other words*, the claimant does not reveal the secret to the verifier; the verifier either has it or finds it.
- The **challenge** is a **time-varying value** such as a **random number** or a **timestamp** which is **sent by the verifier**.
- The **claimant applies a function** to the **challenge** and **sends the result**, called a **response, to the verifier**.
- *The response shows that the claimant knows the secret.*

# Challenge-Response

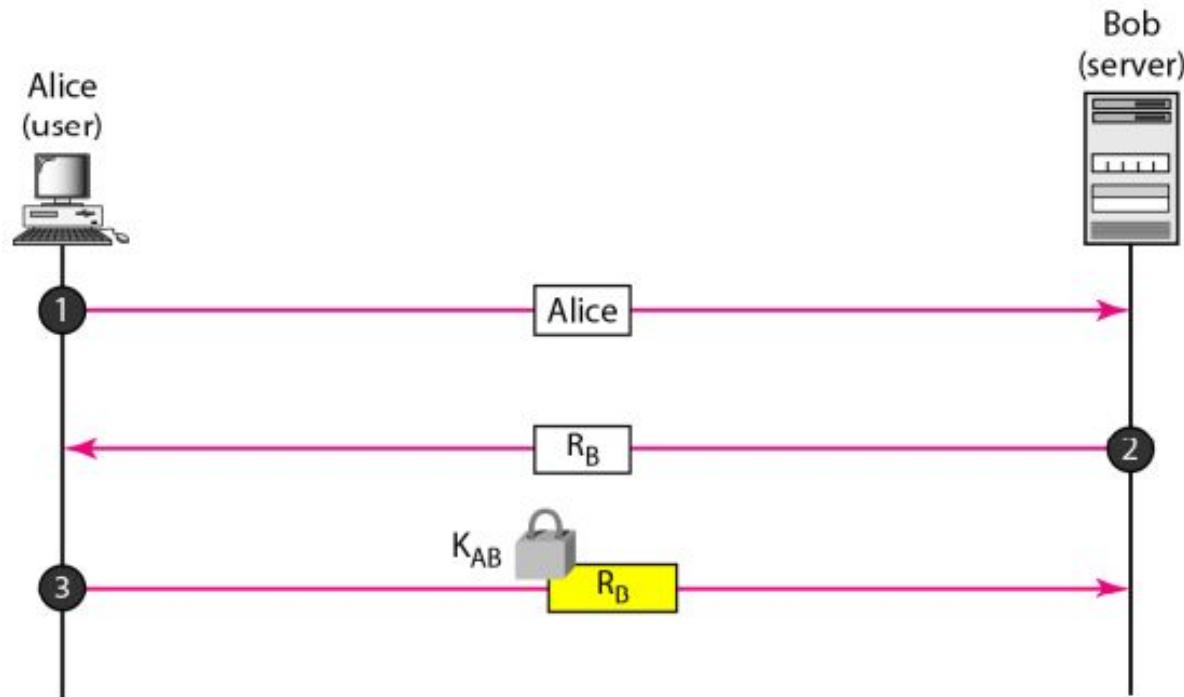
The **Challenge-response authentication** can be achieved in **four different ways**:

1. Challenge-response authentication using a **Symmetric-Key Cipher**
  - a. **Using a Nonce**
  - b. **Using a Timestamp**
2. Challenge-response authentication using a **Keyed-hash function**
3. Challenge-response authentication using an **Asymmetric-Key Cipher**

# Challenge-Response Using a Symmetric-Key Cipher (Using Nonce)

- In the first category, the **challenge-response authentication** is achieved using **symmetric-key encryption**.
- The **secret** here is the **shared secret key**, known by both the **claimant** and the **verifier**.
- The function is the **encrypting algorithm** applied on the **challenge**.
- The **first message** is not part of **challenge-response**, it only **informs** the **verifier** that the **claimant** wants to be **challenged**.
- The **second message** is the **challenge** which contain the **nonce  $R_B$**  randomly chosen by the **verifier** to challenge the **claimant**.
- The **claimant encrypts the nonce** using the **shared secret key** known only to the **claimant** and the **verifier** and sends the result to the **verifier**.

# Challenge-Response Using a Symmetric-Key Cipher(Using Nonce)



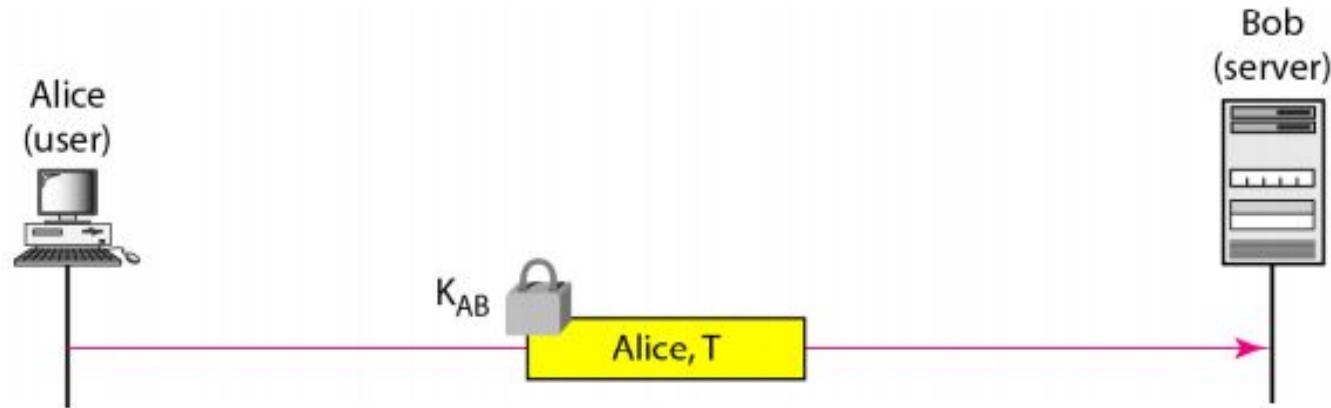
# Challenge-Response Using Symmetric-Key Cipher (Using Nonce)

- The **Verifier** decrypts the message.
- If the **nonce** obtained from **decryption** is the **same** as the one **sent by the verifier**, **Alice** is **granted access**.
- Note that in this process, the claimant and the verifier need to keep the **symmetric key** used in the process **secret**.
- The verifier must also keep the value of the **nonce** for claimant identification until the response is returned.
- Eve cannot **replay** the **third message** and **pretend** that it is a **new request** for authentication **by Alice** because once Bob receives the response, the value of  $R_B$  is not **valid any more**.
- The next time a new value is used as a nonce.

# Challenge-response using a Symmetric-Key Cipher (using Timestamp)

- In this approach, the **time-varying value** is a **timestamp**, which obviously **changes with time**.
- In this approach the **challenge message** is the **current time sent from the verifier to the claimant**.
- However, this supposes that the **client and the server clocks are synchronized**; the **claimant knows the current time**.
- This means that there is **no need for the challenge message**.
- The **first and third messages can be combined**.
- The result is that authentication can be done using **one message**, the **response to an implicit challenge**, the **current time encrypted** using the **shared secret key**.

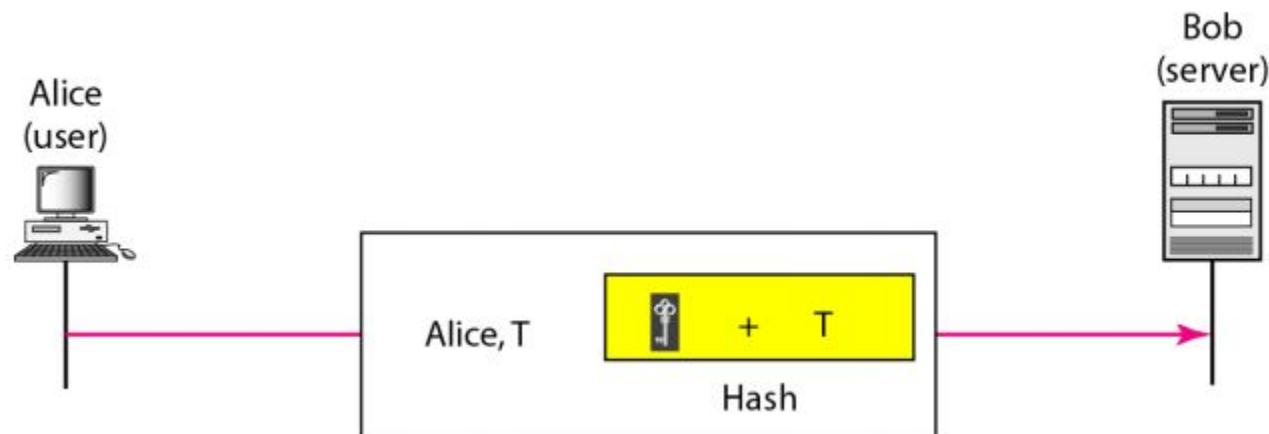
# Challenge-response using a Symmetric-Key Cipher (using Timestamp)



# Challenge-response using a Keyed-Hash function

- Instead of using **encryption and decryption** for entity authentication, we can use a **keyed-hash function** (MAC).
- There are two **advantages** to this scheme.
- First, the **encryption/decryption algorithm** is not **exportable to some countries**.
- Second, in using a **keyed-hash function**, we can **preserve the integrity of challenge and response messages** and at the same time use a **secret**, the **key**.
- Note that in this case, the **timestamp is sent** both as **plaintext T** and as **text** scrambled by the **keyed-hash function**.
- When **Bob** receives the message, he takes the **plaintext T**, applies the **keyed-hash function**, and then **compares** his calculation with what he received to determine the authenticity of Alice.

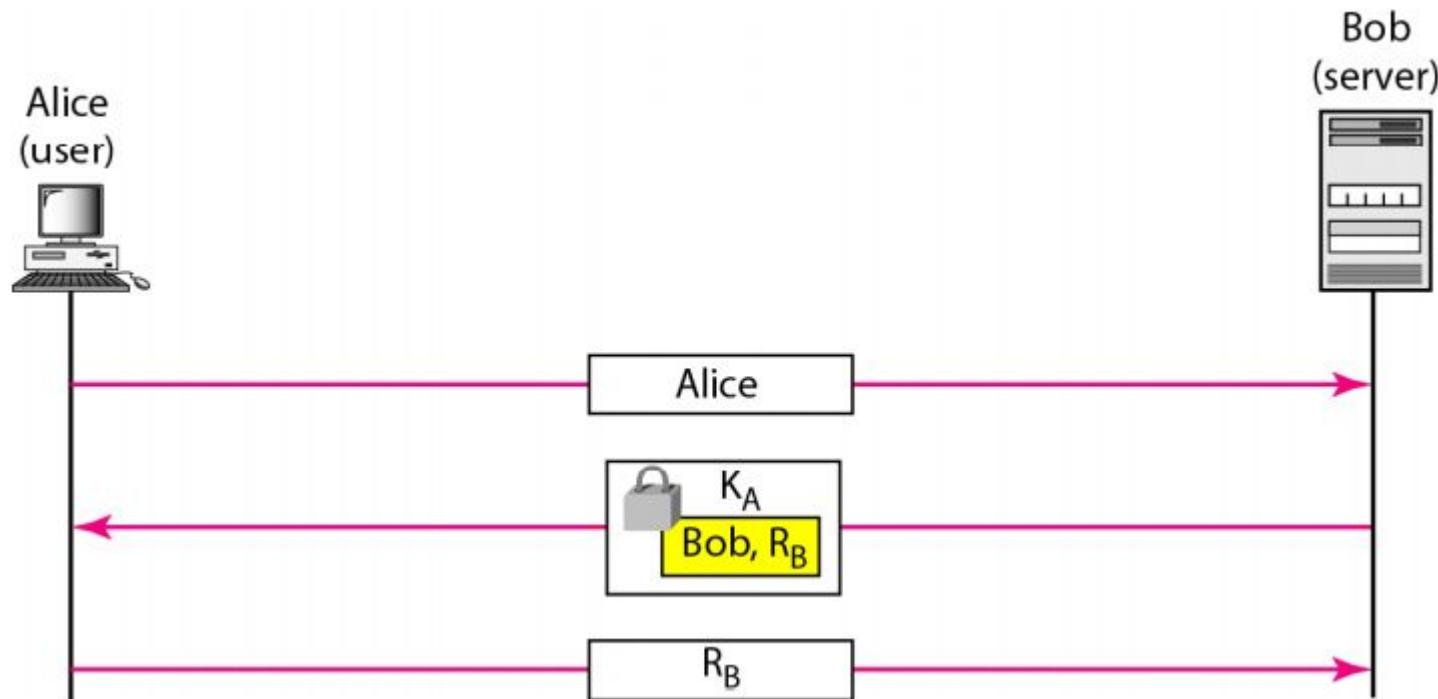
# Challenge-response using a Keyed-Hash function



# Challenge-response using an Asymmetric-Key Cipher

- In **Asymmetric-key cipher for entity authentication** approach the **secret** must be the **private key of the claimant**.
- The **claimant** must show that she owns the **private key related to the public key** that is available to everyone.
- This means that the **verifier** must **encrypt the challenge** using the **public key** of the **claimant**; the **claimant then decrypts the message using her private key**.
- The response to the challenge is the decrypted challenge.
- **Bob encrypts the challenge using Alice's public key.**
- **Alice decrypts the message with her private key and sends the nonce to Bob.**

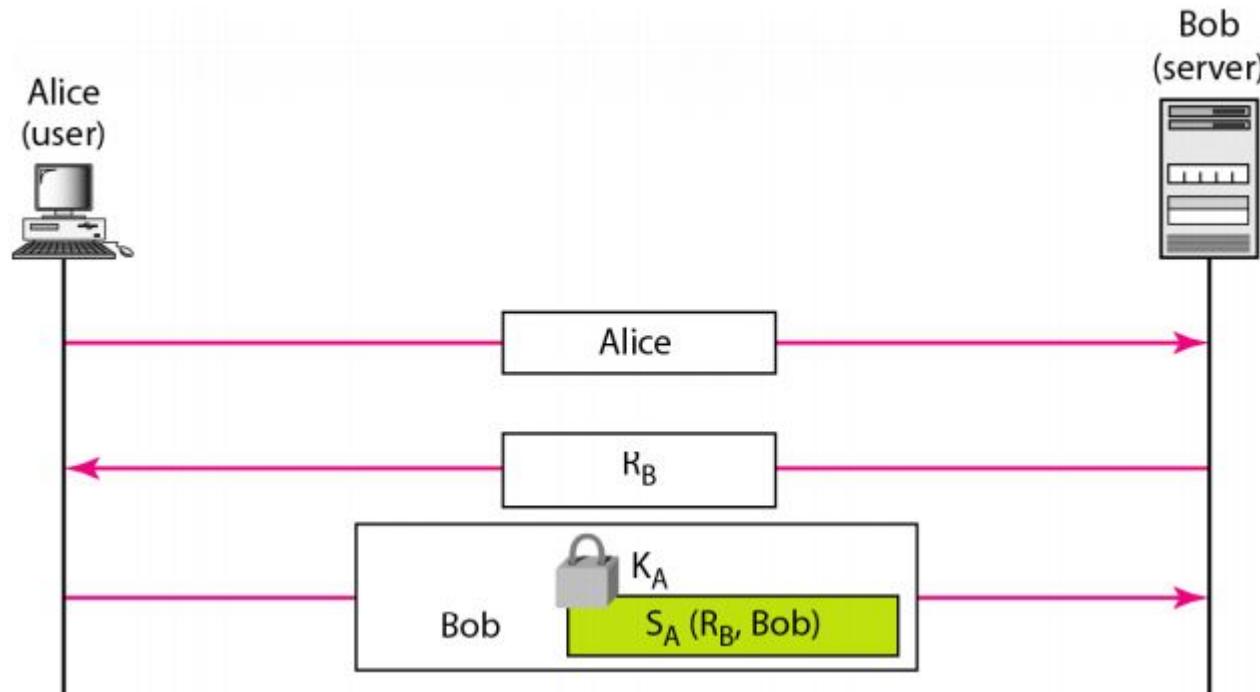
# Challenge-response using an Asymmetric-Key Cipher



# Challenge-response using Digital Signature

- We can use **Digital signature** for entity authentication.
- In this method, we let the **claimant use her private key for signing** instead of using it for decryption.
- There are **four steps**:
  1. **Initialization.** The **claimant** tells the **verifier** it wishes **to be authenticated**.
  2. **Challenge.** The **verifier** generates a **challenge** and issues it to the **claimant**.
  3. **Response.** The **claimant** signs the **challenge** (using his **private key**) and returns it to the verifier.
  4. **Verification.** The **verifier** verifies that the signed version of the **challenge** matches the issued challenge(using claimant public key) and grants access to the client.

# Challenge-response using Digital Signature



# KEY MANAGEMENT

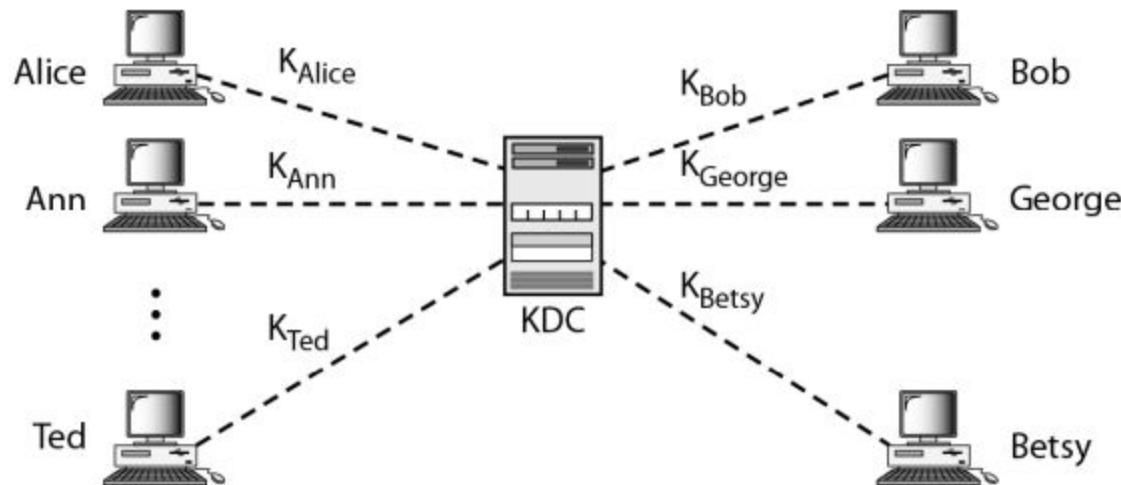
- How **secret keys** in **Symmetric-key cryptography** and how **public keys** in **Asymmetric-key cryptography** are **distributed** and maintained.
- Thus **two issues** are:
  1. Distribution of Symmetric keys
  2. Distribution of Asymmetric keys

# Symmetric-Key Distribution

- Symmetric- key cryptography, however, needs a **shared secret key** between two parties.
- If Alice needs to exchange confidential messages with ***N people, she needs N different keys.***
- What if ***N people need to communicate with one another?***
- A ***total of N(N - 1)/2 keys are needed.***
- Each person needs to have ***N - 1 keys to communicate with each of the other people,*** but because the keys are shared, we need only  $N(N - 1)/2$ .
- If Alice and Bob want to communicate, they need to somehow ***exchange a secret key;*** if Alice wants to communicate with 1 million people, how can she exchange 1 million keys with 1 million people?
- Using the **Internet** is definitely **not a secure method.**

# Key Distribution Centre: KDC

- A practical **solution** is the use of a **trusted party**, referred to as a **key distribution centre (KDC)**.
- To reduce the number of keys, each person establishes a shared secret key with the **KDC**.



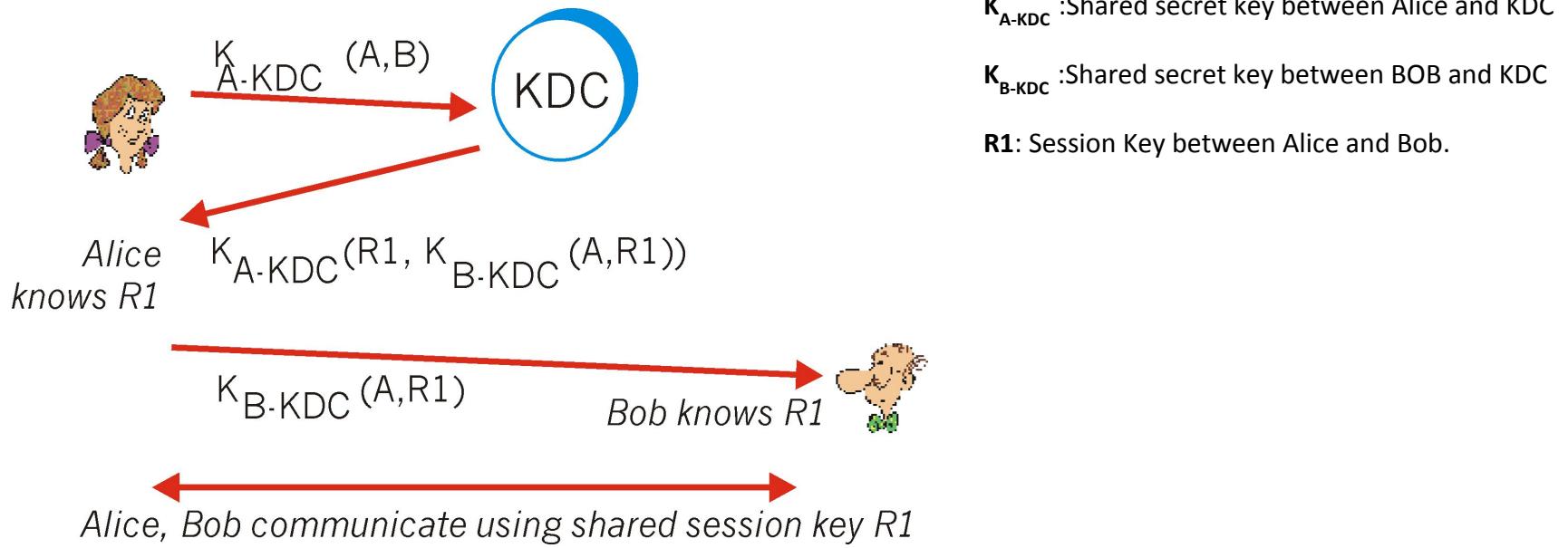
# Key Distribution Centre: KDC

- A **secret key** is established between KDC and each member.
- Alice has a **secret key with KDC**, which we refer to as  $K_{Alice}$
- Bob has a **secret key with KDC**, which we refer  $K_{Bob}$  and so on.
- How can Alice send a **confidential message to Bob?**
- The **process** is as follows:
  1. Alice sends a request to KDC, stating that she needs a **session (temporary) secret key between herself and Bob.**
  2. KDC informs Bob of Alice's request.
  3. If Bob agrees, a **session key is created** between the two.
  4. KDC share the **Session key** with Alice.

# Session Keys

- A KDC creates a **secret key** for each member.
- The **KDC** is a **server** that shares a different **secret symmetric key** with each registered user.
- This **secret key** can be used only between the member and the KDC, not between two members.
- If Alice needs to communicate secretly with Bob, she needs a secret key between herself and Bob.
- A KDC can create a session (**temporary**) key between Alice and Bob.
- After communication is terminated, the session key is no longer valid.

# Setting up a one-time session key using a KDC



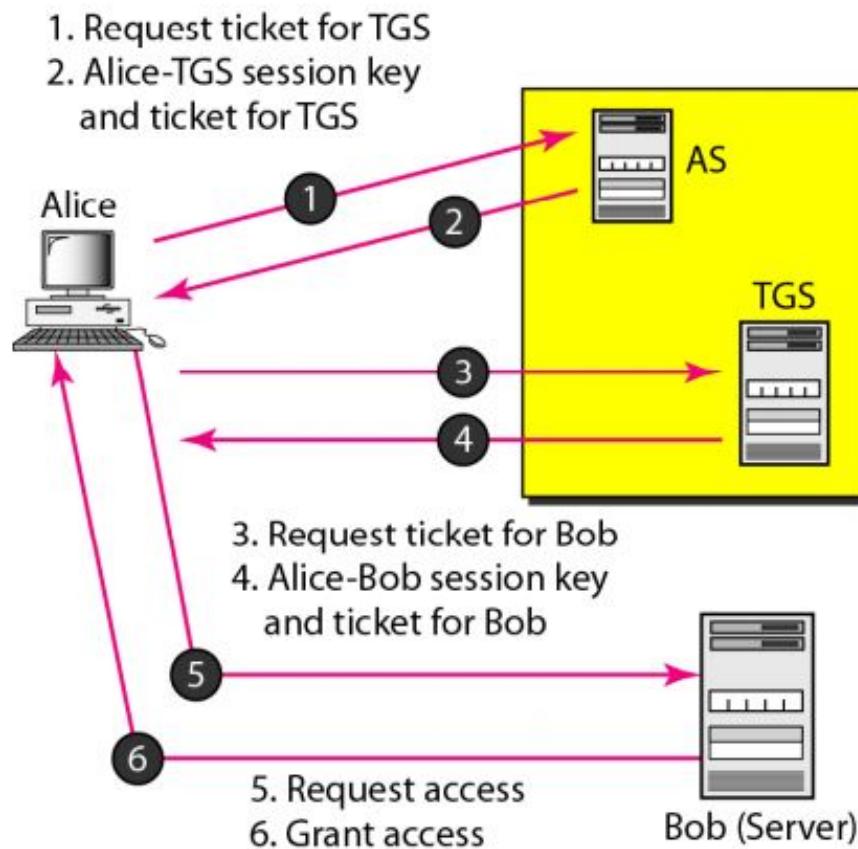
# Kerberos

- Kerberos is an **authentication protocol** .
- Several systems including Windows 2000 use Kerberos.
- It is named after the three-headed dog in Greek mythology that guards the Gates of Hades.

## Servers

- Three servers are involved in the **Kerberos protocol**:
  1. **Authentication server (AS)**,
  2. **Ticket-granting server (TGS)**,
  3. **Real (data) server that provides services** to others.

# Kerberos

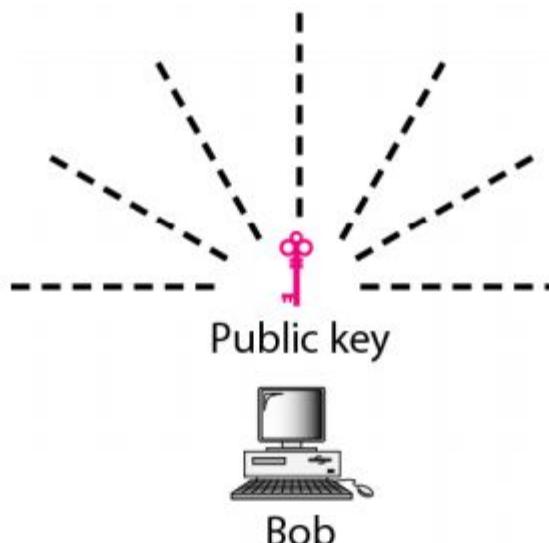


# Public-Key Distribution

- In **asymmetric-key cryptography**, people do not need to know a symmetric shared key.
- If Alice wants to send a message to Bob, she only needs to know Bob's public key, which is open to the public and available to everyone.
- If Bob needs to send a message to Alice, he only needs to know Alice's public key, which is also known to everyone.
- In public-key cryptography, everyone shields a private key and advertises a public key.
- **Public keys**, like secret keys, need to be distributed to be useful.

# Public Announcement

- The naive approach is to announce public keys publicly.
- Bob can put his public key on his website or announce it in a local or national newspaper.
- When Alice needs to send a confidential message to Bob, she can obtain Bob's public key from his site or from the newspaper, or she can even send a message to ask for it.



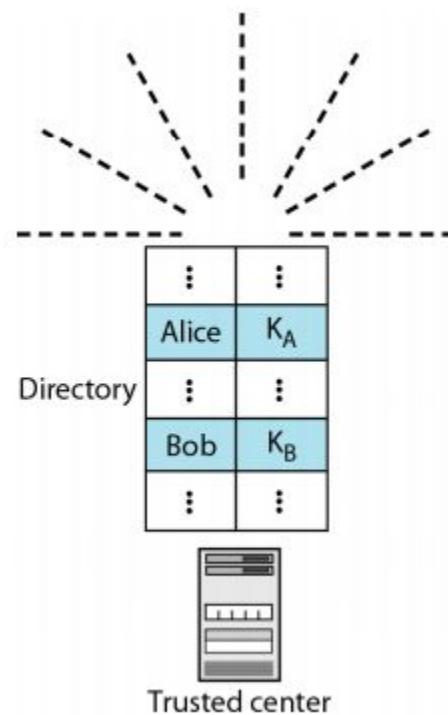
# Public Announcement

- This approach, however, is not secure; it is subject to forgery.
- For example, Eve could make such a public announcement.
- Before Bob can react, damage could be done.
- Eve can fool Alice into sending her a message that is intended for Bob.
- Eve could also sign a document with a corresponding forged private key and make everyone believe it was signed by Bob.
- The approach is also vulnerable if Alice directly requests Bob's public key.
- Eve can intercept Bob's response and substitute her own forged public key for Bob's public key.

# Trusted Center

- A more secure approach is to have a **trusted center** retain a **directory of public keys**.
- The directory, like the one used in a telephone system, is **dynamically updated**.
- Each user can select a **private/public key**, keep the private key, and deliver the **public key for insertion into the directory**.
- The **center requires** that each **user register in the center** and prove his or her identity.
- The **directory** can be **publicly advertised** by the **trusted center**.
- The **center can also respond to any inquiry about a public key**.

# Trusted Center



# Controlled Trusted Center

- A higher level of security can be achieved if there are added controls on the distribution of the public key.
- The **public-key announcements** can include a timestamp and be signed by an authority to prevent interception and modification of the response.
- If Alice needs to know Bob's public key, she can send a request to the **center** including **Bob's name** and a **timestamp**.
- The **center responds** with **Bob's public key**, the **original request**, and the **timestamp signed with the private key of the center**.
- Alice uses the **public key of the center**, known by all, to **decrypt the message** and **extract Bob's public key**.

# Controlled Trusted Center

