

BOOTSTRAP

- Bootstrap is the popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.
- Our Bootstrap tutorial includes all topics of Bootstrap such as jumbotron, table, button, grid, form, image, alert, wells, container, carousel, panels, glyphicon, badges, labels, progress bar, pagination, pager, list group, dropdown, collapse, tabs, pills, navbar, inputs, modals, tooltip, popover and scrollspy.

Bootstrap is mobile friendly

- ◉ Bootstrap 3 is designed to be responsive to mobile devices.
- ◉ Mobile-first styles are part of the core framework of Bootstrap. You have to add the following `<meta>` tag inside the `<head>` element for proper rendering and touch zooming:
- ◉ **`<meta name="viewport" content="width=device-width, initial-scale=1">`**
- ◉ **Note:** The `"width=device-width"` part is used to set the width of the page to follow the screen-width of the device (vary according to the devices).

Containers

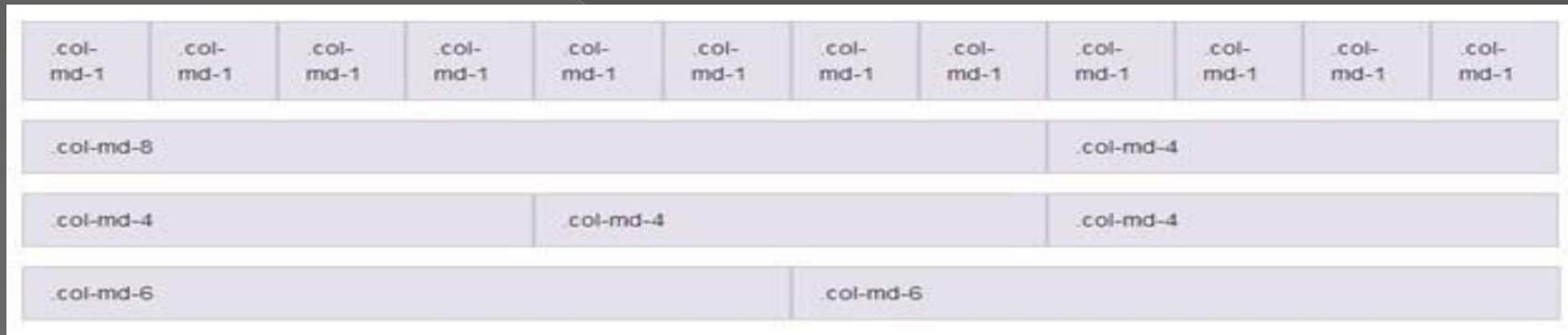
container is used to wrap the site contents.
There are two container classes.

- ◉ The **.container class** provides a responsive fixed width container.
- ◉ The **.container-fluid** class provides a full width container, spanning the entire width of the viewport.

```
<div class="container">  
<h1> First Bootstrap web page</h1>  
<p>Write your text here..</p>  
</div>
```

BOOTSTRAP GRID

The Bootstrap Grid System allows up to 12 columns across the page. You can use all 12 columns individually or you can group the columns together to create wider columns.



The diagram illustrates the Bootstrap Grid System with 12 columns. The columns are arranged in four rows, demonstrating different ways to group them:

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8								.col-md-4			
.col-md-4				.col-md-4				.col-md-4			
.col-md-6						.col-md-6					

Bootstrap Grid System is responsive and the columns are re-arranged automatically according to the screen size.

Grid Classes:

There are four classes in Bootstrap Grid System:

- ◉ xs (for phones)
- ◉ sm (for tablets)
- ◉ md (for desktops)
- ◉ lg (for larger desktops)

SYNTAX:-

- `<div class="container-fluid">`
- `<h1>Unequal Columns</h1>`
- `<p>Resize the browser window to see the effect.</p>`
- `<p>The columns will automatically stack on top of each other when the screen is less than 576px wide.</p>`
- `<div class="row">`
- `<div class="col-sm-4" style="background-color:lavender;">.col-sm-4</div>`
- `<div class="col-sm-8" style="background-color:lavenderblush;">.col-sm-8</div>`
- `</div>`
- `</div>`

Bootstrap Tables

- ◉ Bootstrap Basic Table

A basic Bootstrap table has a light padding and only horizontal dividers.

The `.table` class adds basic styling to a table

- ◉ Striped Rows

The `.table-striped` class adds zebra-stripes to a table

- ◉ Bordered Table

The `.table-bordered` class adds borders on all sides of the table and cells:

◉ Hover Rows

The `.table-hover` class adds a hover effect (grey background color) on table rows

◉ Bootstrap 4 Contextual Table

Contextual classes can be used to color the whole table (`<table>`), the table rows (`<tr>`) or table cells (`<td>`).

The classes that can be used are:

`.table-primary`, `.table-success`, `.table-info`,
`.table-warning`, `.table-danger`, `.table-active`, `.table-secondary`, `.table-light` and
`.table-dark`: [FOR EXAMPLE](#)

Bootstrap Buttons

There are seven styles to add a button in Bootstrap. Use the following classes to achieve the different button styles:

- ◉ `.btn-default`
- ◉ `.btn-primary`
- ◉ `.btn-success`
- ◉ `.btn-info`
- ◉ `.btn-warning`
- ◉ `.btn-danger`
- ◉ `.btn-link`

[For Example](#)

Bootstrap Button Size

In Bootstrap, you can choose a button according to your requirement. It provides four button sizes. The following classes define the different sizes:

- ◉ .btn-lg
- ◉ .btn-md
- ◉ .btn-sm
- ◉ .btn-xs

[For Example](#)

Bootstrap Images

Bootstrap Image Shapes

◉ Rounded Corners:



◉ Circle:



◉ Thumbnail:



Example

- rounded

```

```

- Circle

```

```

- Thumbnail

```

```


Creating a Jumbotron

- ◉ A jumbotron indicates a big box for calling extra attention to some special content or information.
- ◉ A jumbotron is displayed as a grey box with rounded corners. It also enlarges the font sizes of the text inside it.
- ◉ Jumbotron Inside Container

Place the jumbotron inside the `<div class="container">` if you want the jumbotron to NOT extend to the edge of the screen: [For Example](#)

Typography

Typography provides some utilities to add additional styles to texts. These utilities are:

- Text alignment
- Text transform
- Font weight and italics

TEXT ALIGNMENT

- **Align text left:**
- `<p class="text-left">Left aligned text.</p>`
- **Align text center:**
- `<p class="text-center">Center aligned text.</p>`
- **Align text right:**
- `<p class="text-right">Right aligned text.</p>`
- **Align text justify:**
- `<p class="text-justify">Justified text.</p>`
- **Align text no-wrap:**
- `<p class="text-nowrap">No wrap text.</p>`

You can align text on viewports according to their size also.

- Left aligned text on viewports sized SM (small) or wider.
- `<p class="text-sm-left">`Left aligned text on viewports sized SM (small) or wider.`</p>`
- Left aligned text on viewports sized MD (medium) or wider.
- `<p class="text-md-left">`Left aligned text on viewports sized MD (medium) or wider.`</p>`
- Left aligned text on viewports sized LG (large) or wider.
- `<p class="text-lg-left">`Left aligned text on viewports sized LG (large) or wider.`</p>`
- Left aligned text on viewports sized XL (extra-large) or wider.
- `<p class="text-xl-left">`Left aligned text on viewports sized XL (extra-large) or wider.`</p>`

TEXT TRANSFORM

- For lowercase text:

Use "text-lowercase" class to make the text appear in lowercase.

```
<p class="text-lowercase">Lowercased text.</p>
```

- For uppercase text:

Use "text-uppercase" class to make the text appear in uppercase.

```
<p class="text-uppercase">Uppercased text.</p>
```

- For capitalized text:

Use "text-capitalize" class to make the text's first letter appear in uppercase.

```
<p class="text-capitalize">CapItaliZed text.</p>
```

Font weight and italics

It is used to quickly change the weight (boldness) of text or italicize text.

- For bold text:

```
<p class="font-weight-bold">Bold text.</p>
```

- For Italic text:

```
<p class="font-italic">Italic text.</p>
```

Bootstrap Glyphicons

Glyphicons are the icon fonts that are used in web projects. Bootstrap provides 260 Glyphicons from the Glyphicons Halflings set.

- Some examples of Glyphicons are:
- Envelope glyphicon
- Print glyphicon
- Search glyphicon
- Download glyphicon etc.

Glyphicons Syntax

```
<span class="glyphicon glyphicon-name"></span>
```

For Example

THANK YOU



AGENDA

- What is Cloud ?
- What is Cloud Computing?
- History
- Top Benefits of Cloud Computing
- Simple Examples of Cloud Computing
- Essential Characteristics
- Cloud Computing Architecture
- Cloud Models
- Pros and Cons
- What is Microsoft Azure?
- Conclusion

What is Cloud?

- In Cloud Computing, the word cloud is used as a metaphor for “the Internet.” In other words, we can say cloud is something, which is present at remote location. Well it is an abstraction of underlying infrastructures involved.

What is Cloud Computing?

- Simply put, cloud computing is the delivery of computing services – servers, storage, databases, networking, software, and analytics and more- over the Internet(Cloud).
- Cloud Computing consists of hardware and software resources made available on the internet as they are managed by the third party services. These services typically provides access to advanced software applications, high end networks of server computers.

"You don't generate your own electricity. Why generate your own computing?" -Jeff Bezos, Amazon.

History

- It was a gradual evolution that started in the 1950s with mainframe computing
- After some time, around 1970, the concept of virtual machines (VMs) was created.
- In 1999, **Salesforce.com** started delivering of applications to users using a simple website.
- In 2002 Amazon provided First public cloud AWS (Amazon Web Service) , providing services like storage, computation.
- In 2009, **Google Apps** also started to provide cloud computing enterprise applications.
- In 2009, **Microsoft** launched Windows Azure, and companies like Oracle and HP have all joined the game. This proves that today, cloud computing has become mainstream.

Benefits of cloud computing

- Drive down costs
- Accessibility
- Productivity
- Scalability
- Access to automatic updates
- Business Continuity (Back up & Recovery)
- Pay structure

Simple Examples of cloud computing

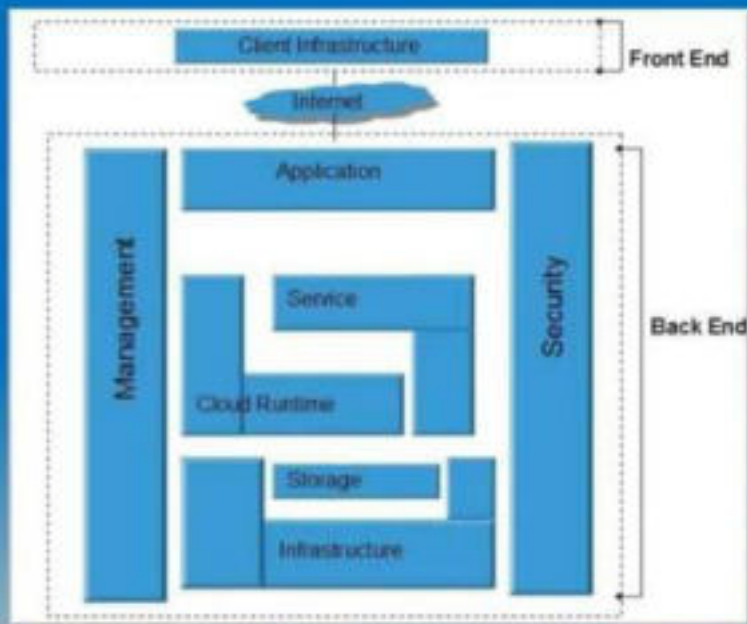
- **Email:** Web-based email services like Gmail and Hotmail deliver a cloud computing service: users can access their email "in the cloud" from any computer with a browser and Internet connection, regardless of what kind of hardware is on that particular computer. The emails are hosted on Google's and Microsoft's servers, rather than being stored locally on the client computer.
- **Office Productivity Software:** Office 365, Google docs and Zoho office. This software allow you to keep and edit your documents online. By doing so, the documents will be accessible anywhere, and you can share the documents and collaborate on them. Multiple people can work in the same document simultaneously.
- **Storage:** One Drive, Google Drive, iCloud and Drop Box.

Common Cloud Characteristics

- On Demand Self Service
- Broad network access
- Multi-Tenancy (Resource Pooling)
- Rapid Elasticity
- Measured service



Cloud Architecture



Cloud Models

- Deployment Models
- Service Models

Deployment Models

- A cloud deployment model represents a specific type of cloud environment, primarily distinguished by ownership, size, and access.
- There are three common cloud deployment models:



Deployment Models

- **Public Cloud:** Public clouds are owned and operated by a third party cloud service provider, which deliver their computing resources like servers and storage over the internet. As the name suggests, **Public cloud** is open to public. Anyone can access and use it by paying accordingly

The logo for Salesforce, consisting of a blue cloud shape with the word "salesforce" written in white lowercase letters inside it.

salesforce

Deployment Models

- **Private Cloud:** The private cloud, in contrast to its public counterpart, isn't available to the public but is built specifically for a single organization to fit its needs. It may be managed internally or by a third-party and be hosted internally or externally.
- **Hybrid Cloud:** A hybrid cloud is a combination of a private cloud combined with the use of public cloud services allowing data and applications to move between private and public clouds. This model gives business greater flexibility and more deployment options

Service Models

In the world of cloud computing, there are three different approaches to cloud-based services:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

Service Models

- **Infrastructure as a service (IaaS)**: is a cloud computing offering in which a vendor provides users access to computing resources such as servers, storage, and networking. Organizations use their own platforms and applications within a service provider's infrastructure.

Key features

- Instead of purchasing hardware outright, users pay for IaaS on demand.
- Infrastructure is scalable depending on processing and storage needs.
- Saves enterprises the costs of buying and maintaining their own hardware.
- Because data is on the cloud, there is no single point of failure.

Service Models

- **Platform as a service (PaaS):** is a cloud computing offering that provides users a cloud environment in which they can develop, manage, and deliver applications. In addition to storage and other computing resources, users are able to use a suite of prebuilt tools to develop, customize and test their own applications.

Key features

- PaaS provides a platform with tools to test, develop, and host applications in the same environment.
- Enables organizations to focus on development without having to worry about underlying infrastructure.
- Providers manage security, operating systems, server software, and backups.
- Facilitates collaborative work even if teams work remotely.

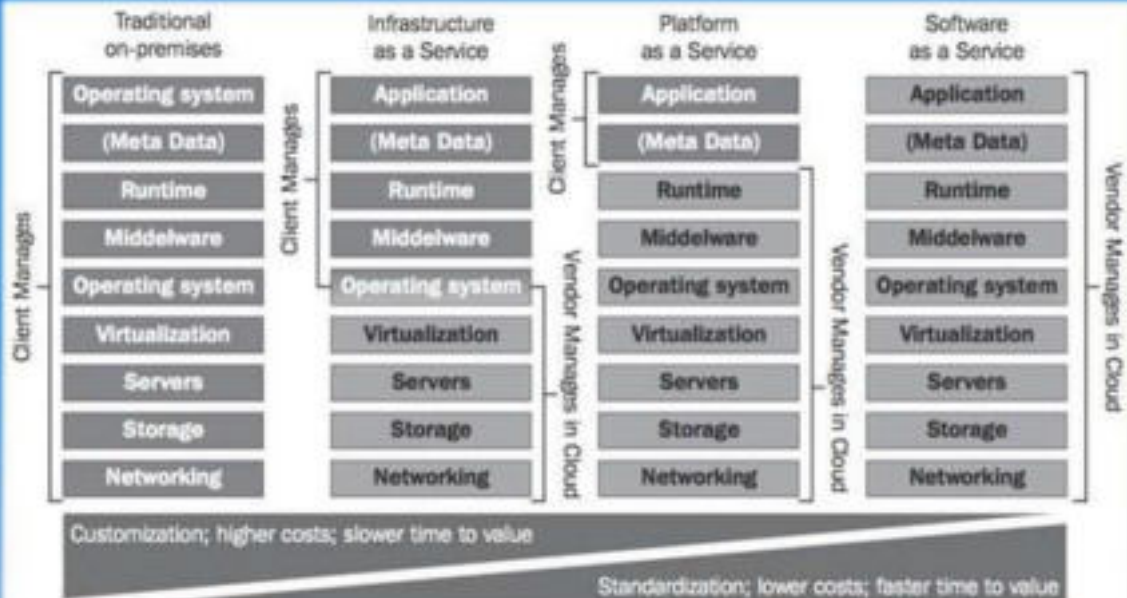
Service Models

- **Software as a service (SaaS):** is a cloud computing offering that provides users with access to a vendor's cloud-based software. Users do not install applications on their local devices. Instead, the applications reside on a remote cloud network accessed through the web or an API. Through the application, users can store and analyze data and collaborate on projects.

Key features

- SaaS vendors provide users with software and applications on a subscription model.
- Users do not have to manage, install, or upgrade software; SaaS providers manage this.
- Data is secure in the cloud; equipment failure does not result in loss of data.
- Use of resources can be scaled depending on service needs.

Service Models



Service Models

- IaaS Providers



Microsoft
Azure



Service Model

- **PaaS Providers**



Service Models

- **SaaS Providers**



Pros and Cons

Pros:

- Reduced hardware equipment for end-users
- Improved performance
- Lower H/W and S/W maintenance
- Instant software updates
- Accessibility
- Metered services
- Less expensive
- Improved Disaster Recovery

Cons:

- Requires good internet speed with good bandwidth
- Security
- Limited control on Infrastructure

What is Microsoft Azure?

- Azure is a flexible cloud platform (PaaS) that enables you to quickly build, deploy and manage applications across a global network of Microsoft – managed datacenters.
- You can build applications using any language, tool or framework.



Microsoft Azure

- **Virtual Machines:**

Azure gives you the ability to create VMs by simply specifying the size and virtual hard disks (VHD) you want to use. Azure provides access to both Windows and Linux VHDs, so the developers has a freedom to choose what they want to work. Developers can use VMs to build and test applications quickly at low cost.

- **Web Sites:**

You can use Azure as a platform for creating and hosting websites and web applications

Microsoft Azure

- **Mobile Services:**

Azure's Mobile services give you the tools to create and deploy applications. The information that gets accessed by the app running on your device is stored in what's called a back-end database, and so Mobile services are referred to as mobile Back-end as a service (mBaaS). With Azure, you can build apps for Android, iOS, HTML / JavaScript and Windows Phone.



Microsoft Azure

- Azure supports the broadest selection of operating systems, programming languages, frameworks, tools, databases and devices. Build apps with JavaScript, Python, .NET, PHP, Java and Node.js; build back-ends for iOS, Android and Windows devices. Azure cloud service supports the same technologies millions of developers and IT professionals already rely on and trust.

Conclusion

- Cloud computing has quickly become one of the most prominent buzzwords in the IT world due to its revolutionary model of computing as a utility. It promises increased flexibility, scalability, and reliability, while promising decreased operational and support costs
- Despite the potential gains achieved from the cloud computing, the organizations are slow in accepting it due to security issues and challenges associated with it. Security is one of the major issues which hamper the growth of cloud. The idea of handing over important data to another company is worrisome; such that the consumers need to be vigilant in understanding the risks of data breaches in this new environment.

Thank you.

**A Seminar
On
Cloud Computing**

INTRODUCTION

Cloud Computing provides us a means by which we can access the applications as utilities, over the Internet. It allows us to create, configure, and customize applications online.

With Cloud Computing users can access database resources via the internet from anywhere for as long as they need without worrying about any maintenance or management of actual resources.

What is Cloud?

The term **Cloud** refers to a **Network** or **Internet**. In other words, we can say that Cloud is something, which is present at remote location.

Cloud can provide services over network, i.e., on public networks or on private networks, i.e., WAN, LAN or VPN.

Applications such as **e-mail**, **web conferencing**, **customer relationship management (CRM)**, all run in cloud.

What is Cloud Computing?

Cloud Computing refers to **manipulating, configuring, and accessing** the applications online. It offers online data storage, infrastructure and application.

Cloud Computing is both a combination of software and hardware based computing resources delivered as a network service.

Cloud Computing Architecture



Basic Concepts

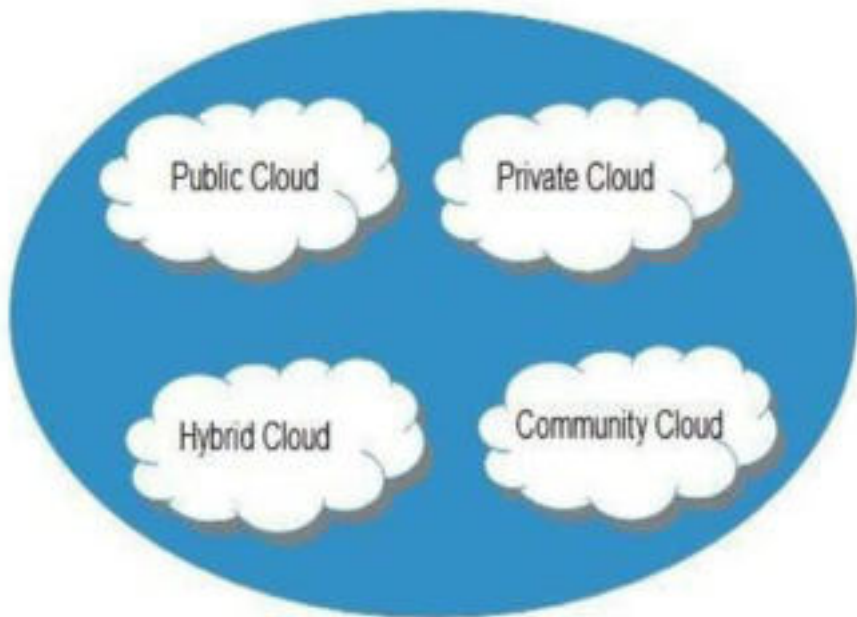
There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users. Following are the working models for cloud computing:

1. Deployment Models

2. Service Models

Deployment Models

Deployment models define the type of access to the cloud, i.e., how the cloud is located? Cloud can have any of the four types of access: Public, Private, Hybrid and Community.



PUBLIC CLOUD : The **Public Cloud** allows systems and services to be easily accessible to the general public, Public cloud may be less secure because of its openness, e.g., e-mail.

PRIVATE CLOUD : The **Private Cloud** allows systems and services to be accessible within an organization. It offers increased security because of its private nature.

COMMUNITY CLOUD : The **Community Cloud** allows systems and services to be accessible by group of organizations.

HYBRID CLOUD : The **Hybrid Cloud** is mixture of public and private cloud. However, the critical activities are performed using private cloud while the non-critical activities are performed using public cloud.

Service Models

Service Models are the reference models on which the Cloud Computing is based. These can be categorized into three basic service models as listed below:

1. Infrastructure as a Service (IaaS)

2. Platform as a Service (PaaS)

3. Software as a Service (SaaS)

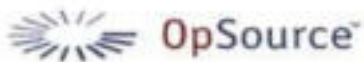
Infrastructure as a Service (IaaS)

IaaS is the delivery of technology infrastructure as an on demand scalable service.

IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc.

- Usually billed based on usage
- Usually multi tenant virtualized environment
- Can be coupled with Managed Services for OS and application support

IaaS Examples



Platform as a Service (PaaS)

PaaS provides the runtime environment for applications, development & deployment tools, etc.

PaaS provides all of the facilities required to support the complete life cycle of building and delivering web applications and services entirely from the Internet.

Typically applications must be developed with a particular platform in mind

- Multi tenant environments
- Highly scalable multi tier architecture

PaaS Examples



Software as a Service (SaaS)

SaaS model allows to use software applications as a service to end users.

SaaS is a software delivery methodology that provides licensed multi-tenant access to software and its functions remotely as a Web-based service.

- Usually billed based on usage
- Usually multi tenant environment
- Highly scalable architecture

SaaS Examples



Do you Use the Cloud?



Advantages

- Lower computer costs
- Improved performance:
- Reduced software costs
- Instant software updates
- Improved document format compatibility
- Unlimited storage capacity
- Increased data reliability
- Universal document access
- Latest version availability
- Easier group collaboration
- Device independence

Disadvantages

- Requires a constant Internet connection
- Does not work well with low-speed connections
- Features might be limited
- Can be slow
- Stored data can be lost
- Stored data might not be secure

Cloud Storage

The logo for Box.com, featuring the word "box" in a lowercase, blue, sans-serif font.The logo for SkyDrive, featuring a blue cloud icon with a yellow swoosh and the text "SkyDrive" in a black, sans-serif font.

SkyDrive

- Create an Account
User name and password.



Dropbox

The logo for Amazon Cloud Drive, featuring the Amazon logo and the text "amazon cloud drive" in a small, black, sans-serif font.

amazon cloud drive

- Content lives with the account in the cloud.
- Log onto any computer with Wi-Fi to find your content

Download For Storage

- Download a cloud based app to on your computer
- The app lives on your Computer
- Save files to the app
- When connected to the Internet it will sync with the cloud
- The Cloud can be accessed from any Internet connection



10:20 AM

2/5/2013

Thank

you...

Question...?

Cloud Computing: An Introduction

Cloud computing is the delivery of computing services over the internet. Learn about different types of clouds, advantages, challenges, implementation, security concerns, and the future of cloud computing.



by **GYANENDRA SHUKLA**



What is Cloud Computing?

1 On-Demand Access

Access to shared resources, like servers, storage, and databases, on-demand over the internet.

2 Scalability & Flexibility

Ability to scale resources up or down, depending on business needs, reducing costs and enabling rapid innovation.

3 Pay-As-You-Go

Usage-based pricing model, allowing businesses to pay only for the resources they use, rather than making large upfront investments.

Types of Clouds

Public Cloud

Resources shared with multiple users.

Private Cloud

Resources dedicated to a single organization.

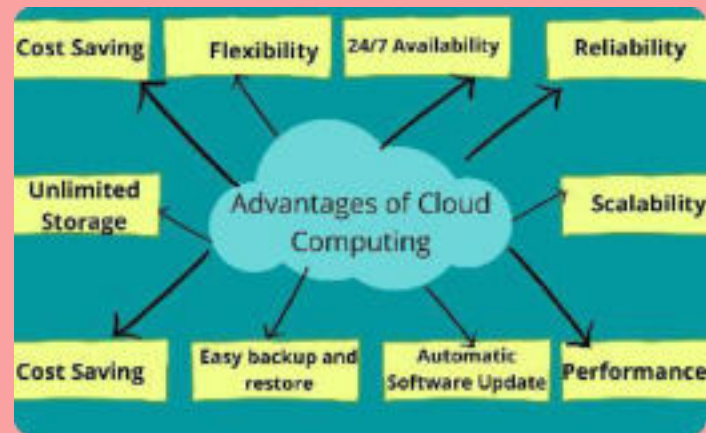
Hybrid Cloud

Combination of public and private cloud.

Community Cloud

Resources shared with specific community members.

Advantages of Cloud Computing



Cost Savings

Eliminate the need for upfront investment in hardware and infrastructure maintenance costs.

Scalability

Scale resources up or down to match your business needs.

Collaboration

Enable collaboration and remote work across teams and locations.

Challenges of Cloud Computing

Data Security

Protecting sensitive data in the cloud.

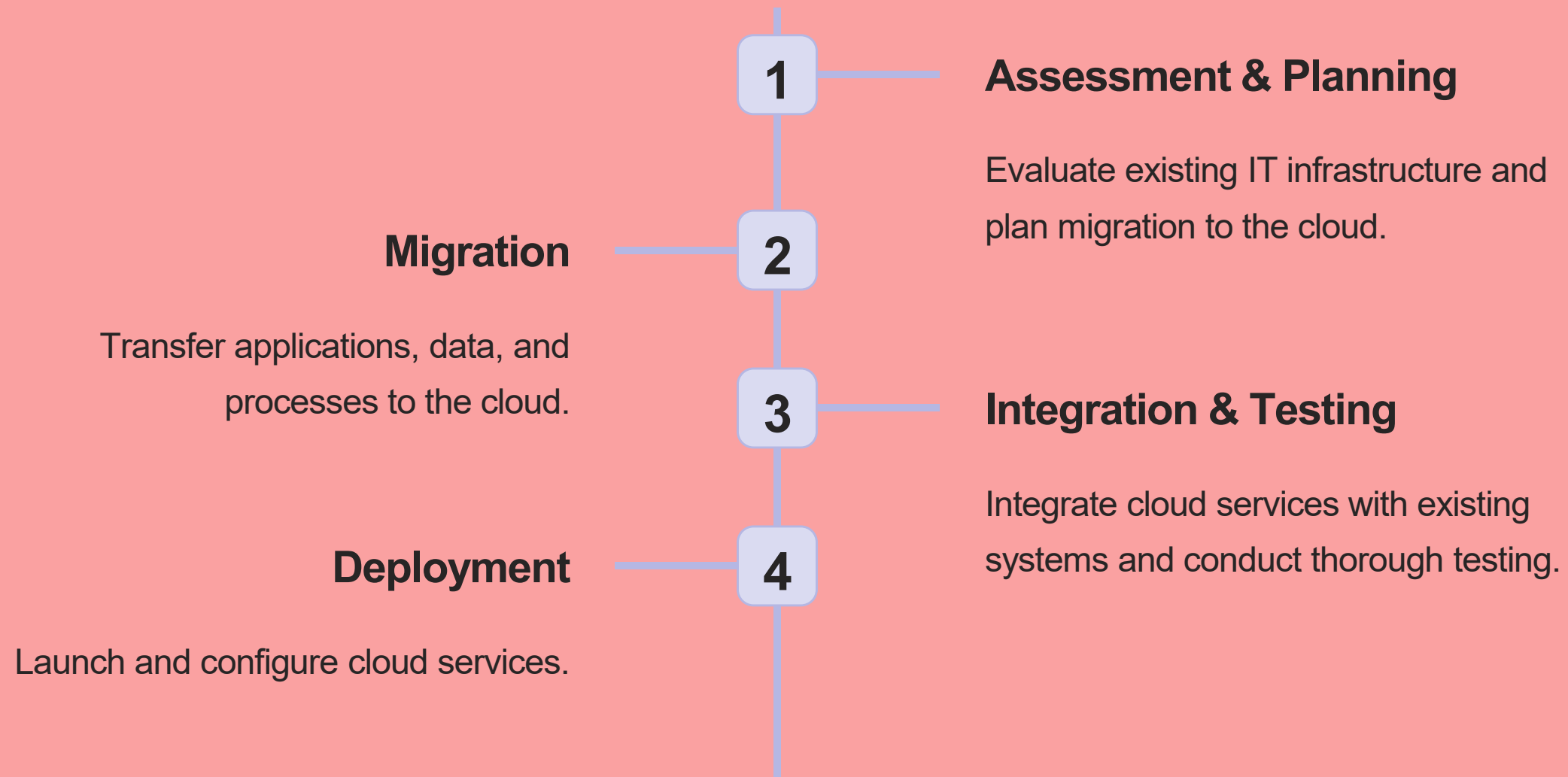
Compliance

Ensuring compliance with regulatory requirements.

Vendor Lock-In

Dependency on a specific cloud provider.

Implementing Cloud Computing



Security Concerns

1 Data Breaches

Protecting data from unauthorized access and breaches.

2 Identity and Access Management

Ensuring proper authentication and authorization controls.

3 Encryption

Securing data in transit and at rest through encryption.

Future of Cloud Computing

Edge Computing

Performing compute and data processing closer to the edge devices, reducing latency.

AI & Machine Learning

Leveraging cloud computing for advanced analytics and intelligent automation.

Serverless Computing

Building and running applications without the need to manage server infrastructure.

Cloud Computing: An Introduction

Cloud computing is the delivery of computing services over the internet. Learn about different types of clouds, advantages, challenges, implementation, security concerns, and the future of cloud computing.



by **GYANENDRA SHUKLA**



What is Cloud Computing?

1 On-Demand Access

Access to shared resources, like servers, storage, and databases, on-demand over the internet.

2 Scalability & Flexibility

Ability to scale resources up or down, depending on business needs, reducing costs and enabling rapid innovation.

3 Pay-As-You-Go

Usage-based pricing model, allowing businesses to pay only for the resources they use, rather than making large upfront investments.

Types of Clouds

Public Cloud

Resources shared with multiple users.

Private Cloud

Resources dedicated to a single organization.

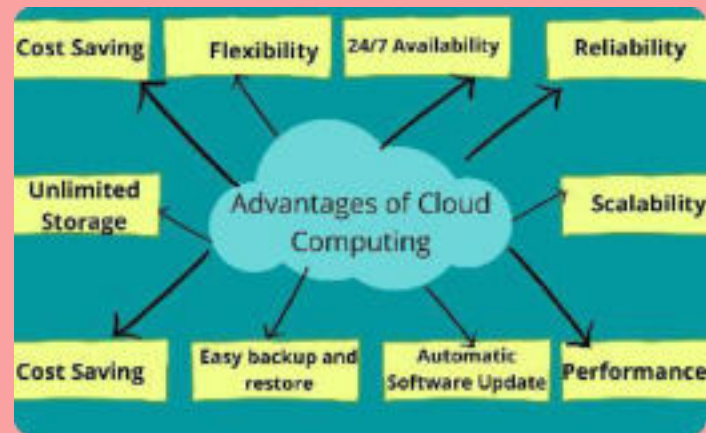
Hybrid Cloud

Combination of public and private cloud.

Community Cloud

Resources shared with specific community members.

Advantages of Cloud Computing



Cost Savings

Eliminate the need for upfront investment in hardware and infrastructure maintenance costs.

Scalability

Scale resources up or down to match your business needs.

Collaboration

Enable collaboration and remote work across teams and locations.

Challenges of Cloud Computing

Data Security

Protecting sensitive data in the cloud.

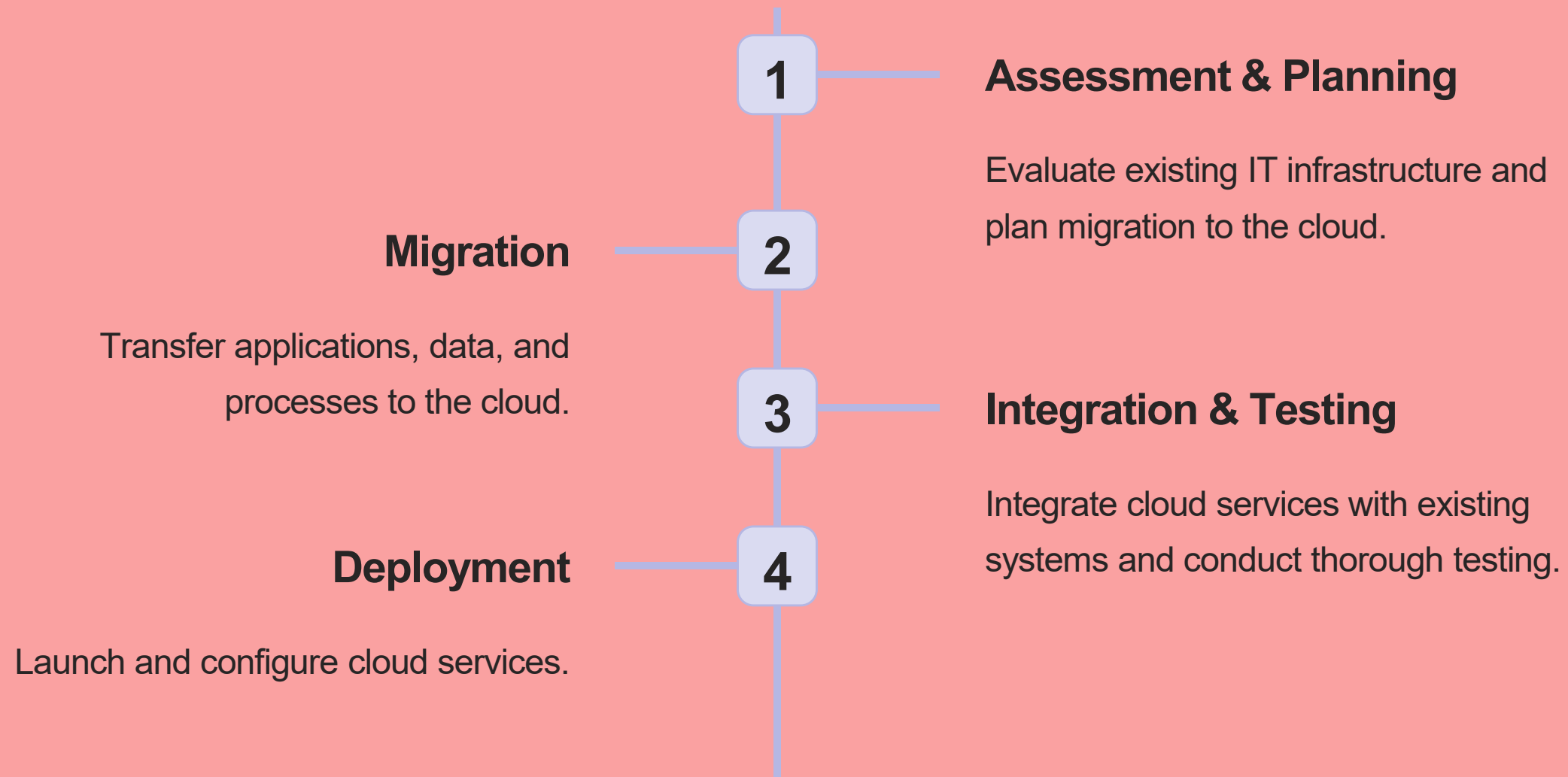
Compliance

Ensuring compliance with regulatory requirements.

Vendor Lock-In

Dependency on a specific cloud provider.

Implementing Cloud Computing



Security Concerns

1 Data Breaches

Protecting data from unauthorized access and breaches.

2 Identity and Access Management

Ensuring proper authentication and authorization controls.

3 Encryption

Securing data in transit and at rest through encryption.

Future of Cloud Computing

Edge Computing

Performing compute and data processing closer to the edge devices, reducing latency.

AI & Machine Learning

Leveraging cloud computing for advanced analytics and intelligent automation.

Serverless Computing

Building and running applications without the need to manage server infrastructure.

Fundamental of Computer Network

Gyanendra Shukla

Block -1
Concepts of Communication
&
Network

UNIT-1

BASICS OF DATA COMMUNICATION

- Introduction
- Objectives
- Concept of Communication System
- Analog & Digital Communication
- Data Communication Mode
- Networking Protocol & Standards
- Application of Computer Networking

What is a Computer Network?

- Computer Network is a group of computers connected with each other through wires, optical fibers or optical links so that various devices can interact with each other through a network.
- The aim of the computer network is the sharing of resources among various devices.
- A computer network is a set of devices connected through links. A node can be computer, printer, or any other device capable of sending or receiving the data. The links connecting the nodes are known as communication channels.

Features Of Computer network

A list Of Computer network features is given below.

- Communication speed
- File sharing
- Back up and Roll back is easy
- Software and Hardware sharing
- Security
- Scalability
- Reliability

Why Computer Networks

- File Sharing
- Hardware Resource sharing
- Communication and collaboration
- Organization
- Remote access
- Data protection
- Internet & Intranet

Introduction of Basics of Data Communication

- Data Communication is a process of exchanging data or information. In case of computer networks, this exchange is done between two devices over a transmission medium.
- Data can be any text, image, audio, video, and multimedia files. Communication is an act of sending or receiving data.
- Data communication refers to the exchange of data between a source and a receiver via a form of transmission media such as a wire cable. Data communication is said to be local if communicating devices are in the same building or a similarly restricted geographical area.

Objectives

After going through this unit, you should be able to:

- Know the concept of communication system
- Understand the communication system and its components
- Differentiate between analog and digital Communication
- Know data communication modes
- Differentiate between synchronous and asynchronous transmission
- Differentiate among Simplex, half-duplex, full duplex communication
- Understand the need of protocols and standard
- Know the functions of OSI layers
- Understand the concepts of encapsulation and End-to-end argument
- Know the different protocol design issues
- Know the applications of computer network

CONCEPT OF COMMUNICATION SYSTEM

Before we discuss about “communication system” and its components, let us understand “communication”. Can you define it, what definition will come first in your mind? When we asked some students, answers were like:

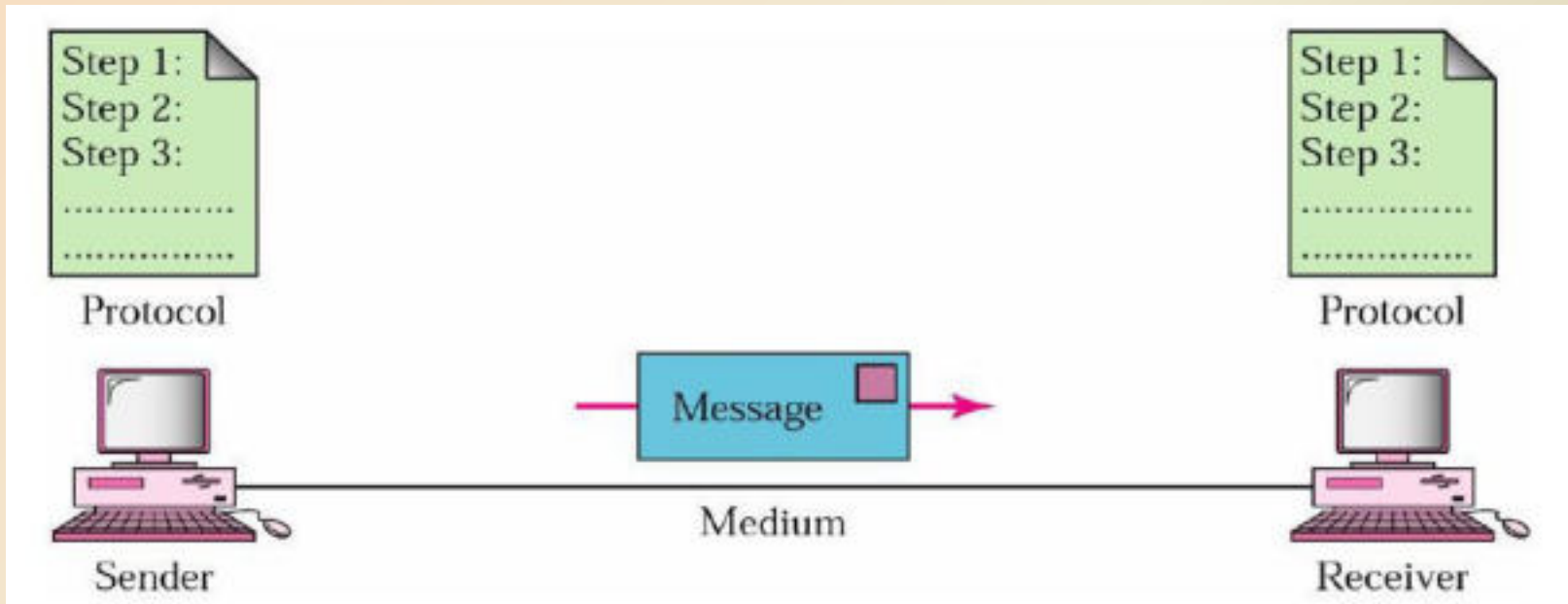
- Delivery of message
- Proper way of passing a signal to the intended user
- Right message, to right person, at right time through right way.

Some definitions of communication:

- ❖ ” Communication is transfer of information from one person to another, whether or not it elicits confidence. But the information transferred must be understandable to the receiver – G.G. Brown
- ❖ The imparting or exchanging of information by speaking, writing, or using some other medium.-Oxford Dictionary

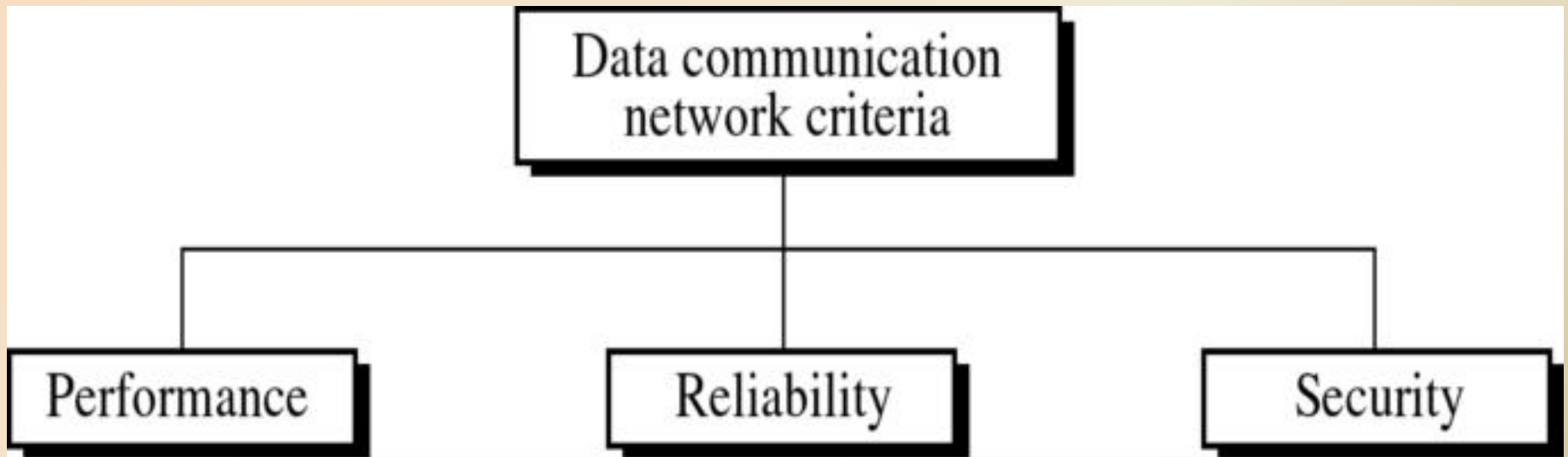
Components of Network Communications

- ✓ **Message** : Data /Information to be communicates
- ✓ **Sender** : Create and send messages
- ✓ **Receiver** : Receive the messages
- ✓ **Medium** : Which carry the message from sender to receiver
- ✓ **Protocol** : A set of rules that govern communications



Networks – Distributed Processing

Network is a set of devices or nodes connected by communication links. A node may be a computer, printer, scanner or any other electronic devices. In network work is to be distributed among all nodes instead of one single computer. It increase the system performances. A network must maintain certain criteria such as *Performance, Reliability* and *Security*



Protocols & Standards

In Computer Networks *protocols* and *standards* are very important.

Protocol : A protocol is a set of rules that govern data communications. A protocol defines what is communicated, how it is communicated, and when it is communicated.

The Key elements of the protocol are *Syntax, Semantics & Timing*

Syntax : It refers Structure and format of the data that implies the order in which it will be presented.

Semantics : It refers Section of bits how the particular pattern to be interpreted.

Timing: It refers Transmission time

Protocols & Standards

Standards : Network has been standardized by number of regulatory body for different activities

ISO – International Standard Organization

ANSI – American National Standard Institute

IEEE – Institute of Electrical and Electronics Engineers

EIA – Electronic Industries Association

Line Configuration

1. Point to Point
2. Multi - Point

As the link is dedicated the question of two devices transmitting at the same time does not arise and so no medium access control protocol needed.

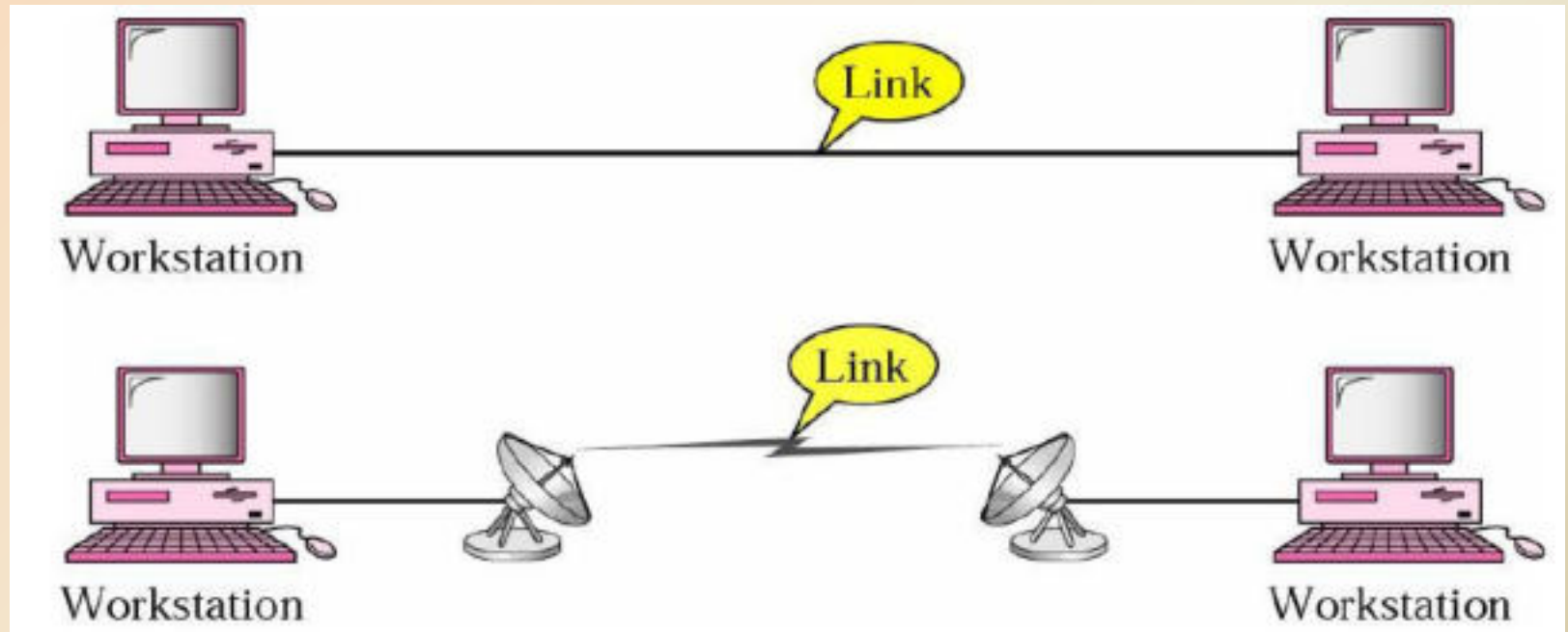


Figure : Point to Point Connection

Two devices transmitting at the same time results in a collision. So Medium Access Control (MAC) protocols are required to avoid collisions and subsequent retransmissions.

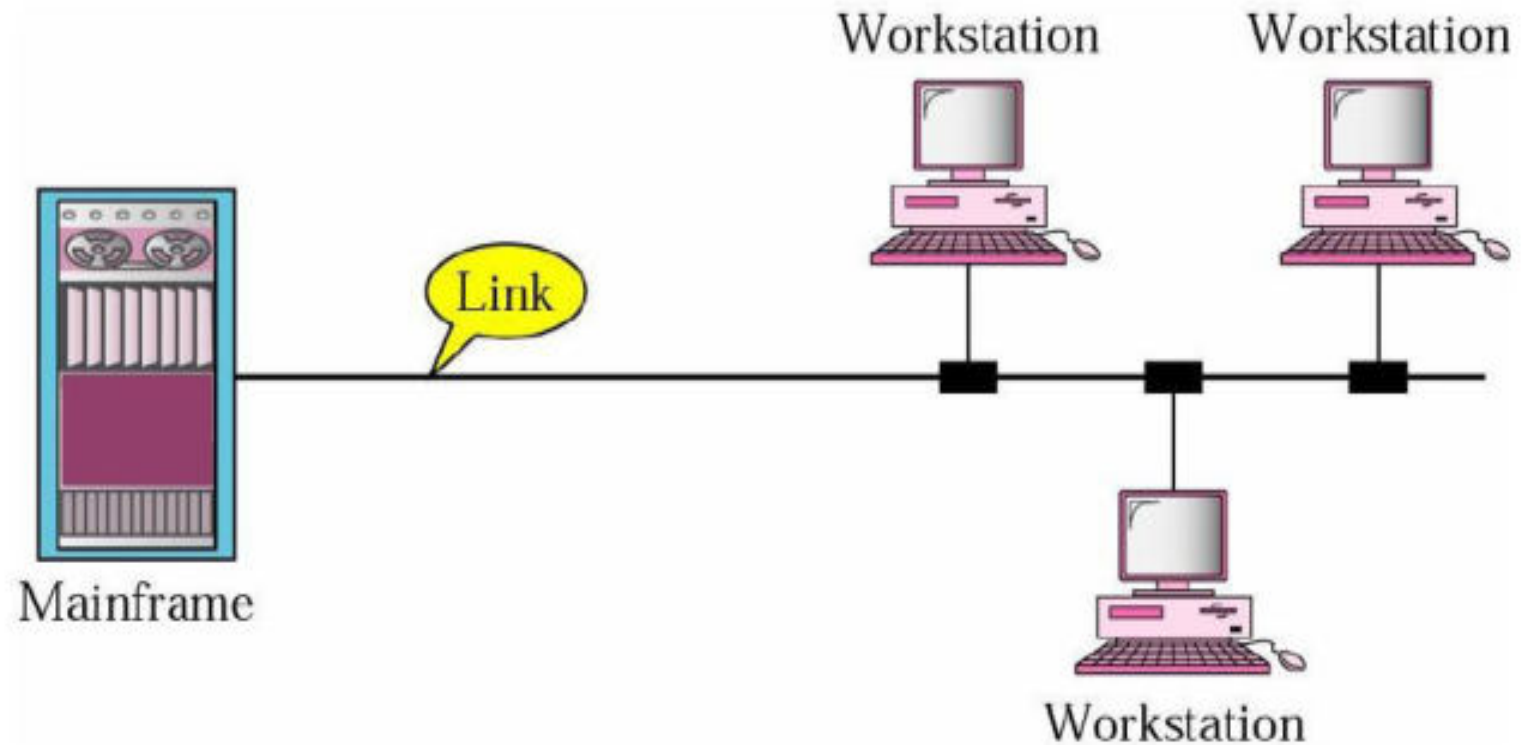
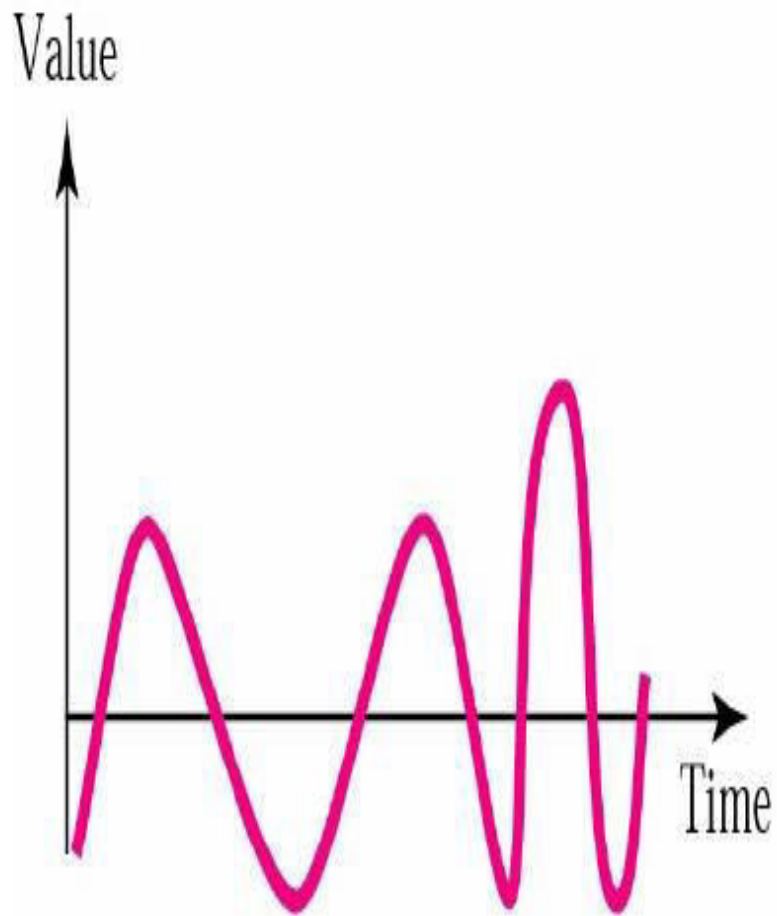


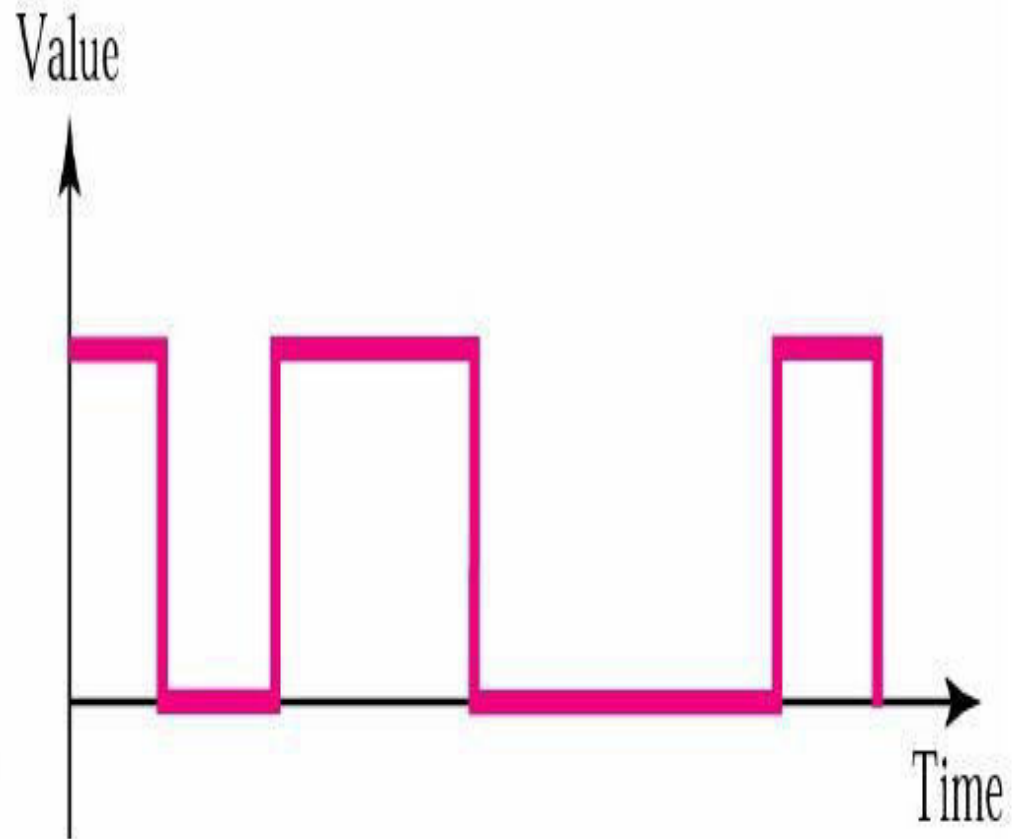
Figure : Multi - Point Connection

ANALOG AND DIGITAL COMMUNICATION

- Analog communication uses analog signals for the transmission of information. Digital communication uses digital signals for the transmission of information. Analog communication uses signals that can be represented by sine waves. Digital communication uses signals that can be represented by square waves.
- A signal which is a continuous function of time and used to carry the information is known as an analog signal. An analog signal represents a quantity analogous to another quantity, for example, in case of an analog audio signal, the instantaneous value of signal voltage represents the pressure of the sound wave.
- A signal that is discrete function of time, i.e. which is not a continuous signal, is known as a digital signal. The digital signals are represented in the binary form and consist of different values of voltage at discrete instants of time.
- Basically, a digital signal represents the data and information as a sequence of separate values at any given time. The digital signal can only take on one of a finite number of values.



a. Analog signal



b. Digital signal

Analog and digital signals are used to transmit information, usually through electric signals. In both these systems, the information, such as any text, audio or video, is transformed into electric signals. Let us see some of the differences between analog and digital systems below in table 1.

Table 1: Comparison between Analog and Digital system

Analog	Digital
Signals are records waveforms as they are. Signal occupies the same order of spectrum as the analog data.	Converts analog waveforms into set of numbers and records them. The numbers are converted into voltage stream for representation. In case of binary it is converted in 1's and 0's.
In analog systems electronic circuits are used for transformation of signals.	In this transformation is done using logic circuits.
About Noise analog signals are more likely to get affected and results in reducing accuracy	Digital signals are less affected, because noise response are analog in nature
Analog signal is a continuous signal which transmits information as a response to changes in physical phenomenon.	Digital signals are discrete time signals generated by digital modulation.
Data transmission is not of high quality	Data transmission has high quality.
Analog devices are not very precise.	Digital systems are very precise.

Data Communication Modes

- In this section, we will learn about some modes of data communication used in computer networking. Because we are going to study computer networking, we assume all data communication is digital. Digital communications is the physical transfer of data/bits over a communication channel.
- As you may know, data are represented as an electromagnetic signal, such as an electrical voltage, radio-wave, microwave, or infrared signal. The channel or medium could be air (for wireless/mobile communication), copper wires, or optical-fibers. Remember, the data transmitted can be pure digital messages generated from a digital-data source, like a computer or a keyboard.
- However, it may also be an analog signal such as a human voice over phone call, which could be digitalized

Data Communication Modes

- In this section, we will learn about some modes of data communication used in computer networking. Because we are going to study computer networking, we assume all data communication is digital. Digital communications is the physical transfer of data/bits over a communication channel.
- As you may know, data are represented as an electromagnetic signal, such as an electrical voltage, radio-wave, microwave, or infrared signal. The channel or medium could be air (for wireless/mobile communication), copper wires, or optical-fibers. Remember, the data transmitted can be pure digital messages generated from a digital-data source, like a computer or a keyboard.
- However, it may also be an analog signal such as a human voice over phone call, which could be digitalized

Data Communication Modes

Serial and parallel transmission

- In digital communication, **serial transmission** of data refers to sequential transmission of bits, where a group of bits over a single channel represents a character. It requires less processing and fewer chances for error. The start and stop of a communication is specified by LSB (lowest significant bit) and MSB (most significant bit) as shown in the Figure.
- **Parallel transmission** refers to simultaneous transmission of the bits over two or more separate channels. Here, we can transmit multiple bits simultaneously as given in Figure 6, which allows for higher data transfer rates than that can be achieved with serial transmission. For example, for internal data communication in a computer system this method of parallel transmission is used. Parallel data transmission is less reliable for long distances because error correction is not very simple and economical in this case.

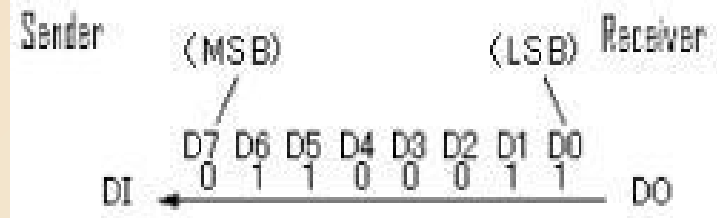


Figure 5: Serial Communication

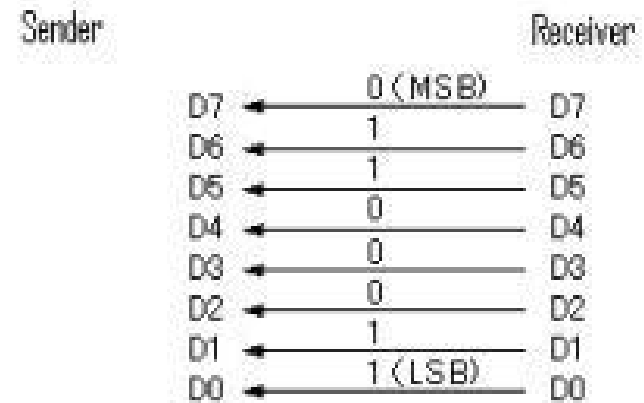


Figure 6: Parallel data transmission

Data Communication Modes

- ❖ Synchronous and Asynchronous Transmission
- ❖ Simplex, Half-Duplex, Full Duplex Communication

Synchronous and Asynchronous Transmission

- Synchronous transmission means both receiver and sender has an agreement (or aware) about timing for the sending data, so that both sender and receiver can coordinate (synchronize) their data signals. Asynchronous means "not synchronous", or no coordination between sender and receiver before transmission. Can you try to explore some examples of Synchronous and asynchronous communication that occurs in your day to day life
- The asynchronous transmission uses start and stop bits to signify the beginning bit. For example, if sender wants to send some data "11100001", it will be appended with the start and stop bit and look like "1 11100001 0". Where, we have assumed that '0' is start bit and '1' is stop bit. Asynchronous transmission works well where the characters are transferred at irregular intervals e.g. data entry from the keyboard.

Synchronous and Asynchronous Transmission

Asynchronous transmission has some advantages and disadvantages, like:

- Each individual character is complete unit, hence if there is an error in a character, other sequence of characters are not affected. However, Error in start and stop bit(s) may cause serious problems in data transfer.
- Doesn't require synchronization of both communication sides.
- It is cost effective
- The speed of transmission is limited.
- Large relative overheads, a high proportion of the transmitted bits are uniquely used for control purposes

Synchronous and Asynchronous Transmission

Synchronous transmission has some advantages and disadvantages, like:

- In comparison to asynchronous communication it has higher speeds, because the system has lesser possibility of error. But, if an error takes place, the complete set of data is lost instead of a single character.
- Serial synchronous transmission is principally used for high-speed communication between computers but is unsuitable where the characters are transferred at irregular intervals.
- It gives lower overheads and thus, greater throughput.
- Process is more complex
- It is not very cost effective as hardware are more expensive

Simplex, Half-Duplex, Full Duplex Communication

The data transmission mode on the channel, can be classified into three ways simplex, half-duplex and full-duplex as given below

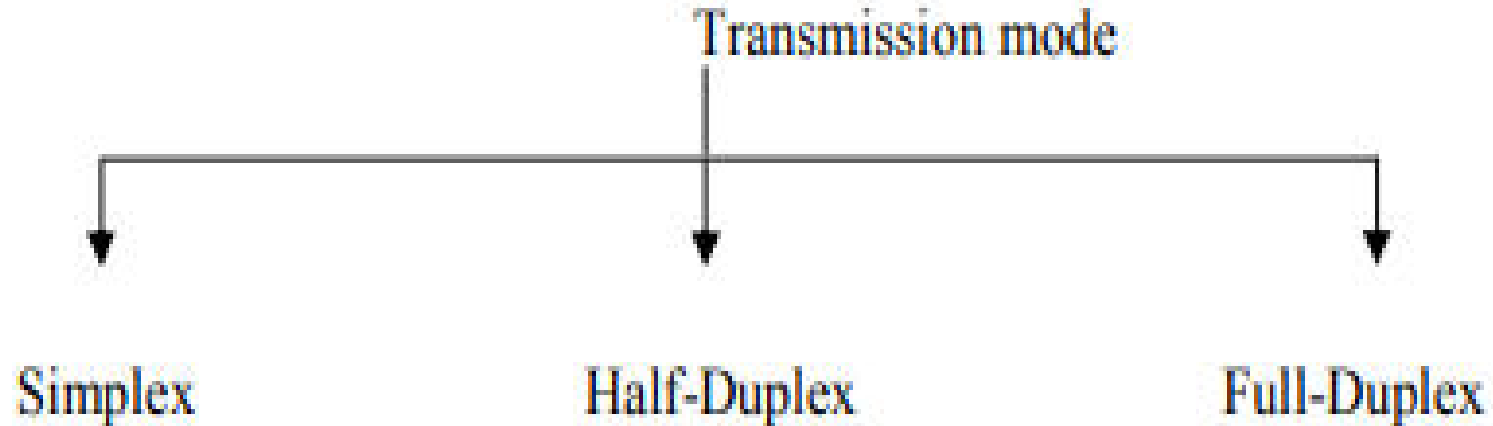


Figure 7: Transmission mode

Simplex Transmission

Simplex transmission is one-way transmission. As the name implies, is simple in term of process and hardware. It is also called unidirectional because the signal travels in only one direction. For example, Radio or TV broadcasting system, which are always in one direction from Radio/TV station to our radio or TV sets.

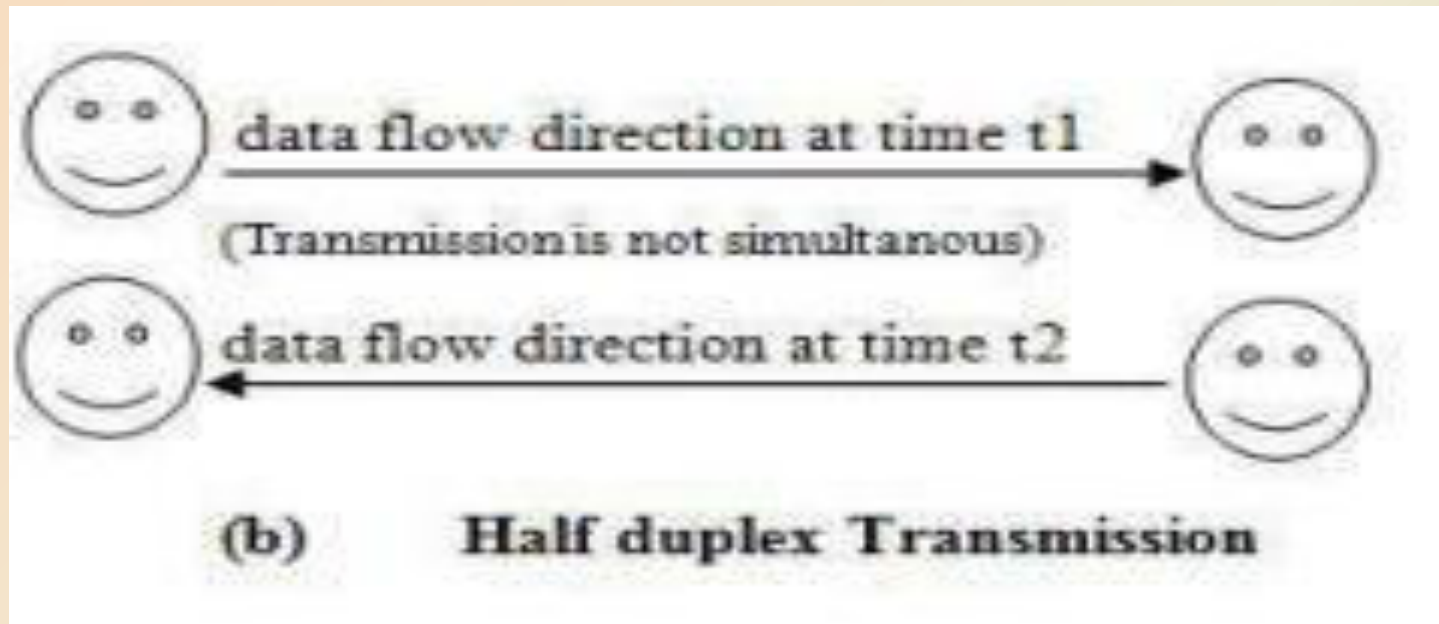


(a)

Simplex Transmission

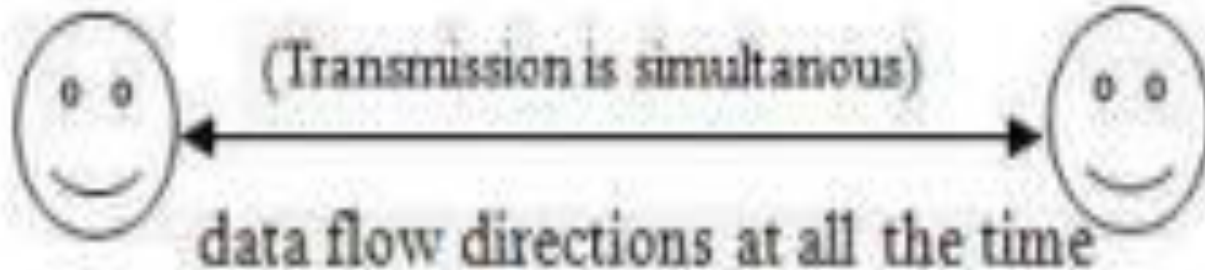
Half-Duplex Transmission

In half-duplex transmission data transmission can be take place in both directions, but not at the same time. This means that only one side can transmit at a time. For example, walky-talky devices used by security agencies are half-duplex as only one person can talk at one time



Full-Duplex Transmission

Full-duplex (also known as Duplex) transmission can take place in both directions at the same time. For example, telephone or mobile conversation is an example of full-duplex communication, where both sender and receiver can hear each other at the same time.



(c) Full duplex Transmission

NETWORKING PROTOCOLS AND STANDARDS

Problems in Computer Communication

- **Hardware Issues** : At the hardware level, an additional component called router is used to connect physically distinct networks. A router connects to the network in the same way as any other computer. Any computer connected to the network has a Network Interface Card (NIC), which has the address (network id+ host id), hard coded into it. A router is a device with more than one NICs. Router can connect incompatible networks as it has the necessary hardware (NIC) and protocols.
- **Software Issues:** The routers must agree about the way information would be transmitted to the destination computer on a different network, since the information is likely to travel through different routers, there must be a predefined standard to which routers must conform. Packet formats and addressing mechanism used by the networks may differ. One approach could be to perform conversion and reconversion corresponding to different networks. But this approach is difficult and cumbersome. Therefore, the Internet communication follows one protocol suite, the TCP/IP.

NETWORKING PROTOCOLS AND STANDARDS

Layering

Since it is difficult to deal with complex set of rules, and functions required for computer networking, these rules and functions are divided into logical groups called layers. Each layer can be implemented interdependently with an interface to other layers providing with services to it or taking its services like data, connection and error control functions are grouped together and make a layer.

A. Speech in telephone conversation is translated, with electrical segments and vice-versa. Similarly in computer system the data or pattern are converted into signals before transmitting and receiving. These function and rules are grouped together and form a layer.

OSI Model

International Standard Organization (ISO), established in 1947, is a multinational body dedicated to worldwide agreement on International Standards. ISO standards cover all aspects of network communications is the Open System Interconnection (OSI) Model.

It consists of Seven separate but related layers, each of which defines a segment of process of moving information across of network.

Features :

- Regardless of underlying network architecture
- Not Vendor Specific
- OSI is not a protocol – A model

Layered Hierarchy

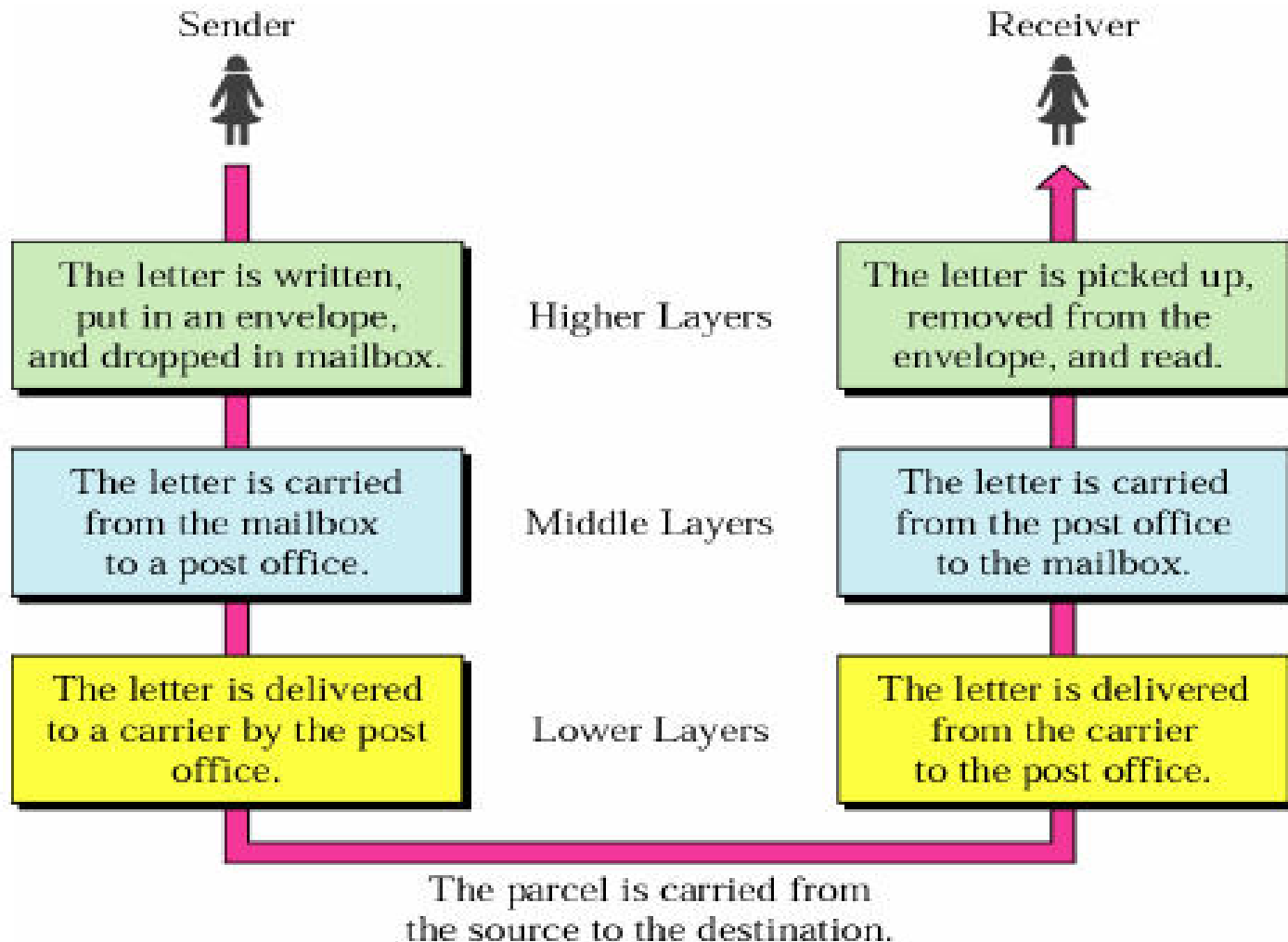


Figure *Sending a letter*

OSI Layers Function

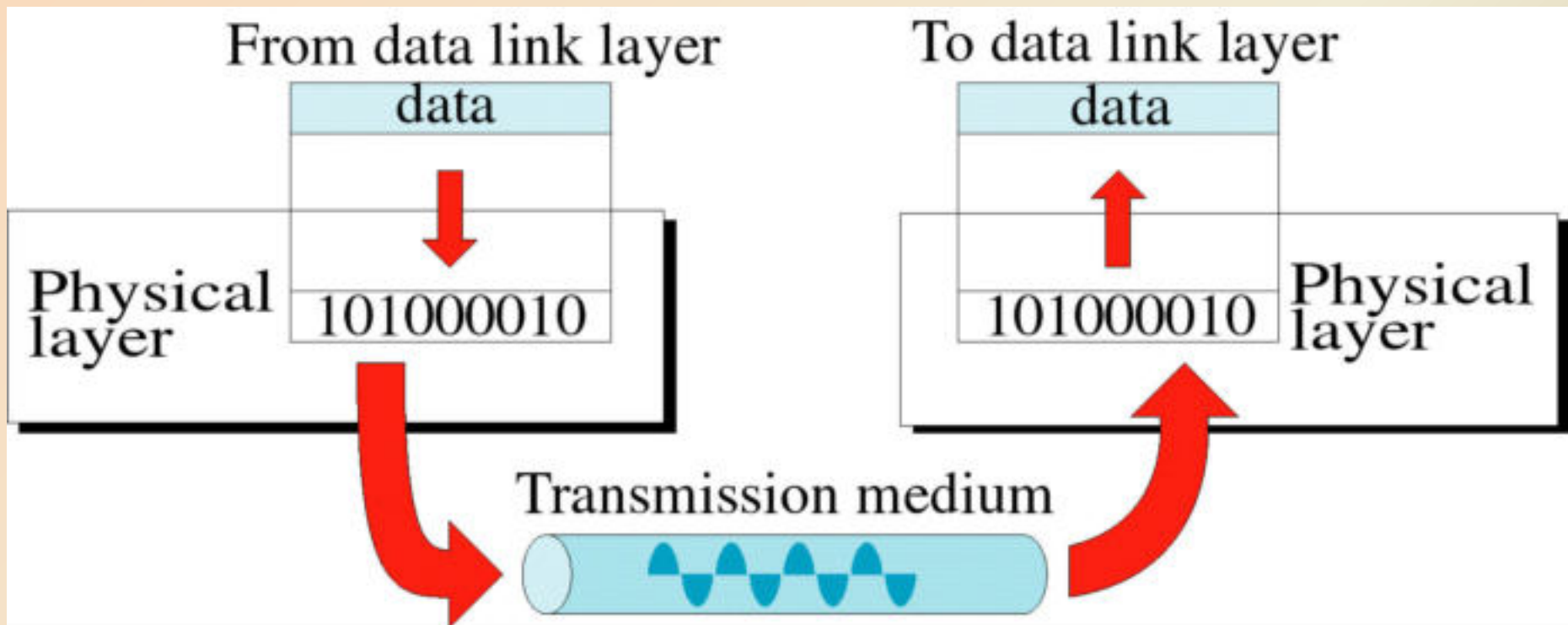
Lower Layers : Layer 1, 2, 3 are the network support layers, they deal with the physical aspects of moving data from one device to another.

Middle Layer : Layer 4 ensure end-to-end reliable data transmission

Upper Layers : Layer 5, 6, 7 are working for user support. They allow interoperability among unrelated software system.

Physical Layer

It is the first layer of OSI Model. It coordinates the functions required to transmit a bit stream over the physical medium.



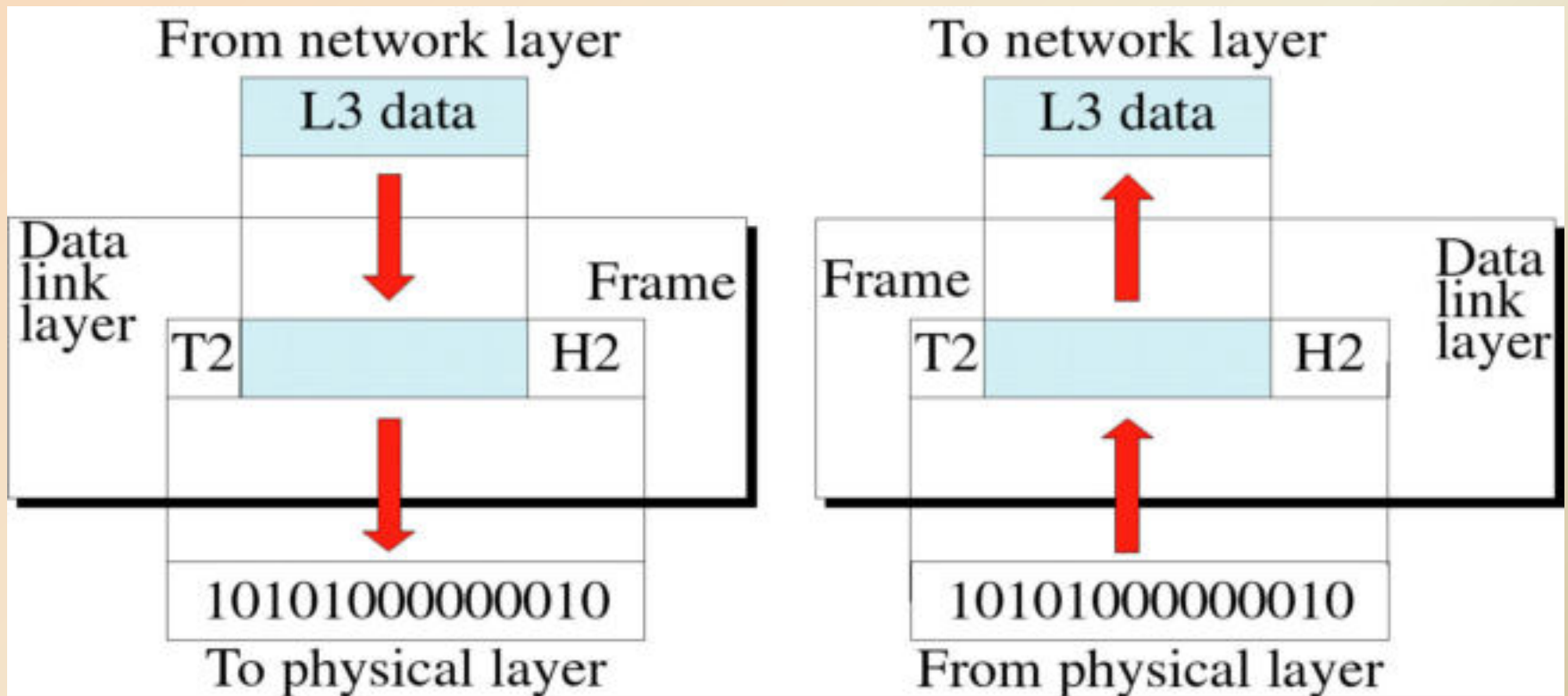
Physical Layer

Functions

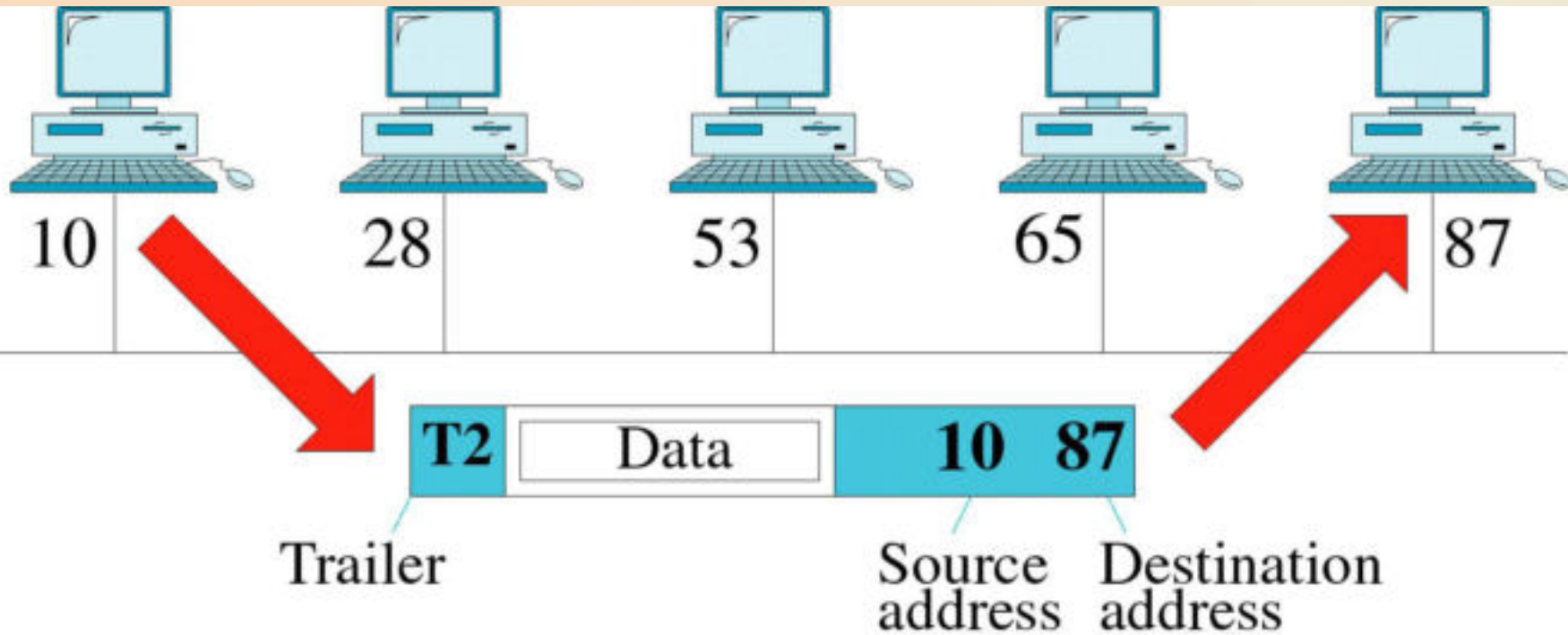
- Physical characteristics of interfaces and media
- Representation of Bits
- Transmission Rate
- Line Configuration
- Protocol
- Transmission Media

Data Link Layer

It is the Second layer of OSI Model. It is responsible for Node-to-Node delivery.



Data Link Layer



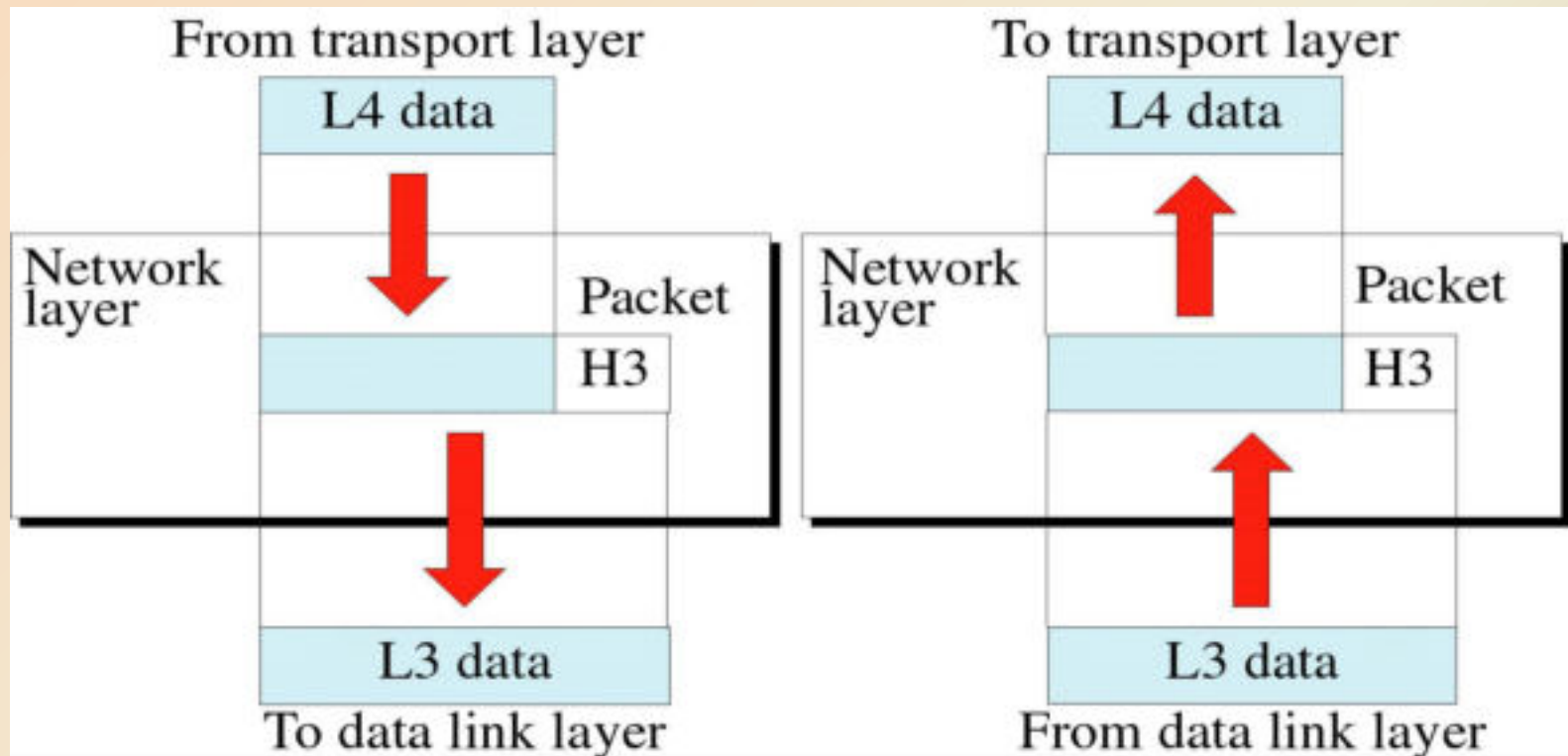
Data Link Layer

Functions

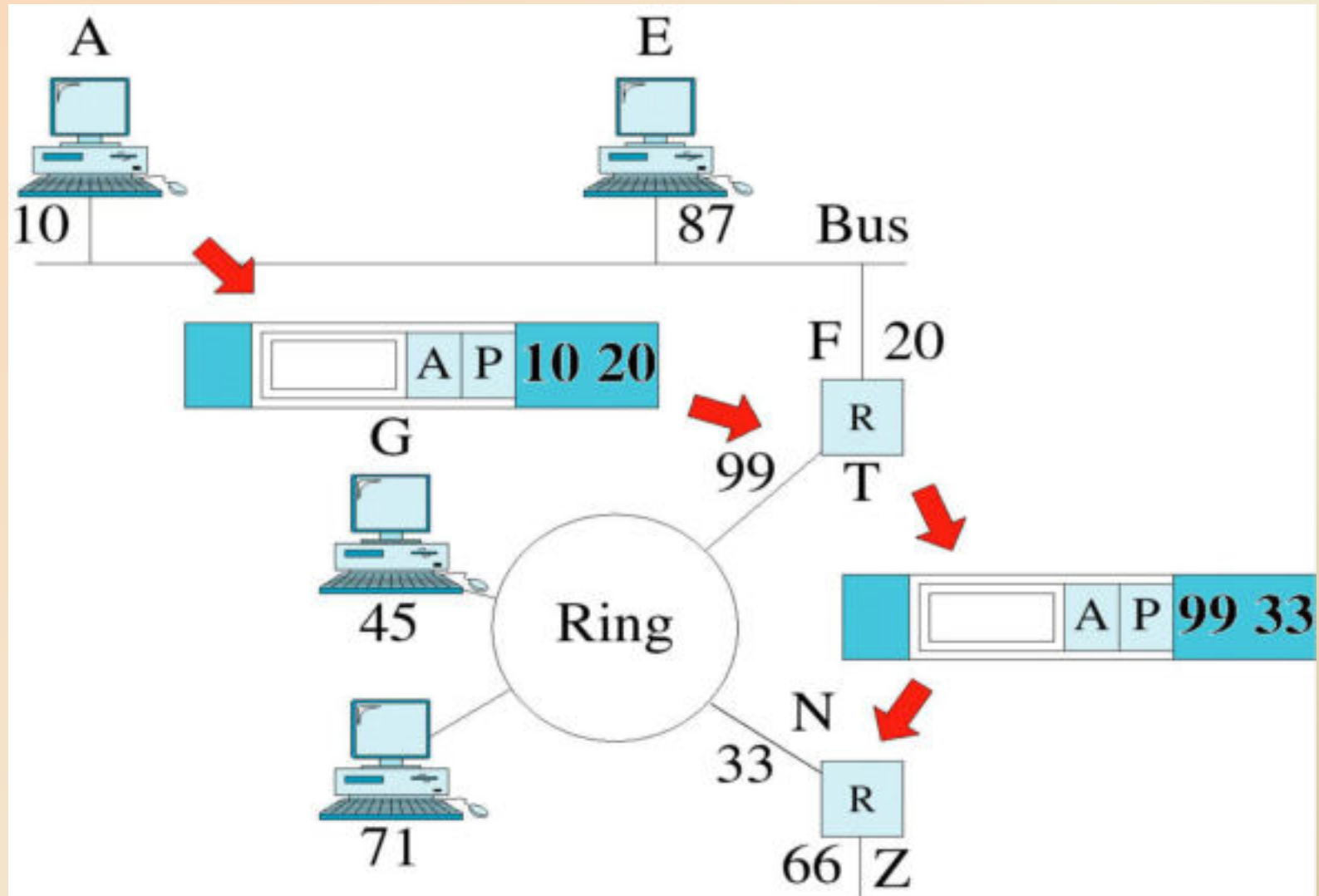
- ❑ **Framing** – Blocking of data into manageable data unit
- ❑ **Physical Addressing** - adding source and destination address
- ❑ **Flow Control** – to avoid overwhelming at receiver end
- ❑ **Error Control** – retransmission of lost data
- ❑ **Access Control** – unique identification of network devices

Network Layer

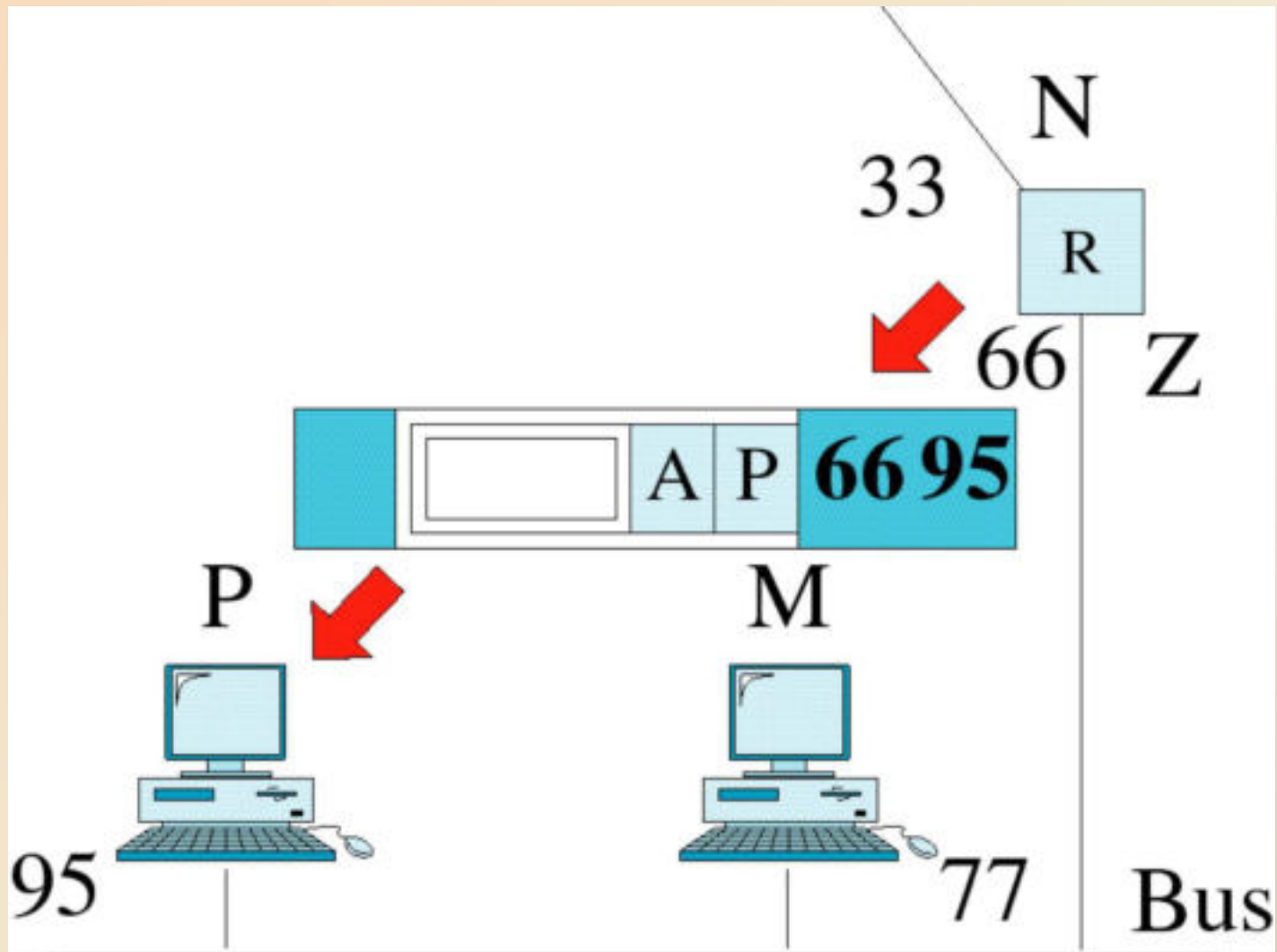
It is the Third layer of OSI Model. It is responsible for source to destination delivery of a packet possibly across multiple networks. It is mainly required to communicate between two different networks



Network Layer



Network Layer



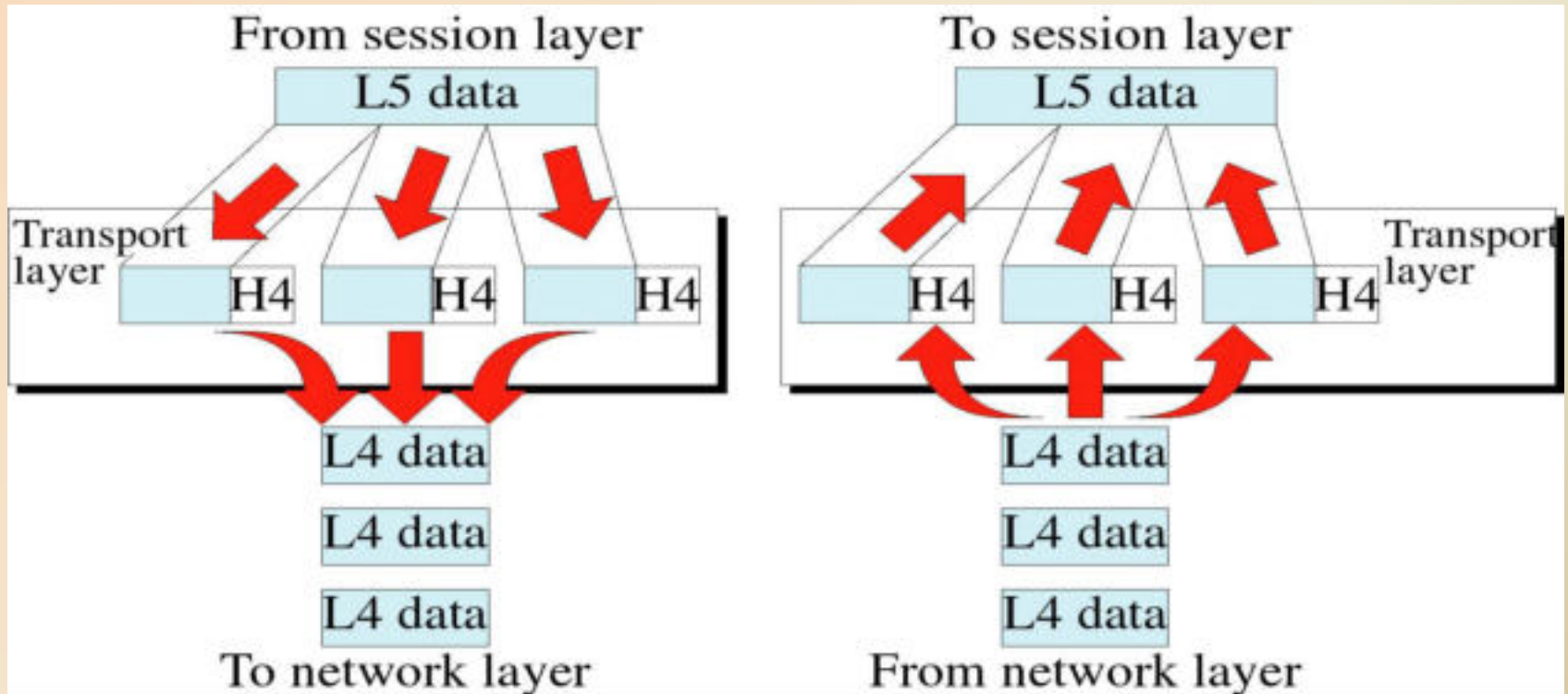
Network Layer

Functions

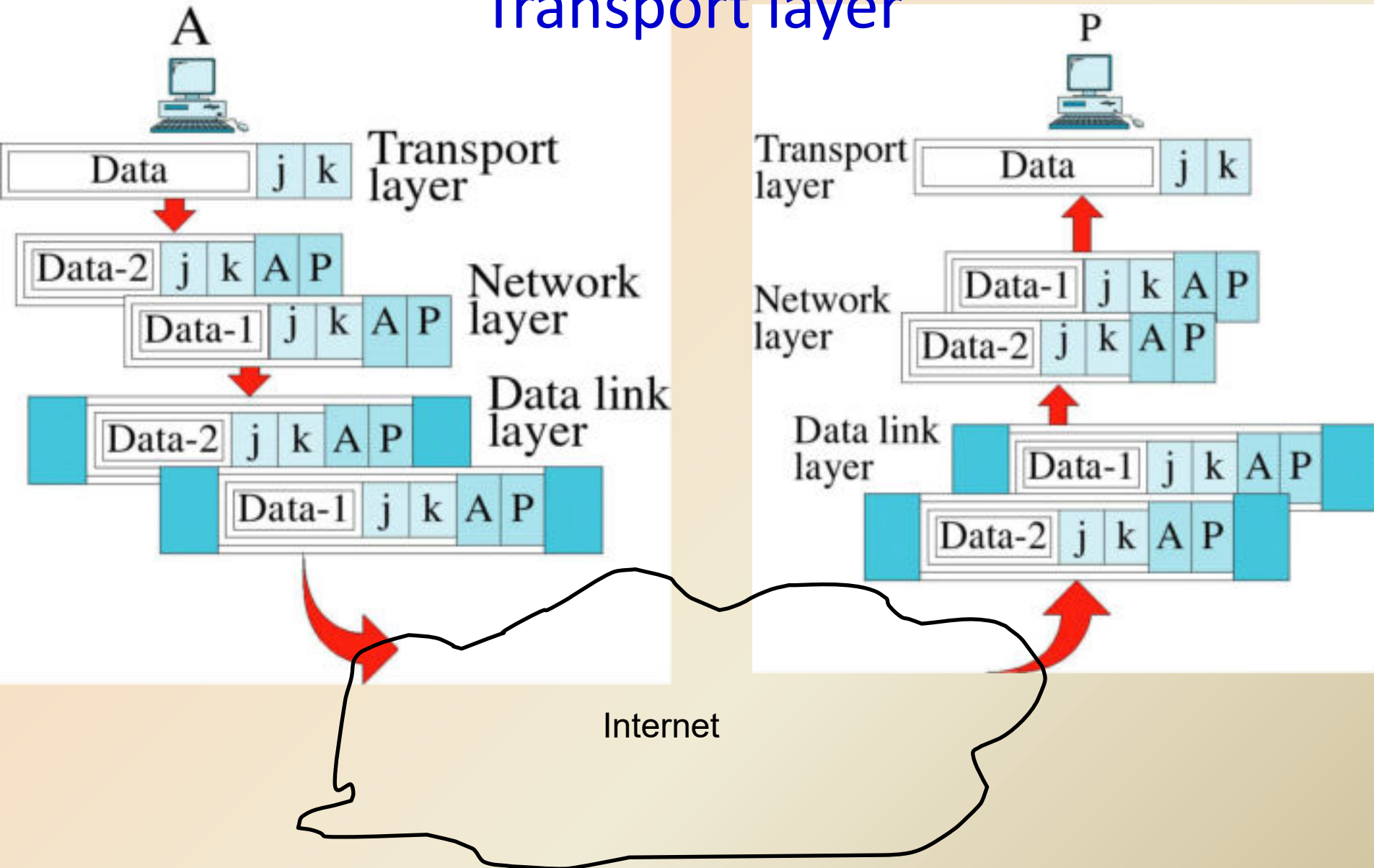
- ❑ **Logical Addressing** – Needed, if packet passes the network boundary
- ❑ **Routing** – It defines routing path within internetwork

Transport Layer

It is the Forth layer of OSI Model. It is responsible for source-to-destination delivery of the entire message. It ensures that the whole message arrives intact and in order.



Transport layer



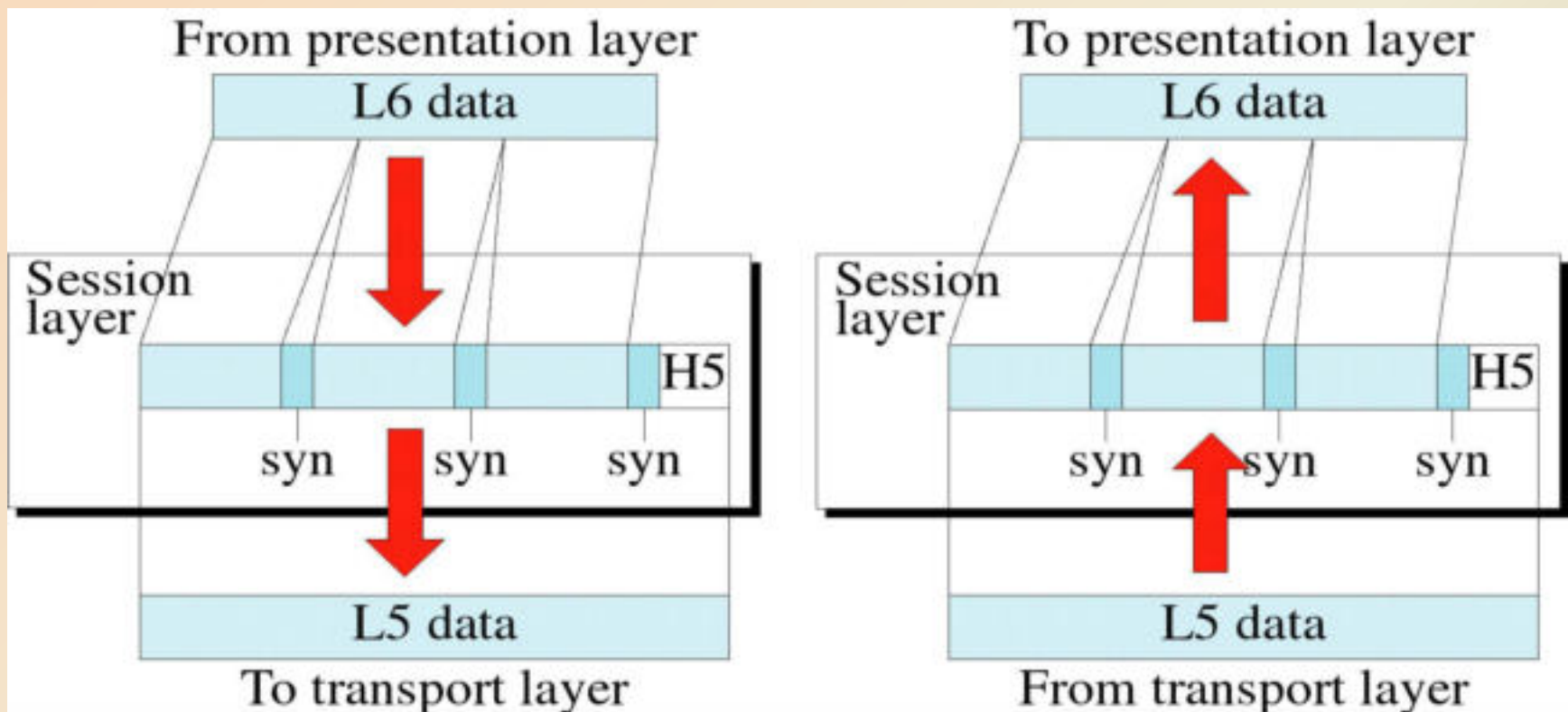
Transport Layer

Functions

- ❑ **Service-point addressing** – message sending to Port Address
- ❑ **Segmentation & Reassembling** – Divide into transmittable segments
- ❑ **Connection Control** – connection oriented / connectionless protocol

Session Layer

It is the Fifth layer of OSI Model. It establishes, maintains and synchronizes the interaction between communicating systems.



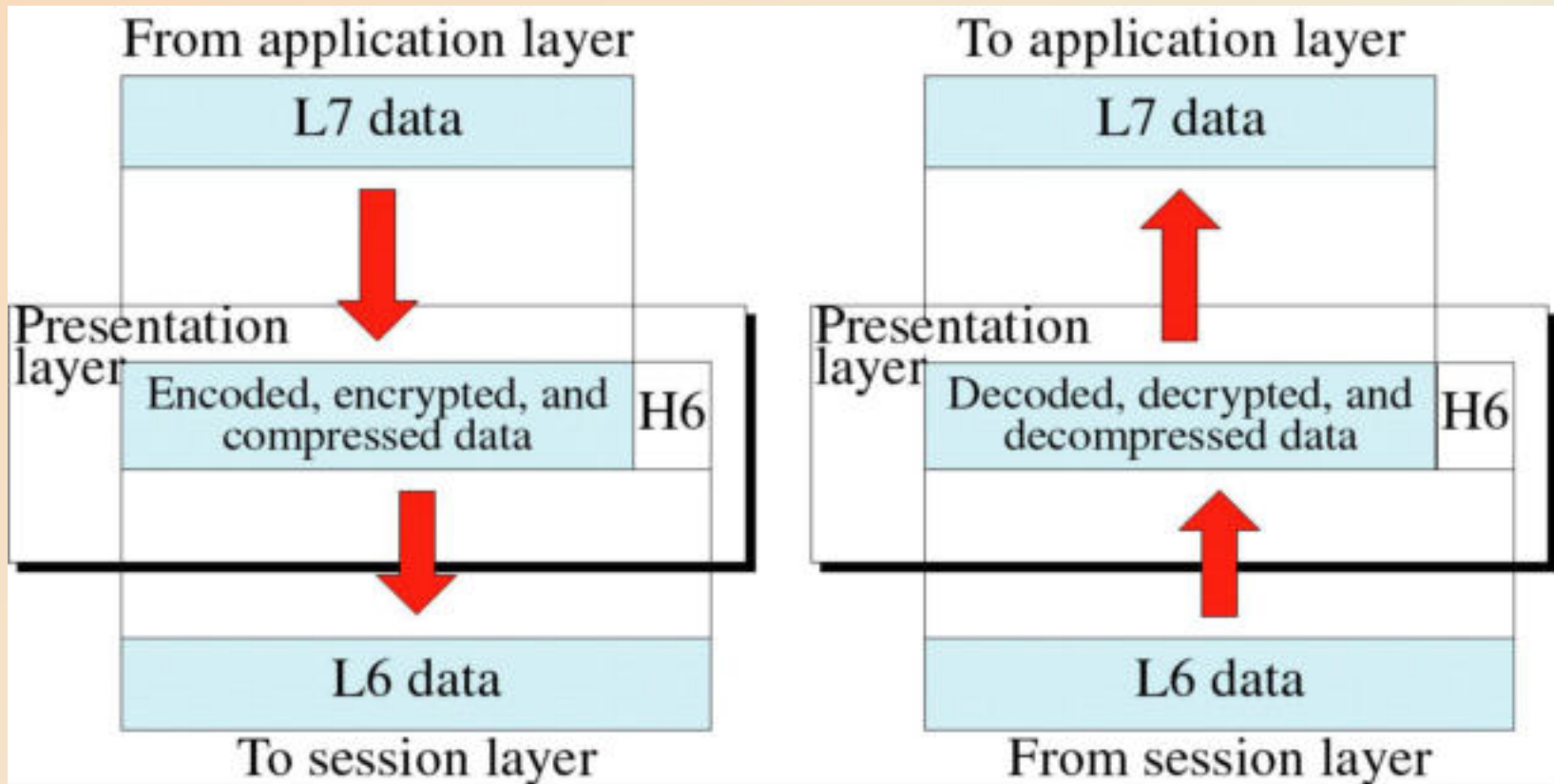
Session Layer

Functions

- ❑ **Dialog Controller** – It allows systems to enter into a dialog. It defines Half / Full duplex communication
- ❑ **Synchronization** – Adds checkpoints into a stream of data.

Presentation Layer

It is the Sixth layer of OSI Model. It is concerned with syntax and semantics of information exchanged between two systems



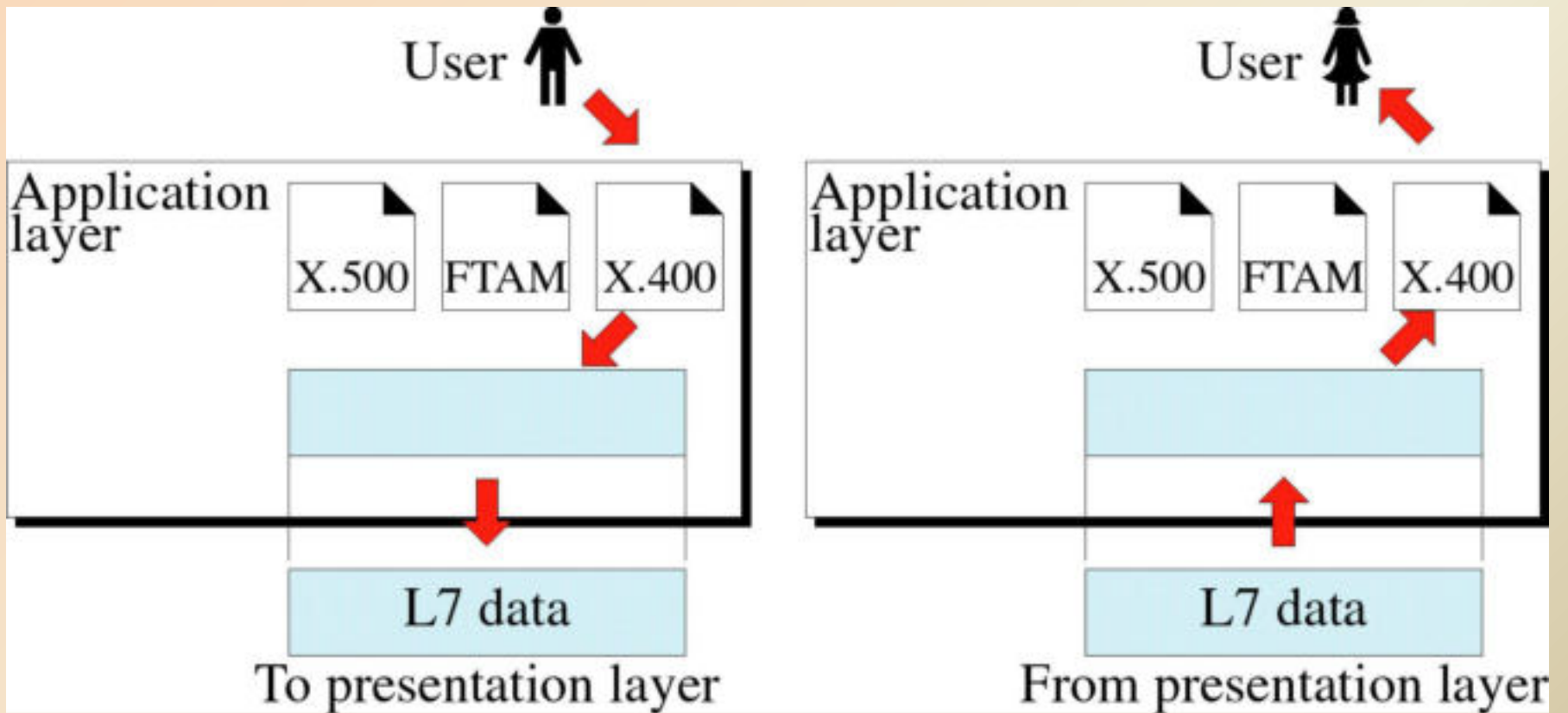
Presentation Layer

Functions

- ❑ **Translation** – Changes information into bit stream
- ❑ **Encryption / Decryption** – Encoded form for security
- ❑ **Compression** – To reduce the number of bits to be transmitted

Application Layer

It is the Seventh layer of OSI Model. It enables user to access the network.

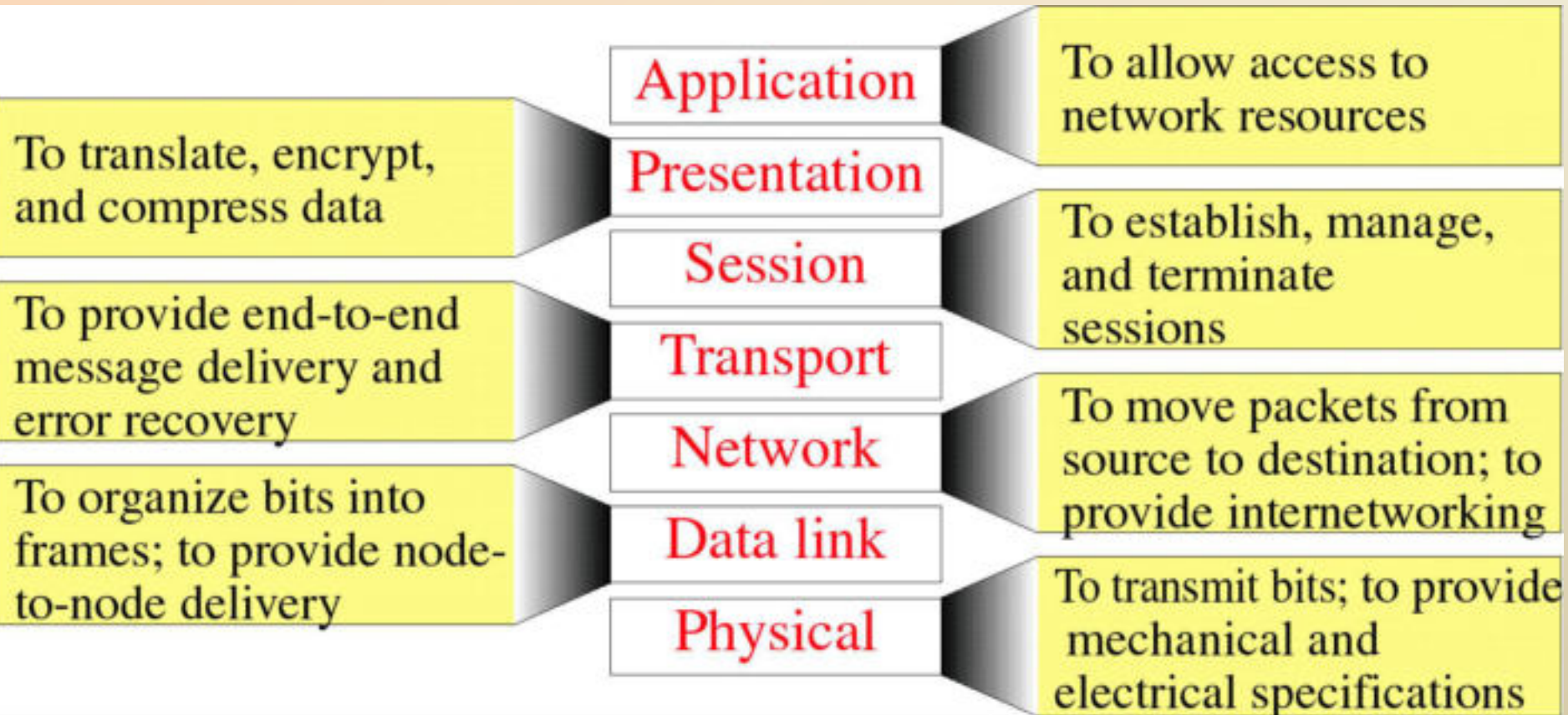


Application Layer

Functions

- Network Virtual Terminal** – Allow user to log on to a remote host
- FTAM** – File Transfer, Access and Management
- Mail Service** – E Mail Service
- Directory Service** – Global information access

Summary of Layers Functions



Encapsulation

- Encapsulation is a technique of implementing layered architecture of a communication system. In OSI model we have separated all the communication functions/services into seven layers. We know that a layer serves the layer above it and is served by the layer below it, so to make it possible encapsulation techniques is followed for sending/receiving data between and through layers. In encapsulation we add some control information or “Header/Trailer” to a Data Unit by a communications protocol. This data along with header/trailer is known as Protocol Data Unit (PDU). This header/trailer actually creates an envelope for the PDU which has its address and addressee.

Unit-2

MODULATION AND ENCODING

Introduction

Objectives

Need for Modulation

Modulation

Amplitude Modulation

Frequency Modulation

Phase Modulation

Digital Communication

Digital Modulation Techniques

Amplitude Shift Keying (ASK)

Frequency Shift Keying

Phase Shift Keying

Introduction

Electronic communication has become an analogy for the communication in the present era of electronic gadgets. As a general concept, we can say that transfer of information from one place to another is communication. A significant point about communication is that it involves a sender (transmitter) and a receiver. Only a sender or a receiver can not complete the process of communication. Therefore dual process of “transmitting and receiving” or “coding and decoding” information can be called as communication making it a two way process. In this unit we will discuss about different modulation and encoding techniques. In this unit both analog and digital modulation will be discussed. Further, this unit we will explore how analog signal are converted into digital system and vice-versa.

OBJECTIVES

After going through this unit, you should be able to:

- Know the concept of modulation
- Understand the different Analog Modulation techniques
- Differentiate between analog and digital modulation
- Know process of analog to digital signal conversion
- Understand the sampling and quantization process
- Know the digital to analog signal conversion process
- Understand the Digital Modulation techniques

MODULATION

Often, the message being communicated is itself a signal, e.g., an audio signal, and to produce a signal that is suitable for transmission through the channel, we effect some transformation on the message signal. Modulation is the Process by which a property or a parameter of a signal is varied in proportion to another signal. The original signal is normally referred as the modulating signal and the high frequency signal, whose properties are changed, is referred as the carrier signal. The resulting signal is finally referred as the modulated signal. For example in case of the amplitude modulation, the amplitude of the carrier wave is varied in accordance with the amplitude of the message signal, whereas in the angle modulation, phase angle of the carrier is varied with respect to the message signal

- .

Benefits of Modulation

1. Modulation can shift the frequency spectrum of a message signal into a band which is better suited to the channel. Antennas only efficiently radiate and admit signals, whose wavelength is similar to their physical aperture. Hence, to transmit and receive, say, voice, by radio we need to shift the voice signal to a much higher frequency band.
1. 2. Modulation permits the use of multiplexing. Multiplexing means allowing simultaneous communication by multiple users on the same channel. For instance, the radio frequency spectrum must be shared and modulation allows users to separate themselves into bands. 3. Modulation can provide some control over noise and interference. For example the effect of noise can be controlled to a large extent by frequency modulation.

AMPLITUDE MODULATION

Amplitude modulation (AM) is a technique used in electronic communication, most commonly for transmitting information via a high frequency carrier wave. AM works by varying the strength of the transmitted signal in relation to the information being sent. For example, changes in signal strength may be used to specify the sounds to be reproduced by a loud speaker, or the light intensity of television pixels. "Undulatory currents" are the initial implementations of the Amplitude modulation. These were the first method to successfully produce quality audio over telephone lines in 1870's

FREQUENCY MODULATION

Frequency modulation, FM is widely used for a variety of radio communications applications. FM broadcasts on the VHF bands still provide exceptionally high quality audio, and FM is also used for a variety of forms of two way radio communications, and it is especially useful for mobile radio communications, being used in taxis, and many other forms of vehicle. in view of its widespread use, frequency modulation, FM, is an important form of modulation, despite many forms of digital transmission being used these days. Since its first introduction the use of frequency modulation, FM has grown enormously. Now wideband FM is still regarded as a very high quality transmission medium for high quality broadcasting. FM, is also widely used for communications where it is resilient to variations in signal strength.

Table 1: Comparison of AM and FM

S. No.	AM Broadcasting	FM Broadcasting
1.	It requires smaller transmission bandwidth	It requires larger bandwidth.
2.	It can be operated in low, medium and high frequency bands.	It needs to be operated in very high and high frequency bands.
3.	It has wider coverage.	Its range is restricted to 50 km.
4.	The demodulation is simple.	The process of demodulation is complex.
5.	The stereophonic transmission is not possible.	In this, stereophonic transmission is possible.
6.	The system has poor noise performance.	It has an improved noise performance.
7.	The AM signal reception does not have any threshold in the useful range of signal noise ratio (SNR).	The FM signal reception exhibits a threshold effect. The useful range of signal noise ratio (SNR) value should be higher than the threshold value.

PHASE MODULATION

Frequency Modulation and the Phase Modulation are the two forms of the angle modulation. The main characteristic of the angle modulation is that the amplitude of the carrier frequency is maintained constant, whereas the frequency or phase is changed. In the phase modulation, the phase of the carrier wave is shifted in accordance with the amplitude of the modulating frequency. Phase modulation is a form of modulation that can be used for radio signals used for a variety of radio communications applications. As it will be seen later that phase modulation and frequency modulation are closely linked together and it is often used in many transmitters and receivers used for a variety of radio communication applications from two way radio communications links, mobile radio communications and even maritime mobile radio communications. Phase modulation is also the basis for many forms of digital modulation based around phase shift keying, PSK which is a form of phase modulation. As various forms of phase shift keying are the favored form of modulation for digital or data transmissions, this makes phase modulation particularly important.

DIGITAL COMMUNICATION

Digital communication is the process of communication in which, the signals are transferred in the form of discrete formats rather than the continuous analog forms. Digital communication is very common in the present day communication systems and the signals are normally transmitted in binary formats. It is always easy to process the digital information as compared to the analog signals, because of their discrete nature and hence they have become more popular in the electronic communication. However, the voice based communication is Analog in nature, the signals needs to be converted into the digital formats to process in through the digital communication systems. The opposite process happens, while reconstructing the voice signal at the receiver end. A device called Modem (Modulator + demodulator) in the process. A modem (modulator-demodulator) is a device that modulates an analog carrier signal to encode digital information, and also demodulates such a carrier signal to decode the transmitted information. The goal is to produce a signal that can be transmitted easily and decoded to reproduce the original digital data

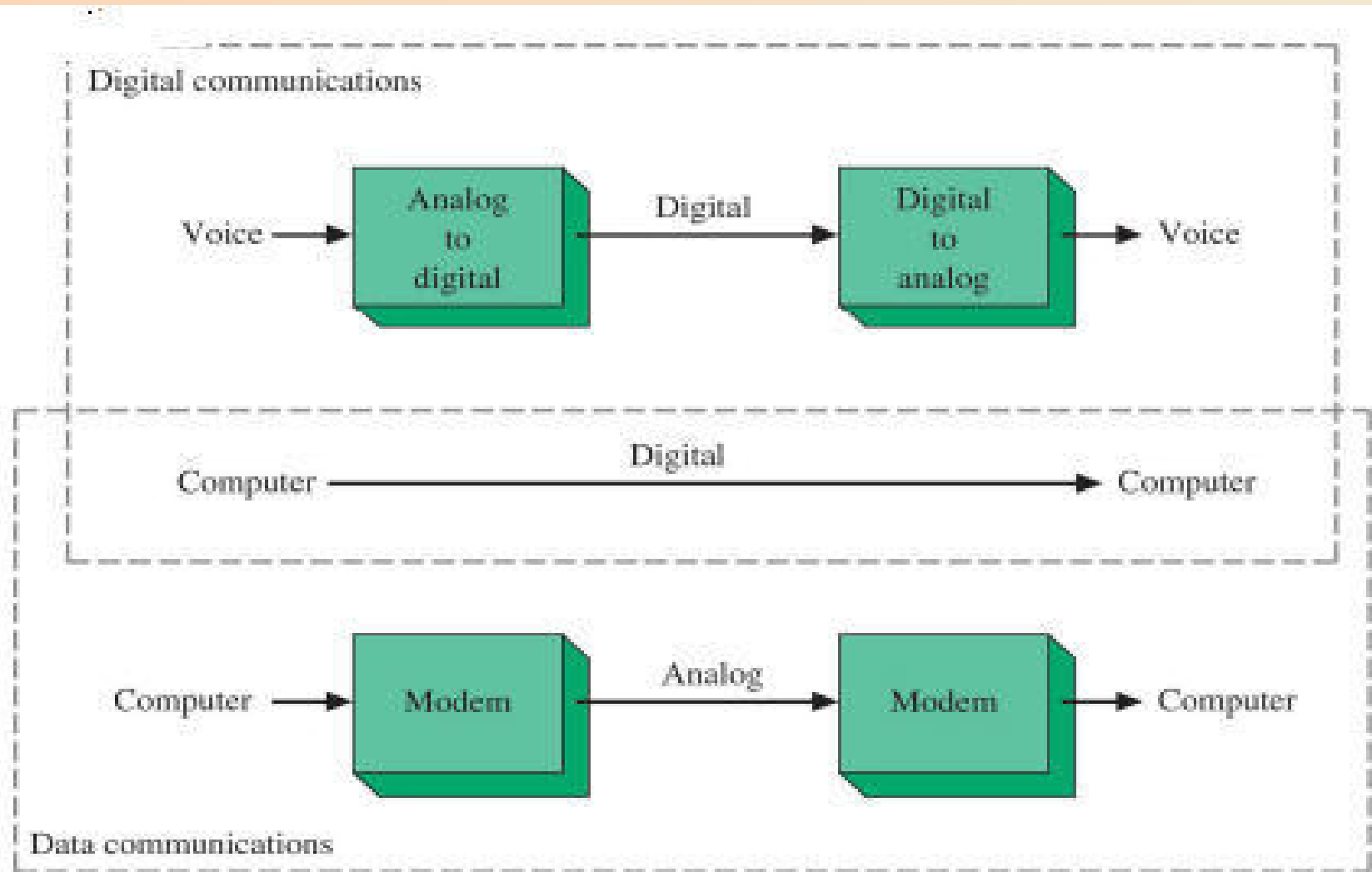


Figure 7: Digital Communication System

Advantages of Digital Communication

1. Reliable communication
2. Less sensitivity to changes in environmental conditions (temperature, etc.)
3. Easy multiplexing
4. Easy signaling
5. Voice and data integration
6. Easy processing like encryption and compression

Disadvantages

1. Increased bandwidth requirement for the communication channels.
2. Need for precision timings (Bit, character, frame synchronization needed)
3. Need for the Analog to Digital and Digital to Analog conversions
4. Higher complexity of the system design

THANK YOU

Fundamental of Computer Network

Gyanendra Shukla

Block -1

Concepts of Communication & Network

Unit-3

Multiplexing & Switching

MULTIPLEXING AND SWITCHING

- Introduction
- Objectives
- Multiplexing concept
- Frequency-Division Multiplexing
- Time-Division Multiplexing
- Code Division Multiplexing
- Space Division Multiplexing
- Switching
- Message Switching
- Circuit Switching
- Packet Switching

Introduction

- The most fundamental need of any communication system design is to cater to large number of users. But this requires a large number of resources and large bandwidths supporting multiple channels.
- Requirement for large number of resources can be met if the resources are available, but this makes it cost ineffective.
- Therefore, the aim is always to use minimum number of resources and make their utilization to their fullest potential. Bandwidth always remains a critical resource due to its limited availability and therefore, communication systems try to harness its fullest potential.
- Networks always require us to accommodate multiple signals utilizing a single piece of cabling to make it cost effective and reduce complexity.
- This need is seen throughout networking whether we are talking about local area networks or wide area ones. Modern telephone systems must place a large number of calls over a limited amount of bandwidth (i.e. a trunk). Broadband LANs must have several different types of data on a single wire at once.
- For these applications, we need to share the resources and in particular the bandwidth. Multiplexing and Switching are the two most important techniques being employed for this purpose in the present day communication systems and have been discussed in the present unit.

OBJECTIVES

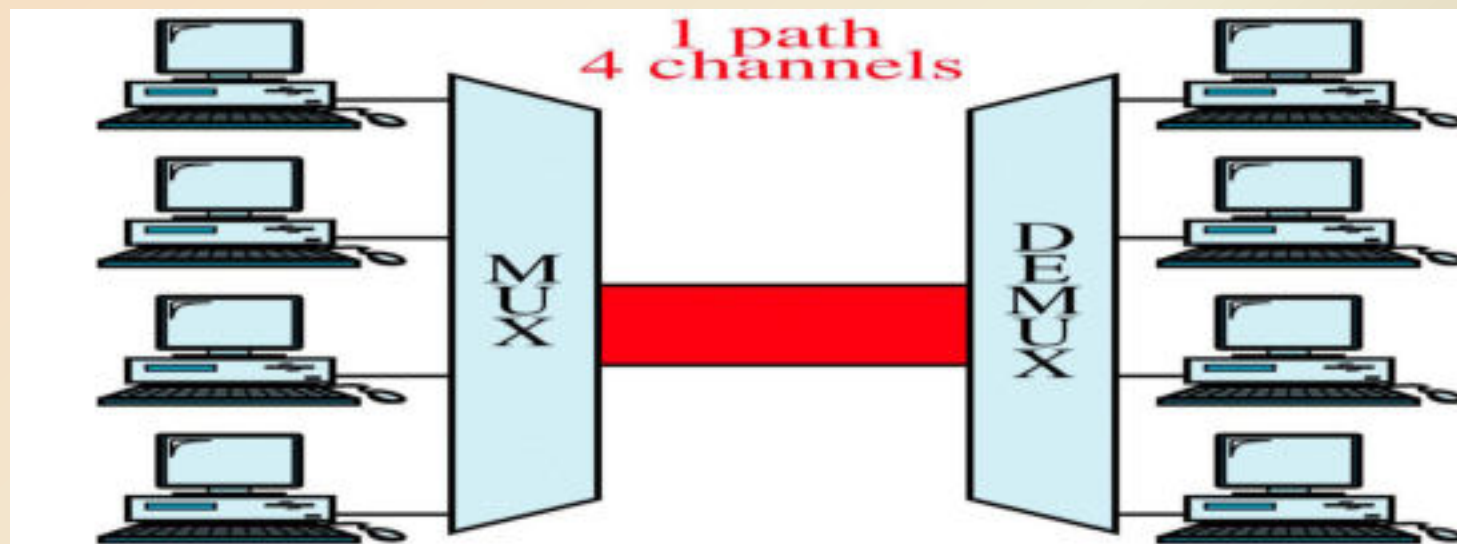
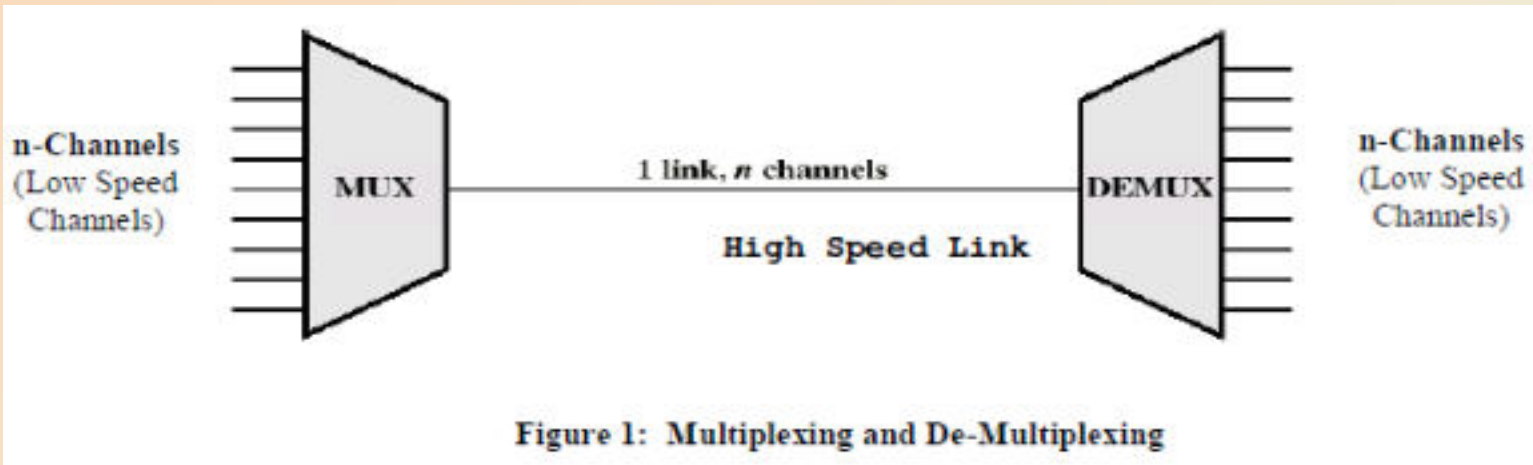
After going through this unit, you should be able to:

- Know the concept of Multiplexing and Switching in computer networks
- Understand the basic multiplexing techniques like FDM, TDM, CDM and SDM
- Differentiate between different types of multiplexing techniques
- Know the switching mechanisms
- Differentiate between packet, circuit and message switching
- Understand the different packet switching mechanisms

Multiplexing Concept

- In general, a medium can carry only one signal at any moment in time.
- For multiple signals to share one medium, the medium must somehow be divided, giving each signal a portion of the total bandwidth. Multiplexing (also known as MUXing) is a method by which multiple analog message signals or digital data streams are combined into one signal over a shared medium.
- The basic aim of the Multiplexing is to share an expensive resource by putting-up multiple signals on the same channel.
- For example, in telecommunications, several telephone calls may be carried using one wire
- Multiplexing originated in telegraphy in the 1870s, and is now widely applied in different streams of communications.
- When several communication channels are needed between the same two points, significant economies may be realized by sending all the messages on one transmission facility – **called multiplexing**.

- In Figure 1, n number of signals from the low speed channels have been combined to one high speed link using a $n:1$ multiplexer.
- Whereas the opposite process is carried out at the other end, where the signals are further separated into n number of low speed channels. This opposite process is referred as demultiplexing.



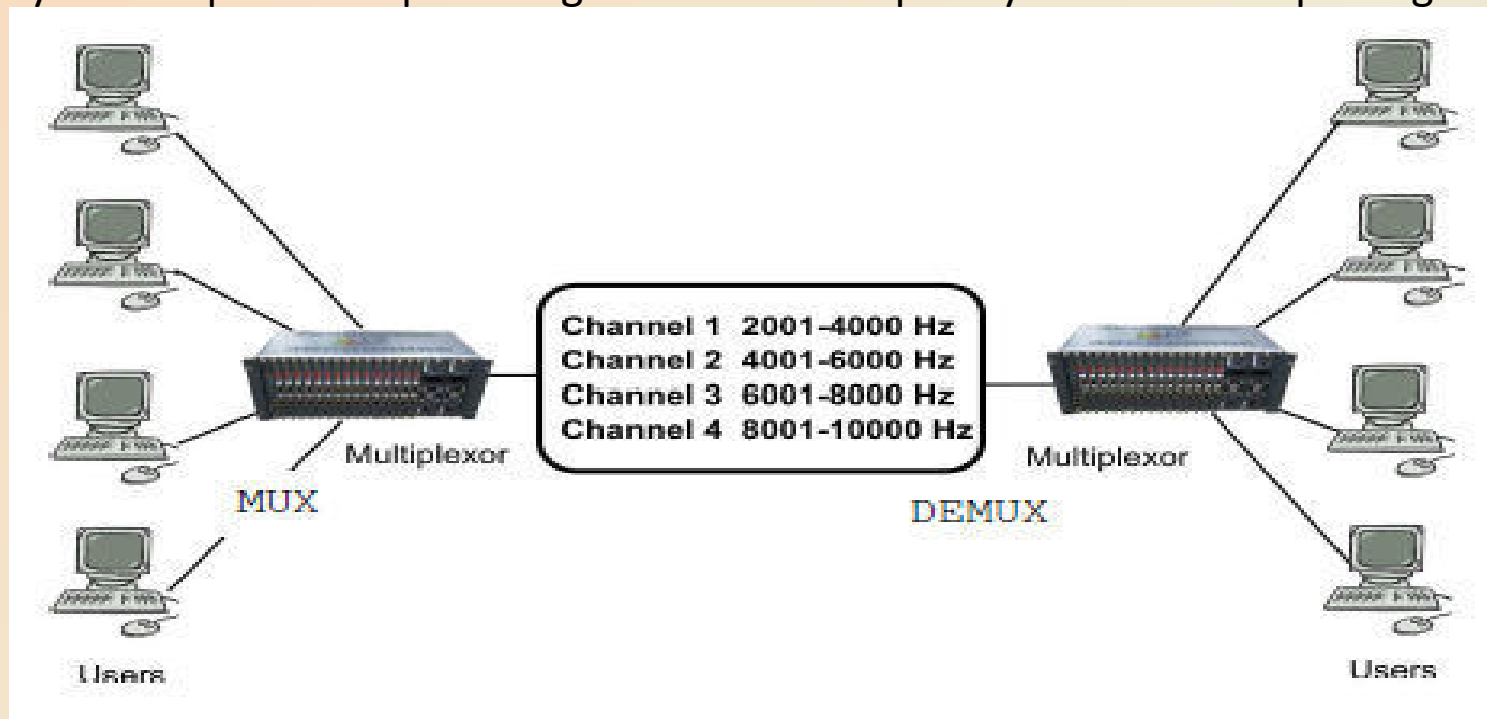
- Multiplexing refers to the ability to transmit data coming from several pairs of equipment (transmitters and receivers) called *low-speed channels on a single physical medium* (called the *high-speed channel*).
- Whereas, a multiplexer is the multiplexing device that combines the signals from the different transmitters and sends them over the high-speed channel.
- A demultiplexer is the device which separates signal received from a high-speed channel into different signal and sends them to receivers.

Four Basic Multiplexing Techniques:

- Frequency division multiplexing (FDM)
- Time division Multiplexing (TDM)
- Code division Multiplexing (CDM)
- Space-division Multiplexing (SDM)
- Frequency division Multiplexing (FDM): Bandwidth is divided into different smaller frequency bands (range).
- Time division Multiplexing (TDM): Time slots are allocated to message signals in a non overlapping manner in the time domain so that individual messages can be recovered from time synchronized switches.
- Quadrature Carrier/amplitude Multiplexing (QAM): Two message signals are transmitted in the same frequency band. The recovery is possible due to the carrier signals being orthogonal.
- Code division Multiplexing (CDM) : Users occupy the same frequency band but modulate their messages with different codes TDMA FDMA CDMA when used for multiple access TDMA, FDMA, e.g., GSM, FM, AM, Wireless networks

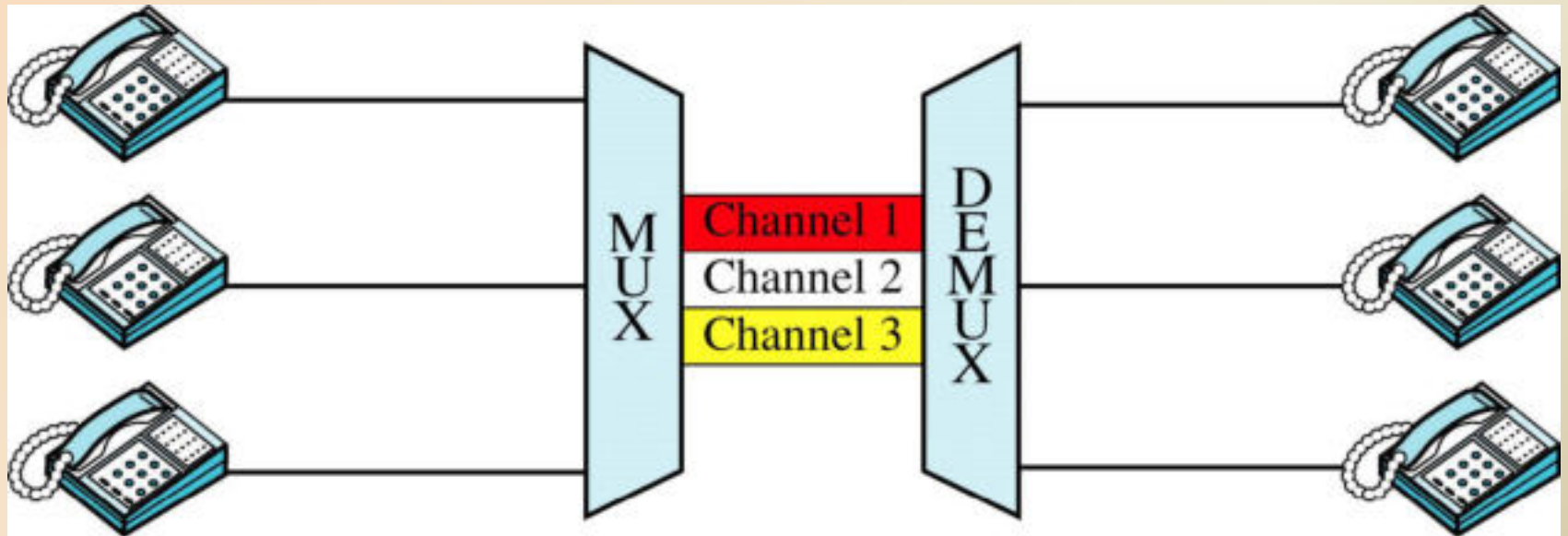
FREQUENCY-DIVISION MULTIPLEXING (FDM)

- Frequency division multiplexing (FDM) is the technique used to divide the available bandwidth into a number of smaller independent logical channels with each channel having a small bandwidth.
- **FDM** is an analog technique that can be applied when the bandwidth of a link is greater than the combined bandwidths of the signals to be transmitted.
- The method of using a number of carrier frequencies each of which is modulated by an independent speech signal is in fact frequency division multiplexing.

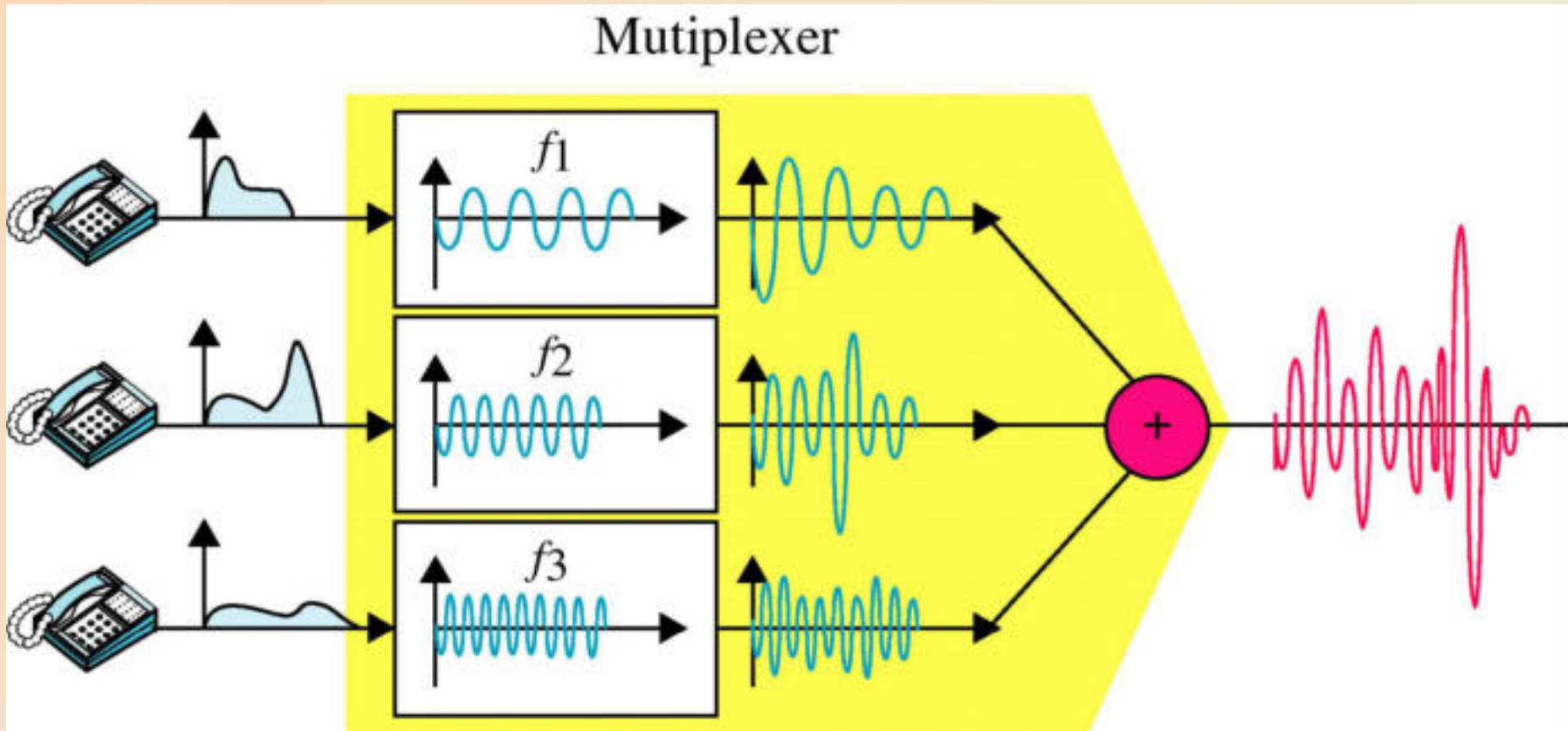


Frequency Division Multiplexing (FDM)

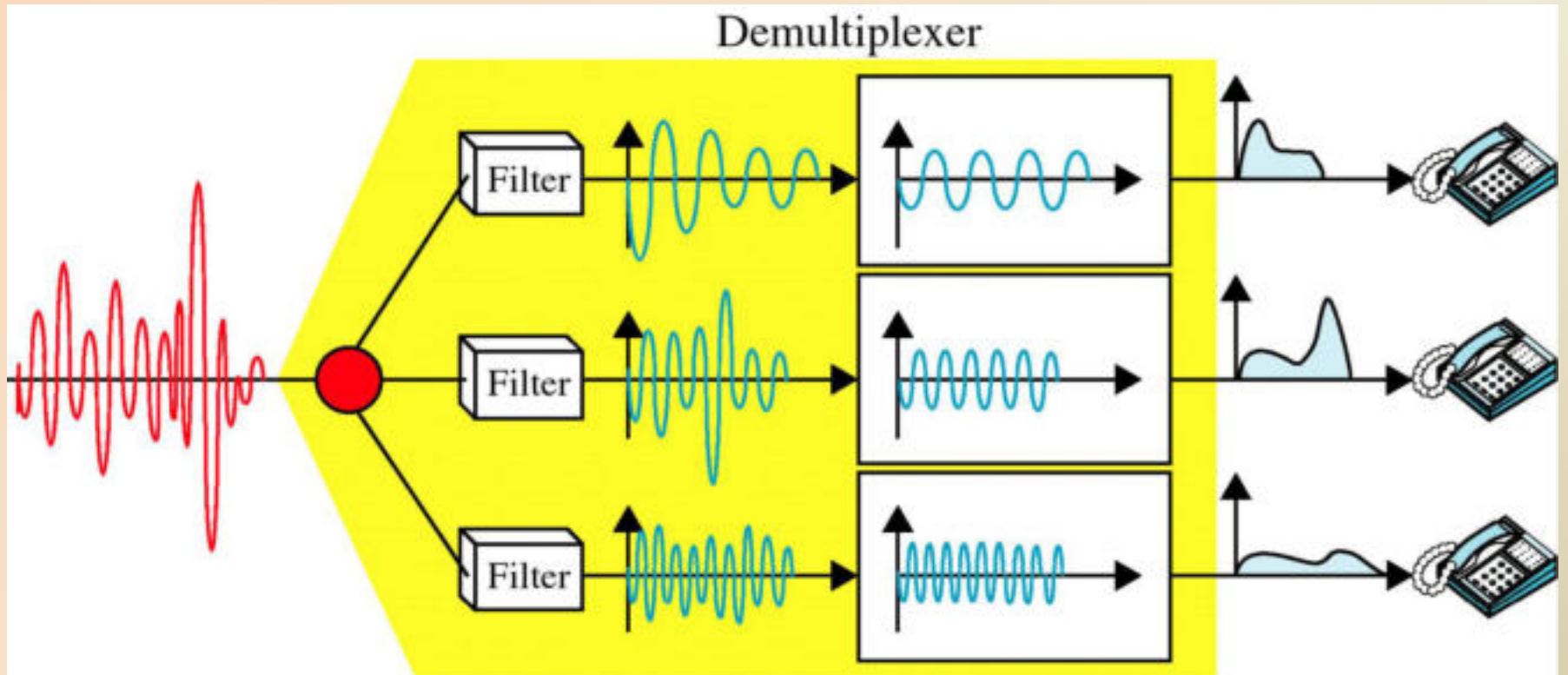
FDM is an analog technique that can be applied when the bandwidth of a link is greater than the combined bandwidths of the signals to be transmitted.



Frequency Division Multiplexing (FDM)



Frequency Division Multiplexing (FDM)



Frequency Division Multiplexing (FDM)

Advantages of FDM:

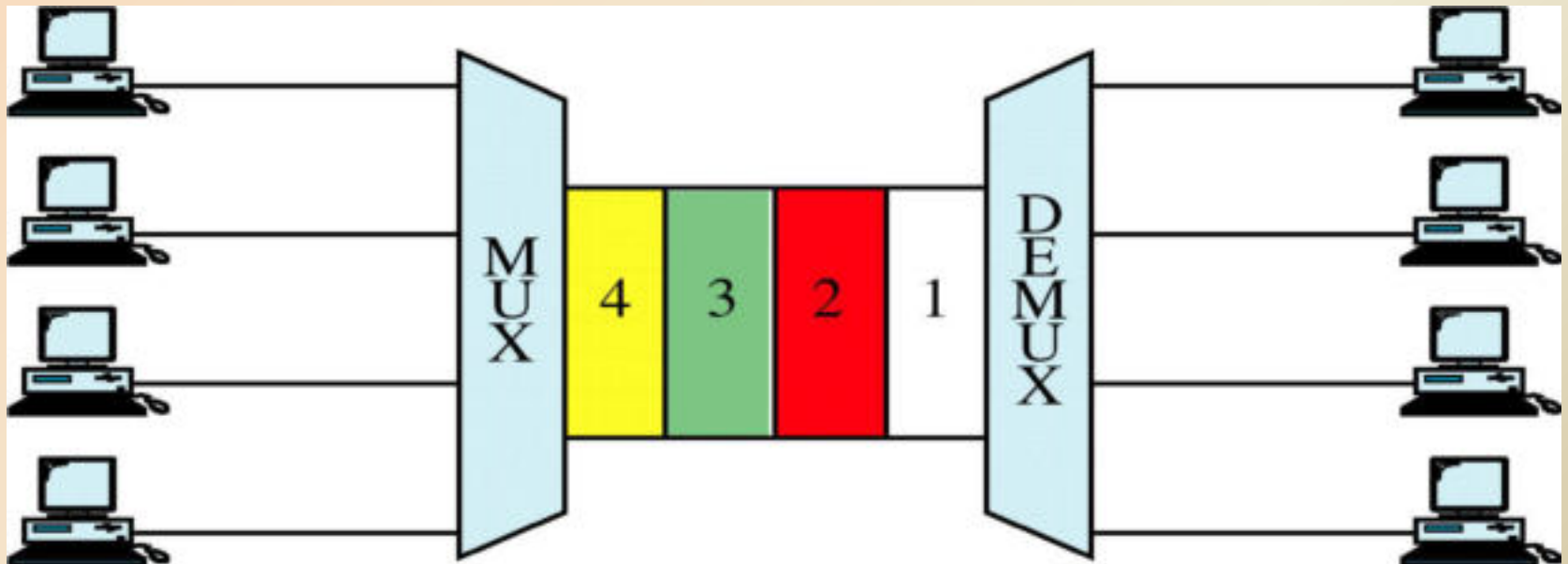
1. The users can be added to the system by simply adding another pair of transmitter modulator and receiver demodulators.
2. FDM system support full duplex information (Both side simultaneous Communication) flow which is required by most of application.

Disadvantages of FDM:

1. In FDM system, the initial cost is high. This may include the cable between the two ends and the associated connectors for the cable.
2. A problem with one user can sometimes affect the others.
3. Each user requires a precise carrier frequency for transmission of the signals.

Time Division Multiplexing (TDM)

TDM is a digital process that can be applied when the data rate capacity of the transmission medium is greater than the data rate required by the sending and receiving devices.



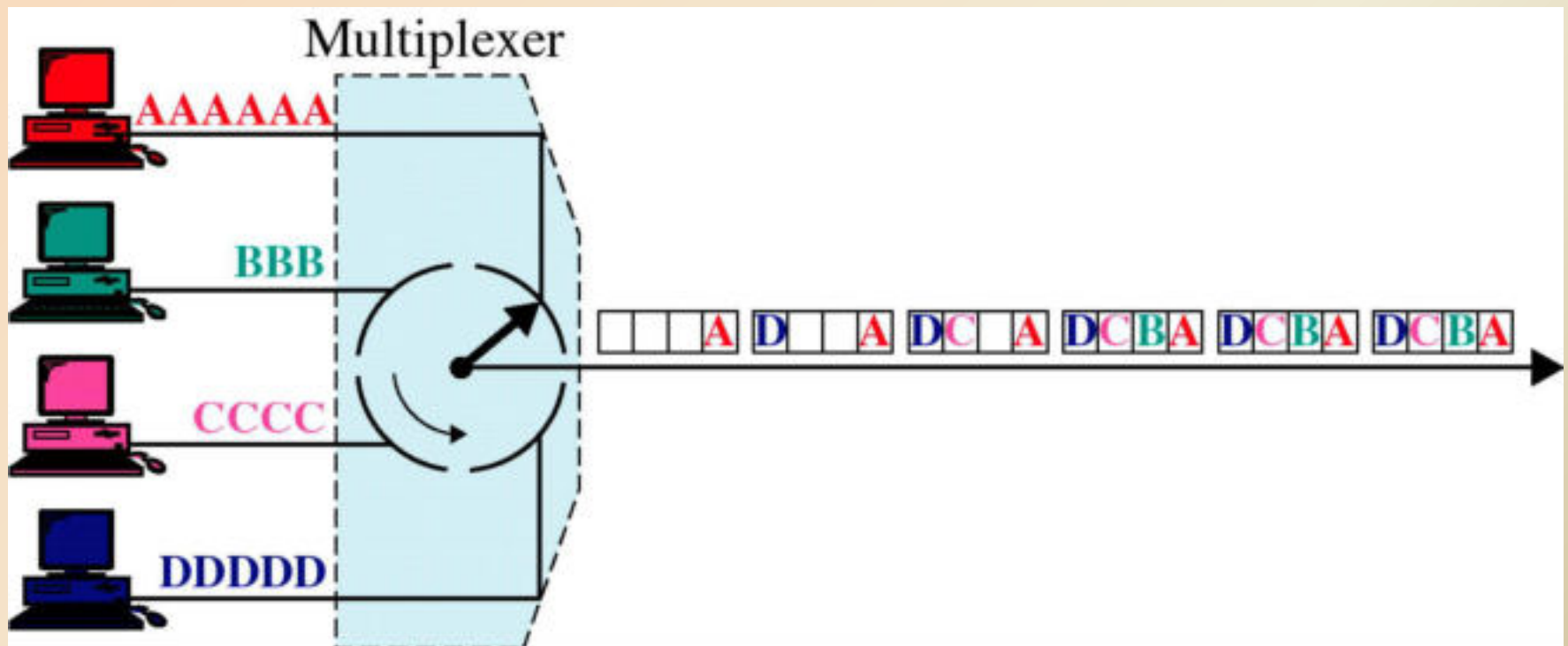
Time Division Multiplexing (TDM)

TDM can be implemented in two ways:

- a) Synchronous TDM
- b) Asynchronous TDM

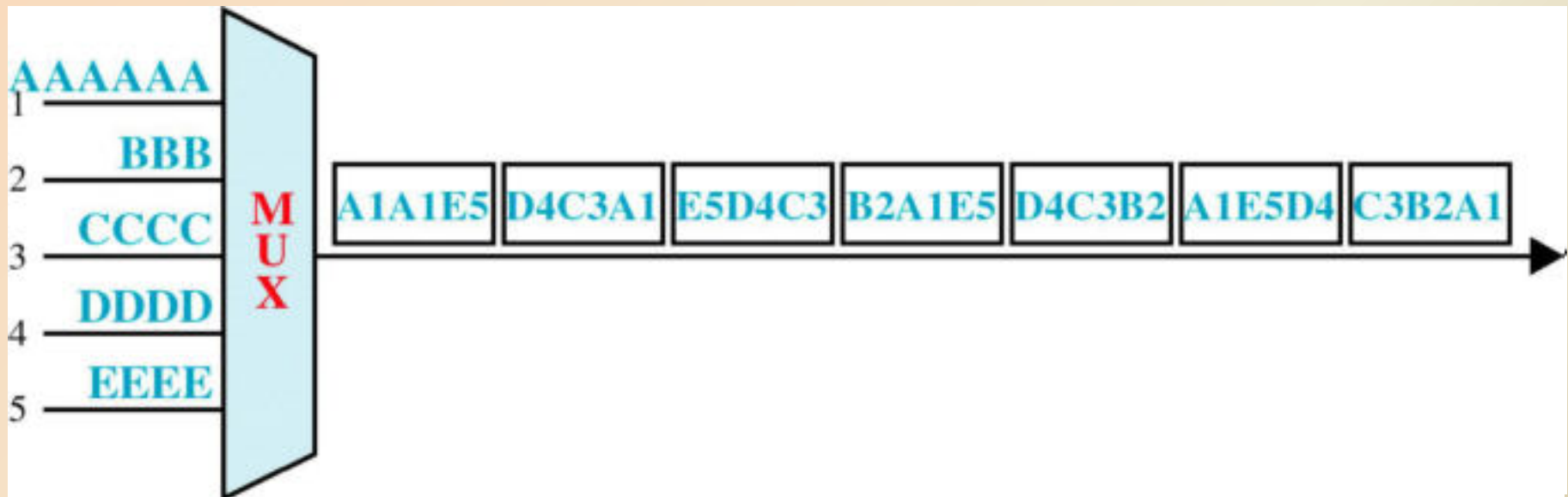
Time Division Multiplexing (TDM)

Synchronous TDM : It allocates exactly the same time slot to each device at all times, whether or not a device has anything to transmit.



Time Division Multiplexing (TDM)

Asynchronous TDM : It allocates time slot to each device, weather a device has anything to transmit. It avoids wasted time.



Advantages of TDM

1. It uses a single link
2. It does not require precise carrier matching at both end of the links.
3. Use of the channel capacity is high.
4. Each to expand the number of users on a system at a low cost.
5. There is no need to include identification of the traffic stream on each packet.

Disadvantages of TDM

1. The sensitivity to other user is very high and causes problems
2. Initial cost is high
3. Technical complexity is more

CODE DIVISION MULTIPLEXING

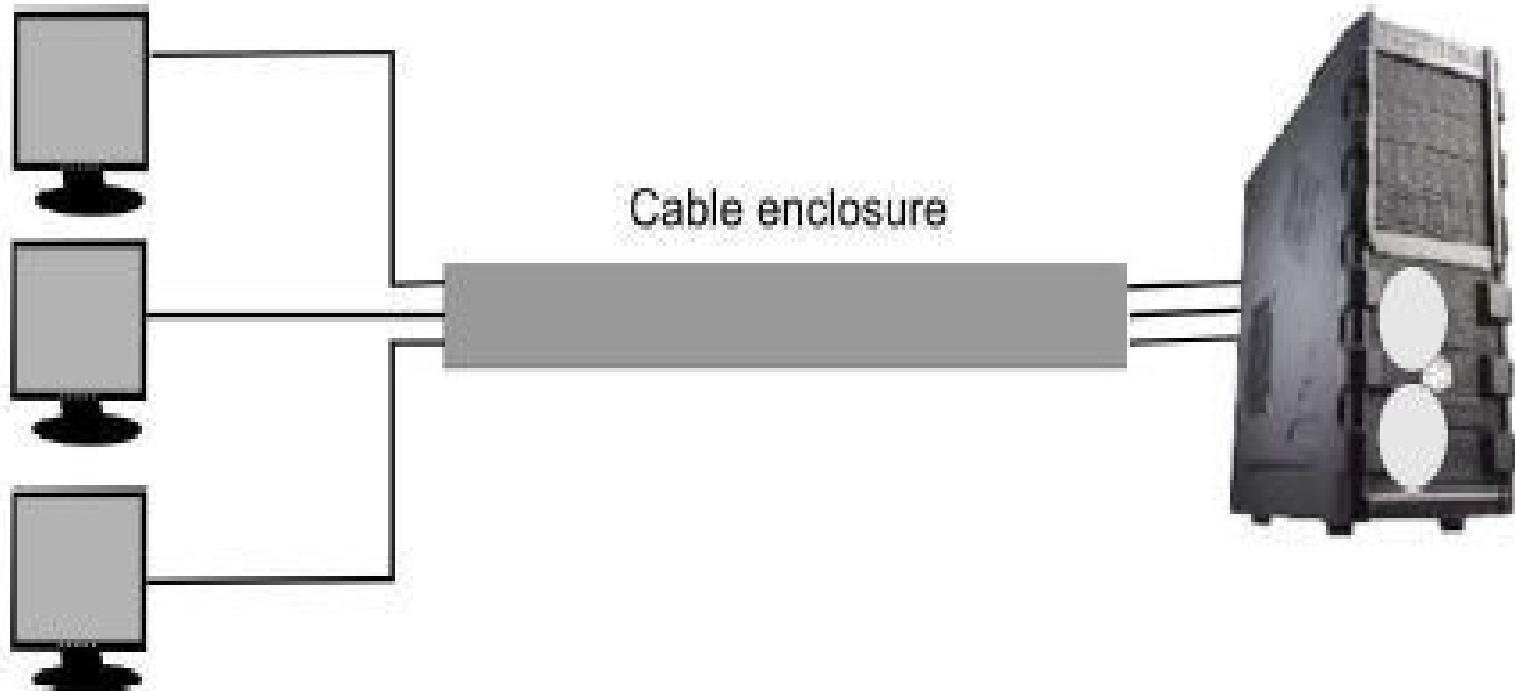
- The concept of multiple access where we can allow several transmitters to send information simultaneously over a single communication channel and it allows several users to share a band of frequencies (or you can say bandwidth).
- CDMA uses spread-spectrum technology and a special coding scheme (where each transmitter is assigned a code generally pseudorandom code) to allow multiple users to be multiplexed over the same physical channel.
- By contrast, time division multiple access (TDMA) divides access by time, while frequency-division multiple access (FDMA) divides it by frequency. CDMA is a form of spread-spectrum signaling, since the modulated coded signal has a much higher data bandwidth than the data being communicated.
- This allows more users to communicate on the same network at one time than if each user was allotted a specific frequency range. Remember that CDMA is a digital technology, so analog signals must be digitized before being transmitted on the network.

SPACE DIVISION MULTIPLEXING

When we want to transmit multiple messages through any of the communication media, the ultimate goal is to maximize the use of the given resources (e.g. time and frequency in general). It involves grouping many separate wires into a common cable enclosure. A cable that has, for example, 50 twisted pairs inside it can support 50 channels. SDM has the unique advantage of not requiring any multiplexing equipment. It is usually combined with other multiplexing techniques to better utilize the individual physical channels. For example, if there are six persons in the office and all of them want to talk at the same time, this will give rise to interference between the conversations. To reduce the interference they may divide themselves into three groups of two, such that the conversation is between each pair of people. If the pairs continue talking whilst sitting next to each other, the interference would still be present. The best way for each pair to converse with minimal interference would be to sit a few feet away from the other pairs (within the same room) and converse. They would still be sharing the same medium for their conversations but the physical space in the room would be divided for each conversation. This is the simplest example of Space Division Multiplexing.

SPACE DIVISION MULTIPLEXING

Space Division Multiplexing is the multiplexing technique in which both the time and frequency can be reused by transmitting our information through a parallel set of channels.

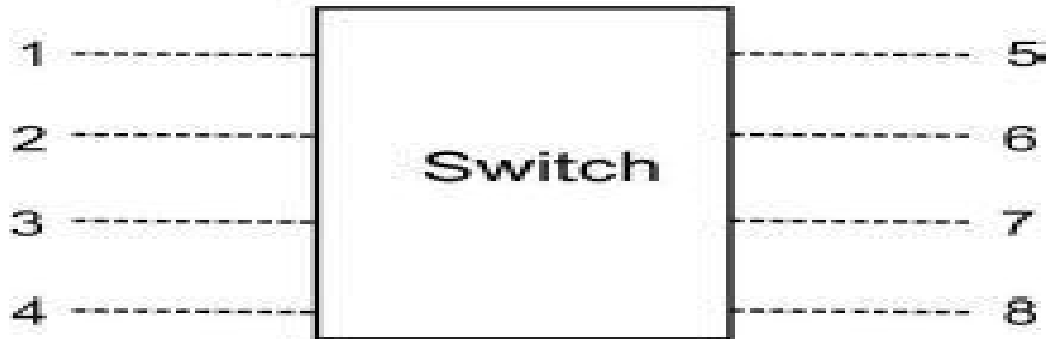


SWITCHING

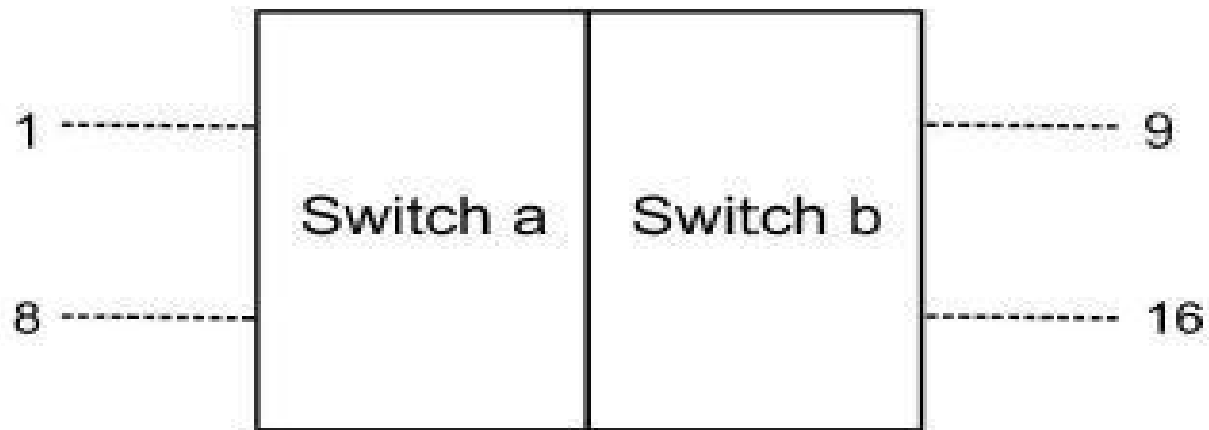
Switching forms a very important process in a communication system. A switch is used to connect the incoming link to the desired outgoing link and directs the incoming message to the appropriate outgoing link.

Consider a group of 8 people with telephones. If we were to use direct lines between all the people, we would need 28 duplex (wires that allow simultaneous two-way conversation) lines. The arithmetic is pretty simple - to connect n subscribers directly, we need $n(n-1)/2$ lines. This is alright as long as the number of subscribers is less and the distances are also small. But in the present day electronic communication systems, we are talking about connecting the entire world - obviously direct connections are not the answer. We need to design a system, which can connect the people from anywhere. Now, if we were to use a switch instead, we could reduce the number of lines needed to just 4, because with 8 subscribers, there would at the most be just 4 conversations simultaneously. The switch would have 4 lines internally and it would use the each line to connect a pair of subscribers.

SWITCHING

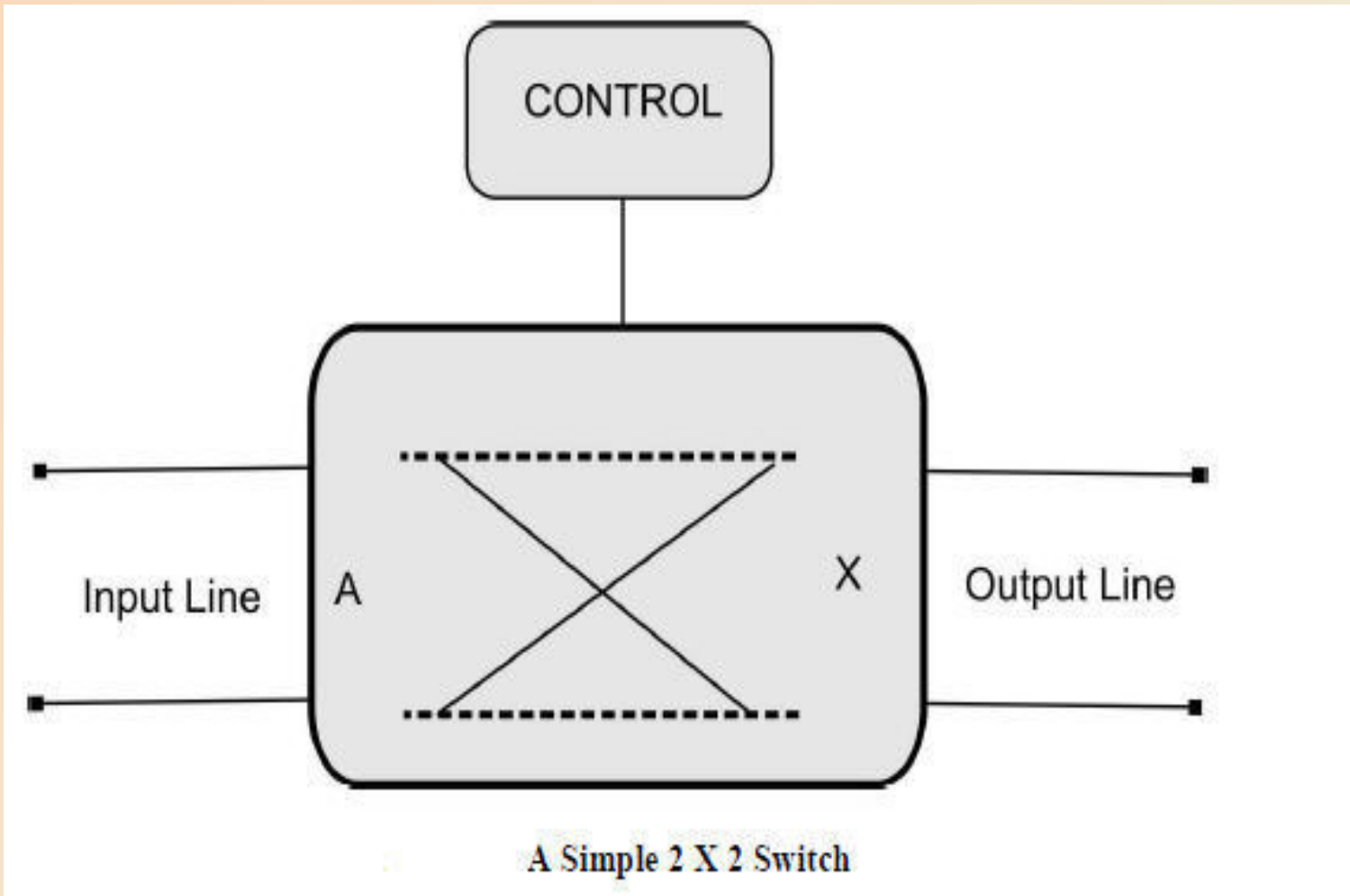


A simple switch with 4-input and 4-output lines



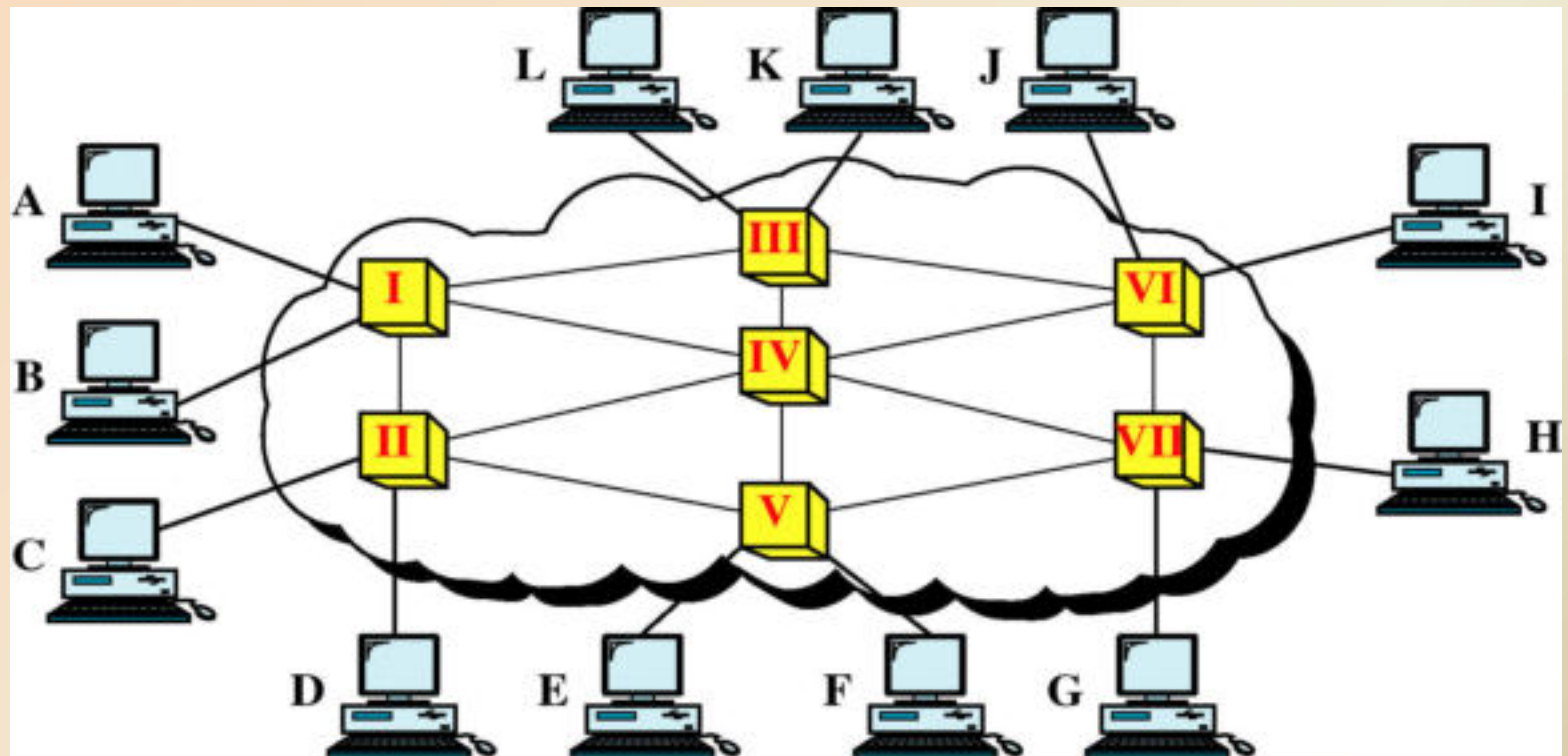
Two Switches with 8-subscribers

SWITCHING



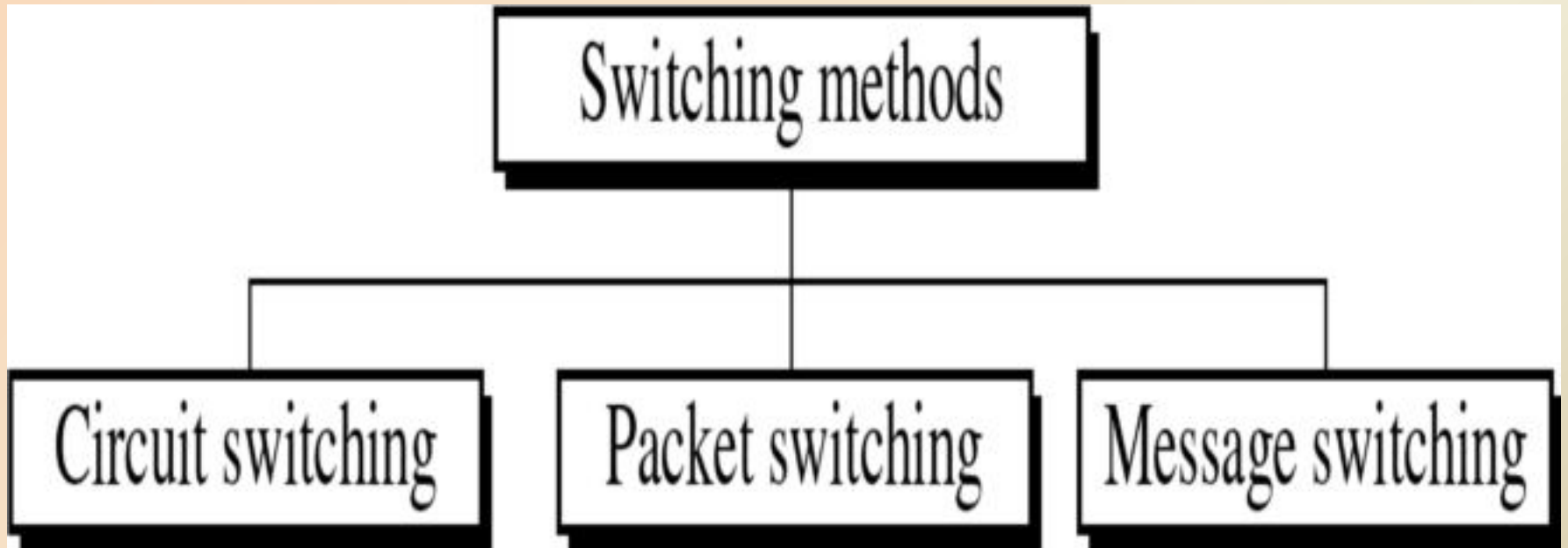
Switching

Switches are hardware and/or software devices capable of creating temporary connections between two or more devices linked to the switch but not to each other.



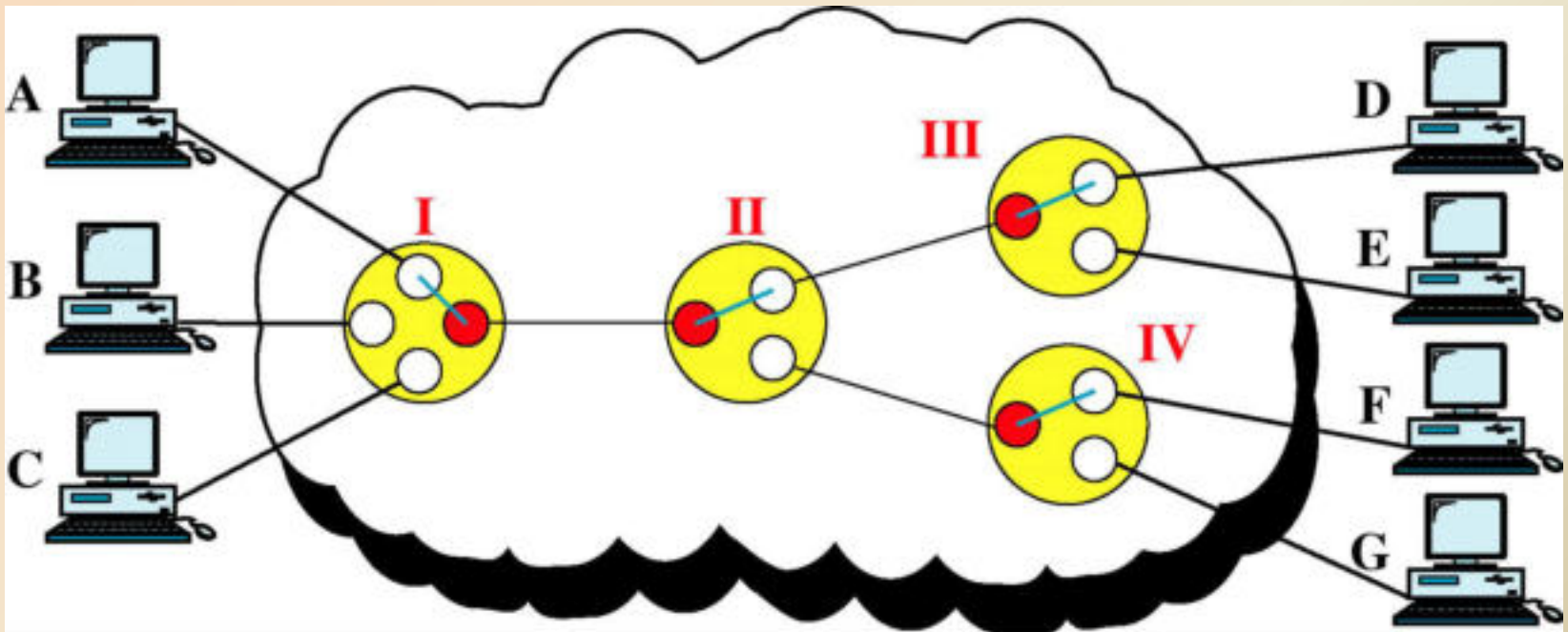
Switching

There are major three types of switching



Circuit Switching

Circuit switching creates a direct physical connection between two devices. It is a device with n inputs and m outputs that creates a temporary connection between an input link and an output link.



Circuit Switching

Circuit switching has been categorized either of two technologies:

a) Space-division Switching

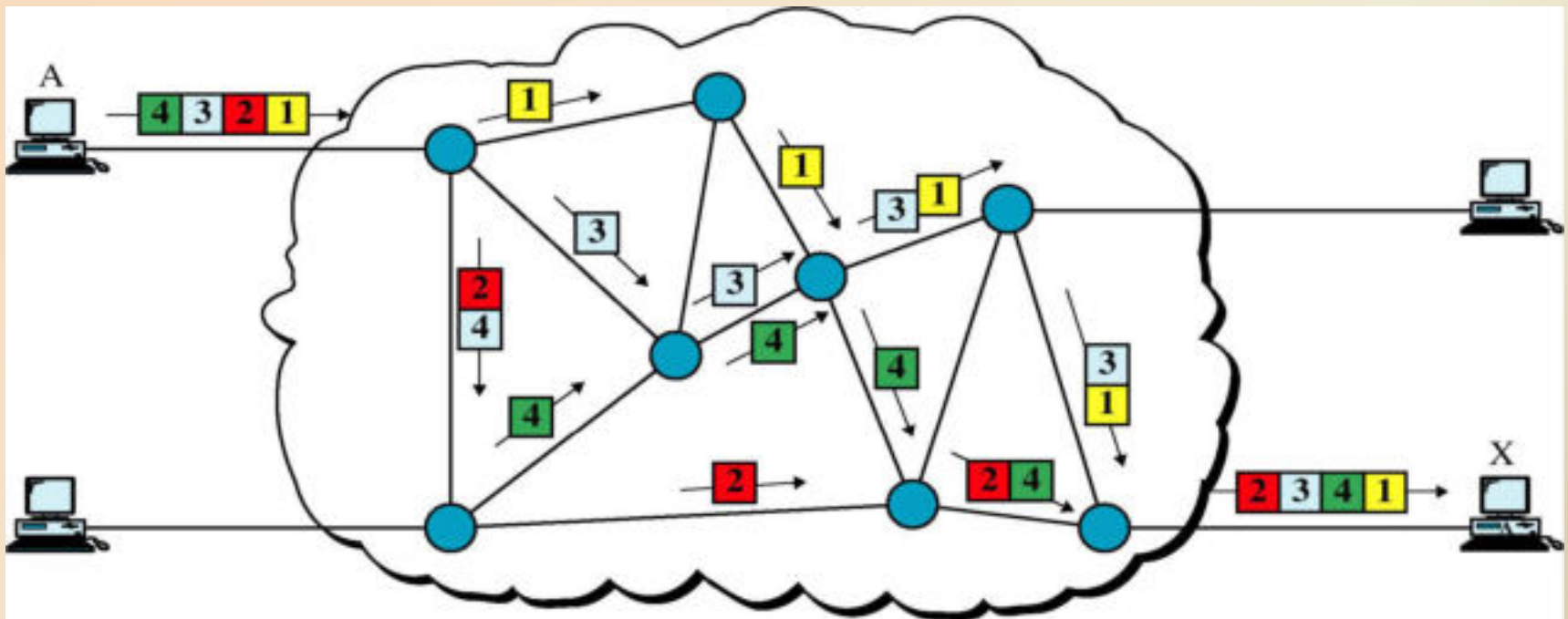
In Space-division Switching, the paths in the circuit are separate from each other. It can be used for both analog and digital communication.

b) Time-division Switching

In Time-division switching, uses time-division multiplexing to achieve switching.

Packet Switching

In Packet switch, data are transmitted in discrete units of potentially variable length blocks called packets. The maximum length of the packet is established by the network.



Packet Switching

There are two popular approaches to packet switching

a) **Datagram Approach**

In this approach each packet is treated independent from all other. Even when one packet represents just a piece of a multi-packet transmission, the network treats it as though it existed alone. Packets in this technology are referred to as **datagram**.

b) **Virtual Circuit Approach**

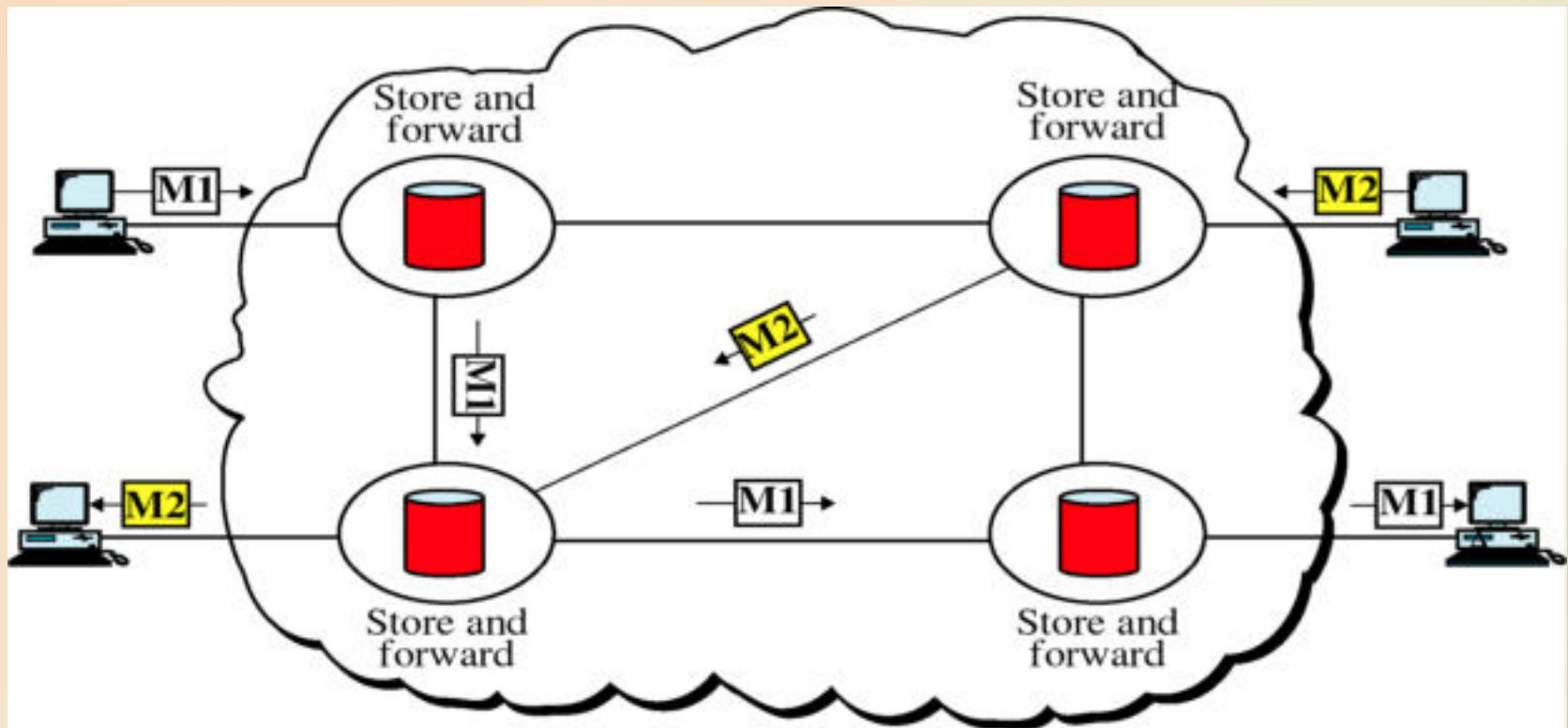
In the virtual circuit approach to packet switching, the relationship between all packets belonging to a message or session is preserved. A single route is chosen between sender and receiver at the beginning of the session.

It has been implemented in two formats:

- a) **Switched Virtual Circuit (SVC)**
- b) **Permanent Virtual Circuit (PVC)**

Message Switching

Message switching is best known by the descriptive term store and forward. In this mechanism, a node receives a message, stores it until the appropriate route is free, then sends it along.



Unit-4

Communication Mediums

Communication Mediums

- Digital Data Transmission
- Objectives
- Serial and Parallel Transmission
- Guided and Unguided Mediums
- Twisted pair
- UTP cable
- STP Cable
- Coaxial Cable
- Fiber Optic Cables
- Unguided Mediums
- Connectors

Digital Data Transmission

- Data transmission, digital transmission, or digital communication is the transfer of data from one point to another or a point-to-point or point-to-multipoint communication channel.
- Data transmitted may be digital messages originating from a data source, such as a computer or a keyboard.
- It may also be an analog signal, such as a phone call or a video signal.
- Examples of such channels include copper wires, optical fibers, wireless communication channels, storage media and computer buses.

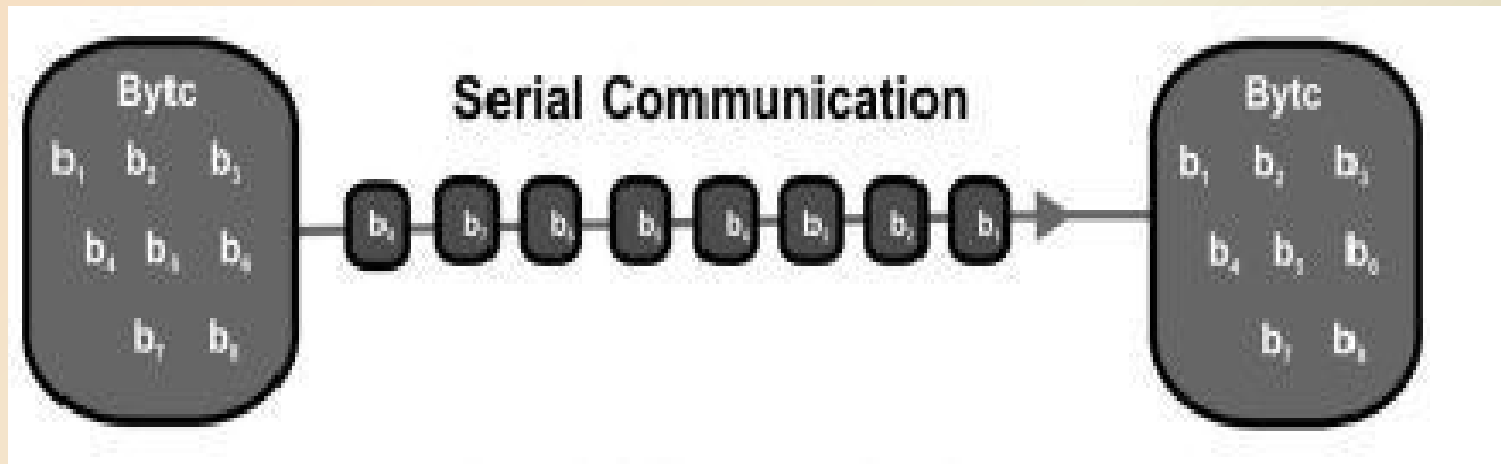
Objectives

- Know the concept of communication mediums
- Differentiate between Serial and Parallel Transmission
- Differentiate Between Guided and Unguided Mediums
- Know the features and limitations of different wired mediums
- Understand the use of Twisted pair, Coaxial and Fiber Optical Cables
- Know the functions of Unguided Mediums
- Understand the use of different connectors

SERIAL AND PARALLEL TRANSMISSION

SERIAL TRANSMISSION

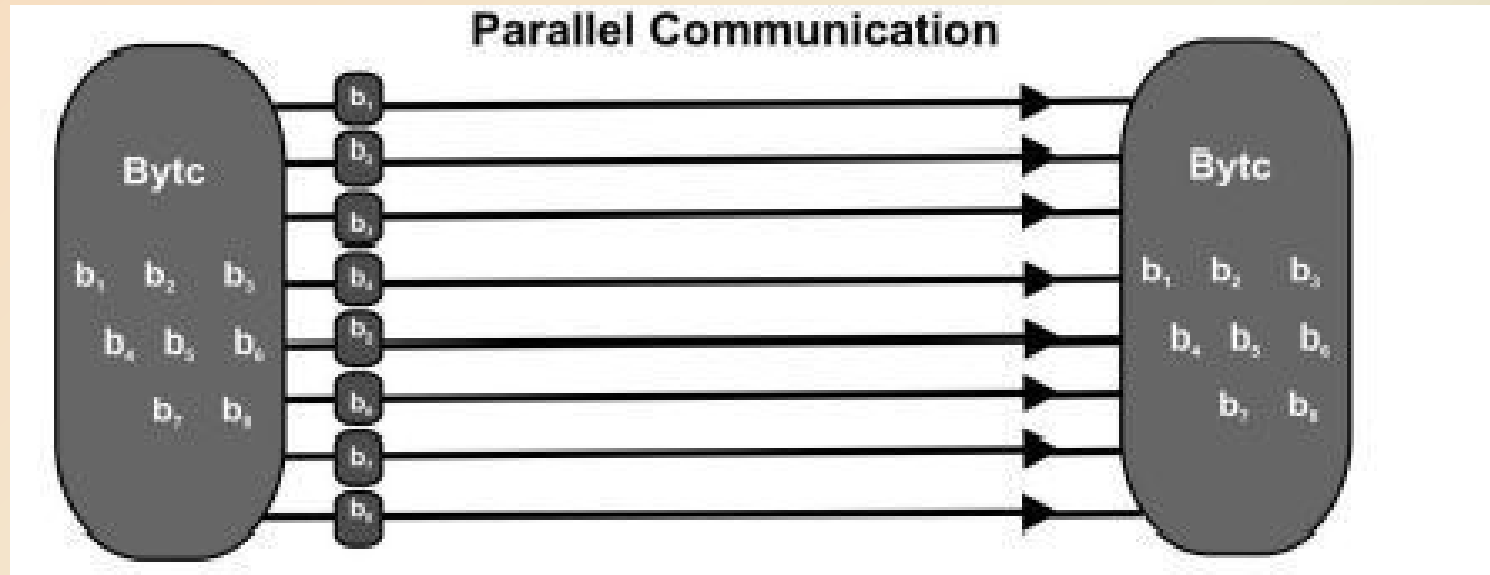
In serial transmission, bits are sent sequentially on the same channel (wire). Serial transmission can be either synchronous or asynchronous. In synchronous transmission, groups of bits are combined into frames and frames are sent continuously with or without data to be transmitted. In asynchronous transmission, groups of bits are sent as independent units with start/stop flags and no data link synchronization, to allow for arbitrary size gaps between frames. However, start/stop bits maintain physical bit level synchronization once detected.



SERIAL AND PARALLEL TRANSMISSION

PARALLEL TRANSMISSION

In parallel transmission, multiple bits (usually 8 bits or a byte/character) are sent simultaneously on different channels (wires, frequency channels) within the same cable as shown in Figure 1, or radio path, and synchronized to a clock. Parallel devices have a wider data bus than serial devices and can therefore, transfer data in words of one or more bytes at a time.

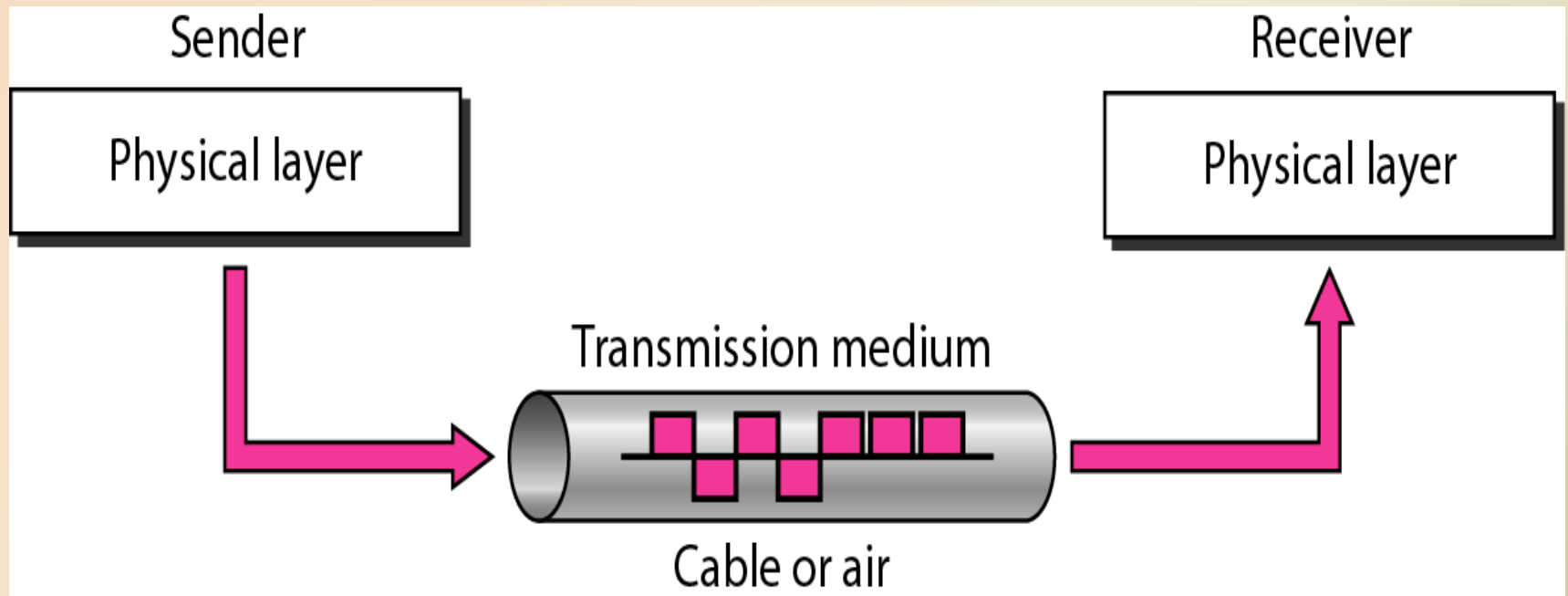


Transmission Media

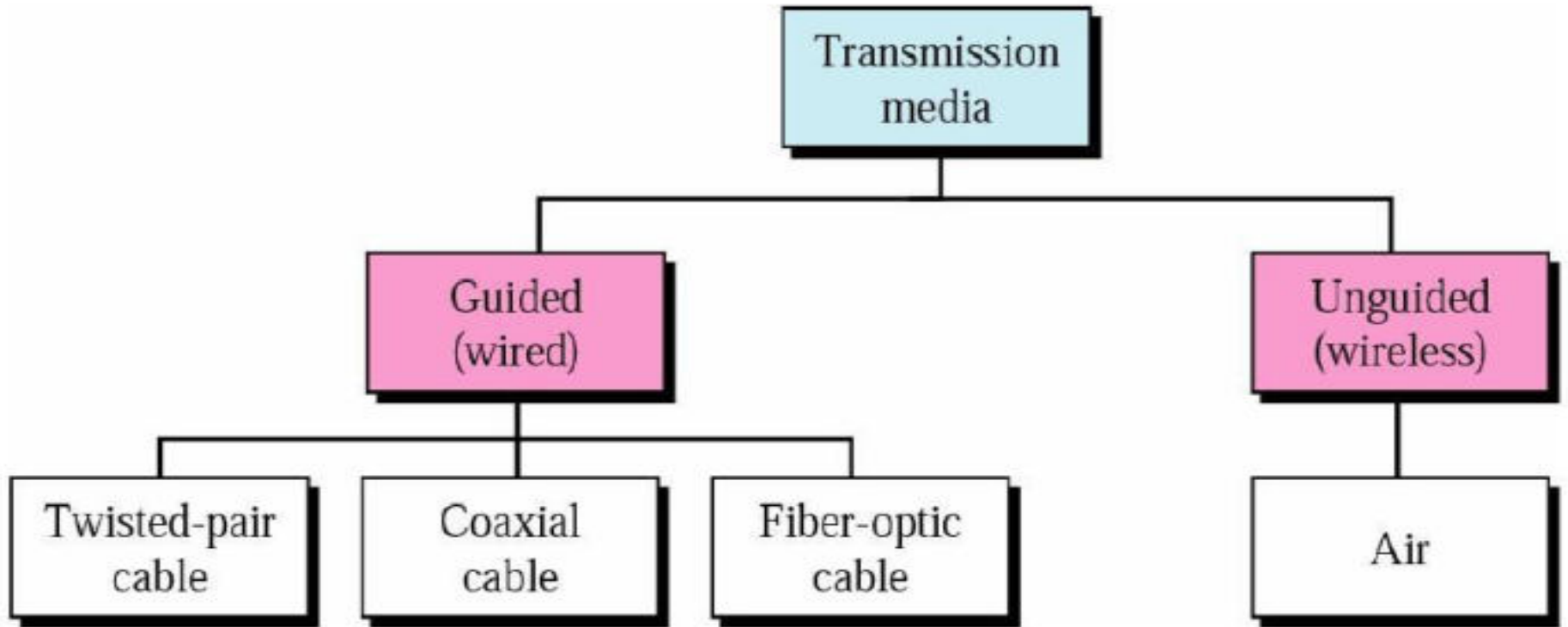
A Transmission media can be broadly defined as which can carry data and information from one place to another. More specifically media may be free space, metallic cable or optical fiber.

In telecommunication media can be divided into two categories

- a) Guided Media
- b) Unguided Media



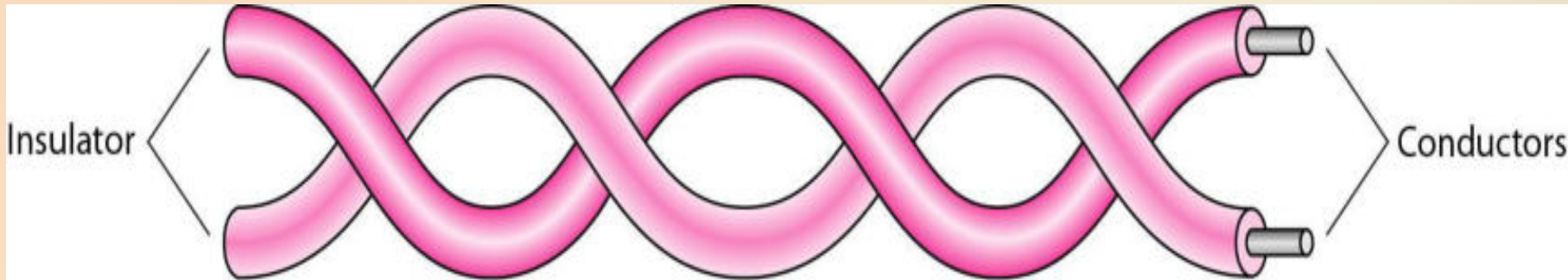
Transmission Media



Guided Media

The media which provide a conduit from one device to another. In the other sense we can say the transmission media which is covered in nature.

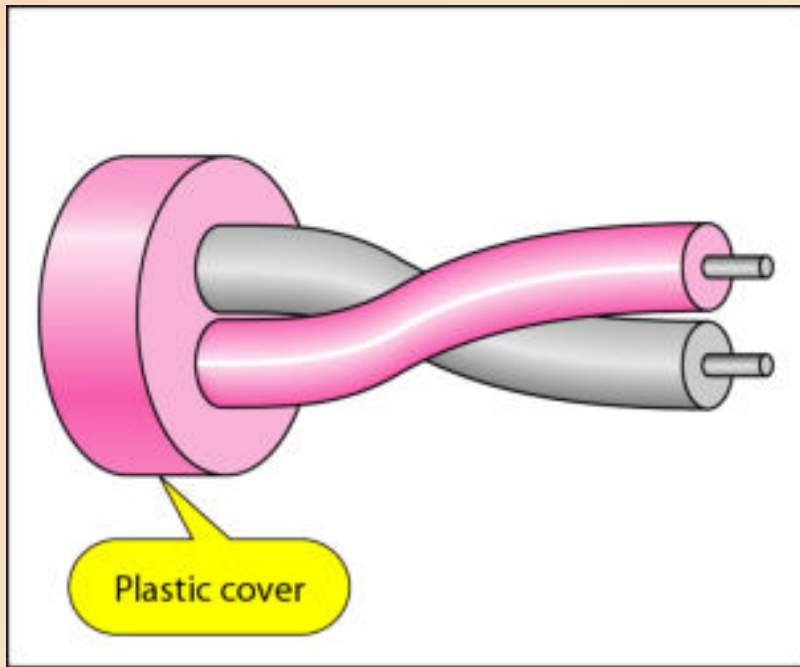
Twisted- pair Cable



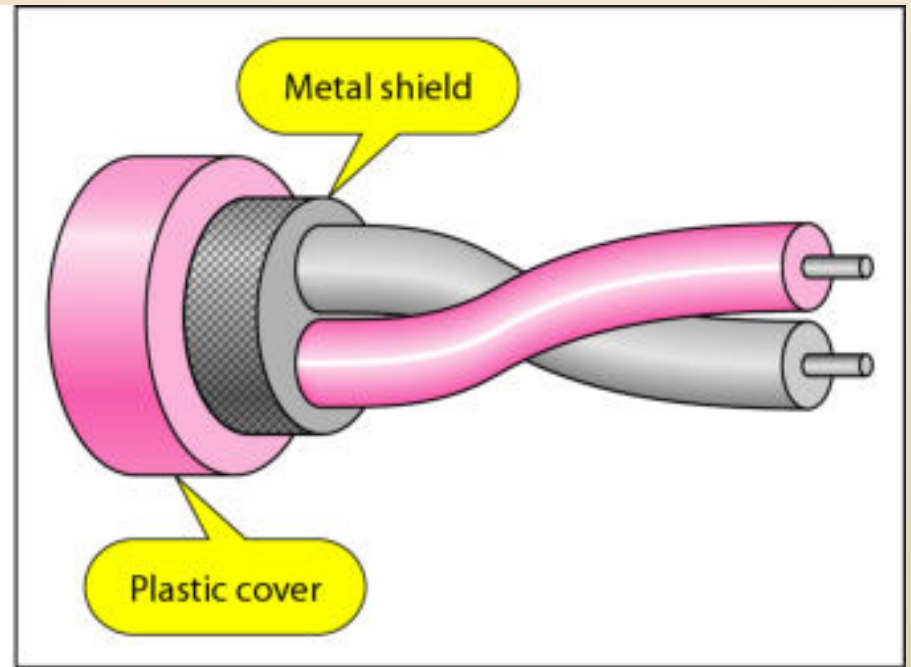
One wire carries signal and the other one is used as a ground reference. The Receiver uses the difference. By twisting in one turn one wire is closer to the noise source and in the next turn the other wire is closer and the noise effect cancels out. Quality determined by number of twists per inch.

Guided Media

UTP Cable (Unshielded Twisted Pair) & STP Cable (Shielded Twisted Pair)



a. UTP

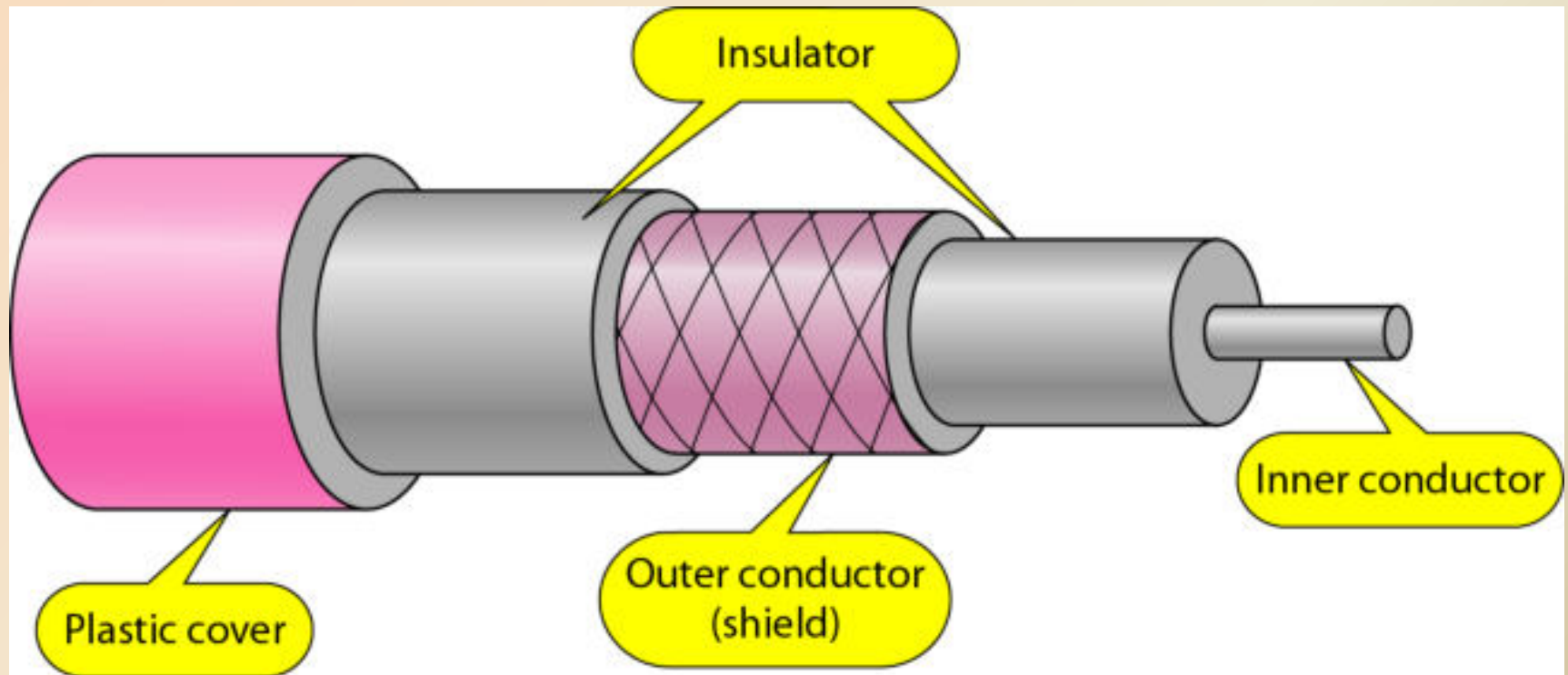


b. STP

Guided Media

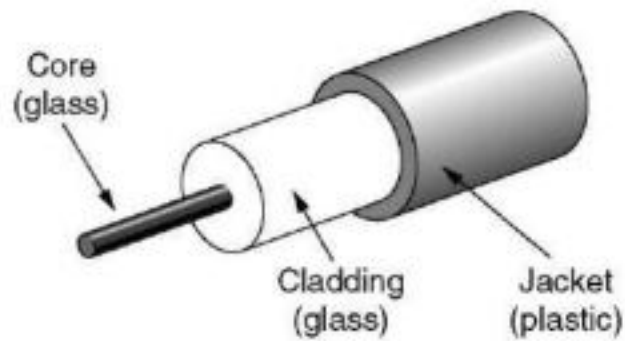
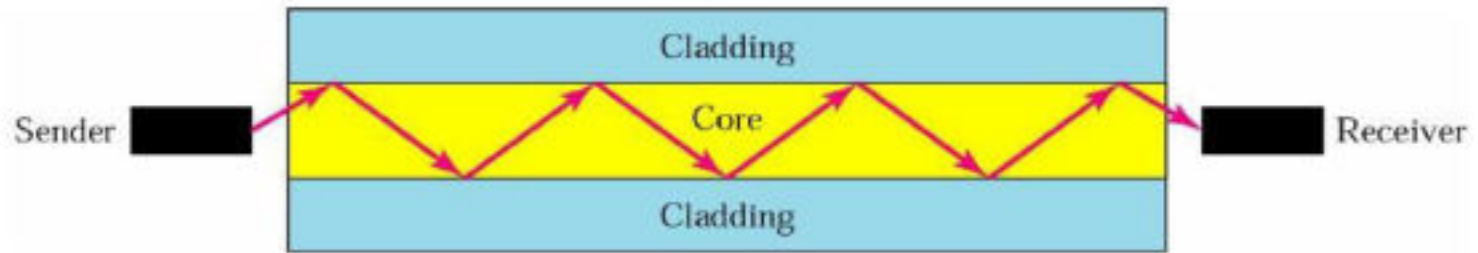
Coaxial cable carries signals of higher frequency ranges than those in twisted-pair cable.

Co-axial Cable

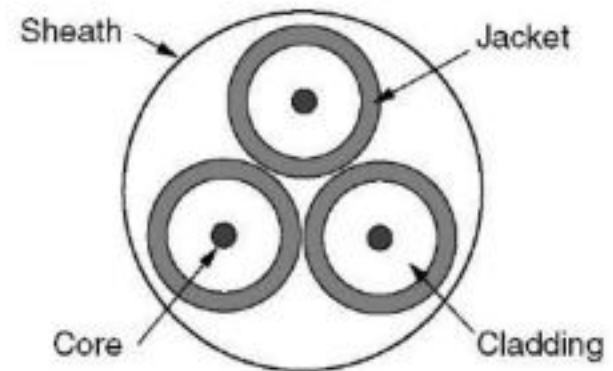


Guided Media

Fiber-Optic Cable : It is one of the good guided media for data transmission in the form of light. It is made of glass or plastic.

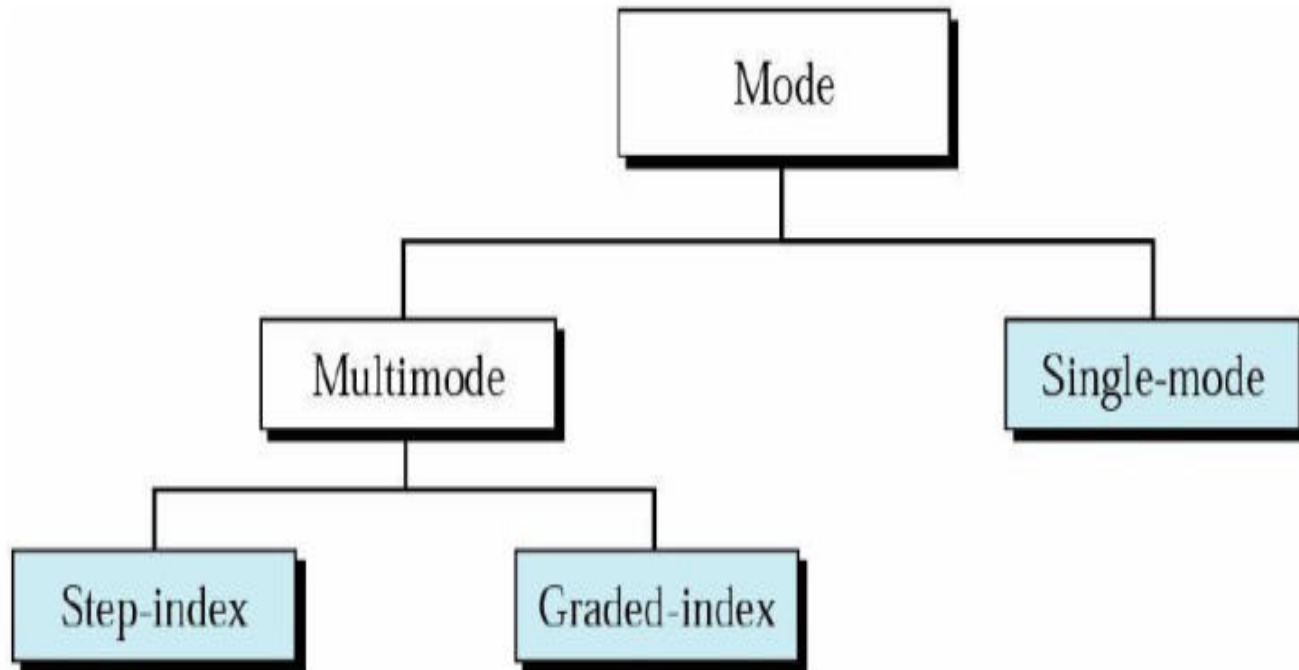


(a)

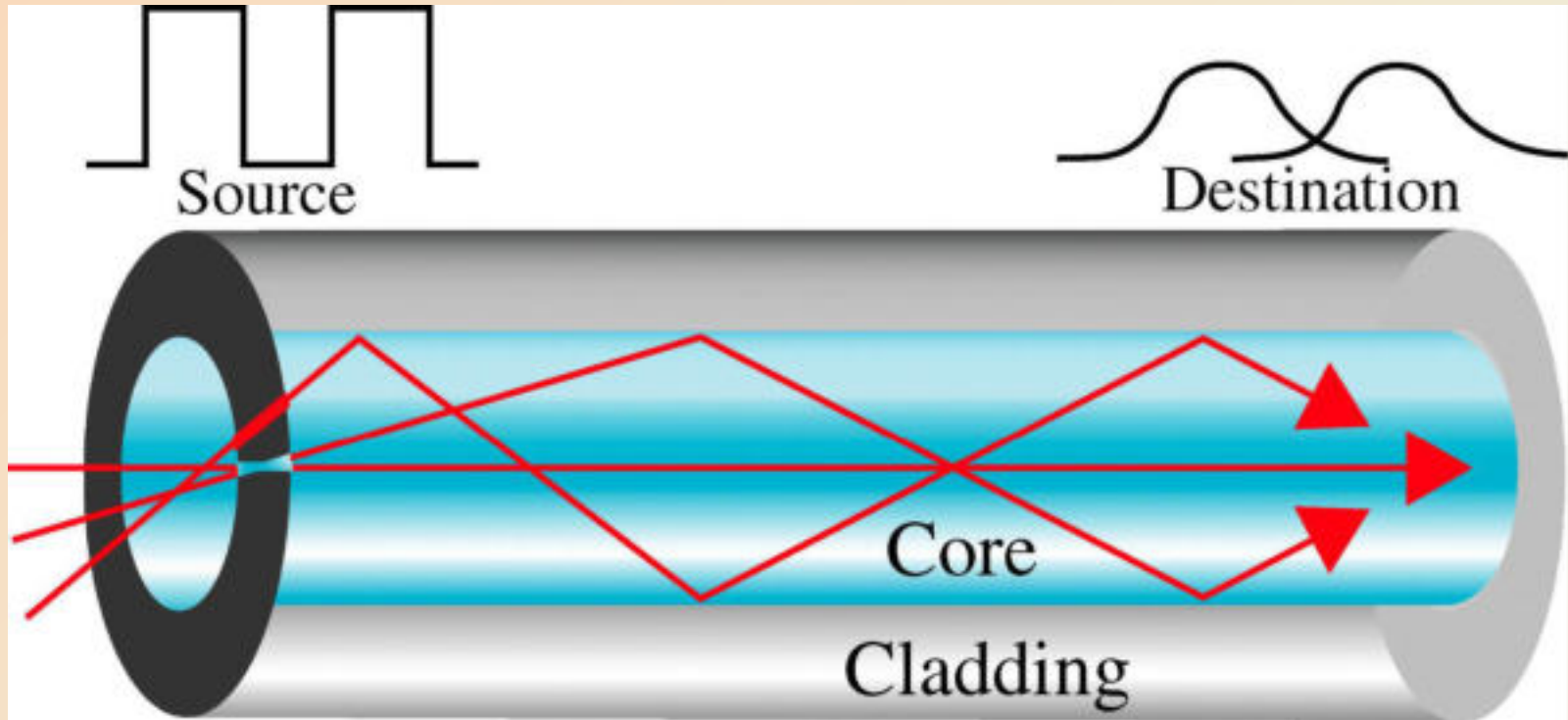


(b)

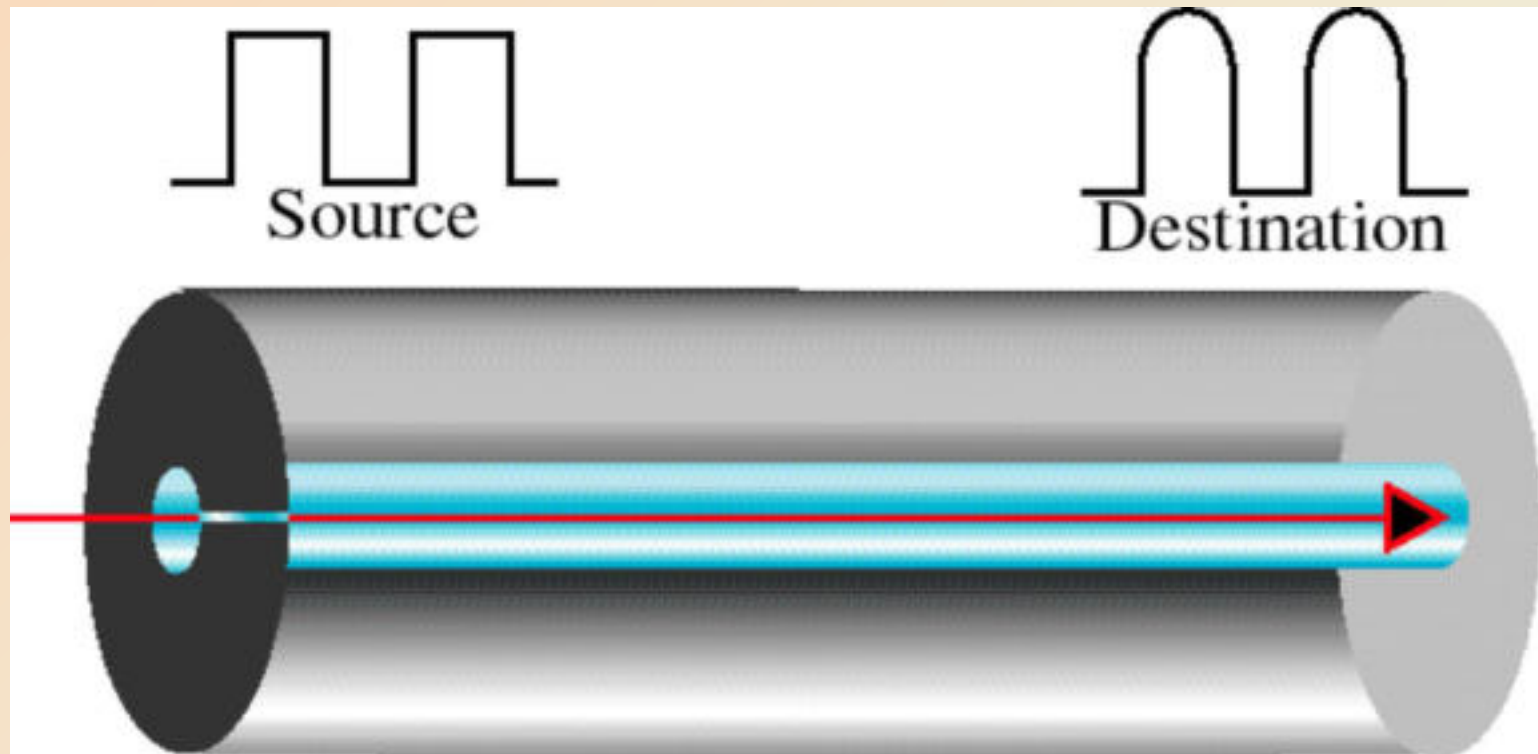
Guided Media



Guided Media

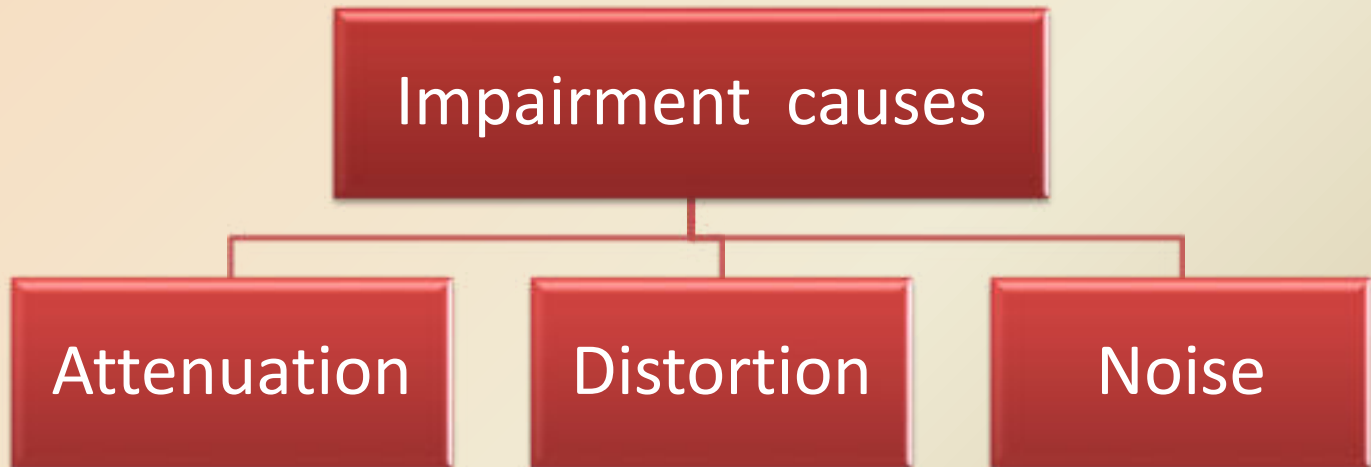


Guided Media



Transmission Impairment

When the signal transmits through the medium from source to destination then source generated data cannot be reached to destination due to the imperfection of media. Causes are given below:



Network Performance

Throughput : Throughput is the measure of how fast we can actually send data through the network.

Latency (Delay): The latency defines how long it takes for an entire message to completely arrive at the destination from the time the first bit is sent out from the source.

- Propagation Time
- Transmission Time
- Queuing Time
- Processing Delay

Thank You

SCHEDULING ALGORITHMS

What is scheduling?

- ▶ One CPU with a number of processes. Only one process can use the CPU at a time which process is going to be execute that thing decide a by scheduler and it's whole thing is called scheduling.

Two types of scheduling:

1. Preemptive
2. Non- Preemptive

Preemptive :

- i. In preemptive scheduling, the currently running process may be interrupted and move to the ready state by OS(forcefully).
- ii. It selects a process and lets it run for a specific time duration, called time quantum.
- iii. If process is **still** running at the end of the time interval, it is suspended and the scheduler selects another process to run.

Non-Preemptive :

- i. In non-preemptive scheduling, the running process can only lose the processor voluntarily by terminating or by requesting and I/O. OR, once CPU given to a process it can not be preempted until the process completes its CPU burst.
- ii. It selects a process to run and then just lets it run until it blocks or terminates.

Types of scheduler

▶ Long-term scheduler:

- ▶ load a job in memory
- ▶ Runs infrequently

▶ Medium-term scheduler:

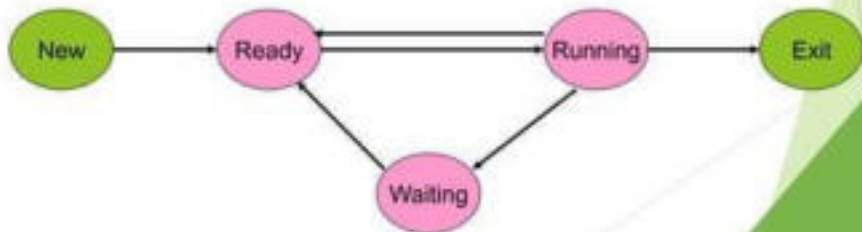
- ▶ Select ready process to run on CPU
- ▶ Should be fast

▶ Short-term scheduler:

- ▶ Reduce multiprogramming or memory consumption

Process Scheduling

- “process” and “thread” used interchangeably
- Which process to run next
- Many processes in “ready” state



FORMULA :

▶ TURN AROUND TIME:

▶ $TAT = (\text{Waiting Time})(W.T) + (\text{Burst Time})(B.T)$

▶ OR

▶ $= (\text{Completion Time})(C.T) - (\text{Arrival Time})(AT)$

▶ WAITING TIME:

▶ $WT = (\text{Turn Around Time})(TAT) - (\text{Burst Time})(B.T)$

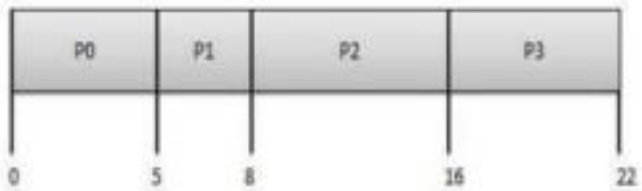
▶ OR

▶ $= (\text{Completion Time})(C.T) - (\text{Arrival Time})(A.T) - (\text{Burst Time})(B.T)$

First-Come First-Serve (FCFS)

- ▶ One ready queue;
- ▶ OS runs the process at head of the queue;
- ▶ New processes come in at the end of the queue;
- ▶ Running process does not give up the CPU until it terminates or it performs IO.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Shortest Job First (SJF)

- ▶ Best approach to minimize waiting time.
- ▶ Impossible to implement.
- ▶ Processer should know in advance how much time process will take.

- ▶ SJF is optimal - gives minimum average waiting time for a given set of processes.

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

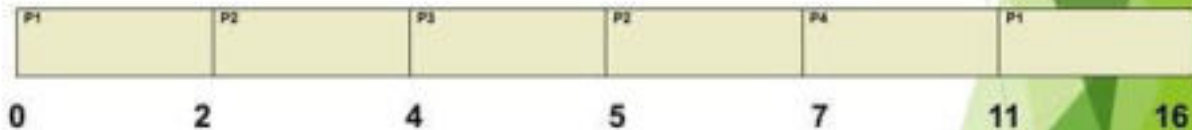
- ▶ SJF (non-preemptive):



Example of Preemptive SJF

Process	Arrival Time	Burst Time
<i>P1</i>	0	7
<i>P2</i>	2	4
<i>P3</i>	4	1
<i>P4</i>	5	4

SJF (preemptive)

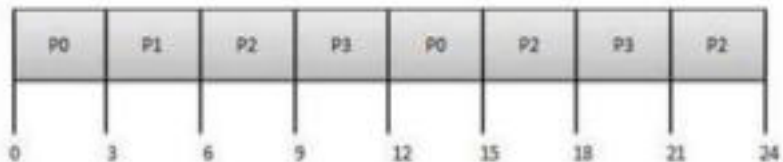


Round Robin Scheduling

- ▶ Each process is provided a fix time to execute called quantum.
- ▶ Once a process is executed for given time period. Process is preempted and other process executes for given time period.
- ▶ Context switching is used to save states of preempted processes.

Process	Arrival Time	Execute Time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

Quantum = 3



Priority Based Scheduling

- ▶ Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- ▶ Processes with same priority are executed on first come first serve basis.
- ▶ Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Priority based non pre-emptive Algo Example

Process	Arrival Time	Burst Time	Priority
P1	0	5	1
P2	1	3	2
P3	2	8	1
P4	3	6	3



Process	Arrival Time	Burst Time	Finish Time	Turnaround time	Waiting time
P1	0	5	14	14	9
P2	1	3	9	8	5
P3	3	8	17	14	12
P4	5	6	6	1	0



Thank You

INTRODUCTION TO LINUX



OVERVIEW



- ❧ What is Linux?
- ❧ History of Linux
- ❧ Features Supported Under Linux
- ❧ The features of Linux

WHAT IS LINUX ?



- Linux is a generic term referring to Unix-like graphical user interface (GUI) based computer operating systems.
- It is Multi-user, Multitasking, Multiprocessor
- Coexists with other Operating Systems
- Runs on multiple platforms

BEFORE LINUX



- ❧ In 80's, Microsoft's DOS was the dominated OS for PC
- ❧ Apple MAC was better, but expensive
- ❧ UNIX was much better, but much, much more expensive. Only for minicomputer for commercial applications
- ❧ People was looking for a UNIX based system, which is cheaper and can run on PC

GNU PROJECT



- Established in 1984 by Richard Stallman
- GNU is a recursive acronym for "GNU's Not Unix"
- Aim at developing a complete Unix-like operating system which is free for copying and modification
- Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed

BEGINNING OF LINUX



- ❧ A famous professor Andrew Tanenbaum developed Minix, a simplified version of UNIX that runs on PC
- ❧ Minix is for class teaching only. No intention for commercial use
- ❧ In Sept 1991, Linus Torvalds, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1

LINUX TODAY



- Linux has been used for many computing platforms
- PC, Supercomputer
- Commercial vendors moved in Linux itself to provide freely distributed code. They make their money by compiling up various software and gathering them in a distributable format
- Red Hat, Slackware, etc
- About 29 million people use Linux worldwide

LINUX-FREE SOFTWARE



- Free software, as defined by the FSF (Free Software Foundation), is a "matter of liberty, not price." To qualify as free software by FSF standards, you must be able to:
- Run the program for any purpose you want to, rather than be restricted in what you can use it for.
- Share the program with others.
- Improve the program and release those improvements so that others can use them.

LINUX SOFTWARES



- ☞ **Red Hat Linux** : One of the original Linux distribution.
- ☞ The commercial, nonfree version is **Red Hat Enterprise Linux**, which is aimed at big companies using Linux servers and desktops in a big way
- ☞ **Debian GNU/Linux** : A free software distribution. Popular for use on servers
- ☞ **Gentoo Linux** : Gentoo is a specialty distribution meant for programmers



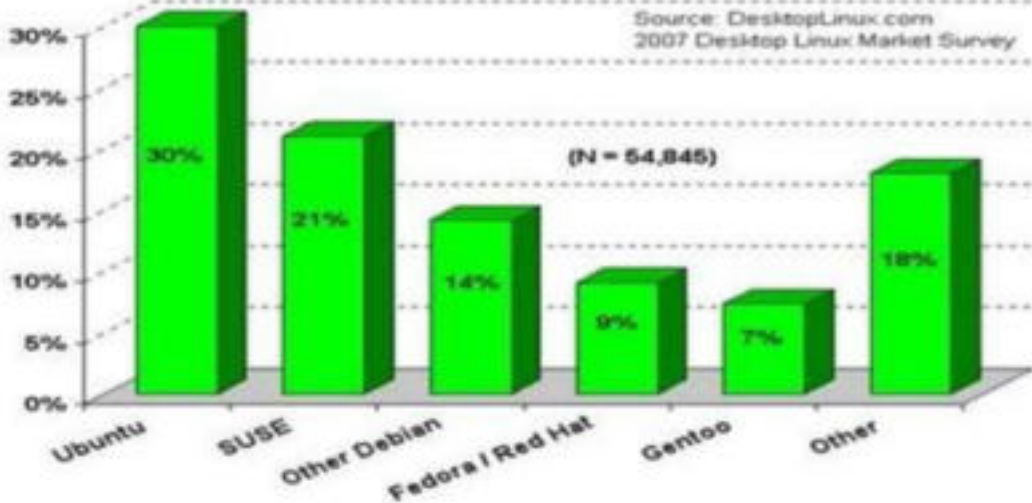
UBUNTU



- ❧ Ubuntu is a free Linux distros
- ❧ Ubuntu is reliable and stable.
- ❧ Ubuntu is the distribution with the biggest software repositories.
- ❧ Ubuntu has a good hardware support for most available companies.
- ❧ Ubuntu have a lot of variants (Kubuntu, Xubuntu, Edubuntu, Goubuntu, MIDI Ubuntu).

Ubuntu is Widely Used

Desktop Linux Distributions





Computer



User's Home



Trash

Default Fedora Desktop

The default desktop has three distinct areas.

From top to bottom, the areas are:

The *menu panel*

The *desktop area*

The *window list panel*



MENU PANEL



10:17 AM

Applications - The Applications menu contains a variety of icons that start software applications. It is similar to the Microsoft **Windows Start menu**.

Places - The Places menu contains a customizable list of directories, mounted volumes, recent documents, and a Search function. Volumes that are mounted may be external USB drives (flash, hard disk, CD, etc.), directories shared across a network, or other media devices such as a portable music player.

SYSTEM MENU




☞ Log Out


☞ About


☞ Help


☞ Lock Screen


☞ Preferences

☞ **Mozilla Firefox** web browser 

☞ **OpenOffice.org Writer** is a word processing program 

☞ **OpenOffice.org Impress** is for creating and giving presentations 

☞ **OpenOffice.org Calc** is a spreadsheet tool 

☞ **Evolution** mail client and personal information manager 

THE DESKTOP AREA



- ❧ **Computer** - This contains all volumes (or disks) mounted on the computer. These are also listed in the Places menu. Computer is equivalent to **My Computer** on Microsoft Windows.
- ❧ **Home** - This is where the logged-in user stores all files by default, such as music, movies, and documents. There is a different home directory for each user, and by default users cannot access each others' home directories. Home is equivalent to **My Documents** on Microsoft Windows.
- ❧ **Trash** - Deleted files are moved to Trash. Empty Trash by right-clicking the icon and clicking Empty Trash

START HERE



PROGRAM WINDOW

The screenshot shows a web browser window titled "Programs" with a menu bar (File, Edit, View, Go, Bookmarks, Preferences, Help) and a toolbar with navigation buttons (Back, Forward, Up, Refresh, Home, Web Search, Stop). The address bar shows "Location: programs:" and a magnifying glass icon. The main content area displays a folder view of the "Programs" folder, which contains 16 items. The items are organized into a grid of categories, each with an icon and a count of items:

- Applications** (9 items): Represented by a keyboard icon.
- Development** (0 items): Represented by a rocket icon.
- Games** (0 items): Represented by a joystick icon.
- Graphics** (3 items): Represented by a pencil icon.
- Help**: Represented by a question mark icon.
- Internet** (16 items): Represented by a globe icon.
- Multimedia** (7 items): Represented by a drum icon.
- Settings** (10 items): Represented by a wrench and screwdriver icon.
- System** (22 items): Represented by a computer monitor icon.

On the left side of the folder view, there is a sidebar with a "Tree" view and a list of items: "Notes", "News", "History", and "Help". Below the folder view, a status bar indicates that the "Programs" folder is selected and has a size of 1.0 K. The browser's taskbar at the bottom shows the "Programs" window is active.

APPLICATION WINDOW



CONTROL CENTRE



OFFICE SOFTWARE



- Word processor
- Spreadsheet
- Presentation
- Database application

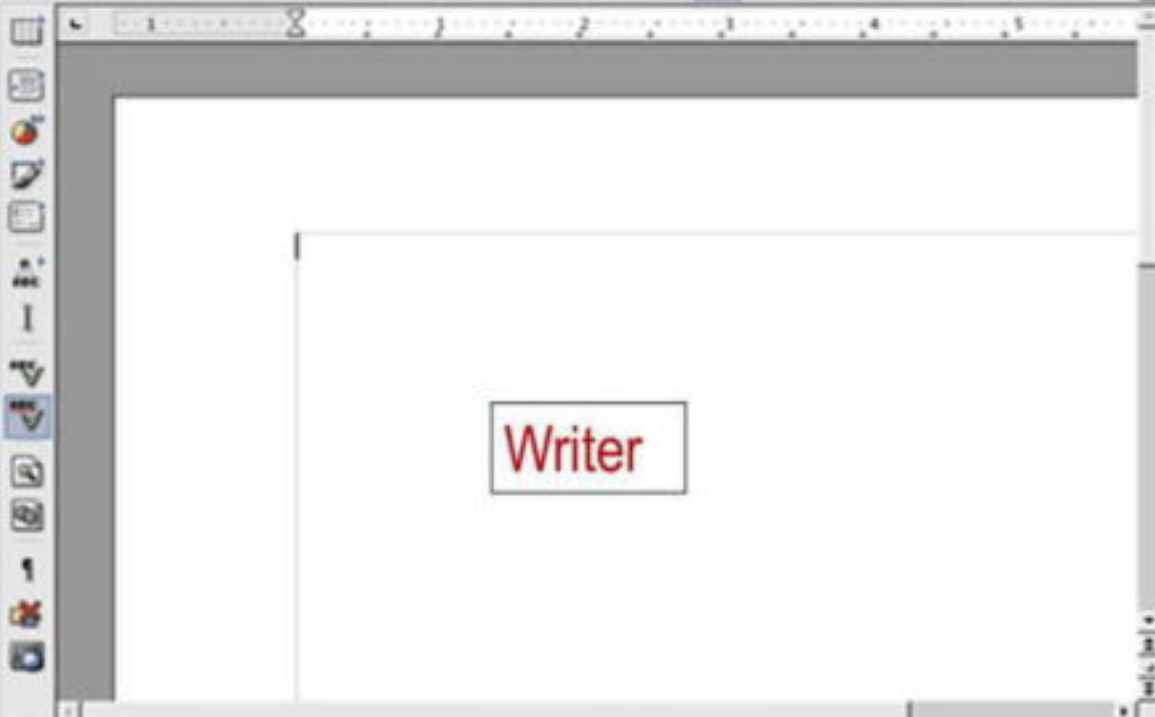
The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I	J	K
1	Fruit	Cost/Pound	Pounds	Total							
2	Apple	\$1.20	2.00	\$2.40							
3	Orange	\$0.80	2.00	\$1.60							
4	Banana	\$1.00	2.00	\$2.00							
5				\$3.28							

A red-bordered box with the text "Calc" is overlaid on the spreadsheet, positioned over cells F13 through G17.



Default Nimbus Roman Nc 12



Writer

WEB BROWSER



 Firefox

[Web](#) [Images](#) [Gmail](#) [News](#) [Local](#) [more...](#)[Google Search](#) | [I'm Feeling Lucky](#)Search:  the web  pages from Canada[Advanced Search](#)
[Feedback](#)
[Language Tools](#)Google.ca offered in: [français](#)[Advertising Programs](#) - [Business Solutions](#) - [About Google](#) - [Go to Google.com](#)

©2006 Google

OTHER SOFTWARES



- ❧ **Audio Player:** The XMMS (X Multimedia System), which is used to play digital sound files
- ❧ **CD Player:** The default CD player
- ❧ **Sound Juicer CD Ripper:** Burn your own CDs
- ❧ **Messaging Client:** GAIM supports AIM, MSN, ICQ, and many other popular IM networks
- ❧ **gFTP:** Useful for grabbing files through FTP (File Transfer Protocol)

WHY LINUX?



Powerful

- Runs on multiple hardware platforms
- Users like its speed and stability
- No requirement for latest hardware
- It's "free"**
- Licensed under GPL
- Vendors are distributors who package Linux



☞ **Multi-user**

☞ A multi-user operating system allows for multiple users to use the same computer at the same time and/or different times.

☞ **Multiprocessing**

☞ An operating system capable of supporting and utilizing more than one computer processor



☞ **Multitasking**

- ☞ An operating system that is capable of allowing multiple software processes to run at the same time

☞ **Multithreading**

- ☞ Operating systems that allow different parts of a software program to run concurrently.

Open Source Software



- ☞ People improve it, people adapt it, people fix bugs. And this can happen at a speed that, compared to conventional software development, seems astonishing



Linux Provide Security

MR. TUX

MR. SWAN

BIB

BIRDS IN BLACK



PROTECTING COMPUTERS
FROM THE SCUM OF
THE UNIVERSE

Linux.conf.au 2003

PERTH
WESTERN AUSTRALIA

<http://conf.linux.org.au/>

- As there is a limited access
- Of user to basic files and folders, in Linux network it provide security to user's privacy. Without disclosing the secured data Linux acts as a efficient server

Linux is Virus Free!!



Linux is "virus-free" in that there are essentially no viruses for Linux in the wild, although research viruses certainly do exist.

Linux Vs Windows



Linux is Cheaper





☞ **Keeping up to date**

- ☞ By Upgrading
- ☞ Linux upgrades faster than Windows

☞ **Compatibility**

- ☞ Linux is Backward Compatible unlike Windows

ADVANTAGES OF LINUX



- ❧ Linux systems are extremely stable
- ❧ Linux is Free
- ❧ No threat of viruses
- ❧ Linux comes with most of the required software pre-installed
- ❧ Update all your software with minimum fuss
- ❧ Linux never gets slow
- ❧ Linux does not need defragmentation
- ❧ Linux can even run on oldest hardware
- ❧ Adding more software is a matter of a few clicks
- ❧ Most Windows-only apps have their either their native version or alternatives for Linux
- ❧ With Linux, you get the highest degree of possible customizability

Thank You!



Linux™





What is exactly needed?



Why GNU/Linux?

- ✓ Software costs \$0
- ✓ Advanced Multitasking
- ✓ Remote tasking ("real networking")
- ✓ Multiuser
- ✓ Easy access to programming languages, databases, open-source projects



Why GNU/Linux?

Contd..

- ✓ Software freedoms
 1. Free to use for any purpose
 2. Free to study and modify the source code
 3. Free to share
 4. Free to share modified versions
- ✓ No dependence on vendors
- ✓ Better performance



Where is Linux Used?

- 75% of respondents were already using Linux and another 14% were evaluating it
- 43% of all web sites use Linux servers running the Apache Web server



Distributions

Unix Distributions

- Linux
- Solaris (Sun Micro Systems)
- HP-UX (Hewlett Packward)
- AIX (IBM)



Distributions

Linux Distributions

- Red Hat Linux
- Debian
- Mandrake
- SUSE



Basics

Shell

- Shell is nothing but a Command prompt like in windows, When you enter the commands, you get your work done
- The Shell is a command interpreter, it takes each command and passes it to the OS kernel to be acted upon. It will displays the results of your operations on screen.
- We have 'n' no.of shells available on any Unix Machine. each one has it's own strengths and weaknesses.



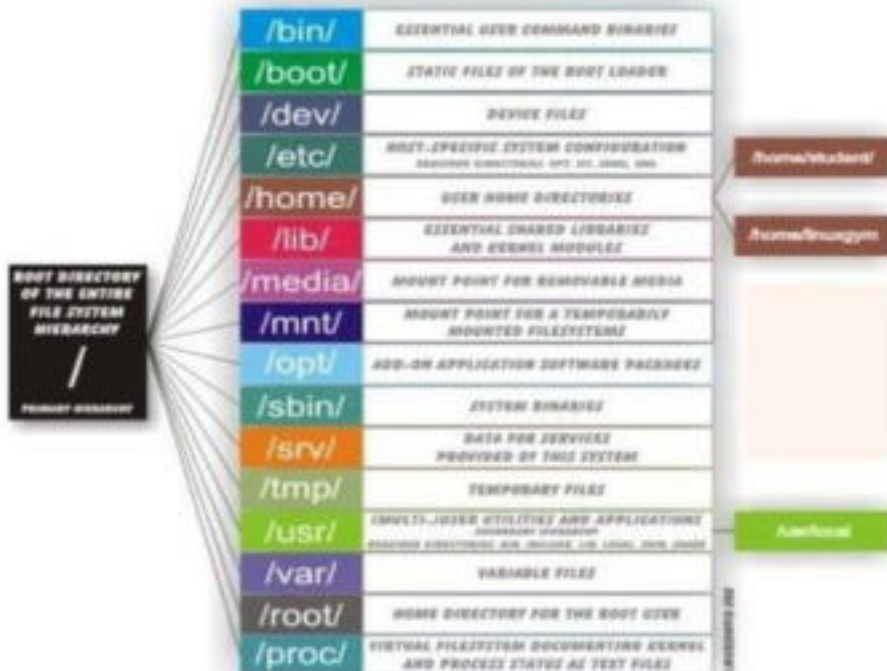
Continued...

➤ The most commonly available Shell(s) are:

1. Bourne Shell (sh)
2. Korn Shell (ksh)
3. C Shell (csh)
4. Bourne Again Shell (bash)
5. TC Shell (tcsh)



Unix Directory Structure/File System Architecture



Logging into the System

- To login to your account type your user name and at the login prompt. Unix is Case Sensitive.
- When the password prompt appears, type your password. Your password will never displayed on screen for security Measure. It is also case sensitive
- Then You will get [username@hostname]#
or [username@hostname <pwd>]\$.
- #, means it is administrative User Login
- \$, means it is Normal user Login
- And for Administrator always Username is 'root', and the User ID and Group Id is 0.



Basic Commands

Commands and their options

- ✓ Each and every command will have generally some options to get more information about the respective outputs.
- ✓ Options will be used by preceding "-". Options will be given after entering the command by giving some space



Working with Basic Commands

Uname: Prints system information

- a print all information
- s print the kernel name
- r print the kernel release
- v print the kernel version
- m print the machine hardware name
- p print the processor type
- o print the operating system

```
root@ubuntu:~# uname -a  
Linux ubuntu 2.6.38-8-generic #42-Ubuntu SMP Mon Apr 11 03:31:58 UTC 2011 i686 i686 i386 GNU/Linux
```



1 2 3 4 5 6 7 8
-rw-r--r-- 1 root root 34457 Jan 29 06:22 install.log

1 – file type

- Normal file

d Directory

l Link file

c character file

b block file

2 – File/Dir Permissions

3 – Owner

4 – Group membership

5 – File/Dir Size

6- File/Dir creation

/modification Date

7 – Time modified/created

8 – File name



Working with Basic Commands

- **man** – manual pages for commands or help for the commands
- Example :- man ls
- Output will be no. of pages.... To view the information line wise or page wise follow these...
- To see the output line by line press "Enter "
- To see the output page by page press "Spacebar"
- To quit from the manual page press the "q" key
- The above three options are useful in all other Unix commands, whenever this kind of scenario occurs.
- **clear** – to clear the screen.



Working with Basic Commands

uptime-Tells how long the system has been running

date – To know present date and time

cal – To know the present month calendar.

if you want specific month calendar in specific year

cal 3 1999 – will display the March 1999 calendar.

pwd – To get the present/current working directory.

mkdir – To create the directories

Ex:- mkdir dir1

To create multiple parent/child directories at a time

mkdir -p dir1/sdir1



Working with Basic Commands

cd – To change the working directory

```
cd <dir_name>    # Switches into specified directory.  
cd ..            # Moves one directory up  
cd ../../       # Moves two directories up (and so on)  
cd /            # Move the / directory.  
cd -            # Go back to you were previously (before the last directory  
change)
```

touch – To create the empty file, if file is not there, otherwise it updates file timestamp to current timestamp.

```
touch f1
```



Working with Basic Commands

cat – To create/view the files.

`cat > file1` - will create a file, it prompts for entering the data, after entering the data, save the file by pressing Ctrl-D.

`cat >> file1` – Will append the data to the file, it also prompts for entering the data, save the file by pressing Ctrl-D

Behavior of cat command

cat	File not exist	File exists
>	Creates it	Overwrites it
>>	Creates it	Append it

You can't edit the files using the cat command...

To view the multiple files

```
cat file1 file2
```



Working with Basic Commands

cp – to copy and paste the files/directories

- Syntax to copy the files

```
cp [options] <source> <destination>
```

To copy file1 as file2 in the same location.

```
cp file1 file2
```

To copy file1 from your present working directory to some other directory

```
cp file1 /opt/dir1/
```

while copying you want to change the file name

```
cp file1 /opt/dir1/file2
```



Working with Basic Commands

Contd..

To copy file1 from some other directory to your present working directory

```
cp /opt/dir1/file1 . ( . Means present working directory)
```

while copying you want to change the file name

```
cp /opt/dir1/file1 file2
```

To copy file1 from some directory to some where else in the system

```
cp /opt/dir1/file1 /home/user1/
```

while copying you want to change the file name

```
cp /opt/dir1/file1 /home/user1/file2
```



Working with Basic Commands

- Syntax to copy the directories
`cp -r [options] <srcdir> <destdir>`

Options:-

- f - to copy files/dir forcefully.. This option will be useful when we want to copy files/dirs, if already exists.
 - i - Interactively copying the files by asking question want to copy or not?
 - p - while copying files/dirs to preserve the file attributes like timestamp, permissions, ownership, groups.
- ```
cp -rf /opt/dir1 /home/user1/
```





# Working with Basic Commands

## rm – removing of files/dirs

- Syntax to remove the files  
rm [options] <filename>
  
- To remove a file  
rm file1
  
- To remove all files starts with "f"  
rm f\*
  
- To remove all files ends "f"  
rm \*f
  
- Syntax to remove the directories  
rm -r [options] <dirname>



# Working with Basic Commands

**mv** – to move files/dirs from one location to another location...And also we can rename the files/dirs

-To rename a file/dir

```
mv file1 file2
```

```
mv dir1 dir2
```

-To move a file/dir

```
mv file1 /opt/dir1
```

```
mv dir1 /home/user1
```

-To move a file/dir and also rename

```
mv file1 /opt/dir1/file2
```

```
mv dir1 /home/user1/dir2
```





# Working with Basic Commands

**wc** – To get the count of lines/words/characters

```
wc file1
```

To get the lines/words/characters output individually.

-l – no.of lines

```
wc -l file1
```

-w – no.of words

```
wc -w file1
```

-c – no.of characters

```
wc -c file1
```



# Working with Basic Commands

- To view the file content page by page
  - more
  - less

`more <my_file>` # views text, use space bar to browse, hit 'q' to exit

`less <my_file>` # a more versatile text viewer than 'more', 'q' exits, 'G' moves to end of text, 'g' to beginning, '/' find forward, '?' find backwards



# Working with Basic Commands

- To view the file top lines based on numerical value

`-head`

`head -n <filename>`

by default you can see first 10 lines of the file

```
head file1
```

to view first 20 lines from file1

```
head -20 file1
```

- To view the file end lines based on numerical value

`-tail`

`tail -n <filename>`

by default you can see end 10 lines of the file

```
tail file1
```

to view last 20 lines from file1

```
tail-20 file1
```



Thank You!  
😊



# Linux: VI editor



- ✓ *Different modes*
- ✓ *Insert commands*
- ✓ *Delete commands*

- ⦿ Linux offers a various types of editors like ex , sed , ed ,vi ,vim ,xvi , nvi , elvis etc.
- ⦿ To create and edit files
- ⦿ Vi editor most commonly used,
- ⦿ Created by Bill Joy at the University of California at Berkeley.

- ⦿ This editor can be invoked by typing vi at the \$prompt .
- ⦿ A filename specified as an argument to vi then vi will edit the specified file , if it exists.

Vi<filename>

Vi modes:

There are 3 modes insert , command and ex.

## Insert mode

The text should be entered in this mode , and any key press in this mode is treated as text.

We can enter into this mode from command mode by pressing any of the keys:

I , r , o , O , R , l , s , S , a , A .

command mode

It is the default mode when we start up vi editor.

All the commands on vi editor should be used in this mode.

We can enter into this mode from Insert mode by pressing the (esc key) and from ex mode by(enter key).



## Ex mode

The ex mode commands can be enter at the last line of the screen in this mode.

We can enter into this mode from command mode by pressing [:] key.

## Insert commands

- i Inserts before cursor
- I Inserts at the beginning of current line
- a Appends after cursor
- A Appends at the end of the current line
- o Inserts a blank line below the current line.

## DELETE COMMANDS

|          |                                                     |
|----------|-----------------------------------------------------|
| x        | Deletes a character at the cursor position          |
| X        | Deletes a character before the cursor position      |
| dd       | Deletes current line                                |
| Dw,dW    | Ignores punctuation that appears with the word.     |
| Db,dB    | Ignores any punctuation that appears with the word. |
| D[enter] | Deletes current line and following line.            |

End of presentation

# UNIT 1



# What is an Operating System?

---

- n A program that acts as an intermediary between a user of a computer and the computer hardware
  
- n Operating system goals:
  - | Execute user programs and make solving user problems easier
  - | Make the computer system convenient to use
  - | Use the computer hardware in an efficient manner



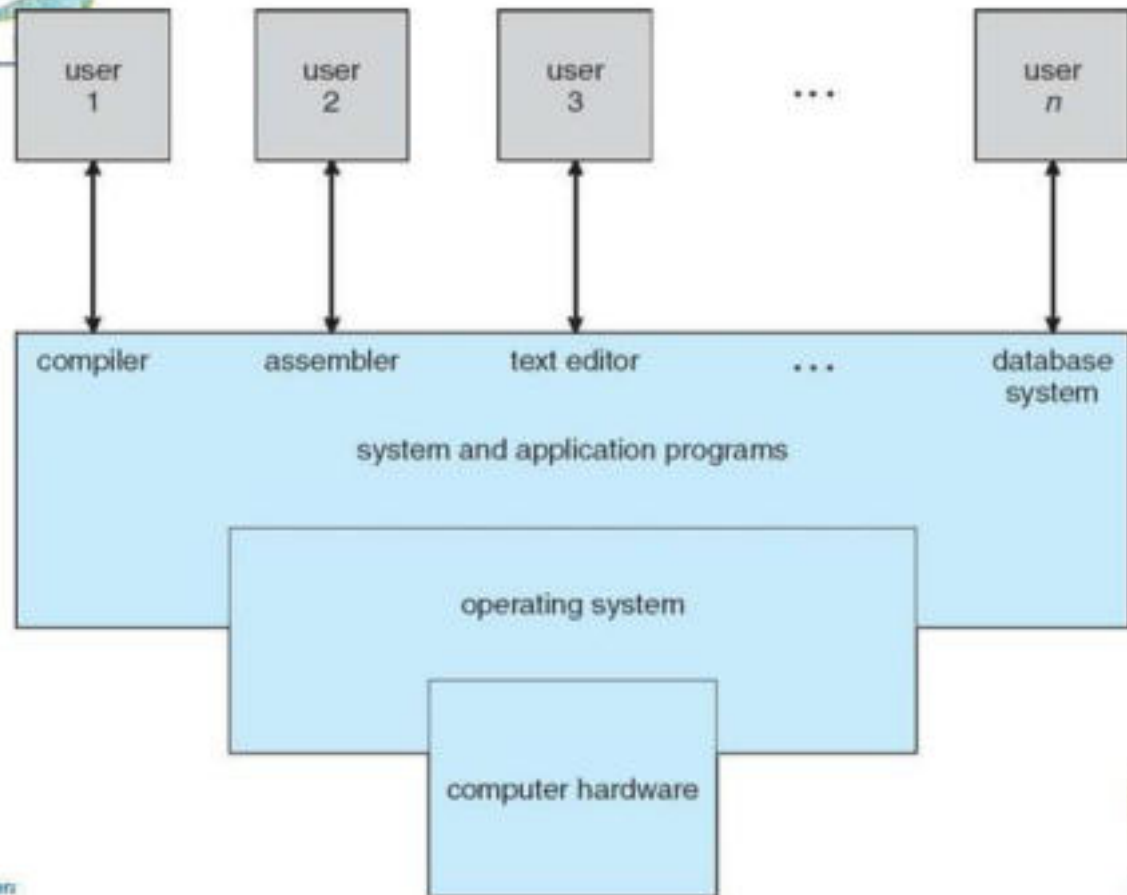


# Computer System Structure

---

- n Computer system can be divided into four components:
  - | Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - | Operating system
    - Controls and coordinates use of hardware among various applications and users
  - | Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - | Users
    - People, machines, other computers









# What Operating Systems Do

---

- n Depends on the point of view
- n Users want convenience, **ease of use**
  - | Don't care about **resource utilization**
- n But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- n Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- n Handheld computers are resource poor, optimized for usability and battery life
- n Some computers have little or no user interface, such as embedded computers in devices and automobiles





# Operating System Definition

---

- n OS is a **resource allocator**
  - | Manages all resources
  - | Decides between conflicting requests for efficient and fair resource use
  
- n OS is a **control program**
  - | Controls execution of programs to prevent errors and improper use of the computer





## Operating System Definition (Cont.)

---

- n No universally accepted definition
  
- n "Everything a vendor ships when you order an operating system" is good approximation
  - | But varies wildly
  
- n "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.





# Computer Startup

---

- n **bootstrap program** is loaded at power-up or reboot
  - | Typically stored in ROM or EPROM, generally known as **firmware**
  - | Initializes all aspects of system
  - | Loads operating system kernel and starts execution



# Types of OS

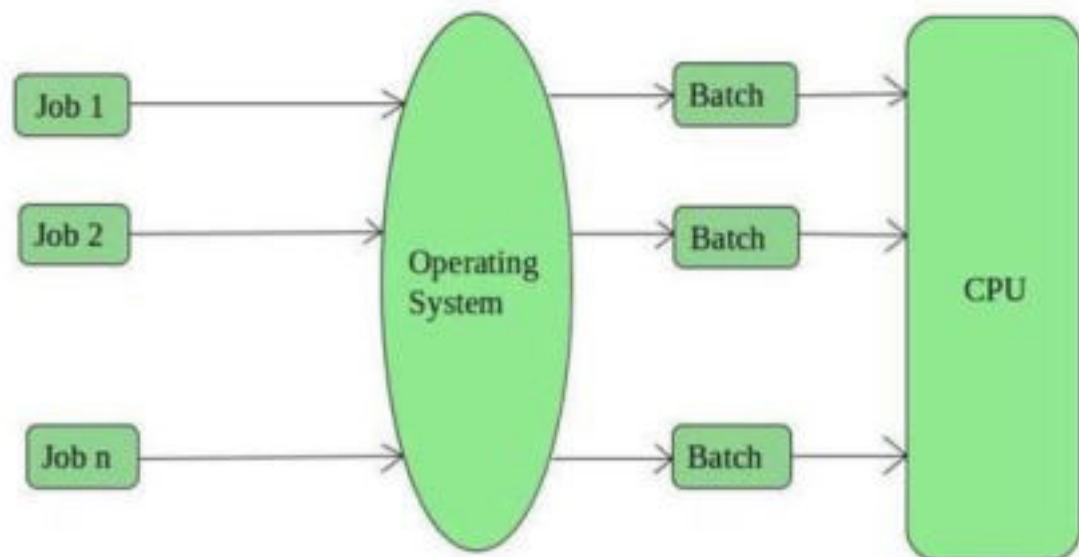
---

- ❑ **Batch Operating System**
- ❑ **Time-Sharing Operating Systems**
- ❑ **Distributed Operating System**
- ❑ **Network Operating System**
- ❑ **Real-Time Operating System**

# Batch Operating System

- This type of operating system does not interact with the computer directly.
- There is an operator which takes similar jobs having same requirement and group them into batches. It is the responsibility of operator to sort the jobs with similar needs.
- **Examples of Batch based Operating System:** Payroll System, Bank Statements etc.

# Cont..



## Cont..

- **Advantages of Batch Operating System:**
- It is very difficult to guess or know the time required by any job to complete.
- Multiple users can share the batch systems
- The idle time for batch system is very less
- It is easy to manage large work repeatedly in batch systems



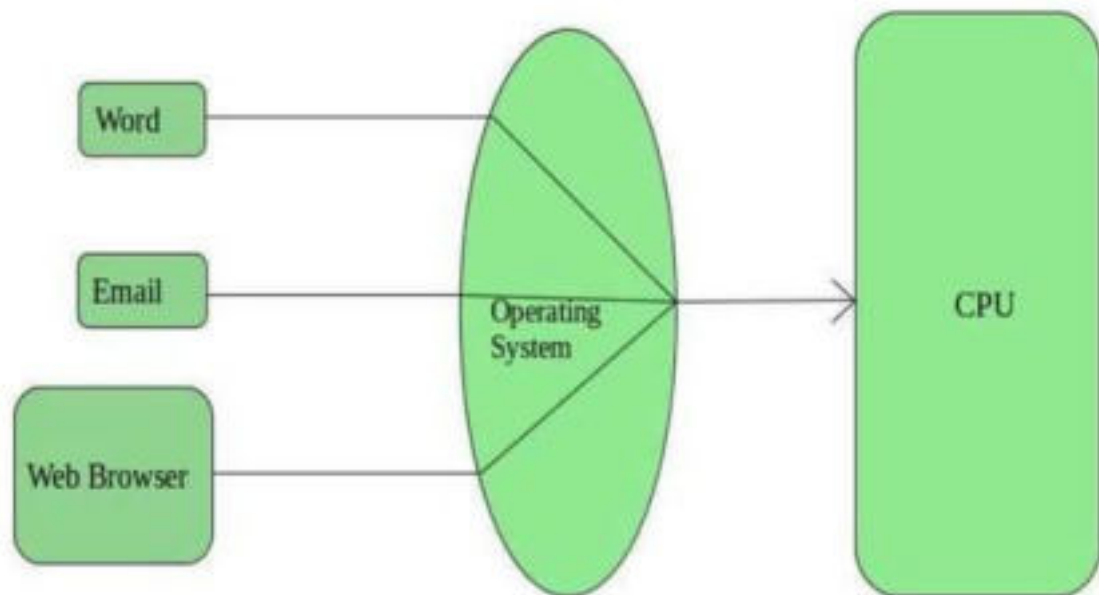
## Cont..

- ❑ **Disadvantages of Batch Operating System:**
- ❑ The computer operators should be well known with batch systems
- ❑ Batch systems are hard to debug
- ❑ It is sometime costly
- ❑ The other jobs will have to wait for an unknown time if any job fails

# Time-Sharing Operating Systems

- Each task is given some time to execute, so that all the tasks work smoothly.
- Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems.
- After this time interval is over OS switches over to next task.
- **Examples of Time-Sharing OSs are:Unix.**

# Cont..



# Cont..

## □ **Advantages of Time-Sharing OS:**

- Each task gets an equal opportunity
- Less chances of duplication of software
- CPU idle time can be reduced

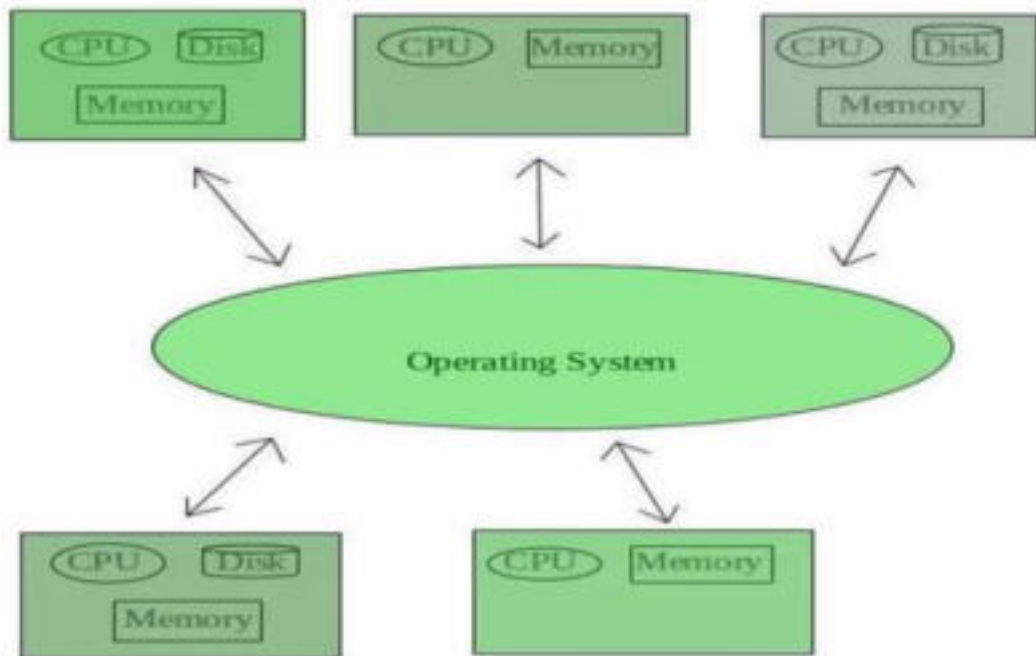
## □ **Disadvantages of Time-Sharing OS:**

- Reliability problem
- One must have to take care of security and integrity of user programs and data
- Data communication problem

# Distributed Operating System

- These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all-over the world.
- These are referred as **loosely coupled systems** or distributed systems.
- It is always possible that one user can access the files or software which are not actually present on his system but on some other system connected within this network i.e., remote access is enabled within the devices connected in that network.

# Cont..



# Cont..

- ❑ **Advantages of Distributed Operating System:**
- ❑ Failure of one will not affect the other network communication, as all systems are independent from each other
- ❑ Electronic mail increases the data exchange speed
- ❑ Since resources are being shared, computation is highly fast and durable
- ❑ Load on host computer reduces
- ❑ These systems are easily scalable as many systems can be easily added to the network
- ❑ Delay in data processing reduces

## Cont..

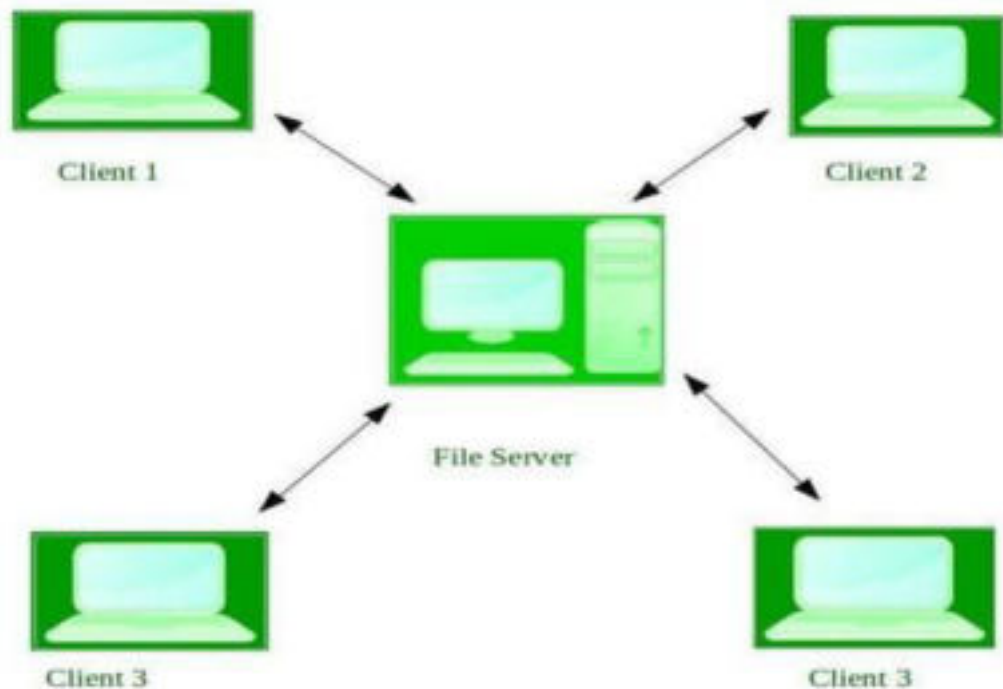
- ❑ **Disadvantages of Distributed Operating System:**
- ❑ Failure of the main network will stop the entire communication
- ❑ To establish distributed systems the language which are used are not well defined yet
- ❑ These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet



# Network Operating System

- ❑ These systems **run on a server** and provide the capability to **manage data, users, groups, security, applications, and other networking functions.**
- ❑ These type of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network.
- ❑ One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections etc. and that's why these computers are popularly known as **tightly coupled systems.**

# Cont..



# Cont..

## □ **Advantages of Network Operating System:**

- Highly stable centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated to the system
- Server access are possible remotely from different locations and types of systems

## □ **Disadvantages of Network Operating System:**

- Servers are costly
- User has to depend on central location for most operations
- Maintenance and updates are required regularly

## Cont..

- **Examples of Network Operating System are:** Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD etc.

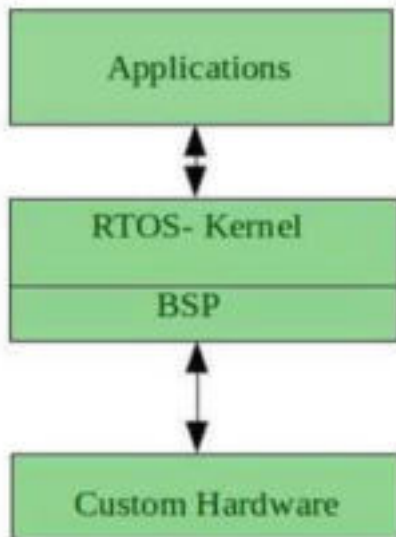
# Real-Time Operating System

- These types of OSs serves the real-time systems.
- The time interval required to process and respond to inputs is very small.
- This time interval is called **response time**.
  
- **Real-time systems** are used when there are time requirements are very strict like **missile systems, air traffic control systems, robots etc.**

## Cont..

- **Two types of Real-Time Operating System which are as follows:**
- **Hard Real-Time Systems:**  
These OSs are meant for the applications where time **constraints are very strict and even the shortest possible delay is not acceptable**. These systems are built for saving life like **automatic parachutes or air bags** which are required to be readily available in case of any accident.
- **Soft Real-Time Systems:**  
These OSs are for applications where for time-constraint is less strict.

# Cont..



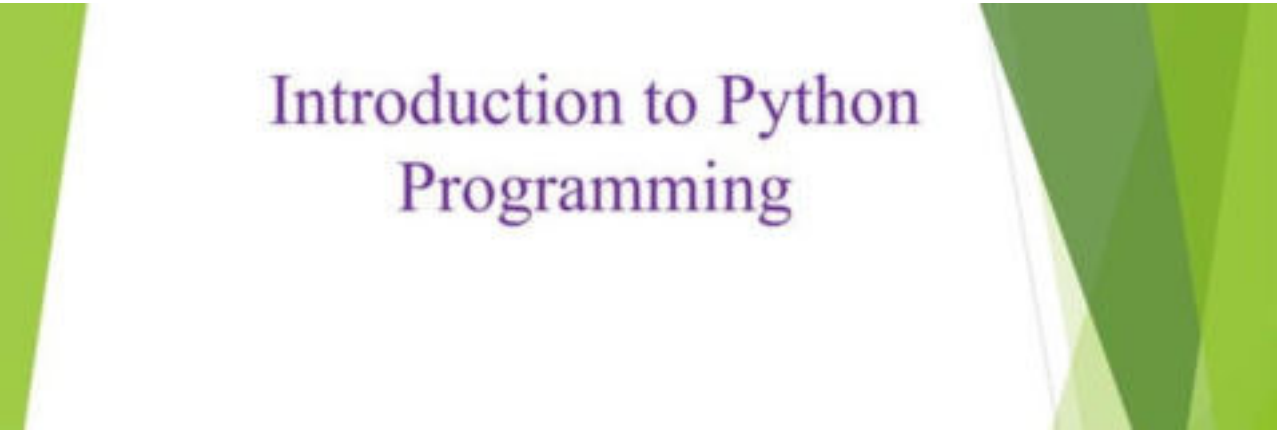
# Operating System Services

- An Operating System provides services to both the users and to the programs.
- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.



# Cont..

- ❑ **Following are a few common services provided by an operating system –**
- ❑ Program execution
- ❑ I/O operations
- ❑ File System manipulation
- ❑ Communication
- ❑ Error Detection
- ❑ Resource Allocation
- ❑ Protection



# Introduction to Python Programming

# Introduction

## What is Python?



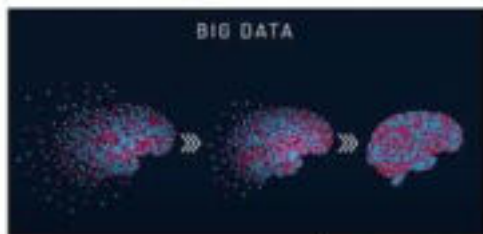
OR



## Who Can Use Python?



► Where Python could be used?



► What Python can do?



## ► Where Python is used?

Data Analysis

Mobile Apps

AI / ML

Desktop Apps

Automation

Testing

Web Apps

Hacking

Wait! I can do all these things using other programming languages too! Then why Python?



## ► Why Python?

► Solves complex problems in less time with fewer lines of code.

► Hello World

► str = 'Hello world'

► print(str[0:3])

► Hel

C#

```
str.Substring(0, 3)
```

JAVASCRIPT

```
str.substr(0, 3)
```

PYTHON

```
str[0:3]
```

## ► Why Python?

- Python is a multi-purpose language with a simple, and beginner-friendly syntax.

High-level

Huge Community

Cross-platform

Large Ecosystem



## So, What we learned so far?

- ▶ Python is a high-level, interpreted, interactive and object-oriented scripting language.
- ▶ Python was designed to be highly readable which uses English keywords frequently where as other languages use punctuation and it has fewer syntactical constructions than other languages.

## So, What we learned so far?

- ▶ **Python is interpreted:** Processed at run time by the interpreter, no need to compile the program before executing it.
- ▶ **Python is Interactive:** It means we can actually sit at a python prompt and interact with the interpreter directly to write programs
- ▶ **Python is Object-Oriented:** Supports OOP that encapsulates code within objects
- ▶ **Python is Beginner's Language:** It is great language for beginner programmers and supports the development of a wide range of applications.



**GUIDO VAN ROSSUM**  
Creator of Python

► Hello world program in Java

```
public class HelloWorld
```

```
{
```

```
 public static void main(String[] args)
```

```
 {
```

```
 System.out.println("Hello World");
```

```
 }
```

```
}
```

► Hello world program in Python

```
print("Hello World")
```

▶ Hello Python

▶ # Online Python compiler (interpreter) to run Python online.

▶ # Write Python 3 code in this online editor and run it.

▶ `print("Hello world")`

▶ `str="Hello world"`

▶ `print(str)`

▶ `print(str[0:3])`

```
Hello world
Hello world
Hel
```

Easy 01

Extensible 07

Expressive 02

Embeddable 08

Free and  
Open Source 03

Interpreted 09

High-Level 04

Large Standard  
Library 10

Portable 05

GUI  
Programming 11

Object  
Oriented 06

Dynamically  
Typed 12

Features of




Python

## ► Java vs Python



## ► Java vs Python



IntelliPost

What is Java?

Java is an object oriented and platform independent programming language used for developing various applications

Java



## ► Java vs Python



**5**

**4**

**3**

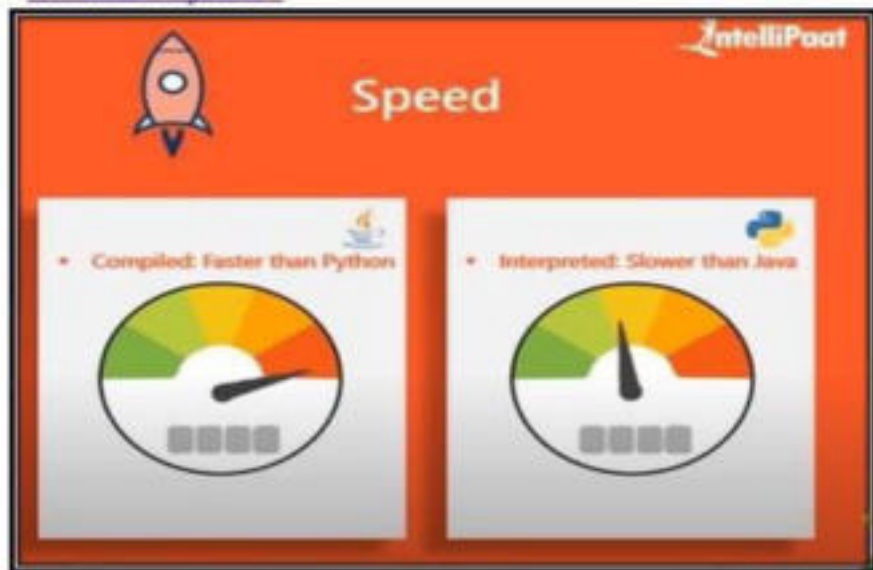
**2**

**1**

### What is Python?

Python is a simple, open source and object oriented programming language used for Artificial Intelligence, Machine Learning, web development and many more

## ► Java vs Python



## ► Comments

**# This line is commented and won't be executed**

.....

This (.....) is called as docstring.

Docstring can act as a multiline comment in Python after few lines of code.

.....

**Please note:** The recommended method for commenting multiple lines is using # on each line. The (.....) method **isn't actually a comment but defines a Text constant of the text between the (.....)**. It isn't displayed, but exists and could potentially cause unexpected errors.

▶ Programming Assignment

▶ Print your name.

▶ Declare two variables (a , b) and assign values.

▶ Print the results of the following

1. Addition of two numbers
2. Subtraction of two numbers
3. Multiplication of two numbers
4. Division of two numbers

▶ Swap the numbers

▶ Print the values of a and b after swapping

▶ Submit the screenshot.

## Data Types

- ▶ Identifiers and Keywords.
- ▶ Integral Types.
- ▶ Floating point types.
- ▶ Strings.

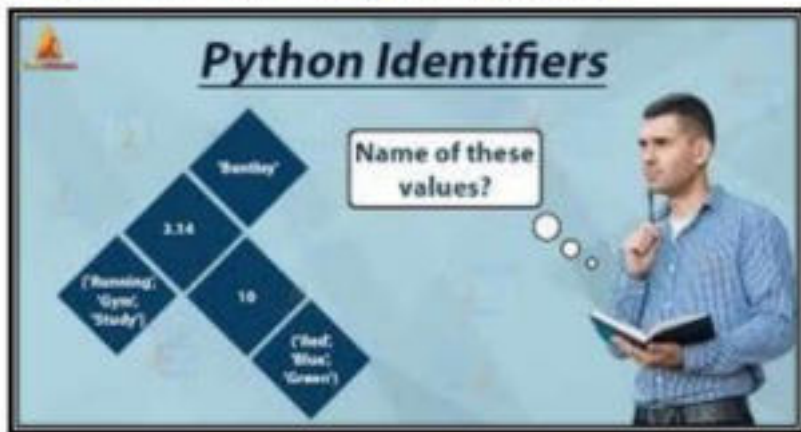
## Data Types - Identifiers and Keywords

### Python Keywords & Identifiers



## Data Types - Identifiers and Keywords

- ▶ **Python Identifiers** : An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.



## Data Types - Identifiers and Keywords

- ▶  $100 \times 5 = 500$
- ▶  $5 \times 5 = 25$
- ▶  $500 - 25 = 475$
  
- ▶  $100 * 5 - 5 * 5$



- ▶ `No_Honey_Cake_Made=100`
- ▶ `Profit_on_each_cake=5`
- ▶ `Unsold_cake=5`
- ▶ `Profit=No_Honey_Cake_Made*Profit_on_each_cake-Unsold_cake*Profit_on_each_cake`

```
ActivePython 3.6.0.3600 (ActiveState Software Inc.) based on
Python 3.6.0 (default, Jan 23 2017, 20:01:14) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 100*5-5*5
475
>>> No_Honey_Cake_Made=100
>>>
>>> Profit_on_each_cake=5
>>> Unsold_cake=5
>>> Profit=No_Honey_Cake_Made*Profit_on_each_cake-Unsold_cake*Profit_on_each_cake
>>> Profit
475
>>>
```



## Data Types - Identifiers and Keywords

### ► Operator precedence :

► ( )

► \* /

► + -

► \* and / or + and -



a = 20

b = 10

c = 15

d = 5

e = 0

```
e = (a + b) * c / d #(30 * 15) / 5 = 90
print ("Value of (a + b) * c / d is ", e)
```

```
e = ((a + b) * c) / d #(30 * 15) / 5 = 90
print ("Value of ((a + b) * c) / d is ", e)
```

```
e = (a + b) * (c / d); # _____ = _____
print("Value of (a + b) * (c / d) is ", e)
```

```
e = a + (b * c) / d; # _____ = _____
print ("Value of a + (b * c) / d is ", e)
```

# Data Types - Identifiers and Keywords

## ► Rules for writing identifiers

1. Identifiers can be a combination of

- Lowercase (a to z)
- Uppercase (A to Z)
- Digits (0 to 9)
- An underscore `_`

► Examples: `myClass`, `var_1`, `print_this_to_screen`.

2. An identifier **cannot start with a digit**.

Example: `1variable` is invalid, but `variable1` is a valid

2. Keywords cannot be used as identifiers.

3. We cannot use special symbols like `!`, `@`, `#`, `$`, `%` etc. in our identifier.

4. An identifier can be of any length.

```
Command Prompt - python
>>> global = 1
 File "<stdin>", line 1
 global = 1
 ^
SyntaxError: invalid syntax
>>> global0 = 1
>>> global0
1
>>> a@ = 0
 File "<stdin>", line 1
 a@ = 0
 ^
SyntaxError: invalid syntax
>>> a_ = 0
>>> a_
0
>>>
```

## Data Types - Identifiers and Keywords

### ► Things to Remember

- ❖ Python is a **case-sensitive** language. This means, Variable and variable are not the same.
- ❖ Always **give the identifiers a name that makes sense**. While `c = 10` is a valid name, writing `count = 10` would make more sense, and it would be easier to figure out what it represents when you look at your code after a long gap.
- ❖ Multiple words can be separated using an underscore, like `this_is_a_long_variable`.

## Data Types - Identifiers and Keywords

- ▶ **Python Keywords** : Keywords are the reserved words in Python.
  1. We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
  2. There are **35 keywords in Python 3.7**. This number can vary slightly over the course of time.
  3. In Python, keywords are **case sensitive**.
  4. All the keywords except **True, False and None** are in **lowercase** and they must be written as they are. The list of all the keywords is given below.

## Data Types - Identifiers and Keywords

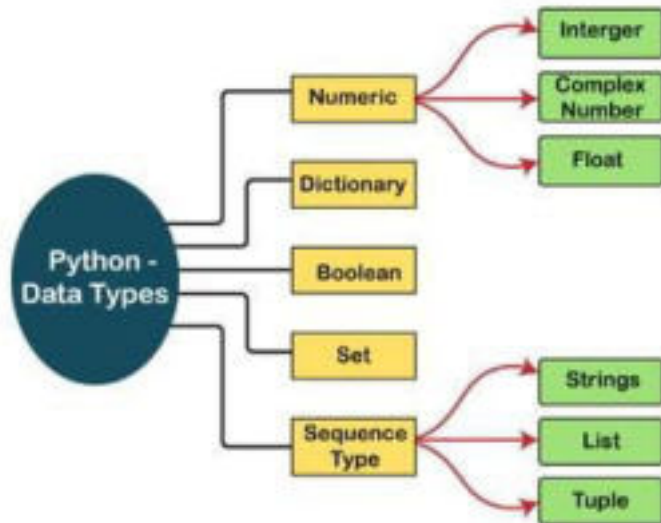
### ► Keywords in python

|              |          |         |          |        |
|--------------|----------|---------|----------|--------|
| <b>False</b> | await    | else    | import   | pass   |
| <b>None</b>  | break    | except  | in       | raise  |
| <b>True</b>  | class    | finally | is       | return |
| and          | continue | for     | lambda   | try    |
| as           | def      | from    | nonlocal | while  |
| assert       | del      | global  | not      | with   |
| async        | elif     | if      | or       | yield  |

# Data Types



# Data Types



# Data Types

## Integers

```
Command Prompt - python
C:\Users\Sri>python
ActivePython 3.6.0.3600 (ActiveState Software Inc.) based on
Python 3.6.0 (default, Jan 23 2017, 20:01:14) [MSC v.1900 64 bit (AMD
64)] on win32
Type "help", "copyright", "credits" or "license" for more information
>>> print(123123123123123123123123123123123123123123123123123123123 + 1)
123123123123123123123123123123123123123123123123123123124
>>>
```



# Data Types

## Integers

- ▶ Python interprets a sequence of decimal digits without any prefix to be a decimal number.

| Prefix                           | Interpretation | Base |
|----------------------------------|----------------|------|
| 0b (zero + lowercase letter 'b') | Binary         | 2    |
| 0B (zero + uppercase letter 'B') |                |      |
| 0o (zero + lowercase letter 'o') | Octal          | 8    |
| 0O (zero + uppercase letter 'O') |                |      |
| 0x (zero + lowercase letter 'x') | Hexadecimal    | 16   |
| 0X (zero + uppercase letter 'X') |                |      |

```
Command Prompt - python
48
>>> print(10)
10
>>> print(0b10)
2
>>> print(0o10)
8
>>> print(0x10)
16
>>>
```

# Data Types

## Floating point types

- ▶ The float type in Python designates a floating-point number. float values are specified with a decimal point.
- ▶ Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation:

## Complex Numbers

- ▶ Complex numbers are specified as `<real part>+<imaginary part>j`.  
Example: `2+3j`

```
Command Prompt - python
>>> type(42)
<class 'int'>
>>> type(4.2)
<class 'float'>
>>> 4.
4.0
>>> .2
0.2
>>> 4.2e-4
0.00042
>>> type(2+3j)
<class 'complex'>
>>>
```

# Data Types

## Strings

- ▶ Strings are sequences of character data.
  - ▶ 'within single quotes'
  - ▶ "within double quotes"
- ▶ A string in Python can contain as many characters as you wish. The only limit is your machine's memory resources. A string can also be empty:
- ▶ What if you want to include a quote character as part of the string itself?
- ▶ Use escape \ sequence.
- ▶ Ex: `print("It's my life ")` → It's my life
- ▶ Ex: `print("The \"Honey Cake\" is delicious")` → "Honey Cake" is delicious

# Data Types

## Strings

```
Command Prompt - python
Python 3.6.0 (default, Jan 23 2017, 20:01:14) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more informa
tion.
>>> print("It\'s my life ")
It's my life
>>> print("The \"Honey Cake\" is delicious")
The "Honey Cake" is delicious
>>> print("a \
... is \
... an alphabet")
a is an alphabet
>>> print("You gave me some\tspace")
You gave me some space
>>> print("First line\nSecond line\nThird line")
First line
Second line
Third line
>>>
```

# Data Types

## Strings

| Escape Sequence | "Escaped" Interpretation                          |
|-----------------|---------------------------------------------------|
| \a              | ASCII Bell (BEL) character                        |
| \b              | ASCII Backspace (BS) character                    |
| \f              | ASCII Formfeed (FF) character                     |
| \n              | ASCII Linefeed (LF) character                     |
| \N{name}        | Character from Unicode database with given <name> |
| \r              | ASCII Carriage Return (CR) character              |
| \t              | ASCII Horizontal Tab (TAB) character              |
| \uxxxx          | Unicode character with 16-bit hex value xxxxx     |
| \Uxxxxxxxx      | Unicode character with 32-bit hex value xxxxxxxx  |
| \v              | ASCII Vertical Tab (VT) character                 |
| \ooo            | Character with octal value ooo                    |
| \hhh            | Character with hex value hh                       |

# Data Types

## Raw Strings

## Triple Quotes

```
Command Prompt - python
C:\Users\Sri>python
ActivePython 3.6.0.3600 (ActiveState Software Inc.) based on
Python 3.6.0 (default, Jan 23 2017, 20:01:14) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print(R"It\'s my Life")
It\'s my Life
>>>
>>> print('''This string has a single (') and a double (") quote.''')
This string has a single (') and a double (") quote.
>>>
>>> print("""This is a
... string that spans
... across several lines""")
This is a
string that spans
across several lines
>>>
```

# Data Types

## Boolean

- ▶ True or False

# Data Types

## Assignment

- ▶ Write a python program using only 4 print statements to get the result as shown below
- ▶ Expected output

It's my Life

The "Birthday cake" was delicious

Give me my       space

My\_Name       Register\_No

PJC Final Year

JNC



The screenshot shows a dark-themed code editor window. At the top right, there are three buttons: a light blue button with a plus sign and the text 'new repl', a grey button with the text 'talk', and a blue button with the text 'sign up'. The main area of the editor is black and contains the following text, which matches the 'Expected output' listed on the left: 'It's my Life', 'The "Birthday cake" was delicious', 'Give me my       space', 'My\_Name       Register\_No', 'PJC Final Year', and 'JNC'. At the bottom left of the editor, there is a prompt character '>' followed by a vertical bar cursor '|'. In the bottom right corner of the editor, there is a small number '11'.

```
It's my Life
The "Birthday cake" was delicious
Give me my space
My_Name Register_No
PJC Final Year
JNC
> |
```



# Data Types

## Assignment

- ▶ Write a python program using only 4 print statements to get the result as shown below
- ▶ Program

```
print("It's my Life")
```

```
print("The \"Birthday cake\" was delicious")
```

```
print("Give me my \t\t space")
```

```
print("My_Name \t \t 18PJC001 \nPJC Final Year \nJNC")
```

## Content: Unit I

### 3. Collection data types:

- ▶ Sequence types.
- ▶ Set types
- ▶ Mapping types
- ▶ Iterating and Copying collections.

## Collection of data types

- ▶ Group of items → Collections
- ▶ Collections has four types
  - ▶ tuple
  - ▶ list
  - ▶ Set
  - ▶ Dictionary
- ▶ Python tuples and lists can be used to hold any number of data items of any data types.
- ▶ **Tuples are immutable**, so once they are created we cannot change them.
- ▶ **Lists are mutable**, so we can easily insert items and remove items whenever we want.

## Collection of data types



```
Command Prompt - system
information.
>>> t=(1,2,3,4,5)
>>> t
(1, 2, 3, 4, 5)
>>> del t[4]
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
>>> t.append(6)
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>>
```

```
Command Prompt - system
>>> print("list demo")
list demo
>>> l=[1,2,3,4,5]
>>> l
[1, 2, 3, 4, 5]
>>> del l[4]
>>> l
[1, 2, 3, 4]
>>> l.append(5)
>>> l
[1, 2, 3, 4, 5]
>>>
```

## Collection of data types

- Some of the operations common for both tuple and list are as follows –

| S.No. | Operation/Functions & Description                                                 |
|-------|-----------------------------------------------------------------------------------|
| 1     | <b>x in seq</b><br>True, when x is found in the sequence seq, otherwise False     |
| 2     | <b>x not in seq</b><br>False, when x is found in the sequence seq, otherwise True |
| 3     | <b>x + y</b><br>Concatenate two sequences x and y                                 |
| 4     | <b>x * n or n * x</b><br>Replicate sequence x with itself n times                 |
| 5     | <b>seq[i]</b><br>ith item of the sequence.                                        |
| 6     | <b>seq[i:j]</b><br>Slice sequence from index i to j                               |

## Collection of data types

- Some of the operations common for both tuple and list are as follows –

| S.No. | Operation/Functions & Description                                     |
|-------|-----------------------------------------------------------------------|
| 7     | <b>seq[i:j:k]</b><br>Slice sequence from index i to j with step k     |
| 8     | <b>len(seq)</b><br>Length or number of elements in the sequence       |
| 9     | <b>min(seq)</b><br>Minimum element in the sequence                    |
| 10    | <b>max(seq)</b><br>Maximum element in the sequence                    |
| 11    | <b>seq.count(x)</b><br>Count total number of elements in the sequence |

## Collection of data types

- Some of the operations applicable for list are as follows –

| S.No. | Operation/Functions & Description                                                                                |
|-------|------------------------------------------------------------------------------------------------------------------|
| 1     | <b>seq.append(x)</b><br>Add x at the end of the sequence                                                         |
| 2     | <b>seq.clear()</b><br>Clear the contents of the sequence                                                         |
| 3     | <b>seq.insert(i, x)</b><br>Insert x at the position i                                                            |
| 4     | <b>seq.pop([i])</b><br>Return the item at position i, and also remove it from sequence. Default is last element. |
| 5     | <b>seq.remove(x)</b><br>Remove first occurrence of item x                                                        |
| 6     | <b>seq.reverse()</b><br>Reverse the list                                                                         |

## Collection of data types

### Tuple in Python

- ▶ A tuple is a collection of objects which ordered and immutable.
- ▶ Tuples are sequences, just like lists.
- ▶ The differences between tuples and lists are, the **tuples cannot be changed** unlike lists and **tuples use parentheses**, whereas **lists use square brackets**.



## Collection of data types

- ▶ Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```
tuple1 = ('physics', 'chemistry', 1997, 2000)
```

```
tuple2 = (1, 2, 3, 4, 5)
```

```
tuple3 = "a", "b", "c", "d"
```

- ▶ The empty tuple is written as two parentheses containing nothing –

```
tuple0=()
```

- ▶ To write a tuple containing a single value you have to include a comma, even though there is only one value –

```
tup1 = (50,)
```

## Collection of data types

### Accessing Values in Tuples

- ▶ To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index.

```
tuple1 = ('physics', 'chemistry', 1997, 2000)
```

```
tuple2 = (1, 2, 3, 4, 5)
```

```
tuple3 = "a", "b", "c", "d", "e"
```

```
print(tuple1[1:2]) # ('chemistry',)
```

```
print(tuple2[1:3]) # (2,3)
```

```
print(tuple3[1:4]) # ('b','c','d')
```

## Collection of data types

### Updating Tuples

- ▶ Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates

```
tuple1 = ('physics', 'chemistry', 1997, 2000)
```

```
tuple2 = (1, 2, 3, 4, 5)
```

```
tuple3 = "a", "b", "c", "d", "e"
```

```
tuple1.append(2020)
```

```
tuple1.pop()
```

```
tuple1.remove(1997)
```

```
tuple4=tuple2+tuple3[2:3] # (1,2,3,4,5, 'c')
```

## Collection of data types

### Delete Tuple Elements

- ▶ Removing individual tuple elements is not possible.

```
tuple1 = ('physics', 'chemistry', 1997, 2000)
```

```
tuple2 = (1, 2, 3, 4, 5)
```

```
tuple3 = "a", "b", "c", "d", "e"
```

```
tuple4=tuple2+tuple3[2:3] # (1,2,3,4,5, 'c')
```

```
del tuple4
```

```
print(tuple4) #NameError: name 'tuple4' is not defined
```

## Collection of data types – Mapping Types

### Dictionaries

- ▶ A dict is an **unordered collection of zero or more key–value pairs** whose keys are object references that refer to hashable objects, and whose values are object references referring to objects of any type.
- ▶ **Dictionaries are mutable**, so we can easily add or remove items, but since they are **unordered** they have **no notion of index position** and so **cannot be sliced** or strided.

## Collection of data types – Mapping Types

### Defining a Dictionary

- ▶ Dictionaries are Python's implementation of a data structure that is more generally known as an associative array. A dictionary consists of a collection of key-value pairs.
- ▶ Each key-value pair maps the key to its associated value.
- ▶ We can define a dictionary by enclosing a comma-separated list of key-value pairs in curly braces (`{ }`).
- ▶ A colon (`:`) separates each key from its associated value:

## Collection of data types – Mapping Types

### Defining a Dictionary

```
myDictionary = {
 <key>: <value>,
 <key>: <value>,
 <key>: <value>
}
```

## Collection of data types – Mapping Types

### Defining a Dictionary

- ▶ We can also construct a dictionary with the built-in `dict()` function. The argument to `dict()` should be a sequence of key-value pairs. A list of tuples works well for this:

```
myDictionary = dict([
 (<key>, <value>),
 (<key>, <value>),
 .
 .
 .
 (<key>, <value>)
])
```



## Collection of data types – Mapping Types

### Accessing Dictionary Values

- ▶ You don't get them by index, then how do you get them?
- ▶ A **value** is retrieved from a dictionary by specifying its corresponding **key in square brackets** ([ ]):

`myDictionary['existing_key']`

- ▶ If you refer to a key that is not in the dictionary, Python raises an exception:

`myDictionary['non_existing_key'] #ERROR`

## Collection of data types – Mapping Types

### Accessing Dictionary Values

- ▶ **Adding** an entry to an existing dictionary is simply a matter of **assigning a new key and value**:

```
myDictionary['new_key']='value'
```

- ▶ If you want to **update** an entry, you can just **assign a new value to an existing key**:

```
myDictionary['existing_key']='new_value'
```

- ▶ To **delete** an entry, use the **del** statement, **specifying the key to delete**:

```
del myDictionary['existing_key']
```

## Collection of data types – Mapping Types

| Syntax                               | Description                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>d.clear()</code>               | Removes all items from dict <code>d</code>                                                                                                                                                                                                                                                                                                                   |
| <code>d.copy()</code>                | Returns a shallow copy of dict <code>d</code>                                                                                                                                                                                                                                                                                                                |
| <code>d.fromkeys(<br/>s, v)</code>   | Returns a dict whose keys are the items in sequence <code>s</code> and whose values are <code>None</code> or <code>v</code> if <code>v</code> is given                                                                                                                                                                                                       |
| <code>d.get(k)</code>                | Returns key <code>k</code> 's associated value, or <code>None</code> if <code>k</code> isn't in dict <code>d</code>                                                                                                                                                                                                                                          |
| <code>d.get(k, v)</code>             | Returns key <code>k</code> 's associated value, or <code>v</code> if <code>k</code> isn't in dict <code>d</code>                                                                                                                                                                                                                                             |
| <code>d.items()</code>               | Returns a view* of all the (key, value) pairs in dict <code>d</code>                                                                                                                                                                                                                                                                                         |
| <code>d.keys()</code>                | Returns a view* of all the keys in dict <code>d</code>                                                                                                                                                                                                                                                                                                       |
| <code>d.pop(k)</code>                | Returns key <code>k</code> 's associated value and removes the item whose key is <code>k</code> , or raises a <code>KeyError</code> exception if <code>k</code> isn't in <code>d</code>                                                                                                                                                                      |
| <code>d.pop(k, v)</code>             | Returns key <code>k</code> 's associated value and removes the item whose key is <code>k</code> , or returns <code>v</code> if <code>k</code> isn't in dict <code>d</code>                                                                                                                                                                                   |
| <code>d.popitem()</code>             | Returns and removes an arbitrary (key, value) pair from dict <code>d</code> , or raises a <code>KeyError</code> exception if <code>d</code> is empty                                                                                                                                                                                                         |
| <code>d.setdefault(<br/>k, v)</code> | The same as the <code>dict.get()</code> method, except that if the key is not in dict <code>d</code> , a new item is inserted with the key <code>k</code> , and with a value of <code>None</code> or of <code>v</code> if <code>v</code> is given                                                                                                            |
| <code>d.update(a)</code>             | Adds every (key, value) pair from <code>a</code> that isn't in dict <code>d</code> to <code>d</code> , and for every key that is in both <code>d</code> and <code>a</code> , replaces the corresponding value in <code>d</code> with the one in <code>a</code> — <code>a</code> can be a dictionary, an iterable of (key, value) pairs, or keyword arguments |
| <code>d.values()</code>              | Returns a view* of all the values in dict <code>d</code>                                                                                                                                                                                                                                                                                                     |

## Collection of data types – Iterating and Copying Collections

- ▶ Once we have collections of data items, it is natural to want to **iterate** over all the items they contain.
- ▶ In this section's first subsection we will **introduce some of Python's iterators and the operators and functions that involve iterators.**
- ▶ Another common requirement is to **copy a collection.** There are some subtleties involved here because of Python's use of object references (for the sake of efficiency), so in this section's second subsection, we will examine how to copy collections and get the behavior we want.

## Collection of data types – Iterating and Copying Collections

### **Iterators and Iterable Operations and Functions**

- ▶ An iterable data type is one that can return each of its items one at a time.
- ▶ Any object that has an `__iter__()` method, or any sequence (i.e., an object that has a `__getitem__()` method taking integer arguments starting from 0) is an iterable and can provide an iterator.
- ▶ An iterator is an object that provides a `__next__()` method which returns each successive item in turn, and raises a `StopIteration` exception when there are no more items.

**Table 3.4** Common Iterable Operators and Functions

| Syntax                                | Description                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>s + t</code>                    | Returns a sequence that is the concatenation of sequences <code>s</code> and <code>t</code>                                                                                                                                                                                                                                                                                |
| <code>s * n</code>                    | Returns a sequence that is <code>int n</code> concatenations of sequence <code>s</code>                                                                                                                                                                                                                                                                                    |
| <code>x in i</code>                   | Returns True if item <code>x</code> is in iterable <code>i</code> ; use <code>not in</code> to reverse the test                                                                                                                                                                                                                                                            |
| <code>all(i)</code>                   | Returns True if every item in iterable <code>i</code> evaluates to True                                                                                                                                                                                                                                                                                                    |
| <code>any(i)</code>                   | Returns True if any item in iterable <code>i</code> evaluates to True                                                                                                                                                                                                                                                                                                      |
| <code>enumerate(i, start)</code>      | Normally used in <code>for ... in</code> loops to provide a sequence of <code>(index, item)</code> tuples with indexes starting at 0 or <code>start</code> ; see text                                                                                                                                                                                                      |
| <code>len(x)</code>                   | Returns the "length" of <code>x</code> . If <code>x</code> is a collection it is the number of items; if <code>x</code> is a string it is the number of characters.                                                                                                                                                                                                        |
| <code>max(i, key)</code>              | Returns the biggest item in iterable <code>i</code> or the item with the biggest <code>key(item)</code> value if a <code>key</code> function is given                                                                                                                                                                                                                      |
| <code>min(i, key)</code>              | Returns the smallest item in iterable <code>i</code> or the item with the smallest <code>key(item)</code> value if a <code>key</code> function is given                                                                                                                                                                                                                    |
| <code>range(start, stop, step)</code> | Returns an integer iterator. With one argument ( <code>stop</code> ), the iterator goes from 0 to <code>stop - 1</code> ; with two arguments ( <code>start</code> , <code>stop</code> ) the iterator goes from <code>start</code> to <code>stop - 1</code> ; with three arguments it goes from <code>start</code> to <code>stop - 1</code> in steps of <code>step</code> . |
| <code>reversed(i)</code>              | Returns an iterator that returns the items from iterator <code>i</code> in reverse order                                                                                                                                                                                                                                                                                   |
| <code>sorted(i, key, reverse)</code>  | Returns a list of the items from iterator <code>i</code> in sorted order; <code>key</code> is used to provide DSU (Decorate, Sort, Undecorate) sorting. If <code>reverse</code> is True the sorting is done in reverse order.                                                                                                                                              |
| <code>sum(i, start)</code>            | Returns the sum of the items in iterable <code>i</code> plus <code>start</code> (which defaults to 0); <code>i</code> may not contain strings                                                                                                                                                                                                                              |
| <code>zip(i1, ..., iN)</code>         | Returns an iterator of tuples using the iterators <code>i1</code> to <code>iN</code> ; see text                                                                                                                                                                                                                                                                            |



## Collection of data types – Iterating and Copying Collections

### Copy an Object in Python

- ▶ In Python, we use = operator to create a copy of an object. You may think that this creates a new object; it doesn't. It only creates a new variable that shares the reference of the original object.
- ▶ Let's take an example where we create a list named **old\_list** and pass an object reference to **new\_list** using = operator.

**Thank You**



# Python Control Structures

# Python If-else statements

- Decision making is the most important aspect of almost all the programming languages. As the name implies, decision making allows us to run a particular block of code for a particular decision. Here, the decisions are made on the validity of the particular conditions. Condition checking is the backbone of decision making.
- Indentation  
Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

| Statement           | Description                                                                                                                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| If Statement        | The if statement is used to test a specific condition. If the condition is true, a block of code (if-block) will be executed.                                                                                                                                      |
| If - else Statement | The if-else statement is similar to if statement except the fact that, it also provides the block of the code for the false case of the condition to be checked. If the condition provided in the if statement is false, then the else statement will be executed. |
| Nested if Statement | Nested if statements enable us to use if ? else statement inside an outer if statement.                                                                                                                                                                            |

## The if statement

The if statement is used to test a particular condition and if the condition is true, it executes a block of code known as if-block. The condition of if statement can be any valid logical expression which can be either evaluated to true or false.

## Syntax

```
if expression:
 statement
```

## Example 1

```
num = int(input("enter the number?"))
if num%2 == 0:
 print("Number is even")
```

## Output:

```
enter the number?
10 Number is even
```

### Program to print the largest of the three numbers

```
a = int(input("Enter a? "));
b = int(input("Enter b? "));
c = int(input("Enter c? "));
if a>b and a>c:
 print("a is largest");
if b>a and b>c:
 print("b is largest");
if c>a and c>b:
 print("c is largest");
```

### Output:

Enter a? 100 Enter b? 120 Enter c? 130 c is largest

# Using Else

Program to check whether a person is eligible to vote or not.

```
age = int (input("Enter your age? "))
if age>=18:
 print("You are eligible to vote !!");
else:
 print("Sorry! you have to wait !!");
```

## Output:

- Enter your age? 90 You are eligible to vote !!

# The elif statement

- The elif statement enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them. We can have any number of elif statements in our program depending upon our need. However, using elif is optional.

The syntax of the elif statement is given below.

**if** expression 1:

    # block of statements

**elif** expression 2:

    # block of statements

**elif** expression 3:

    # block of statements

**else:**

    # block of statements

## Example

```
marks = int(input("Enter the marks? "))
```

```
if marks > 85 and marks <= 100:
```

```
 print("Congrats ! you scored grade A ...")
```

```
elif marks > 60 and marks <= 85:
```

```
 print("You scored grade B + ...")
```

```
elif marks > 40 and marks <= 60:
```

```
 print("You scored grade B ...")
```

```
elif (marks > 30 and marks <= 40):
```

```
 print("You scored grade C ...")
```

```
else:
```

```
 print("Sorry you are fail ?")
```



# Python Loops

- The flow of the programs written in any programming language is sequential by default. Sometimes we may need to alter the flow of the program. The execution of a specific code may need to be repeated several numbers of times.

## Why we use loops in python?

- The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times.

**For example**, if we need to print the first 10 natural numbers then, instead of using the print statement 10 times, we can print inside a loop which runs up to 10 iterations.

## Advantages of loops

There are the following advantages of loops in Python.

It provides code re-usability.

Using loops, we do not need to write the same code again and again.

Using loops, we can traverse over the elements of data structures (array or linked lists).

# Loops

## For Loop

The **for loop in Python** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like list, tuple, or dictionary.

The syntax of for loop in python is given below.

```
for iterating_var in sequence:
 statement(s)
```

## Iterating string using for loop

```
str = "Python"
for i in str:
 print(i)
```

## Output

P  
Y  
T  
H  
O  
N

Program to print the sum of the given list.

```
list = [10,30,23,43,65,12]
```

```
sum = 0
```

```
for i in list:
```

```
 sum = sum+i
```

```
print("The sum is:",sum)
```

## Output:

The sum is: 183

# For loop Using range() function

## The range() function

The **range()** function is used to generate the sequence of the numbers. If we pass the `range(10)`, it will generate the numbers from 0 to 9. The syntax of the `range()` function is given below.

### Syntax:

`range(start,stop,step size)`

- The start represents the beginning of the iteration.
- The stop represents that the loop will iterate till stop-1. The **range(1,5)** will generate numbers 1 to 4 iterations. It is optional.
- The step size is used to skip the specific numbers from the iteration. It is optional to use. By default, the step size is 1. It is optional.

**Program to print numbers in sequence.**

```
for i in range(10):
 print(i,end = ' ')
```

**Output:**

0 1 2 3 4 5 6 7 8 9

What does end do:

The print() function inserts a new line at the **end**, by default. In **Python 2**, it can be suppressed by putting ',' at the **end**. In **Python 3**, "**end = ' '**" appends space instead of newline.

**Program to print table of given number.**

```
n = int(input("Enter the number "))
for i in range(1,11):
 c = n*i
 print(n,"*",i,"=",c)
```

**Output:**

Enter the number 10

10 \* 1 = 10

10 \* 2 = 20

10 \* 3 = 30

10 \* 4 = 40

10 \* 5 = 50

10 \* 6 = 60

10 \* 7 = 70

10 \* 8 = 80

10 \* 9 = 90

10 \* 10 = 100

LOOPING STATEMENTS: WHILE,  
FOR, NESTED LOOP  
CONTROL STATEMENTS: BREAK,  
CONTINUE, PASS;



LOOPING STATEMENTS: WHILE, FOR,  
NESTED LOOP



- WHILE LOOP
- FOR LOOP
- NESTED LOOP



## WHILE LOOP

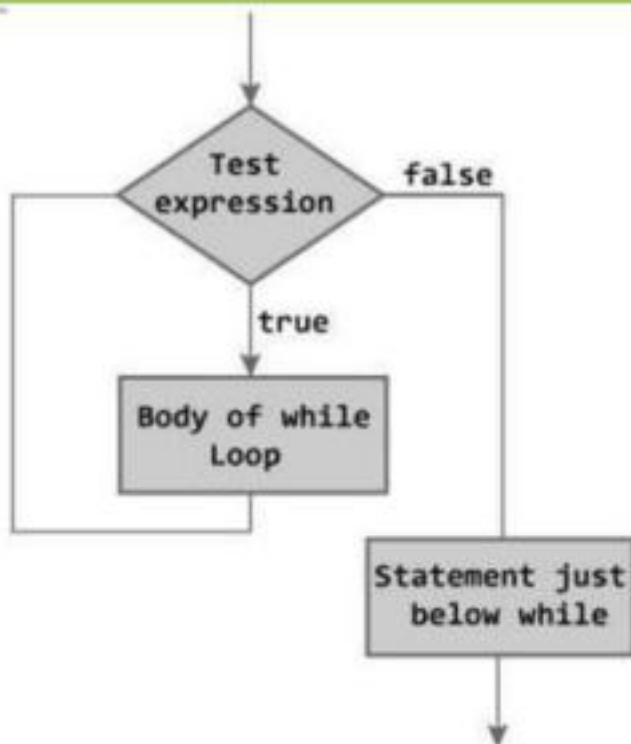
- While loop statement in Python is used to repeatedly executes set of statement as long as a given condition is true.
- In while loop, test expression is checked first.

- The body of the loop is entered only if the test\_expression is True. After one iteration, the test expression is checked again.
- This process continues until the test\_expression evaluates to False.
- In Python, the body of the while loop is determined through indentation.
- The statements inside the while starts with indentation and the first unindented line marks the end.

## Syntax

```
initial value
while(condition):
 body of while loop
 increment
```

# Flowchart



## Examples:

1. Program to find sum of n numbers:
2. Program to find factorial of a number
3. Program to find sum of digits of a number:
4. Program to Reverse the given number:
5. Program to find number is Armstrong number or not
6. Program to check the number is palindrome or not

|                                                                                                                  |                                 |
|------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <pre>n=eval(input("enter n")) i=1 sum=0 while(i&lt;=n):     sum=sum+i     i=i+1 print(sum)</pre>                 | <pre>enter n 10 55</pre>        |
| <b>Factorial of a numbers:</b>                                                                                   | <b>output</b>                   |
| <pre>n=int(input("enter n")) i=1 fact=1 while(i&lt;=n):     fact=fact*i     i=i+1 print(fact)</pre>              | <pre>enter n 5 120</pre>        |
| <b>Sum of digits of a number:</b>                                                                                | <b>output</b>                   |
| <pre>n= int (input("enter a number")) sum=0 while(n&gt;0):     a=n%10     sum=sum+a     n=n//10 print(sum)</pre> | <pre>enter a number 123 6</pre> |

| Reverse the given number:                                                                                                                                                                                                                    | output                                                               |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|--|
| <pre> n= int (input("enter a number")) sum=0 while(n&gt;0):     a=n%10     sum=sum*10+a     n=n//10 print(sum) </pre>                                                                                                                        | <pre> enter a number 123 321 </pre>                                  |  |
| Armstrong number or not                                                                                                                                                                                                                      | output                                                               |  |
| <pre> n=eval(input("enter a number")) org=n sum=0 while(n&gt;0):     a=n%10     sum=sum+a*a*a     n=n//10 if(sum==org):     print("The given number is Armstrong number") else:     print("The given number is not Armstrong number") </pre> | <pre> enter a number 153 The given number is Armstrong number </pre> |  |

| <b>Palindrome or not</b>                                                                                                                                                                                       | <b>output</b>                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
| <pre>n=eval(input("enter a number")) org=n sum=0 while(n&gt;0):     a=n%10     sum=sum*10+a     n=n //10 if(sum==org): print("The given no is palindrome") else: print("The given no is not palindrome")</pre> | <pre>enter a number121 The given no is palindrome</pre> |



# FOR LOOP

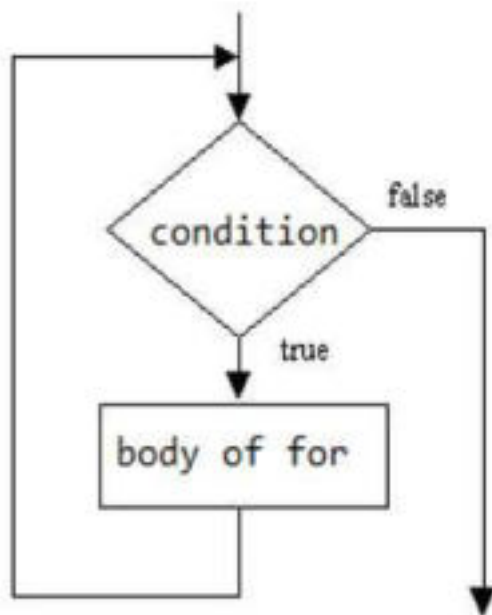
## **for in range:**

- We can generate a sequence of numbers using range() function. range(10) will generate numbers from 0 to 9 (10 numbers).
- In range function have to define the start, stop and step size
- as range(start,stop,step size). step size defaults to 1 if not provided.

## Syntax:

```
for i in range(start,stop,steps):
 body of for loop
```

# Flowchart



## For in sequence

- The for loop in Python is used to iterate over a sequence ([list](#), [tuple](#), [string](#)).
- Iterating over a sequence is called traversal. Loop continues until we reach the last element in the sequence.
- The body of for loop is separated from the rest of the code using indentation

```
for i in sequence:
 print(i)
```

# Sequence can be a list, strings or tuples:

| S.No | Sequences                 | Example                               | Output                           |
|------|---------------------------|---------------------------------------|----------------------------------|
| 1.   | <i>For loop in string</i> | <i>for i in "Ramu":<br/>print(i)</i>  | <i>R<br/>A<br/>M<br/>U</i>       |
| 2.   | <i>For loop in list</i>   | <i>for i in [2,3,5,6,9]: print(i)</i> | <i>2<br/>3<br/>5<br/>6<br/>9</i> |
| 3.   | <i>For loop in tuple</i>  | <i>for i in (2,3,1): print(i)</i>     | <i>2<br/>3<br/>1</i>             |

## Examples:

1. print nos divisible by 5 not by 10
2. Program to print fibonacci series
3. Program to find factors of a given number
4. check the given number is perfect number or not
5. check the no is prime or not
6. Print first n prime numbers
7. Program to print prime numbers in range

| Fibonacci series                                                                                                                                                      | output                                                                    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| <pre> a=0 b=1 n=int(input("Enter the number of terms: ")) print("Fibonacci Series: ") print(a,b) for i in range(1,n,1):     c=a+b     print(c)     a=b     b=c </pre> | <pre> Enter the number of terms: 6 Fibonacci Series: 0 1 1 2 3 5 8 </pre> |
| find factors of a number                                                                                                                                              | Output                                                                    |
| <pre> n=int(input("enter a number:")) for i in range(1,n+1,1):     if(n%i==0):         print(i) </pre>                                                                | <pre> enter a number:10 1 2 5 10 </pre>                                   |

| Check The No Is Prime Or Not                                                                                                                                                                                                          | Output                                                     |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <pre> n=int(input("enter a number")) for i in range(2,n):     if(n%i==0):         print("The num is not a prime") break     else:         print("The num is a prime number.") </pre>                                                  | <pre> enter a no:7 The num is a prime number. </pre>       |
| Check A Number Is Perfect Number Or Not                                                                                                                                                                                               | Output                                                     |
| <pre> n=int(input("enter a number:")) sum=0 for i in range(1,n,1):     if(n%i==0):         sum=sum+i if(sum==n):             print("the number is perfect number")     else:         print("the number is not perfect number") </pre> | <pre> enter a number:6 the number is perfect number </pre> |

# NESTED LOOPS

- Python programming language allows using one loop inside another loop.
- We can use one or more loop inside any another while, for or do...while loop.

## Syntax - nested for loop

```
for iterating_var in sequence:
 for iterating_var in sequence:
 statements(s)
statements(s)
```

## Syntax - nested while loop

```
session:
 while expression:
 statement(s)
statement(s)
```



**Example**

```
Find the prime numbers from 2 to 100:
```

```
i = 2
```

```
while(i < 100):
```

```
 j = 2
```

```
 while(j <= (i/j)):
```

```
 if not(i%j): break
```

```
 j = j + 1
```

```
 if (j > i/j) : print i, " is prime"
```

```
 i = i + 1
```

```
print "Good bye!"
```

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime
Good bye!
```

## ELSE STATEMENT IN LOOPS

### else in for loop:

- If else statement is used in for loop, the else statement is executed when the loop has reached the limit.
- The statements inside for loop and statements inside else will also execute

| example                                                                                               | output                                          |  |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------|--|
| <pre>for i in range(1,6):<br/>    print(i)<br/>else:<br/>    print("the number greater than 6")</pre> | 1<br>2<br>3<br>4<br>5 the number greater than 6 |  |

### else in while loop:

- If else statement is used within while loop , the else part will be executed when the condition become false.
- The statements inside for loop and statements inside else will also execute

| Program                                                                                                       | output                                         |
|---------------------------------------------------------------------------------------------------------------|------------------------------------------------|
| <pre>i=1 while(i&lt;=5 ):     print     (i)     i=i+     1 else:     print("the number greater than 5")</pre> | <pre>1 2 3 4 5 the number greater than 5</pre> |

# CONTROL STATEMENTS

## BREAK

- Break statements can alter the flow of a loop.
- It terminates the current loop and executes the remaining statement outside the loop.
- If the loop has else statement, that will also gets terminated and come out of the loop completely.

# Syntax:

```
while (test Expression):
```

```
 // codes
```

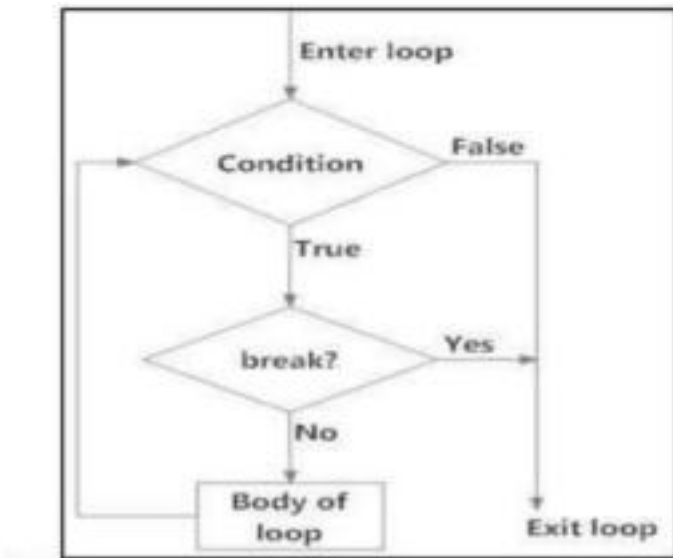
```
 if (condition for break):
```

```
 break
```

```
 // codes
```



# Flowchart



| example                                                                               | Output      |
|---------------------------------------------------------------------------------------|-------------|
| <pre>for i in "welcome":<br/>    if(i=="c"):<br/>        break<br/>    print(i)</pre> | w<br>e<br>l |



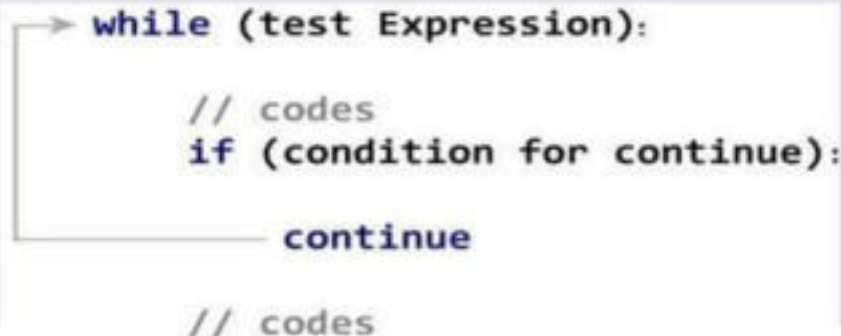
# CONTINUE

- It terminates the current iteration and transfer the control to the next iteration in the loop.

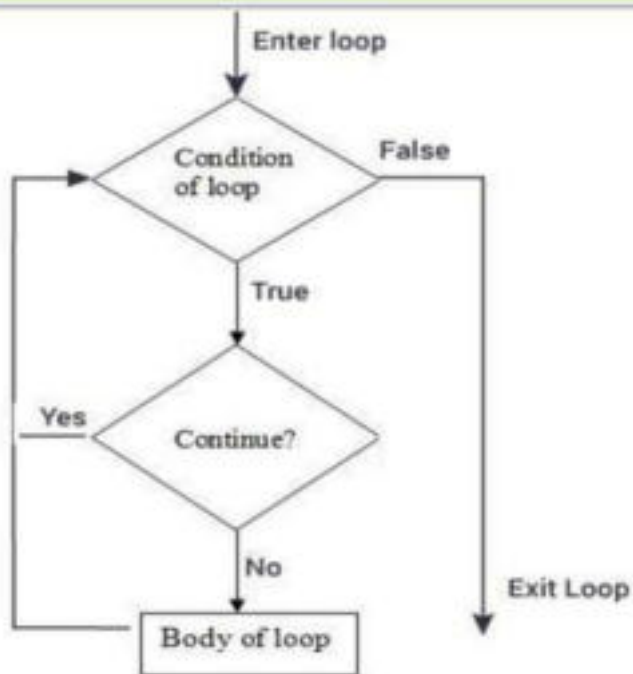
## Syntax:

Continue

```
→ while (test Expression):
 // codes
 if (condition for continue):
 continue
 // codes
```



# Flowchart



| Example:                                                                         | Output                     |
|----------------------------------------------------------------------------------|----------------------------|
| <pre>for i in     "welcome":     if(i=="c"):         continue     print(i)</pre> | w<br>e<br>l<br>o<br>m<br>e |

# PASS

- It is used when a statement is required syntactically but you don't want any code to execute.
- It is a null statement, nothing happens when it is executed.

## Syntax:

pass

break

**sample**

```
for i in "welcome":
 if (i == "c"):
 pass
 print(i)
```

**Output**

```
w

e

l

c

o

m
e
```

# DIFFERENCE BETWEEN BREAK AND CONTINUE

| <u>Break</u>                                                                       | <u>Continue</u>                                                                                 |
|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| terminates the current loop and executes the remaining statement outside the loop. | It terminates the current iteration and transfer the control to the next iteration in the loop. |
| <b>syntax:</b><br>break                                                            | <b>syntax:</b><br>continue                                                                      |
| for i in "welcome":<br>if(i=="c"):<br>break<br>print(i)                            | for i in<br>"welcome":<br>if(i=="c"):<br>continue<br>print(i)                                   |
| l                                                                                  | w<br>e<br>l<br>c<br>o<br>m<br>e                                                                 |



python

# STRING MANIPULATION IN PYTHON

PROF POOJA B S

# String Manipulation in Python

- ▶ Getting Length of a string
- ▶ Traversal through a string with a loop
- ▶ String slices
- ▶ Strings are immutable
- ▶ Looping and Counting
- ▶ The in operator



- ▶ String is a sequence of characters.
- ▶ Enclosed within pair of single/double quotes.
- ▶ Character has an index Number.
- ▶ Eg: "Hello World"

|           |   |   |   |   |   |   |   |   |   |   |    |
|-----------|---|---|---|---|---|---|---|---|---|---|----|
| Character | H | e | l | l | o |   | W | o | r | l | D  |
| Index     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

- ▶ Access characters using Index.
- ▶ Eg: 

```
word1='Hello'
word2='Hi'
x=word1[1]
print(x)
y=word2[0]
print(y)
```

- ▶ End of the string can be accessed using Negative Index.
- ▶ Eg: "Hello World"

|                |     |     |    |    |    |    |    |    |    |    |    |
|----------------|-----|-----|----|----|----|----|----|----|----|----|----|
| Character      | H   | e   | l  | l  | o  |    | W  | o  | r  | l  | d  |
| Negative Index | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

- ▶ Eg:  
word="Hello"  
x=word[-1]  
print(x)  
x=word[-5]  
print(x)

## Getting Length of a String using len()

- ▶ Built-in function
- ▶ Obtain length of a string.
- ▶ Eg: word="Python"

```
l=len(word)
```

```
print(l)
```

## Traversal through String with a loop

- ▶ Extracting every character and performing some action is called Traversal.
- ▶ It can be done using while or for loop

## 1. Using for loop

```
s="Python"
for i in s:
 print(i, end='^')
```

## 2. Using while loop

```
s="Python"
i=0
while i<len(s):
 print(s[i],end='^')
```

**#FORWARD TRAVERSAL**

```
s="Python"
i=-1
while i>=-len(s):
 print(s[i],end='^')
```

**#BACKWARD TRAVERSAL**

## String Slices

- ▶ Portion of a string is a string slice.
- ▶ Extract a required number of characters using colon (:)
- ▶ **Syntax:**

**st[i:j:k]**

- ▶ i is the first index or beginning index or start index
- ▶ j is the last index or end index. If j index is not present, means slice should be till the end of the string.
- ▶ k is the **stride**, to indicate no. of steps incremented. Default value is 1.
- ▶ Eg:

```
st="Hello World"
print(st[0:5:1])
```

1. `print (st[0:5])`
2. `print (st[3:8])`
3. `print (st[7:])`
4. `print (st[:])`
5. `print (st[::2])`
6. `print (st[4:4])`
7. `print (st[3:8:2])`
8. `print (st[1:8:3])`
9. `print (st[-4:-1])`
10. `print (st[-1])`
11. `print (st[>-1])`
12. `print (st[:])`
13. `print (st[::-1])`
14. `print (st[::-2])`

## String are immutable

- ▶ Strings are immutable [cannot change]
- ▶ To modify string, create a new string.
- ▶ Eg:

```
1. s="hello world"
 s[3]='t'
```

**#TypeError: 'str' object does not support item assignment**

```
2. s="hello world"
 s1=s[:3]+'t'+s[4:]
 print(s1)
```

**#helto world**

## Looping and Counting

- ▶ Using loops we can count the frequency of character.
- ▶ Eg:

```
w='Book'
count=0
for letter in w:
 if letter=='o':
 count=count+1
print("The Occurences of Character 'o' is %d"%(count))
```



## The in operator

- ▶ Boolean operator takes 2 string operands.
- ▶ Returns True if 1<sup>st</sup> operand appears in 2<sup>nd</sup> Operand. Else False.
- ▶ Eg:

'e' in 'hello'      #True

'x' in 'hello'      #False

'EL' in 'Hello'    #False

## String Comparison

- ▶ Comparison operators like: <, > and == applied to string objects.
- ▶ Result in True or False
- ▶ Comparisons happens using ACSII codes
- ▶ Eg:

```
1. s='hello'
 if s=='hello':
 print('Same') #Same
```

```
2. s='hello'
 if s<='Hello':
 print('Lesser')
 else:
 print('Greater') #Greater
```

- ▶ A-Z 65-90
- ▶ a-z 97-122
- ▶ 0-9 48-57
- ▶ Space 32
- ▶ Enter 13

**UNIT-1**  
**INTRODUCTION TO WEB APPLICATIONS**

The term web application refers to a software system that provides a user interface through a web browser. Examples of web applications include blogs, online shopping, search engines, etc.

Static web pages are stored in the file system of web server usually displays the same information to all visitors. Whereas dynamic pages are constructed by a program that produce the HTML. This type of web application provide individual information to the user and let them personalize the content according to their preferences.

## Client Side Scripting Vs Server Side Scripting

A script is generally a series of program or instruction, which has to be executed on other program or application. The scripts can be written in two forms, at the server end (back end) or at the client end (server end).

Basis for comparison

|                    |                                                                         |                                                                |
|--------------------|-------------------------------------------------------------------------|----------------------------------------------------------------|
| Basic              | Works in the back end which could not be visible at the client end.     | Works at the front end and script are visible among the users. |
| Processing         | Requires server interaction.                                            | Does not need interaction with the server.                     |
| Languages involved | PHP, ASP.net, Ruby on Rails, ColdFusion, Python, etcetera.              | HTML, CSS, JavaScript, etc.                                    |
| Affect             | Could effectively customize the web pages and provide dynamic websites. | Can reduce the load to the server.                             |
| Security           | Relatively secure.                                                      | Insecure                                                       |

## Conclusion:

Client-side scripting and server-side scripting works in a coordinated manner with each other. However, both the scripting techniques are very different, where the client-side scripting emphasize on making the interface of the web application or website more appealing and functional. Conversely, server-side scripting emphasizes on the data accessing methods, error handling and fast processing etc.

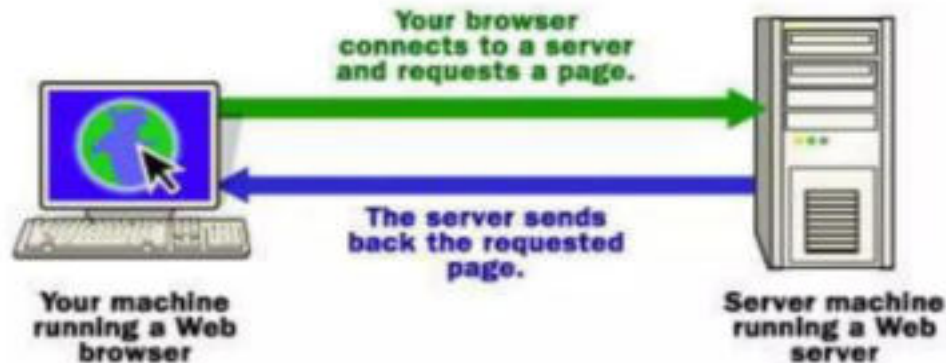
## **LOCAL SERVER AND REMOTE SERVER**

Understanding the difference between a Local Server and a Remote server is very important. If you are referring to a Local Server, this means that you have a server setup on your current machine. When the server is Remote, this just means that it is on another computer.

You may have heard the phrase, "Working Remotely." What this means is that the person will not be working in the office, instead they could be working at home or from their favorite corner coffee shop. So since that person is away from the office, they are considered "Remote." In the same case if a server hosting your files is not in-house or on your computer, it is considered to be a Remote Server.

## LOCAL SERVER AND REMOTE SERVER

First let's talk about what happens when you visit a web page in your browser. Well, your computer needs to contact a (remote) server on the web and then send the data back to your computer:



But when you are running a local server, your computer contacts itself since it is running a local server and returns the data back to the browser:





# INSTALL PHP

To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

- **WAMP** for Windows
- **LAMP** for Linux
- **MAMP** for Mac
- **SAMP** for Solaris
- **FAMP** for FreeBSD
- **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.

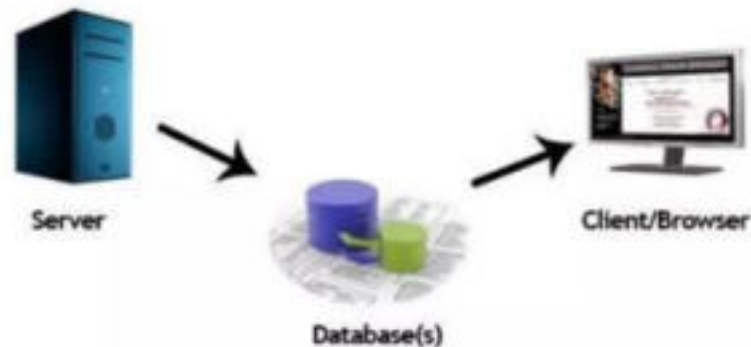
If you are on Windows and don't want Perl and other features of XAMPP, you should go for WAMP. In a similar way, you may use LAMP for Linux and MAMP for Macintosh.

# STATIC V/S DYNAMIC WEB SITE DEVELOPMENT

## Dynamic website

Dynamic website is a collection of dynamic web pages whose content changes dynamically. It accesses content from a database or Content Management System (CMS). Therefore, when you alter or update the content of the database, the content of the website is also altered or updated.

### Dynamic Website



# STATIC V/S DYNAMIC WEB SITE DEVELOPMENT

## Static website vs Dynamic website development.

| Static Website                                                                                        | Dynamic Website                                                                                                      |
|-------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| Prebuilt content is same every time the page is loaded.                                               | Content is generated quickly and changes regularly.                                                                  |
| It uses the <b>HTML</b> code for developing a website.                                                | It uses the server side languages such as <b>PHP, SERVLET, JSP, and ASP.NET</b> etc. for developing a website.       |
| It sends exactly the same response for every request.                                                 | It may generate different HTML for each of the request.                                                              |
| The content is only changed when someone publishes and updates the file (sends it to the web server). | The page contains "server-side" code which allows the server to generate the unique content when the page is loaded. |
| Flexibility is the main advantage of static website.                                                  | Content Management System (CMS) is the main advantage of dynamic website.                                            |

# INTRODUCTION TO PHP

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP Syntax is C-Like.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.

# INTRODUCTION TO PHP

## *Characteristics of PHP:*

Five important characteristics make PHP's practical nature possible –

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity



# EXAMPLE OF PHP SCRIPT

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML you'll have PHP statements like this –

```
<html>
 <head>
 <title>Hello World</title>
 </head>
 <body>
 <?php echo "Hello, World!";?>
 </body>
</html>
```

**It will produce following result –**

Hello, World!

# START AND END TAG OF PHP

The PHP syntax is a set of rules that define how a program should be written. All code written in PHP must be identified as PHP code. A set of tags are used to mark the beginning and end of a block of code, in between which any amount of code can be written.

- The standard *opening tag* is:

```
<?php
```

- The standard *closing tag* is:

```
?>
```

**Note:** PHP statements end with a semicolon (;).

# DATA TYPES IN PHP

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL



## PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

```
$x = "Hello world!";
$y = 'Hello world!';
```

## PHP Integer

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- Rules for integers:
  - An integer must have at least one digit
  - An integer must not have a decimal point
  - An integer can be either positive or negative
  - Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation
- In the following example \$x is an integer. The PHP var\_dump() function returns the data type and value:

```
<?php
$x = 5985;
var_dump($x);
?>
```

## PHP Float

- A float (floating point number) is a number with a decimal point or a number in exponential form.

e.g. \$x = 10.365;

## PHP Boolean

- A Boolean represents two possible states: TRUE or FALSE.
- Booleans are often used in conditional testing.

```
$x = true;
$y = false;
```

## PHP Array

- An array stores multiple values in one single variable.
- In the following example \$cars is an array.

```
$cars = array("Volvo","BMW","Toyota");
```

---

## PHP Object

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.
- First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

```
<?php
class Car {
 function Car() {
 $this->model = "VW";
 }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

## PHP NULL Value

- Null is a special data type which can have only one value: NULL.
  - A variable of data type NULL is a variable that has no value assigned to it.
- Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.  
Variables can also be emptied by setting the value to NULL:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

# VARIABLES IN PHP

- Variables are "containers" for storing information.

## Creating (Declaring) PHP Variables

- In PHP, a variable starts with the \$ sign, followed by the name of the variable:
- ```
<?php  
$txt = "Hello world!";  
$x = 5;  
$y = 10.5;  
?>
```

PHP Variables

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Rules for PHP variables:
 1. A variable starts with the \$ sign, followed by the name of the variable
 2. A variable name must start with a letter or the underscore character
 3. A variable name cannot start with a number
 4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

Variable names are case-sensitive (\$age and \$AGE are two different variables)

Output Variables

- The PHP echo statement is often used to output data to the screen.
- The following example will show how to output text and a variable:
- Example

```
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
 - Local
 - Global
 - Static

Local variable

The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.

A variable declaration outside the function with the same name is completely different from the variable declared inside the function. Let's understand the local variables with the help of an example:

File: local_variable1.php

```
<?php
function local_var()
{
    $num = 45; //local variable
    echo "Local variable declared inside the function is: ". $num;
}
local_var();
?>
```

Output:

```
Local variable declared inside the function is: 45
```


Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

Let's understand the global variables with the help of an example:

Example:

File: global_variable1.php

```
<?php
$name = "Sanaya Sharma";    //Global Variable
function global_var()
{
    global $name;
    echo "Variable inside the function: ". $name;
    echo "</br>";
}
global_var();
echo "Variable outside the function: ". $name;
?>
```

Output:

```
Variable inside the function: Sanaya Sharma
Variable outside the function: Sanaya Sharma
```

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script except for **magic constants**, which are not really constants. PHP constants can be defined by 2 ways:

1. Using `define()` function
2. Using `const` keyword

Constants are similar to the variable except once they defined, they can never be undefined or changed. They remain constant across the entire program. PHP constants follow the same PHP variable rules. **For example**, it can be started with a letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

PHP constant: define()

Use the define() function to create a constant. It defines constant at run time. Let's see the syntax of define() function in PHP.

```
define(name, value, case-insensitive)
```

1. **name:** It specifies the constant name.
2. **value:** It specifies the constant value.
3. **case-insensitive:** Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

Let's see the example to define PHP constant using define().

File: constant1.php

```
<?php  
define("MESSAGE","Hello JavaTpoint PHP");  
echo MESSAGE;  
?>
```

Output:

```
Hello JavaTpoint PHP
```

PHP constant: const keyword

PHP introduced a keyword **const** to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are **case-sensitive**.

File: constant4.php

```
<?php
const MESSAGE="Hello const by JavaTpoint PHP";
echo MESSAGE;
?>
```

Output:

```
Hello const by JavaTpoint PHP
```

PHP OPERATORS AND EXPRESSIONS

PHP Operators can be categorized in following forms:

- [Arithmetic Operators](#)
- [Assignment Operators](#)
- [Bitwise Operators](#)
- [Comparison Operators](#)
- [Incrementing/Decrementing Operators](#)
- [Logical Operators](#)
- [String Operators](#)
- [Array Operators](#)

PHP COMMENTS

PHP comments can be used to describe any line of code so that other developer can understand the code easily. It can also be used to hide any code.

PHP supports single line and multi line comments. These comments are similar to C/C++ and Perl style (Unix shell style) comments.

- // (C++ style single line comment)
- /**/Multiline comment

Process Management in OS

A Program does nothing unless its instructions are executed by a CPU. A program in execution is called a process. In order to accomplish its task, process needs the computer resources.

There may exist more than one process in the system which may require the same resource at the same time. Therefore, the operating system has to manage all the processes and the resources in a convenient and efficient way.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.

The operating system is responsible for the following activities in connection with Process Management

- 1.Scheduling processes and threads on the CPUs.
- 2.Creating and deleting both user and system processes.
- 3.Suspending and resuming processes.
- 4.Providing mechanisms for process synchronization.
- 5.Providing mechanisms for process communication.

Attributes of a process

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them. This is also called context of the process. Attributes which are stored in the PCB are described below.

1. Process ID

When a process is created, a unique id is assigned to the process which is used for unique identification of the process in the system.

2. Program counter

A program counter stores the address of the last instruction of the process on which the process was suspended. The CPU uses this address when the execution of this process is resumed.

3. Process State

The Process, from its creation to the completion, goes through various states which are new, ready, running and waiting. We will discuss about them later in detail.

4. Priority

Every process has its own priority. The process with the highest priority among the processes gets the CPU first. This is also stored on the process control block.

5. General Purpose Registers

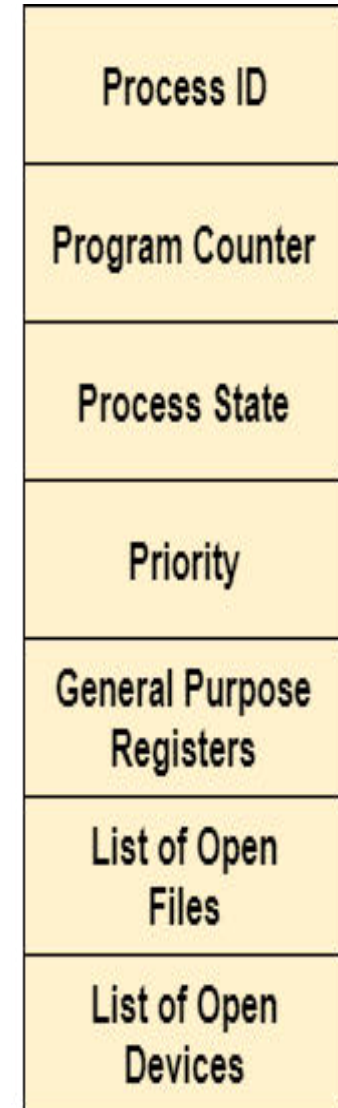
Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

6. List of open files

During the Execution, Every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.

7. List of open devices

OS also maintain the list of all open devices which are used during the execution of the process.



Process Attributes

Process States

The names of the states are not standardized although the process may be in one of the following states during execution.

1. **New:-** A program which is going to be picked up by the OS into the main memory is called a new process.

2. **Ready:-** Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned. The OS picks the new processes from the secondary memory and put all of them in the main memory.

The processes which are ready for the execution and reside in the main memory are called ready state processes. There can be many processes present in the ready state.

3. **Running:-** One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm. Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one. If we have n processors in the system then we can have n processes running simultaneously.

4. **Block or wait:-** From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process. When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes.

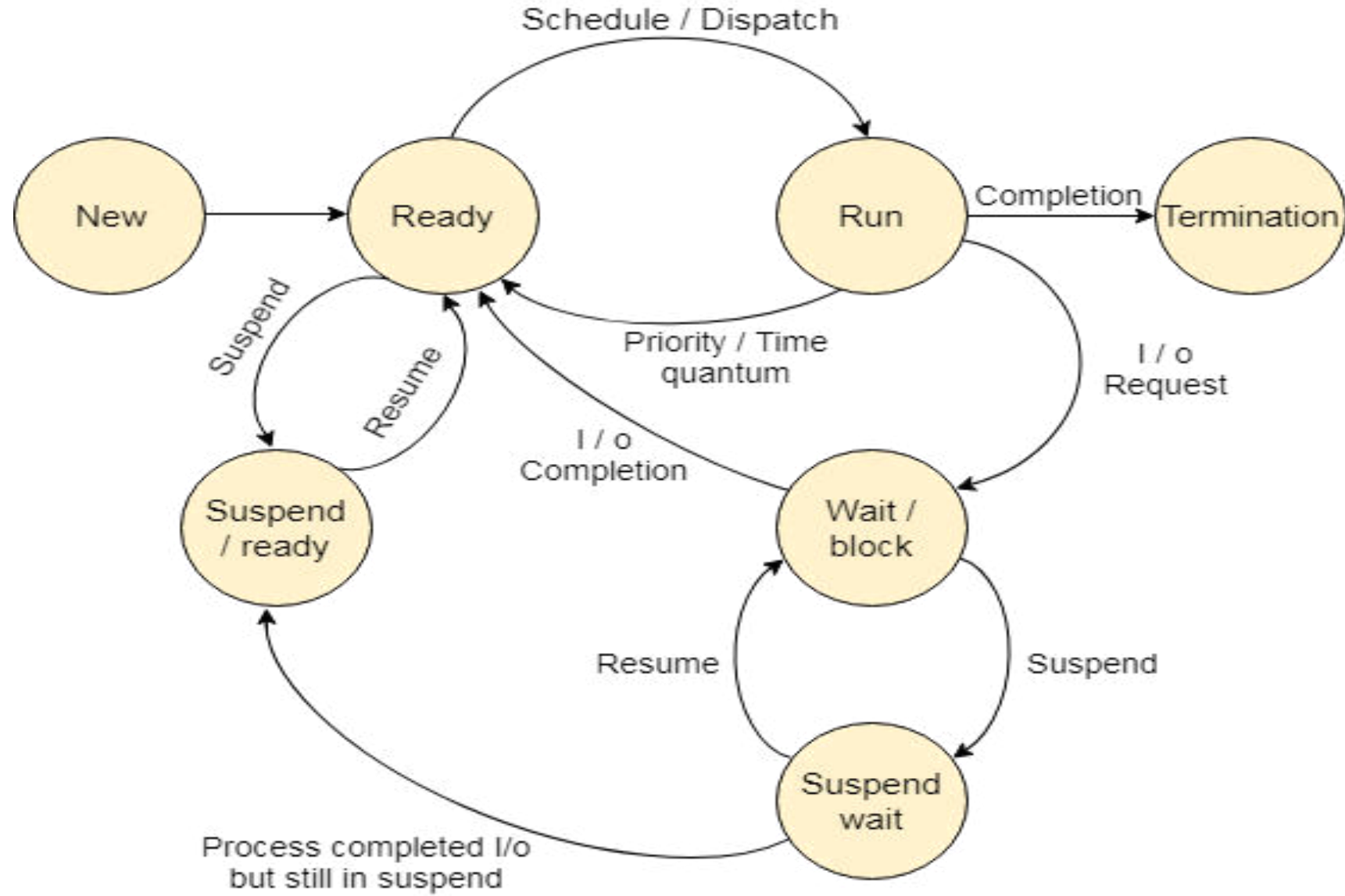
5. **Completion or termination:-** When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.

6. **Suspend ready:-** A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.

If the main memory is full and a higher priority process comes for the execution then the OS have to make the room for the process in the main memory by throwing the lower priority process out into the secondary memory. The suspend ready processes remain in the secondary memory until the main memory gets available.

7. **Suspend wait:-** Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory. Since it is already waiting for some resource to get available hence it is better if it waits in the secondary memory and make room for the higher priority process. These processes complete their execution once the main memory gets available and their wait is finished.

State Diagram



Process Scheduling in OS (Operating System)

Operating system uses various schedulers for the process scheduling described below.

1. Long term scheduler:- Long term scheduler is also known as job scheduler. It chooses the processes from the pool (secondary memory) and keeps them in the ready queue maintained in the primary memory.

Long Term scheduler mainly controls the degree of Multiprogramming. The purpose of long term scheduler is to choose a perfect mix of IO bound and CPU bound processes among the jobs present in the pool.

If the job scheduler chooses more IO bound processes then all of the jobs may reside in the blocked state all the time and the CPU will remain idle most of the time. This will reduce the degree of Multiprogramming. Therefore, the Job of long term scheduler is very critical and may affect the system for a very long time.

2. Short term scheduler:- Short term scheduler is also known as CPU scheduler. It selects one of the Jobs from the ready queue and dispatch to the CPU for the execution.

A scheduling algorithm is used to select which job is going to be dispatched for the execution. The Job of the short term scheduler can be very critical in the sense that if it selects job whose CPU burst time is very high then all the jobs after that, will have to wait in the ready queue for a very long time.

This problem is called starvation which may arise if the short term scheduler makes some mistakes while selecting the job.

3. Medium term scheduler:- Medium term scheduler takes care of the swapped out processes. If the running state processes needs some IO time for the completion then there is a need to change its state from running to waiting.

Medium term scheduler is used for this purpose. It removes the process from the running state to make room for the other processes. Such processes are the swapped out processes and this procedure is called swapping. The medium term scheduler is responsible for suspending and resuming the processes.

It reduces the degree of multiprogramming. The swapping is necessary to have a perfect mix of processes in the ready queue.

Process Queues

The Operating system manages various types of queues for each of the process states. The PCB related to the process is also stored in the queue of the same state. If the Process is moved from one state to another state then its PCB is also unlinked from the corresponding queue and added to the other state queue in which the transition is made.

There are the following queues maintained by the Operating system.

1. Job Queue

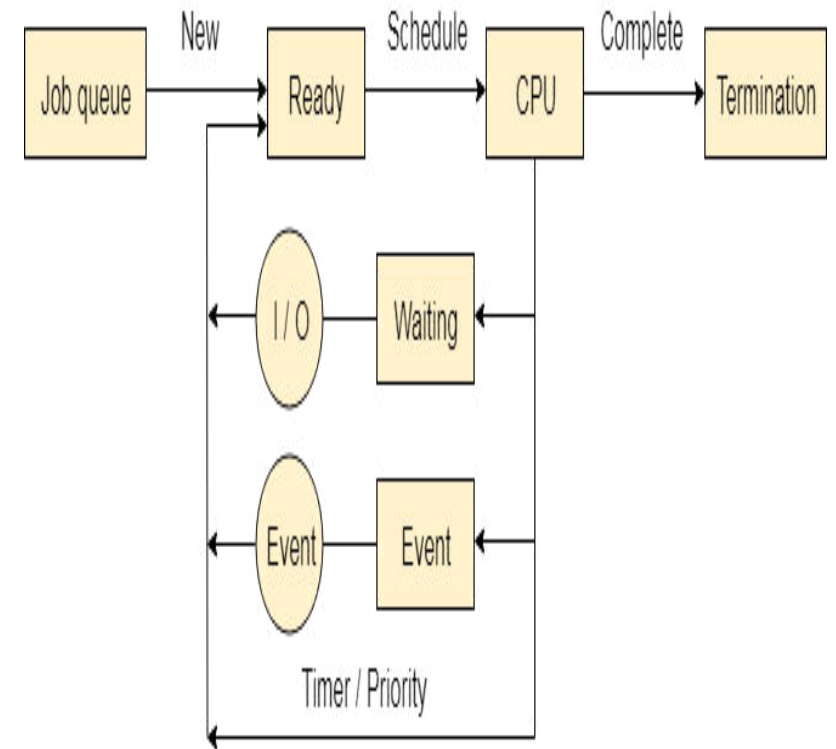
In starting, all the processes get stored in the job queue. It is maintained in the secondary memory. The long term scheduler (Job scheduler) picks some of the jobs and put them in the primary memory.

2. Ready Queue

Ready queue is maintained in primary memory. The short term scheduler picks the job from the ready queue and dispatch to the CPU for the execution.

3. Waiting Queue

When the process needs some IO operation in order to complete its execution, OS changes the state of the process from running to waiting. The context (PCB) associated with the process gets stored on the waiting queue which will be used by the Processor when the process finishes the IO.



Operation on Processes

The operations of process carried out by an operating system are primarily of two types:

1. **Process creation**
2. **Process termination**

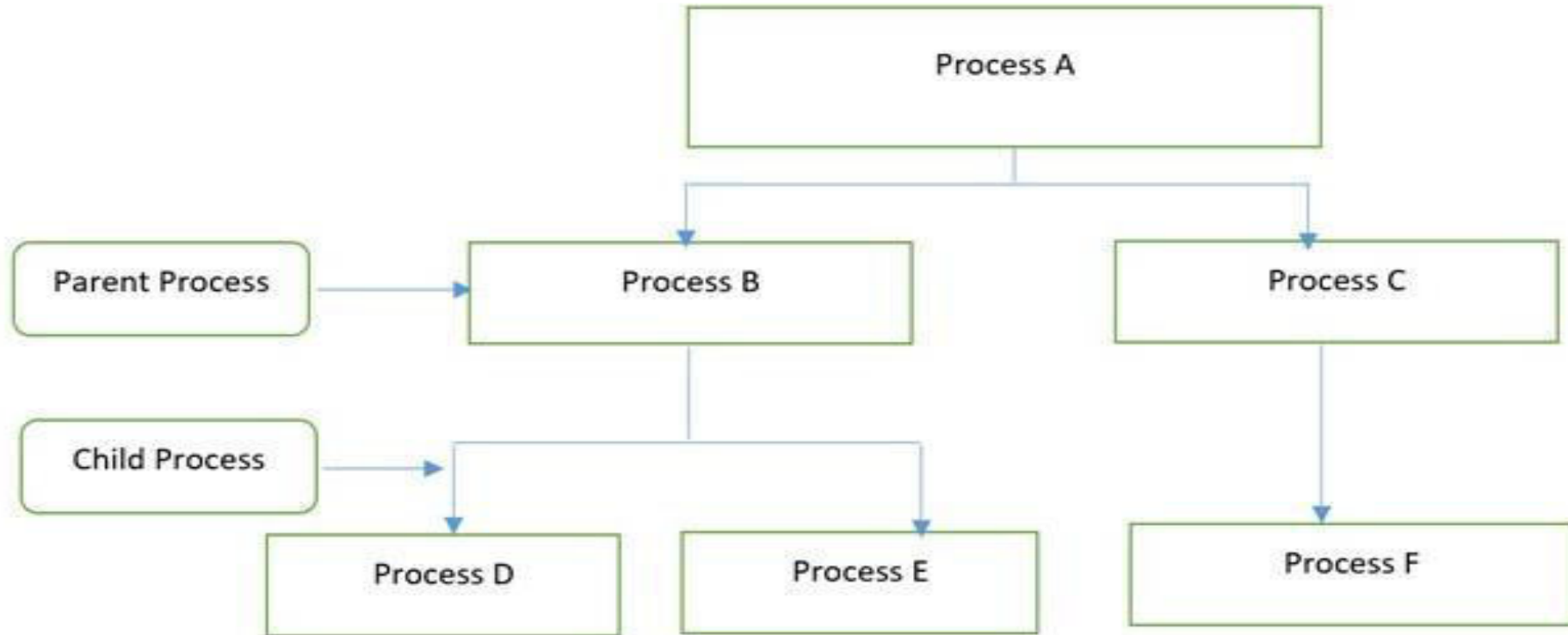
1) Process Creation

Process creation is a task of creating new processes. There are different situations in which a new process is created. There are different ways to create new process. A new process can be created at the time of initialization of operating system or when system calls such as **fork ()** are initiated by other processes. The process, which creates a new process using system calls, is called **parent process** while the new process that is created is called **child process**. The **child processes** can create new processes using system calls. A new process can also create by an operating system based on the request received from the user.

The process creation is very common in running computer system because corresponding to every task that is performed there is a process associated with it. For instance, a new process is created every time a user logs on to a computer system, an application program such a MS Word is initiated, or when a document printed.

2) Process termination

Process termination is an operation in which a process is terminated after the execution of its last instruction. This operation is used to terminate or end any process. When a process is terminated, the resources that were being utilized by the process are released by the operating system. When a child process terminates, it sends the status information back to the parent process before terminating. The child process can also be terminated by the parent process if the task performed by the child process is no longer needed. In addition, when a parent process terminates, it has to terminate the child process as well because a child process cannot run when its parent process has been terminated.



The above figure shows the hierarchical structure of processes.

The termination of a process when all its instruction has been executed successfully is called normal termination. However, there are instances when a process terminates due to some error. This termination is called as abnormal termination of a process.

MICROSOFT WORD
OR
OFFICE WORD

By:

Gyanendra Kumar Shukla
gyanendrashukla01@gmail.com
+91-9718246894

Microsoft Word

Microsoft Word is word processing software. It is developed by Microsoft and is part of Microsoft Office Suite. It enables you to create, edit and save professional documents like letters and reports.

Brief History:

Microsoft word was released in 1983 as Multi-Tool Word. Its first version was based on the framework of Bravo which was world's first graphical writing program.

Microsoft renamed Multi Tool Word to Microsoft Word, and then in October 1983, Microsoft released its first version for the IBM PC.

In 1985, Microsoft ported it to the Macintosh which was different from its DOS-based counterpart, i.e. Macintosh offered various major interface changes.

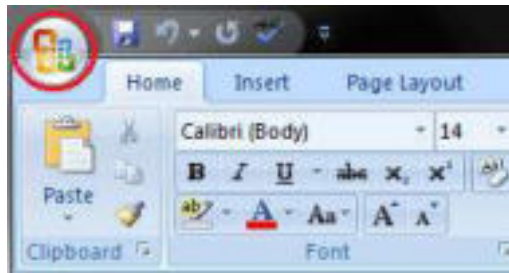
In 1989, Microsoft released a new version of Word for its Windows operating systems. It was the Microsoft Word who introduced the concept of WYSIWYG (What You See Is What You Get), i.e. it allowed to create and display bold and italics text.

In 2014, Microsoft developed the source code for Microsoft Word for Windows 1.1a.

Microsoft Office Button

Microsoft Office Button is located on the top left corner of the window. It is a new user interface feature that replaced the traditional "File" menu. You can also see this button in Outlook while creating a new message, task, contact, etc.

See the image:



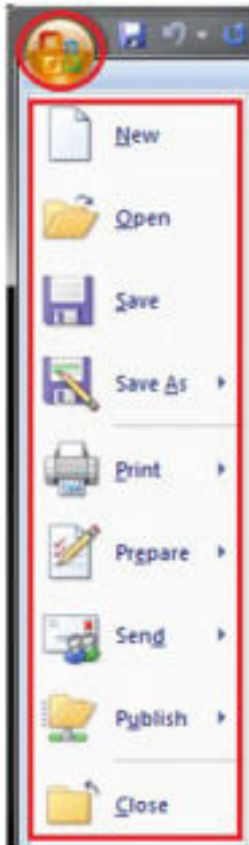
When you click the button it offers a list of commands to perform different tasks which are New, Open, Save, Save As, Print, Prepare, Send, Publish and Close. These commands are described below the following image.

See the image:

GYANENDRA SHUKLA

gyanendrashukla01@gmail.com

+91-9718246894



New: This command enables you create a new file, i.e. Word document.

Open: This command allows you to open an existing file on the computer.

Save: This command is used to save a file after completing the work. You can also save the changes made to the currently open file.

Save As: This command helps you save a new file with a desired file name to a desired location on the hard drive.

Print: This command is used to print a hard copy of the currently open document.

Prepare: This command allows you to prepare the document for distribution, i.e. you can view and edit the document properties and inspect the hidden metadata.

Send: This command allows you share the document with other users, i.e. you can send a document through e-mail or by posting to a blog.

Publish: This command allows you distribute the document to other people, i.e. you can create a blog with the content of the document.

Close: This command is used to close the currently open file.

GYANENDRA SHUKLA

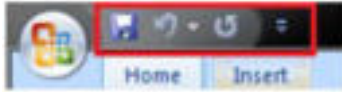
gyanendrashukla01@gmail.com

+91-9718246894

Quick Access Toolbar

Quick Access Toolbar lies next to the Microsoft Office Button. It is a customizable toolbar that comes with a set of independent commands. It gives you quick access to commonly used commands such as Save, Undo, Redo, etc.

See the image:



When you click the drop-down arrow next to toolbar it offers more commands. With a left click you can add any of these commands to Quick Access Toolbar. You can also remove the commands added to the tool bar. The indent, spacing values, individual styles and other features that appear on the ribbon cannot be added to quick access toolbar. The following image is showing the menu of quick access toolbar.

See the image:



GYANENDRA SHUKLA

gyanendrashukla01@gmail.com

+91-9718246894

Title Bar

It lies next to the Quick Access Toolbar. It displays the title of the currently open document or application. It is present on almost all windows displayed on your computer. So, if there are several windows across the screen, you can identify each window by looking at the title bar. In many graphical user interfaces, you can also move a window by dragging the title bar.

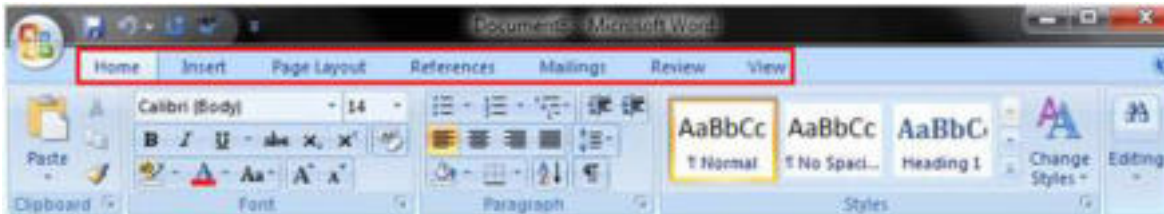
See the image:



Ribbon and Tabs

The Ribbon is a user interface element which was introduced by Microsoft in Microsoft Office 2007. It is located below the Quick Access Toolbar and the Title Bar. It comprises seven tabs; Home, Insert, Page layout, References, Mailing, Review and View. Each tab has specific groups of related commands. It gives you quick access to the commonly used commands that you need to complete a task.

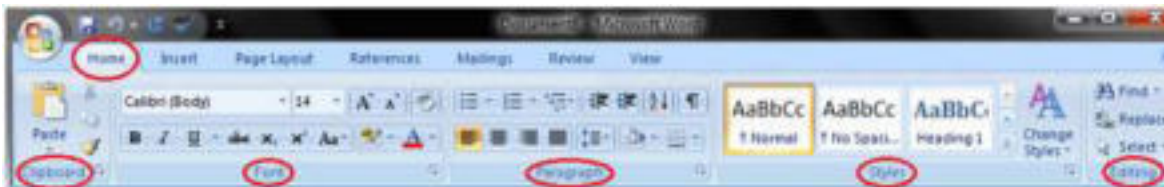
See the image:



Home tab:

The Home tab is the default tab in Microsoft Word. It has five groups of related commands; Clipboard, Font, Paragraph, Styles and Editing. It helps you change document settings like font size, adding bullets, adjusting styles and many other common features. It also helps you to return to the home section of the document.

See the image:



GYANENDRA SHUKLA

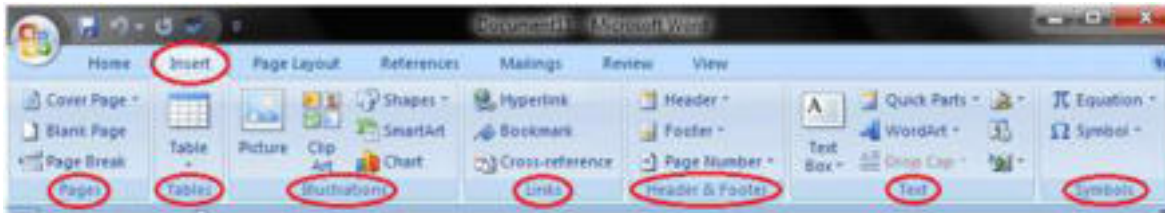
gyanendrashukla01@gmail.com

+91-9718246894

Insert tab:

Insert Tab is the second tab in the Ribbon. As the name suggests, it is used to insert or add extra features in your document. It is commonly used to add tables, pictures, clip art, shapes, page number, etc. The Insert tab has seven groups of related commands; Pages, Tables, Illustrations, Links, Header & Footer, Text and Symbols.

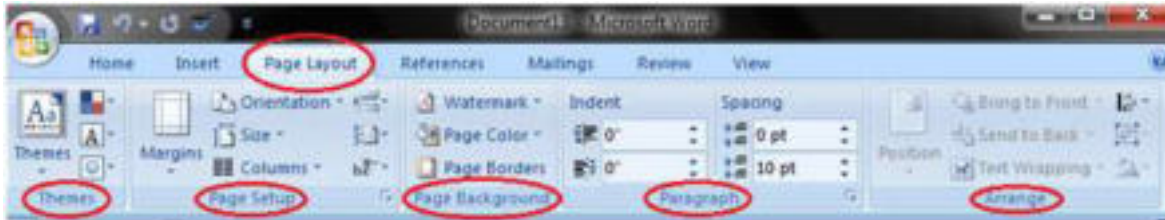
See the image:



Page Layout tab:

It is the third tab in the Ribbon. This tab allows you to control the look and feel of your document, i.e. you can change the page size, margins, line spacing, indentation, documentation orientation, etc. The Page Layout tab has five groups of related commands; Themes, Page Setup, Page Background, Paragraph and Arrange.

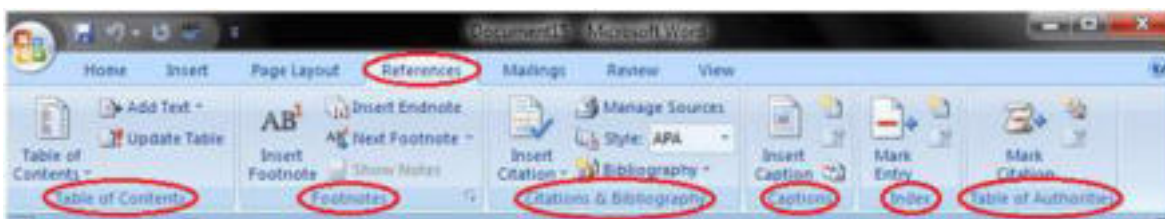
See the image:



References tab:

It is the fourth tab in the Ribbon. It allows you to enter document sources, citations, bibliography commands, etc. It also offers commands to create a table of contents, an index, table of contents and table of authorities. The References tab has six groups of related commands; Table of Contents, Footnotes, Citations & Bibliography, Captions, Index and Table of Authorities.

See the image:



GYANENDRA SHUKLA

gyanendrashukla01@gmail.com

+91-9718246894

Mailings tab:

It is the fifth tab in the ribbon. It is the least-often used tab of all the tabs available in the Ribbon. It allows you merge emails, writing and inserting different fields, preview results and convert a file into a PDF format. The Mailings tab has five groups of related commands; Create, Start Mail Merge, Write & Insert Fields, Preview Results and Finish.

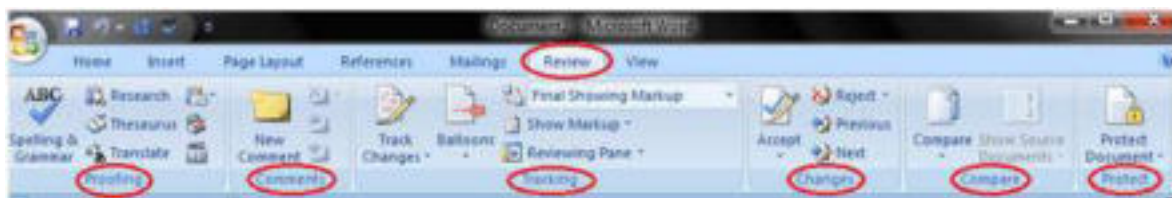
See the image:



Review tab:

It is the sixth tab in the Ribbon. This tab offers you some important commands to modify your document. It helps you proofread your content, to add or remove comments, track changes, etc. The Review tab has six groups of related commands; Proofing, Comments, Tracking, Changes, Compare and Protect.

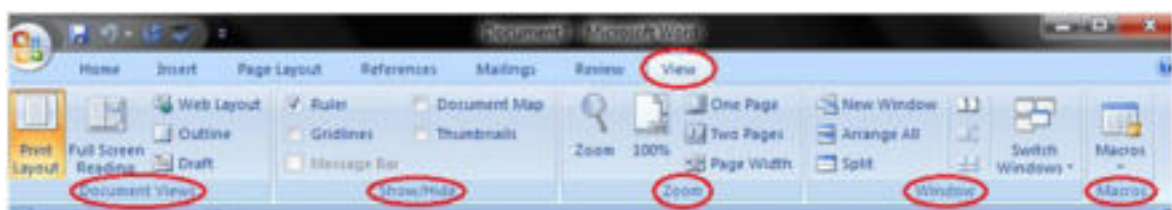
See the image:



View tab:

The View tab is located next to the Review tab. This tab allows you to switch between Single Page and Two Page views. It also enables you to control various layout tools like boundaries, guides, rulers. Its primary purpose is to offers you different ways to view your document. The View tab has five groups of related commands; Document Views, Show/Hide, Zoom, Window and Macros.

See the image:



GYANENDRA SHUKLA

gyanendrashukla01@gmail.com

+91-9718246894

Ruler

The Ruler is located below the Ribbon around the edge of the document. It is used to change the format of the document, i.e. it helps you align the text, tables, graphics and other elements of your document. It uses inches or centimeters as the measurements unit and gives you an idea about the size of the document.

See the image:



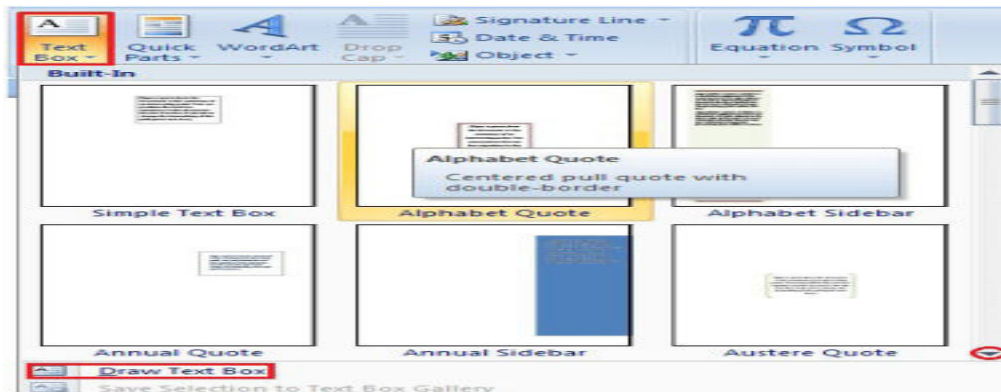
TEXT BASICS

How to Insert Text in MS Word

The basic steps to insert text or to create a new document in Word are listed below;

- Go to the start menu and look for Microsoft Word icon
- Click the icon to open the Microsoft Word
- You will see a blinking cursor or insertion point in the text area below the ribbon
- Now, as you start typing, the words will appear on the screen in the text area
- To change the location of insertion point press spacebar, Enter or Tab keys

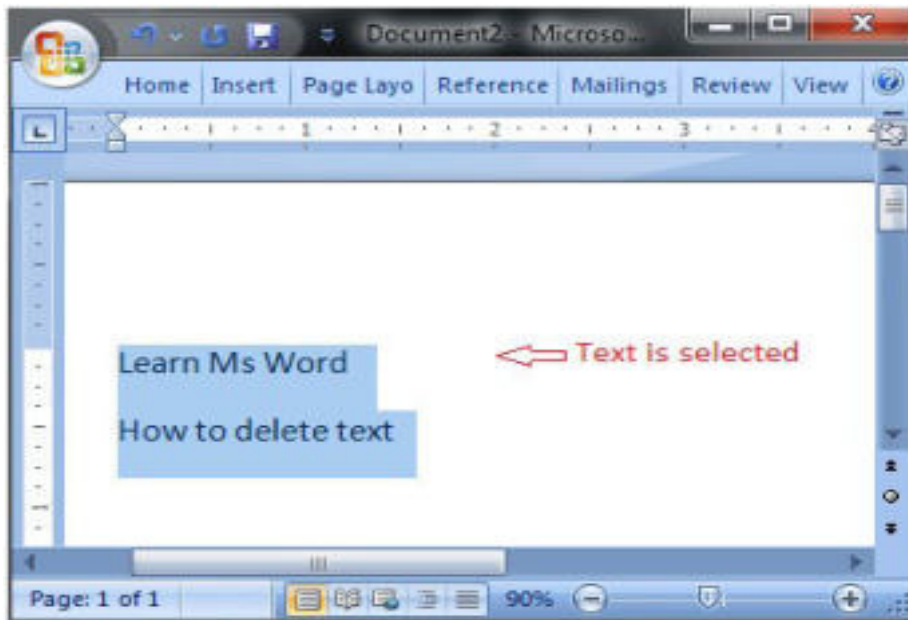
See the image:



How to Delete Text in MS Word

You can easily delete the text in Word including characters, paragraphs or all of the content of your document. Word offers you different methods to delete the text; some of the commonly used methods are given below;

- Place the cursor next to the text then press Backspace key
- Place the cursor to the left of the text then press Delete key
- Select the text and press the Backspace or Delete key
- Select the text and type over it the new text.



How to Select Text in MS Word

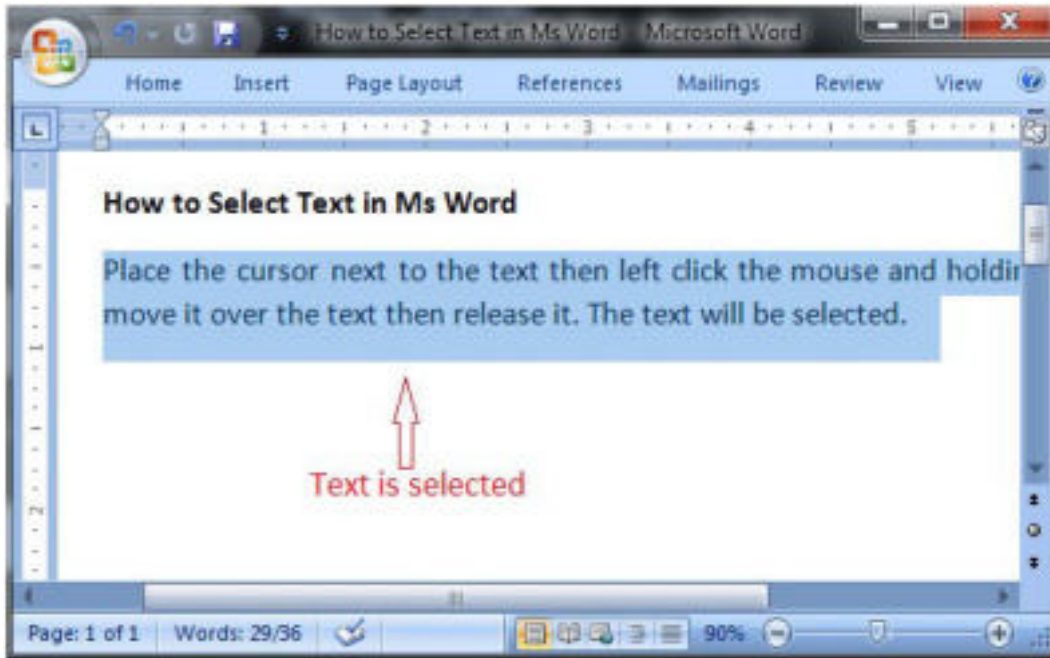
Place the cursor next to the text then left click the mouse and holding it down move it over the text then release it. The text will be selected.

Some shortcuts for selecting text are:

- To select a single word double click within the word
- To select the entire paragraph triple click within the paragraph
- To select entire document, in Home tab, in Editing group click Select then choose Select All option or press CTRL+A

- Shift + Arrow; hold down the shift key then press the arrow key, the word will select the text in the direction of the arrow key. There are three arrow keys, so you can select the text in three different directions.

See the image:



How to Copy and Paste Text in MS Word

Word offers different methods to copy and paste text. Some of the popular methods are given below.

Method 1:

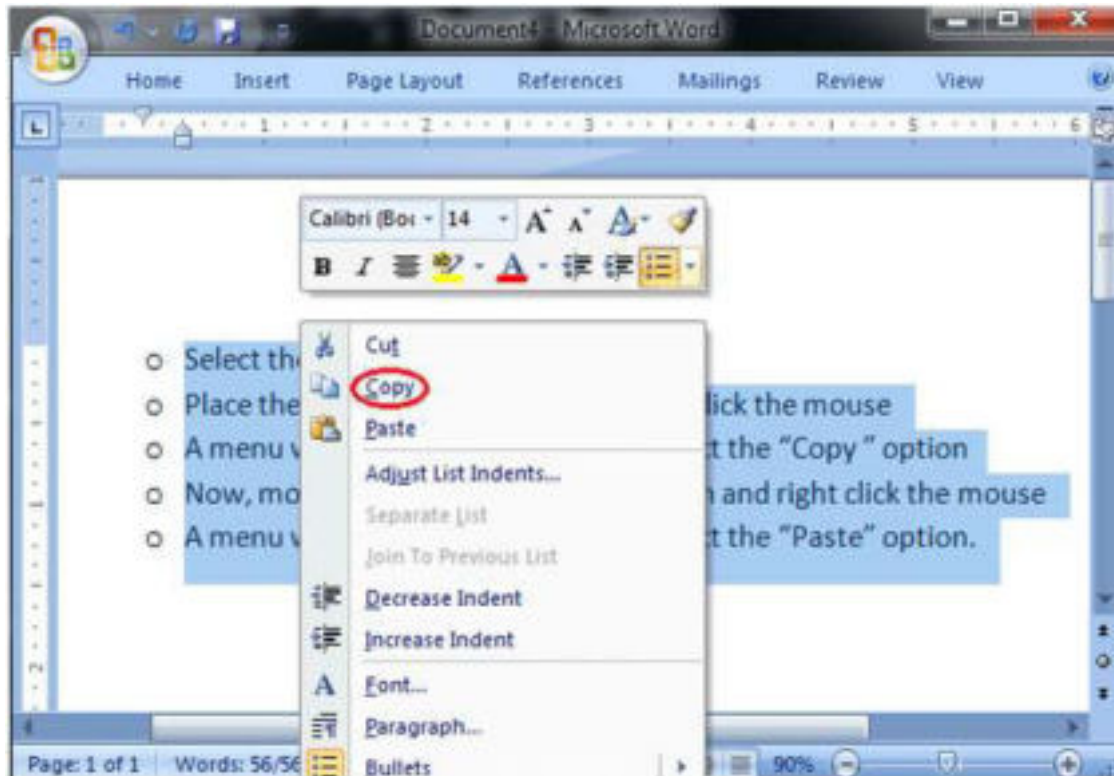
- Select the text you want to copy
- Select the Home tab and click the Copy command
- Place the cursor where you want to paste the text
- Click the Paste command in Home tab

Method 2:

- Select the text
- Place the cursor over the text and right click the mouse

- A menu will appear; with a left click select the "Copy" option
- Now, move the cursor to a desired location and right click the mouse
- A menu will appear; with a left click select the 'Paste" option.

See the image:



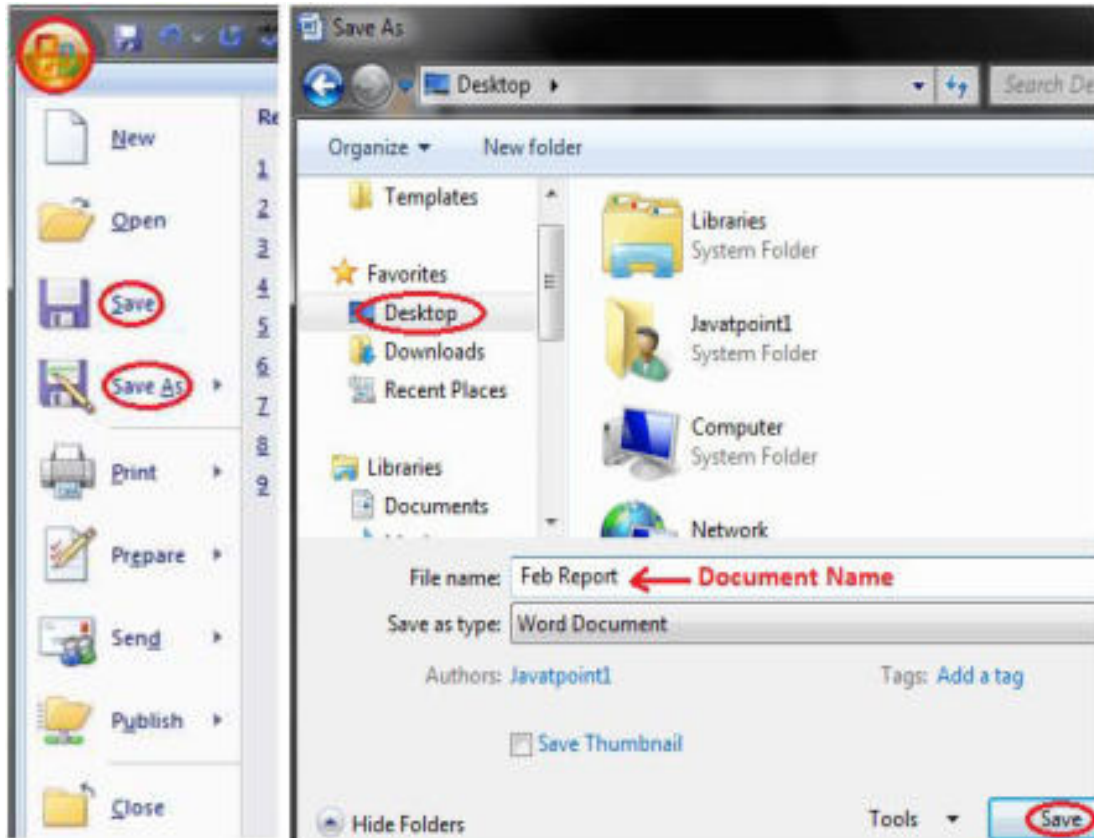
How to Save the Document in MS Word

When you create a document it is important to save the document so that it can be viewed or reused later. The basic steps to save a document are listed below;

- Click the Microsoft Office Button
- A list of different commands appears
- Click the 'Save As' command
- it displays 'Save As' Dialogue Box
- Save the document to desired location with a desired name

You can also choose 'Save' command from the list to save the document to its current location with same title. If you are saving a fresh document it displays 'Save As' dialogue box.

The shortcut method to save a document is to press "Ctrl+S" keys. It opens the 'Save As' dialogue box where you can name you document and save it to a desired location.



Proofing Features

How to Correct Errors in MS Word

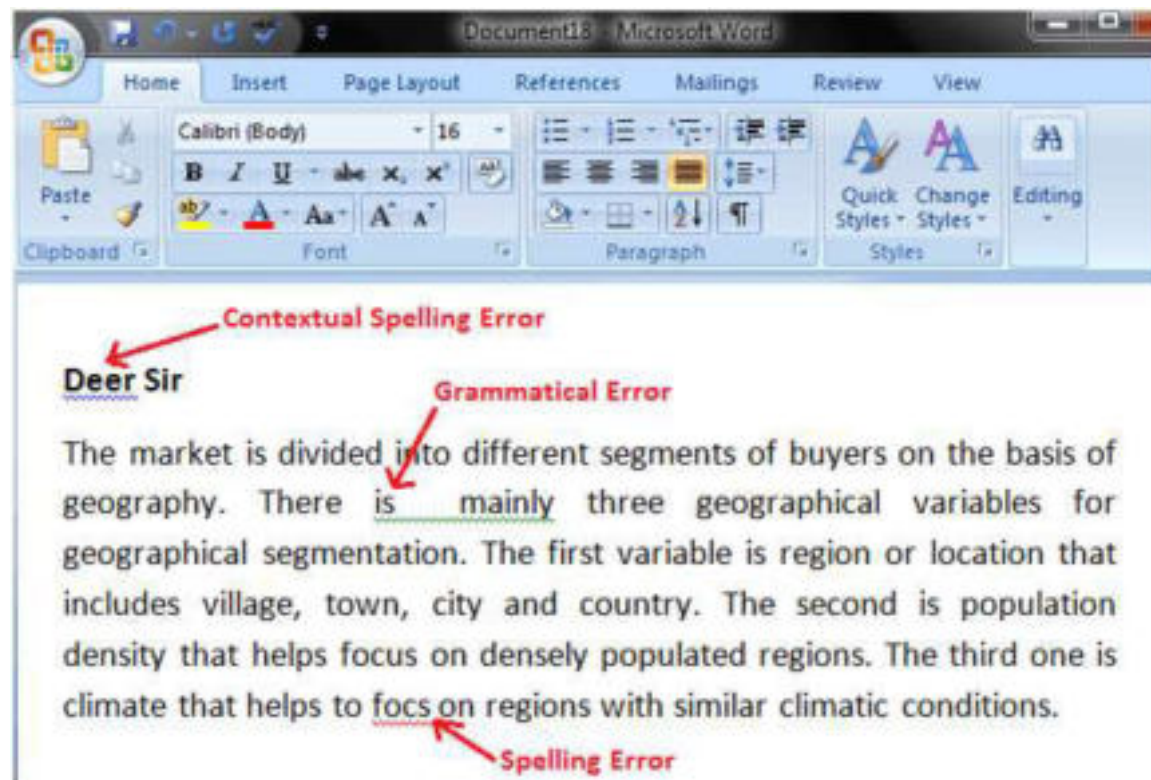
When you type text in a document, by default the Word informs you if there is any contextual, spelling or grammatical error. Word informs you in different ways for different errors;

If there is any contextual error in the document, it will underline the text with blue line.

If there is any spelling error in the document, it will underline the text with red line.

If there is any grammar error in the document, it will underline the text with green line.

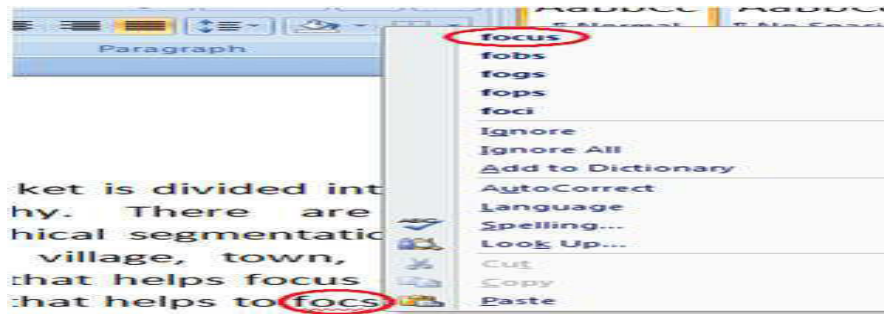
See the image:



Steps to correct errors:

- Place the cursor over the text that you want to correct
- Right click the mouse
- A list of suggestions appears
- Choose the correct word with a left click

See the image:

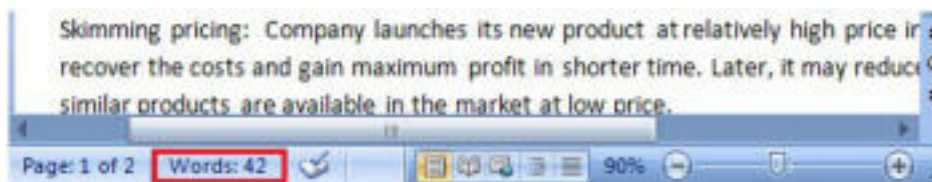


How to Check Word Count in Ms Word

When you start typing, the Word automatically counts the number of words and number of pages and displays the information on the status bar. If the word count is not available on the status bar, you can right click the status bar and select the Word Count option from the menu. Basic steps to check the word count are listed below;

- Open the document
- Look at left bottom corner of the document
- You will see the total word count and the number of pages

See the image:



To know the word count of a specific line or paragraph you have to select it then Word will display its word count along with the total word count, e.g. 15/40. In this example, the selection has 15 words out of the total number of words (40).

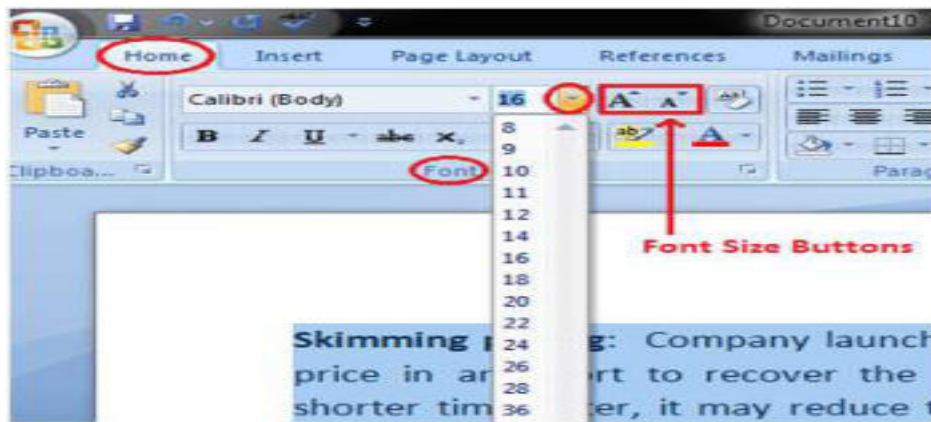
Formatting Text

How to Change Font Size in MS Word

You can easily change the font size of your text in the document. The basic steps to change the Font size are listed below:

- Select the text that you want to modify
- In Home tab locate the Font group
- In Font group click the drop-down arrow next to font size box
- Font size menu appears
- Select the desired font size with a left click
- Select the text and click the increase or decrease font size buttons

See the image:

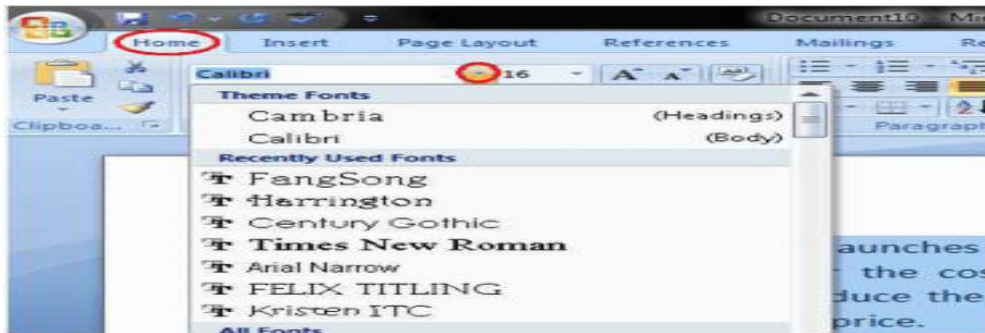


How to Change Font Style in MS Word

The basic steps to change the font of a text in a document are given below;

- Select the text you want to modify
- Select the Home tab and locate the Font group
- Click the drop-down arrow next to font style box
- Font style menu appears
- With a left click select the desired font style
- If you want to change the font to bold or italic, click the 'B' or 'I' icons on the format bar.

See the image:

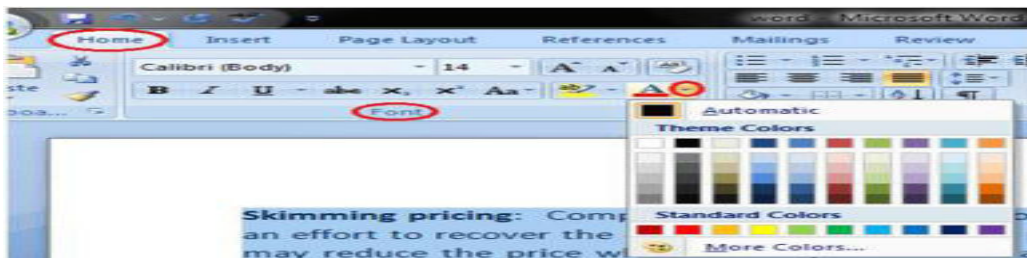


How to Format Font Color in MS Word

MS Word allows you to change the Font color of your text. If you want to emphasize a particular word or phrase, you can change its font color. The basic steps to change the Font color are given below;

- Select the text you want to modify
- In Home tab locate the Font group
- Click the drop-down arrow next to Font color button
- Font color menu appears
- Select the desired font color with a left click
- Word will change the Font color of the selected text.

See the image:



How to Change Text Case in MS Word

You can easily change the text case in your document by following the steps given below;

- Select the text you want to change
- In Home tab locate the Font group
- Click the drop-down arrow in 'Change Case' button
- It displays text case menu
- Select the desired case with a left click

The case menu offers four options;

Sentence case: It capitalizes the first letter of each sentence.

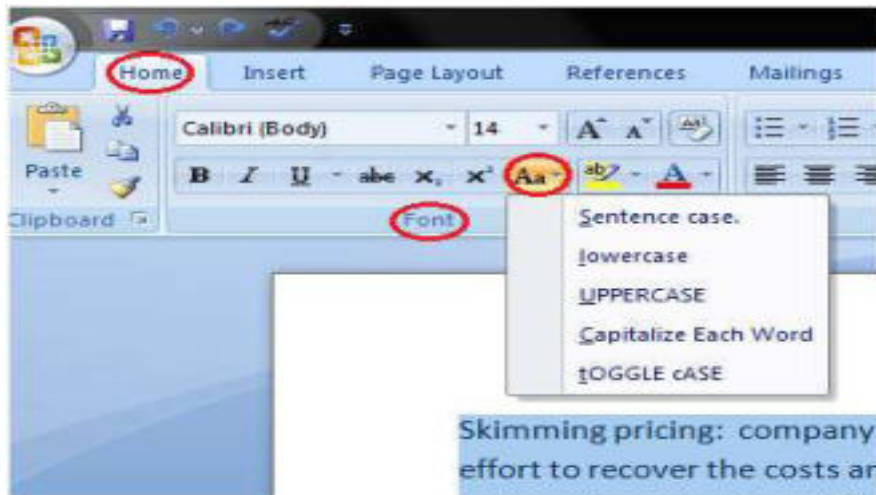
Lowercase: It changes the text from uppercase to lowercase.

Uppercase: It capitalizes all the all letters of your text.

Capitalize Each Word: It capitalizes the first letter of each word.

Toggle Case: It allows you to shift between two case views, e.g. to shift between Capitalize Each Word and cAPITALIZE eACH wORD.

See the image:



How to Change Text Alignment in MS Word

You can change the text alignment in your document to make it more presentable and readable. The basic steps to change the text alignment are given below;

- Select the content you want to modify
- In Home tab locate the Paragraph group
- It has four alignment options ;

Align Text Left: Aligns the text towards left margin

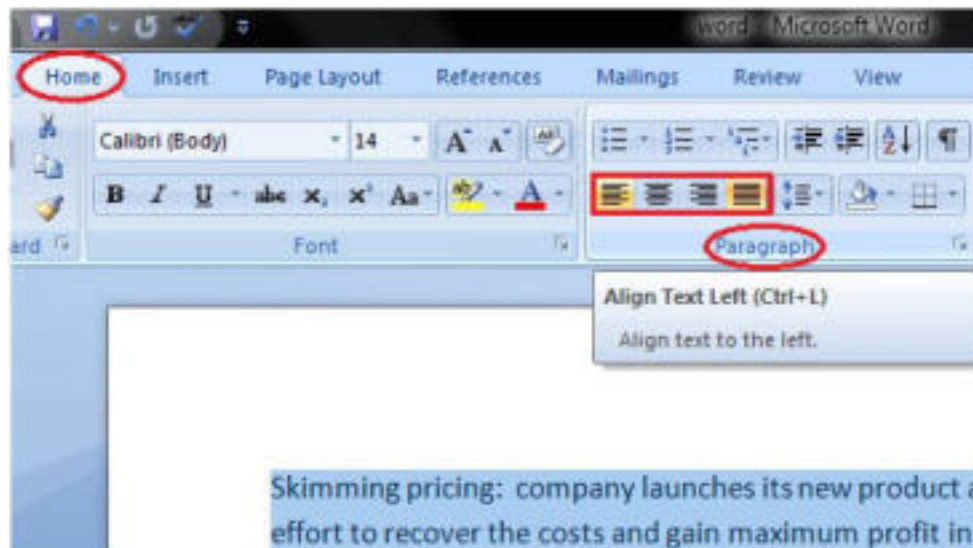
Center: Brings the text at center

Align Text Right: Aligns the text towards right margin

Justify: Aligns the text to both left and right margins

- Select the desired alignment option with a left click

See the image:



How to Insert a Text Box in MS Word

Text box allows you to control the position of a block of text in your document. You can also format them with borders and shading. The two commonly used methods to insert Text Boxes are given below:

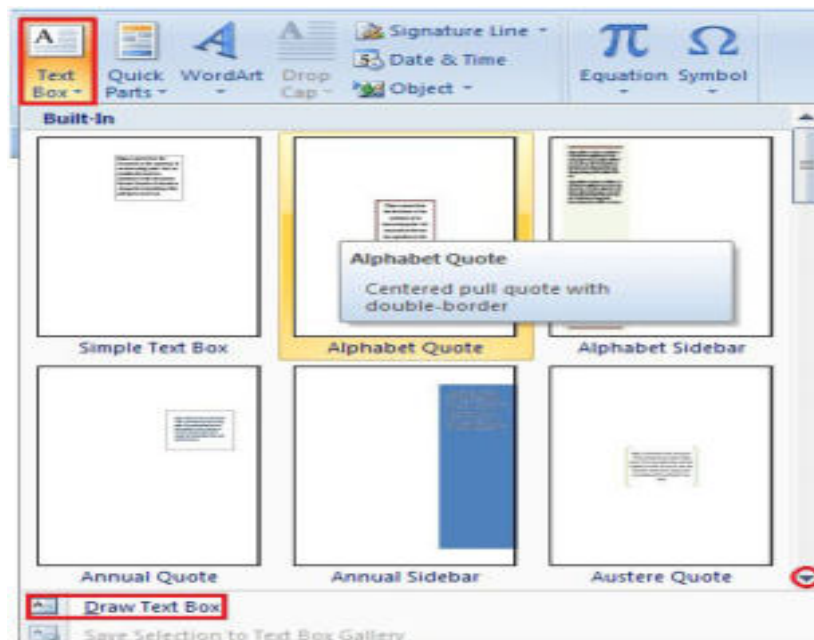
Method 1:

- Select the Insert tab
- Locate the Text group
- Click the Text Box button
- It displays Built-In text box menu and an option to draw table
- With a left click select the desired text box format from the menu

Method 2:

- Select 'Draw Text Box' option
- A cross shaped cursor appears
- Left click the mouse and holding it down drag it to draw the box of desired dimensions

See the image:

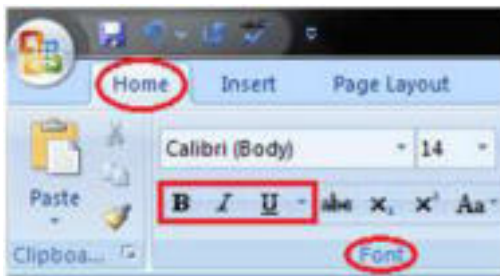


Bold, Italic and Underline Commands in MS Word

These commands are given in the Font group in the Home tab. Their functions are given below;

- **Bold:** It allows you to Bold the text of your document
- **Italic:** It allows you to Italicize the text of your document
- **Underline:** It allows you to underline the text of your document

See the image:



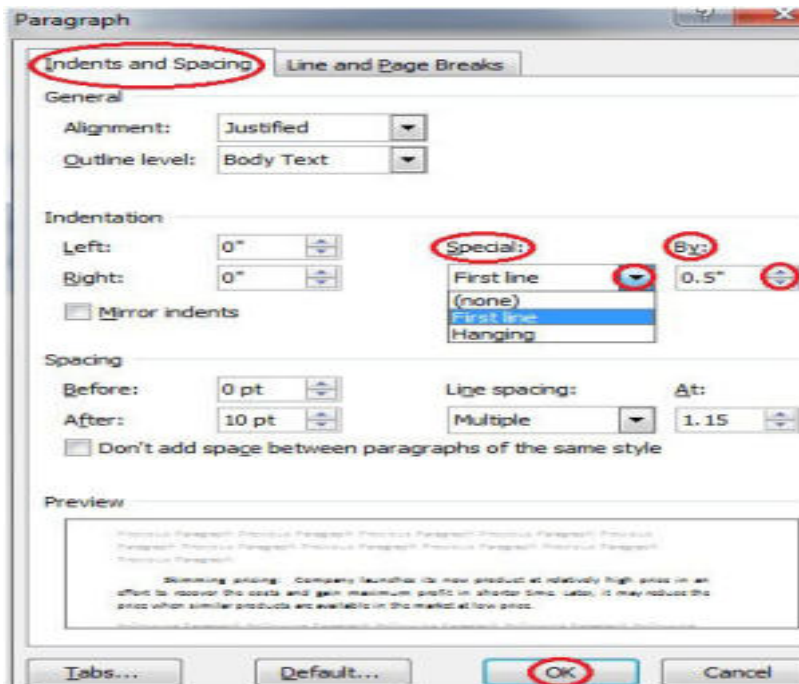
Formatting Paragraph

How to Create First Line Indent in MS Word

You can create indent within your paragraph by following these steps:

- Select the paragraph or place the cursor anywhere within the paragraph
- Select the Home tab
- Locate the Paragraph group and click the arrow at right bottom corner
- 'Paragraph' dialog box appears
- In 'Indents and Spacing' section click the drop down arrow in 'Special' field
- Select the 'First Line' option
- Enter desired indent in 'By' field and click Ok

See the image:



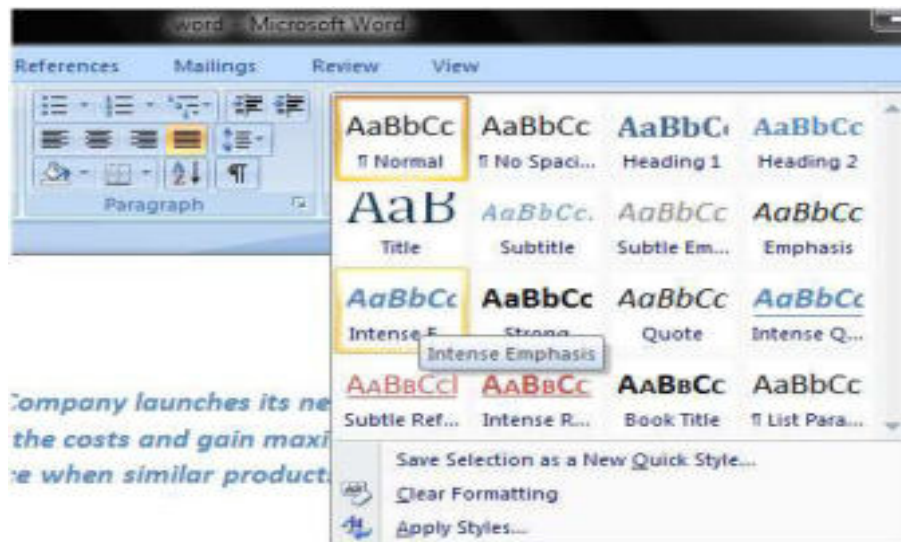
How to Apply Style in MS Word

You can create professional and presentable documents in MS Word by applying different styles. The basic steps to apply a style in a document are listed below;

- Select the text to which you want to apply the style
- Select the Home tab
- In Styles group you will see different styles



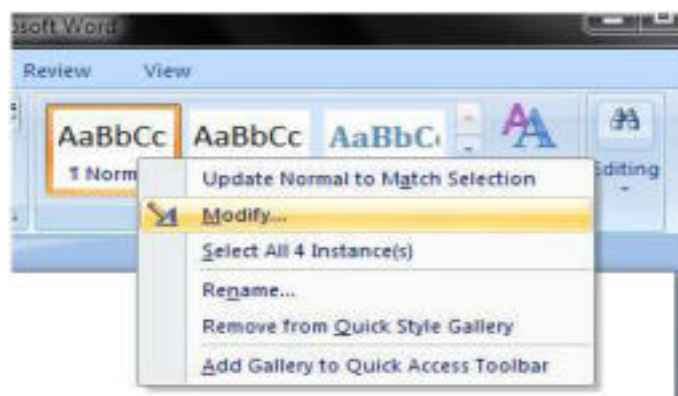
- To view more styles click the drop-down arrow
- It displays style menu
- Select the desired style with a left-click



How to Customize Style in MS Word

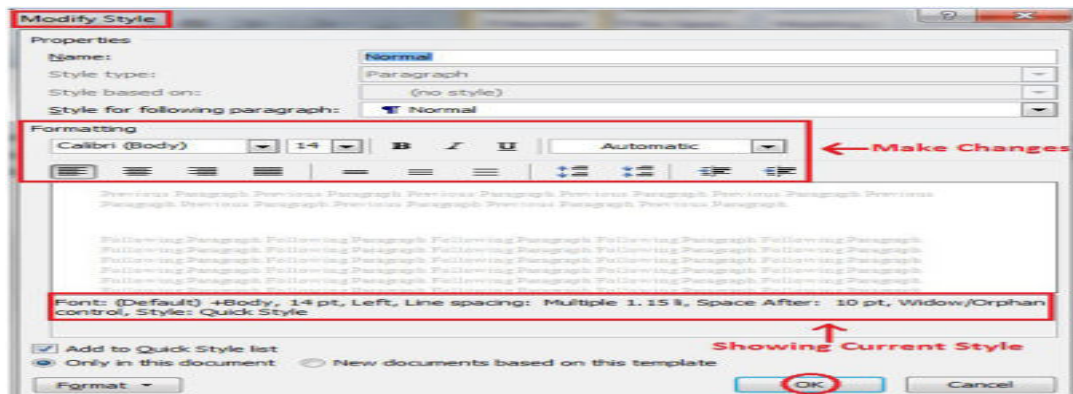
If you are looking for formatting options that are not given in the built-in styles, you can modify or customize an existing style to fulfill your needs. The steps to customize a style are as follows;

- Select the style that you want to modify
- Right click the mouse
- It displays a list of different commands
- Select the 'Modify' option



- 'Modify Style' dialogue box appears

See the image:

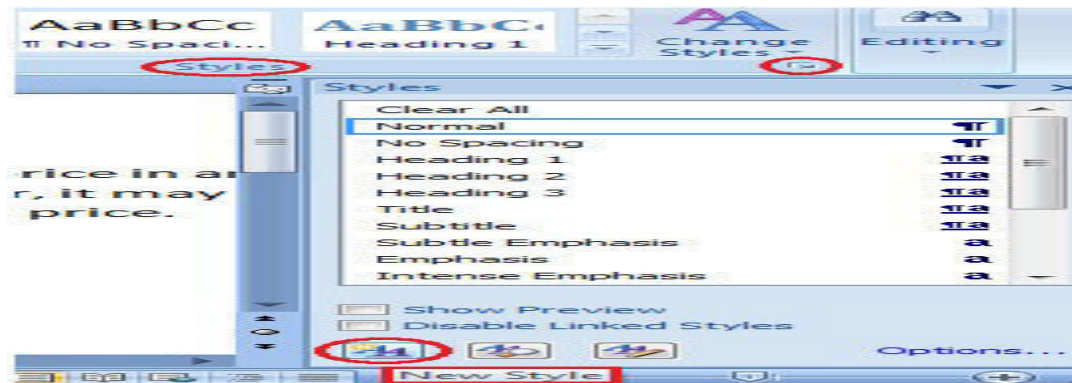


- Make the desired changes to formatting
- Click OK to apply the changes to style

How to Create New Style in MS Word

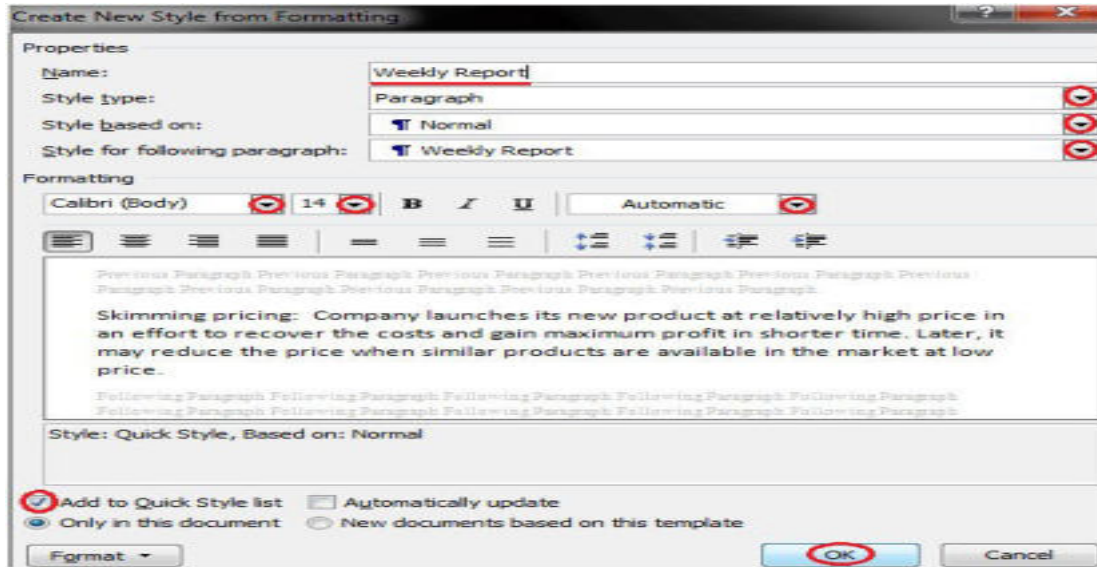
You can add new styles to your list of styles, i.e. Word allows you to set the styles for font, figure, paragraph, etc. It helps you to keep consistency in all the documents of a topic or subject. The steps to create new styles are given below;

- Select the Home tab
- In Styles group click the arrow at the right bottom corner of the group
- It displays the 'Styles' task pane



In 'Styles' task pane click the 'New Style' button

It displays 'Create New Style from Formatting' dialogue box



- Enter the name for new style and make all the desired changes
- Click OK, the new style will be added to the list of styles

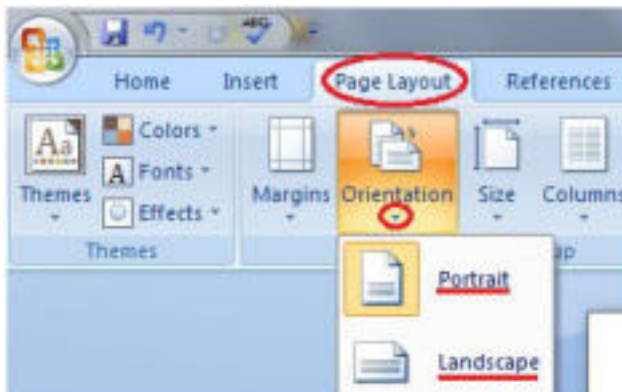
Modify Page Layout

How to Change Page Orientation in MS Word

Page Orientation refers to the direction in which a document is displayed. It is of two types; portrait (vertical) and landscape (horizontal). The default orientation is portrait; it can be changed to landscape by following these steps;

- Select the Page Layout tab
- Locate the Page Setup group
- In Page Setup group click the Orientation command
- It displays two options, Portrait and Landscape
- Select the desired page orientation

See this images:



How to Change Page Size in MS Word

The default paper size in Word is 8.5 x 11 inches which easily fits in printers. This size is not fixed; you can change it if you want a document with different paper size. The steps to change the paper size are given below;

- Click the Page Layout tab
- In Page Setup group click the Size command
- Paper size menu will appear
- With a left click select the desired paper size
- To customize page size click the 'More Paper Sizes' option

See this images:

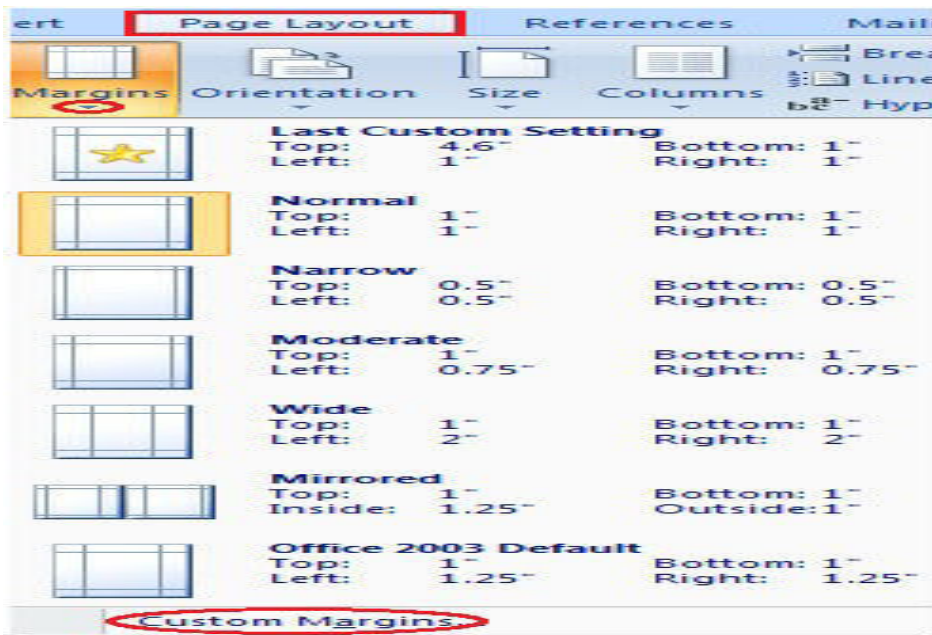


How to Change Page Margins in MS Word

The margin is the space between the text and border of a document. By default, it is a one-inch space. Depending on your needs, it can be changed by following the below-listed steps;

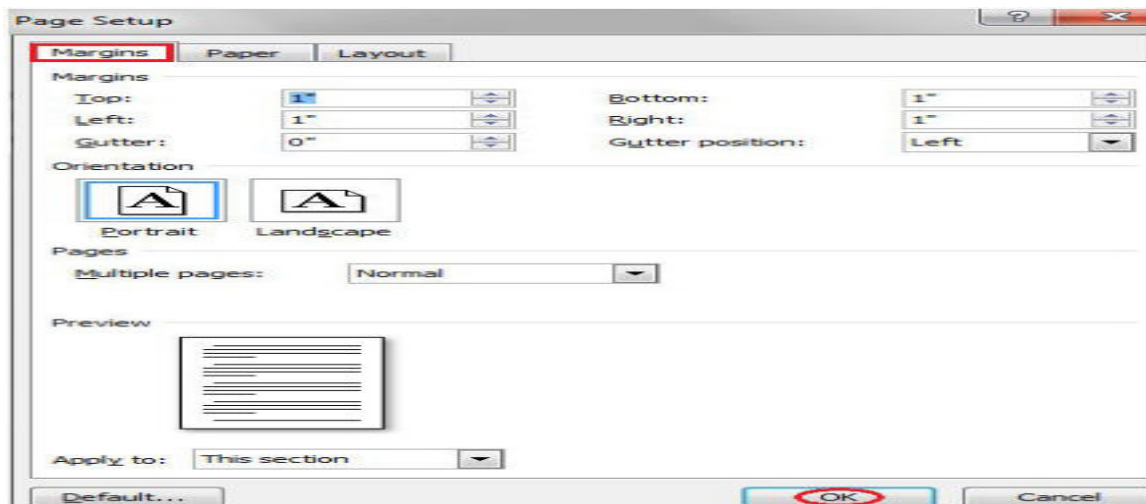
- Select the Page Layout tab
- In Page Setup group click the Margins command
- A list of Margins appears
- Select the desired Margin with a left click

See the image:



To customize Margins select 'Custom Margins'. It displays a 'Page Setup' dialog box. Enter the desired margin size and click Ok.

See the image:



How to Insert Page Break in MS Word

Word inserts a page break at the end of each page. It also allows you to insert a page break at some other place in the document. The steps to insert page break are given below;

- Place the cursor where you want to insert the break
- Select the Page Layout tab
- In Page Setup group click the 'Breaks' command
- A list of Page Breaks appears
- With a left click select the desired page break from the list

See the image:



How to insert a header and footer in Microsoft Word document

In Microsoft Word, Headers and Footers are used to insert additional information such as **title, file name, date, page numbers, etc.** The presence of both header and footer in the Word document makes your document more professional and easier to read as well as understand.

Headers appear at the **top margin of the Word document**, while **Footers** appear at the **bottom margin of the Word document**.

To insert a header and footer in Microsoft Word, follow the below given basic steps –

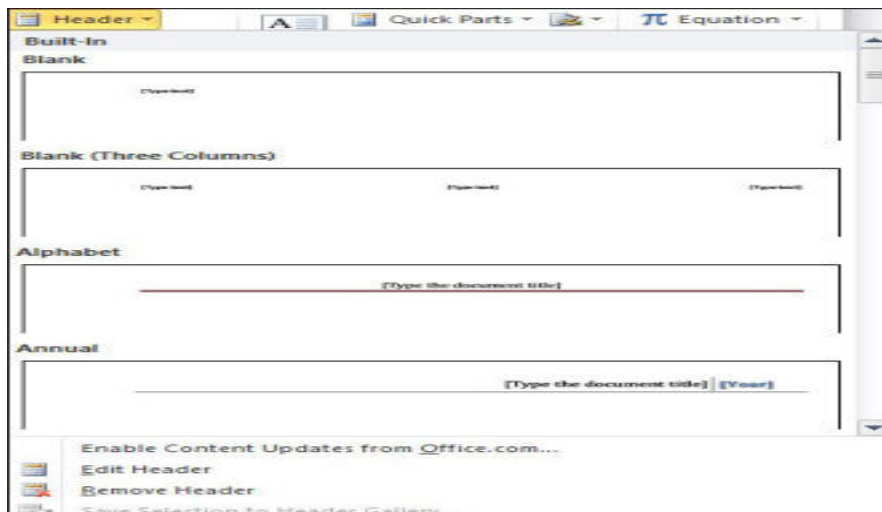
Step 1: Open the new or an existing Word document in which you want to insert header and footer.

Step 2: Go to the **Insert** tab at the top of the Ribbon.

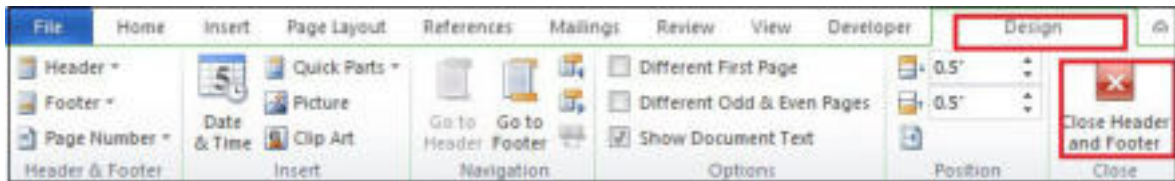
Step 3: Click on **either header or footer** drop-down menu in the **Header & Footer** section.



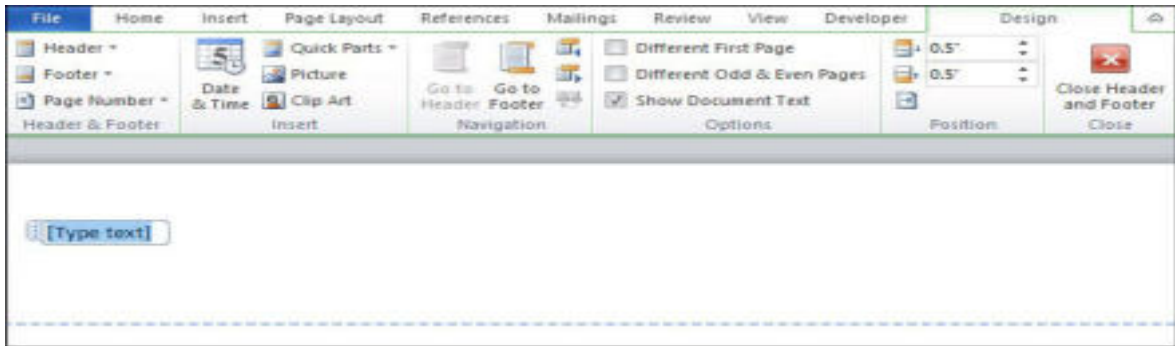
Step 4: A Header or Footer drop-down menu will display on the screen with a list of built-in Header or Footer options. Select your desired option from the Built-in list.



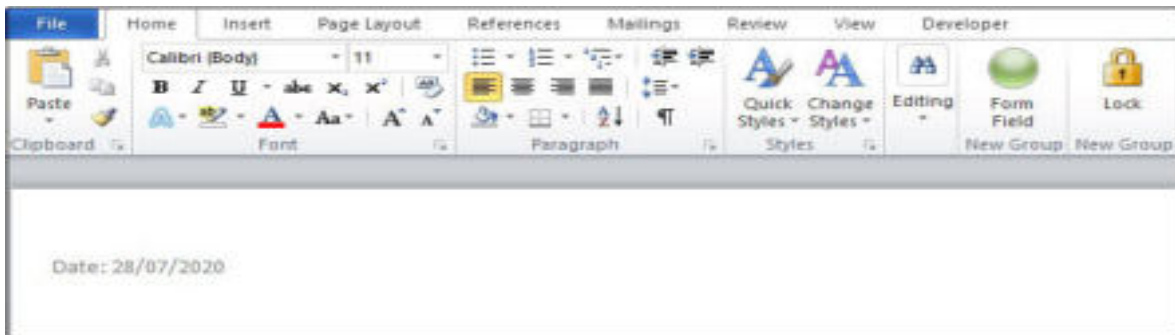
Step 5: A **Design** tab with Header & Footer option will appear at the top of the document (on the Ribbon), as shown in the below screenshot.



Step 6: Type your desired information into the header or footer section.



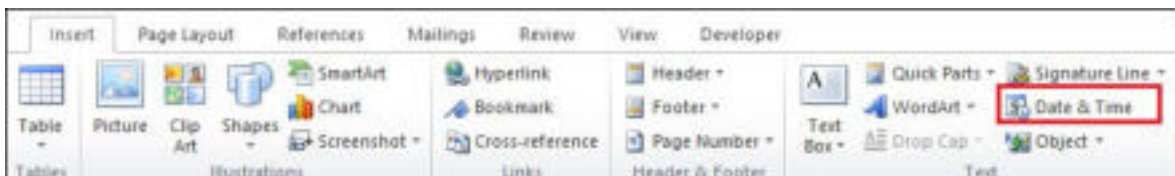
Step 7: Once you type your desired text in the Header section, click on Close Header and Footer under the Design section on the Ribbon or press the Esc key from the keyboard to remove the dotted underline. Now, you can see that the Header is inserted to the Word document.



Insert the Date or Time in a Header or Footer

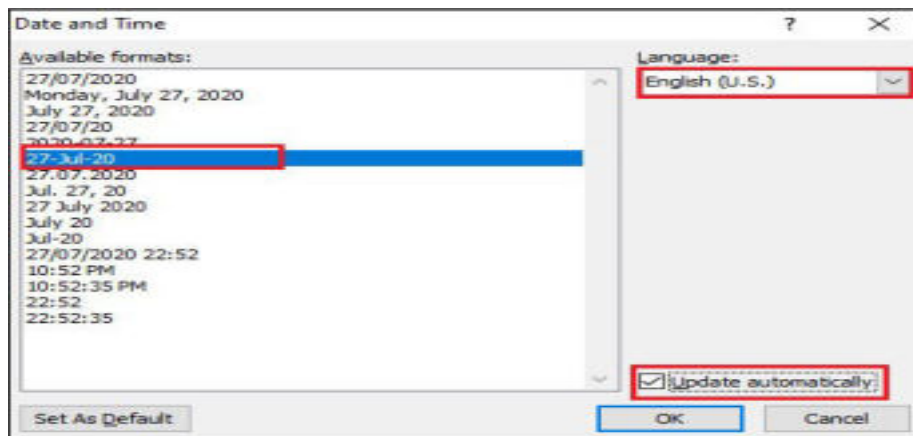
To insert the Date or Time in a Header or Footer, follow the below instructions -

1. Go to the **Insert** tab on the Ribbon and click on the **Date & Time** option in the **Text** section.



2. A Date and Time dialog box will appear on the screen in which do the following -

- Select Date format from the Available format.
- Select your desired language.
- Tick on the Update automatically checkbox.
- Click on the OK button at the bottom of the dialog box.



Now, you can see that your selected format will appear on the Word document.

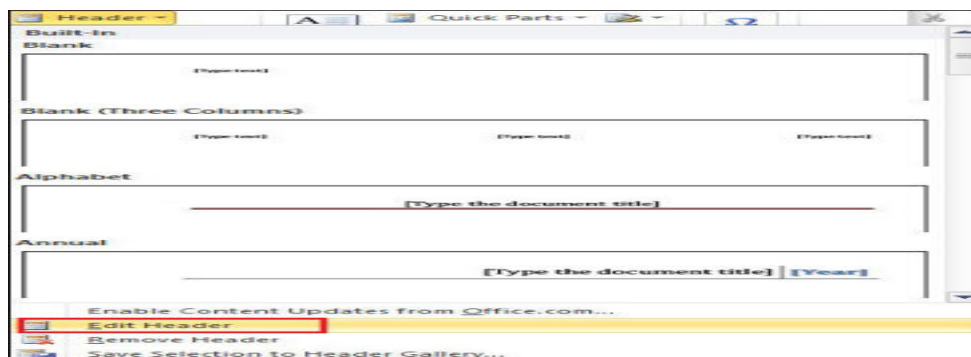
Edit Header and Footer in Word document

Once you create Header and Footer in Word document, you can also edit it based on your requirement.

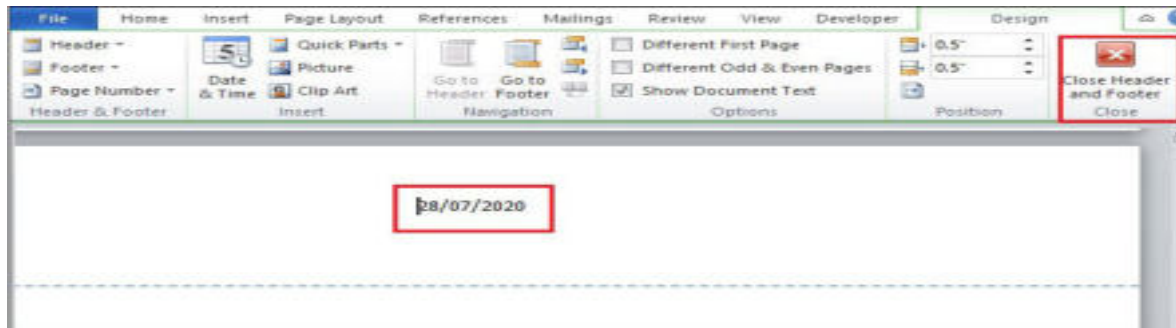
There are the following steps to edit Header and Footer in Word document.

Step 1: Go to the **Insert** tab on the Ribbon and click on **either Header or Footer** drop-down menu that you want to Edit.

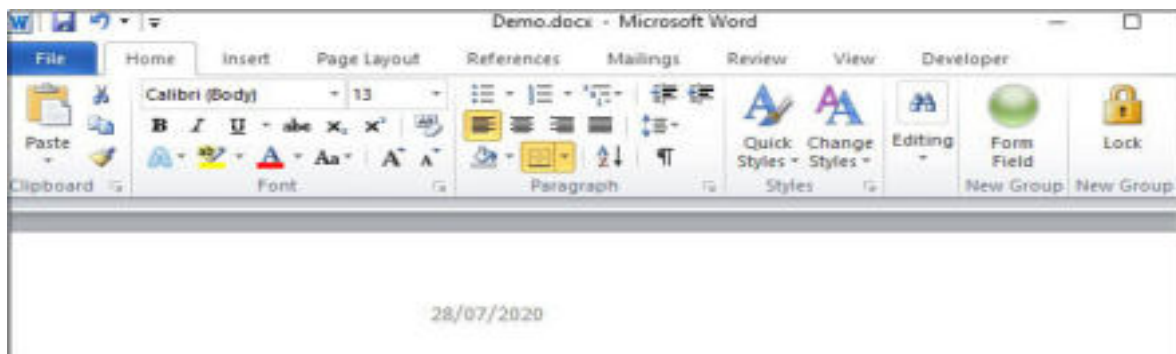
Step 2: A **Built-In** Header option window will appear on the screen. Click on the **Edit Header** option.



Step 3: Edit Header based on your requirement. Once you edit Header, click on the **Close** Header and Footer option at the top right corner of the document to disappear the blue dotted lines.



Now, you can see that Header is edit based on your requirement.



Delete Header and Footer from Word document

To Delete Header and Footer from Word document, follow the below steps -

1. Go to the **Insert** tab on the Ribbon and click on the **Header & Footer** option.
2. A Header or Footer dialog will appear on the screen. Click on the **Remove Header** or **Remove Footer** option.



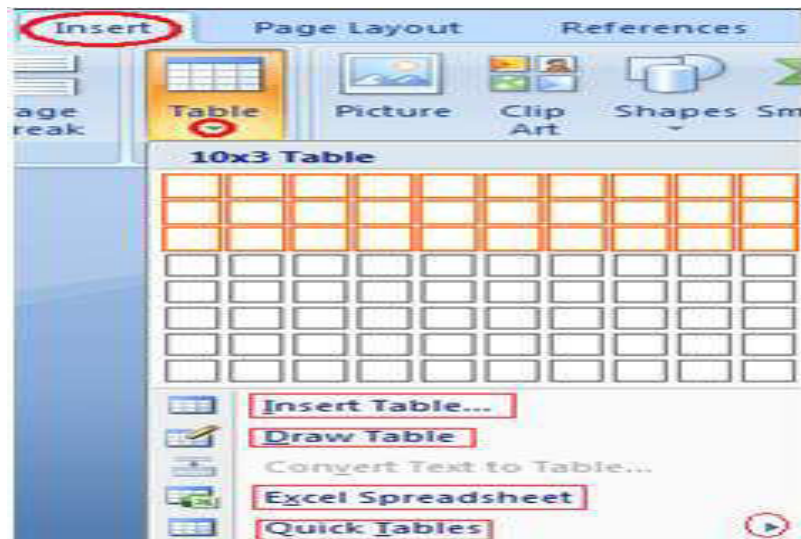
Working with Tables

How to Insert Table in MS Word

Table is a versatile tool of MS Word. It allows you to organize your information, i.e. you can align text, present numerical data and create forms and calendar. The steps to insert table are given below;

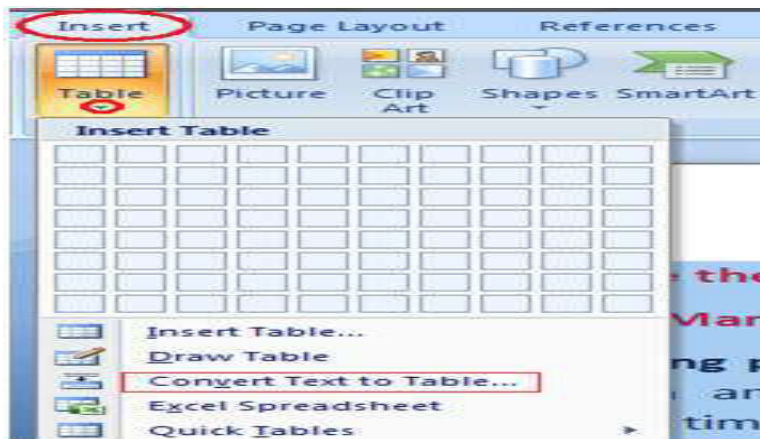
- Place the cursor where you want to insert the table
- Select the Insert tab
- In Tables group click the Table command
- It displays different options to insert the table
- Select the desired option to insert the table

See the image:



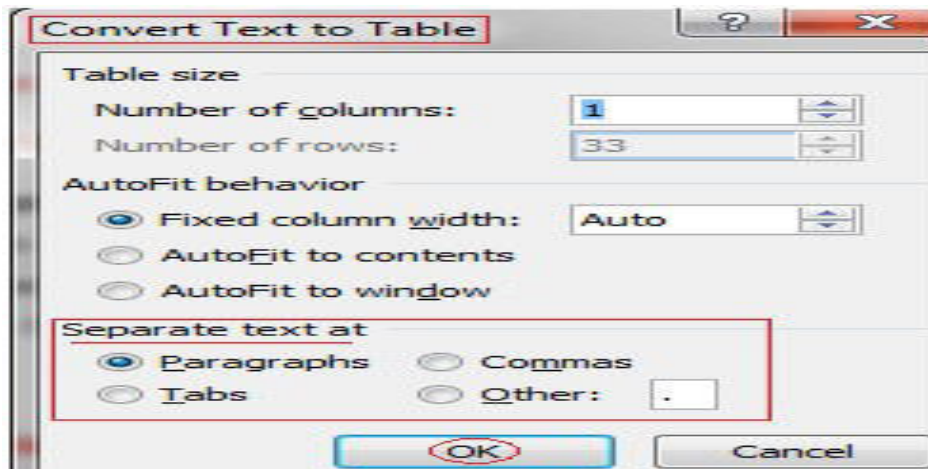
To Convert Text to Table

- Select the text
- Select the Insert tab
- In Tables group click the Table command
- Select the 'Convert Text to Table' option



- It displays a dialog box
- In 'Separate text at' section select the desired option
- Click OK, the text will convert to a table

See this image:



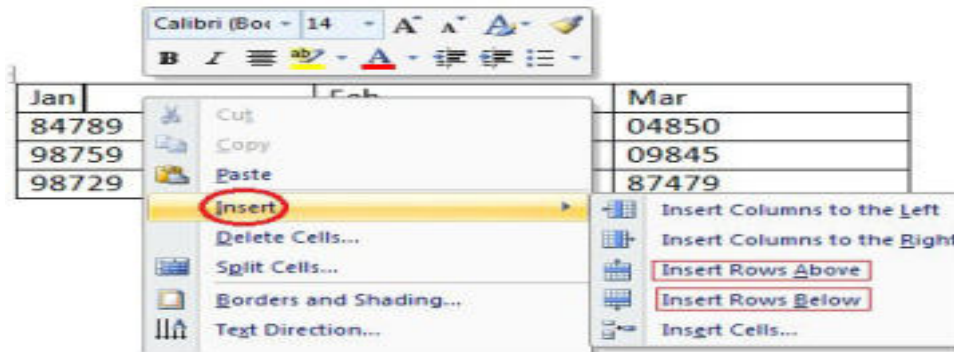
How to Add Row in Table

If you want to increase or add a new row in your table, you can follow the steps given below;

- Place the cursor in a row above or below which you want to add row
- Right click the mouse
- A menu appears

- Place the arrow over Insert option
- It will display a menu
- As required select 'Insert Rows Above' or 'Insert Rows Below'

See this images:

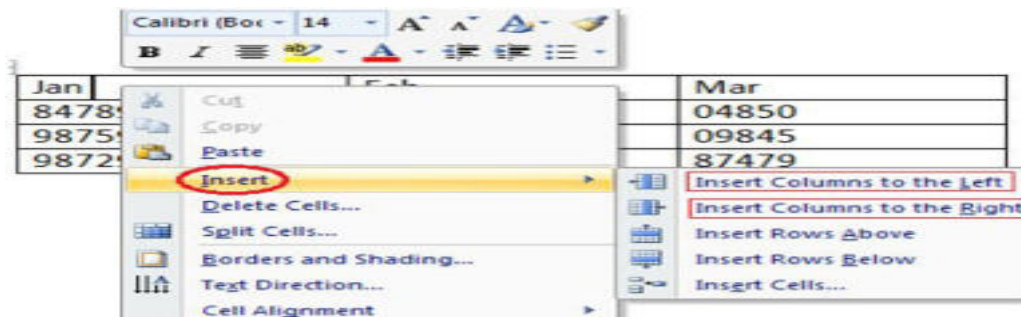


How to Add Column in Table

If you want to increase or add a new column in your table, you can follow these steps;

- Place the cursor in the column adjacent to which you want to add the column
- Right click the mouse
- It displays a menu
- Place the arrow over Insert option
- It shows a list of commands
- As required select 'Insert Columns to the Right' or 'Insert Columns to the Left'

See the image:

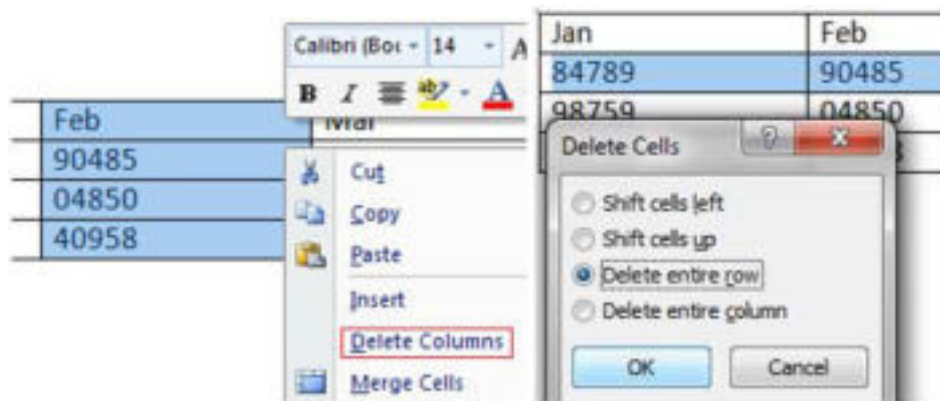


How to Delete Column or Row in Table

The table command also allows you to delete a column or row in your table. You can delete the unwanted columns or rows by following these steps;

- Select the column or row of the table
- Right click the mouse
- A menu appears
- As required select 'Delete Columns' or 'Delete Rows'

See the image:



How to Modify Table

Word allows you to customize tables as per your requirement. You can modify your table in different ways, i.e. you can choose a table style, table design, draw borders. The steps to modify a table are given below;

- Select the table
- Two new tabs Design and Layout appear on the Ribbon
- On Design tab you will see three groups of commands to modify table; Table Style Options, Table Styles and Draw Borders:



- Layout tab has six groups of commands to format table

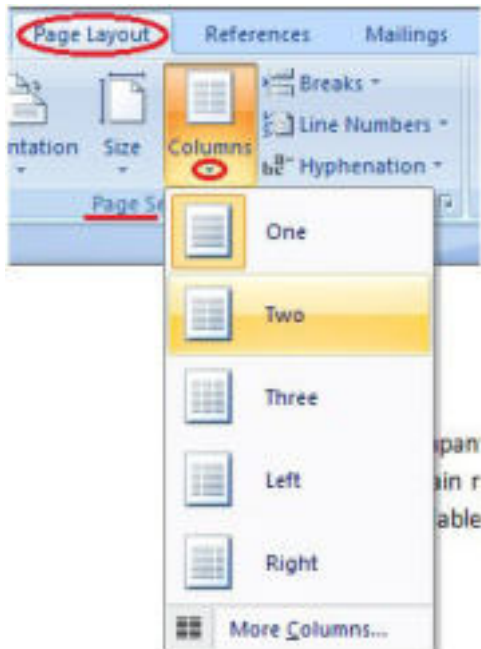


How to Split Text to Columns

You can split your text to columns as shown in the image given below. It helps you make your text more readable and presentable. The steps involved in this process are given below:

- Open the document
- Select the Page Layout tab
- In Page Setup group click the Columns command
- It displays a list of options to split text into columns
- Select the desired option

See the image:

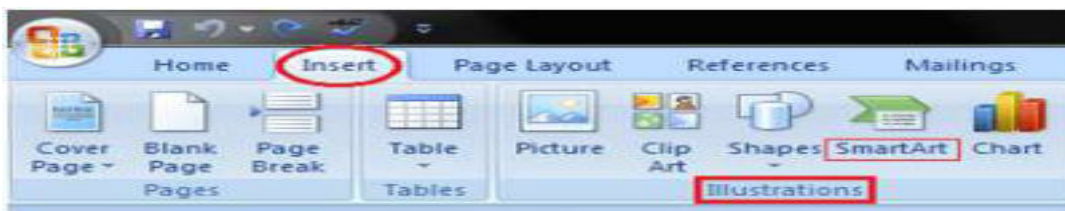


Inserting Illustrations

How to Insert Smart Art Graphics

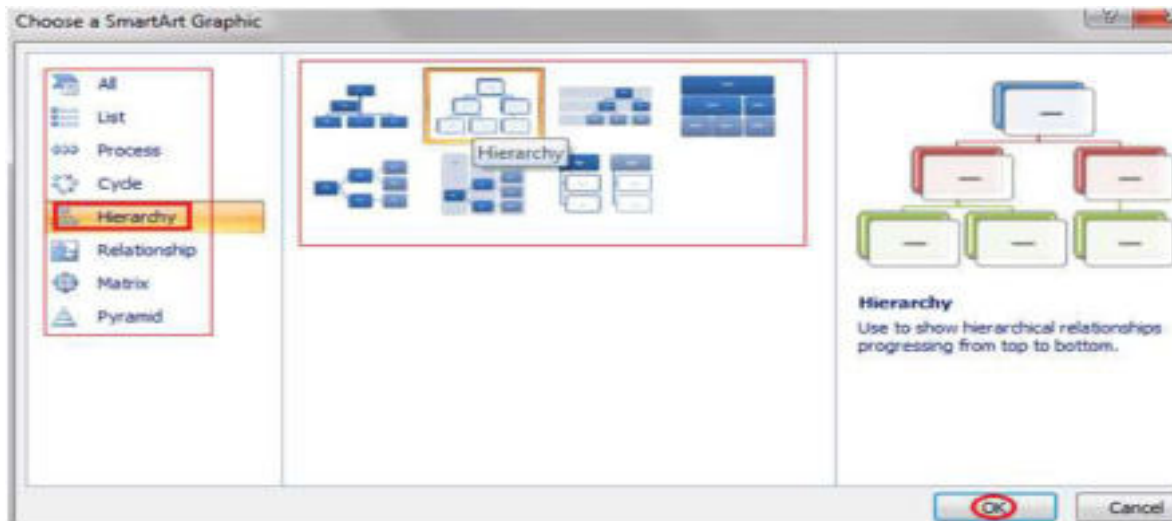
You can insert Smart Art Graphics in your document to effectively communicate your message. You can insert a process flow, a relationship or an organization hierarchy. The steps to insert smart art graphics are given below;

- Place the cursor in the document where you want to insert graphics
- Select the Insert tab
- In Illustrations group select the 'Smart Art' command



- On left side, a dialog box appears with list of categories
- In the center you will see the illustrations available in a category
- Select the desired illustration and click Ok

See the image:

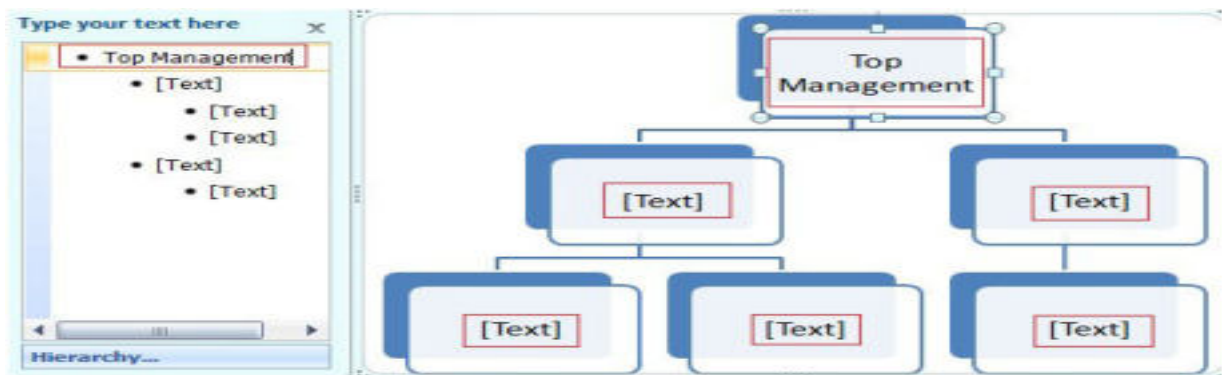


How to Add Text in Smart Art Graphics

Word also allows you to add text in Smart Art Graphics; you can add facts, figures and any other information. The steps to add smart art graphics are given below;

- Select the graphic
- Task pane appears on the left side if it is not visible then click the arrow on left side of graphic
- Type the text in task pane fields it will appear in the graphic
- You can also directly type the text in the graphic in text area
- Close the task pane after entering the text and click outside the graphic

See the image:



How to Insert Picture in Document

Pictures make our text more attractive and readable. You can insert relevant pictures in your text by following these steps:

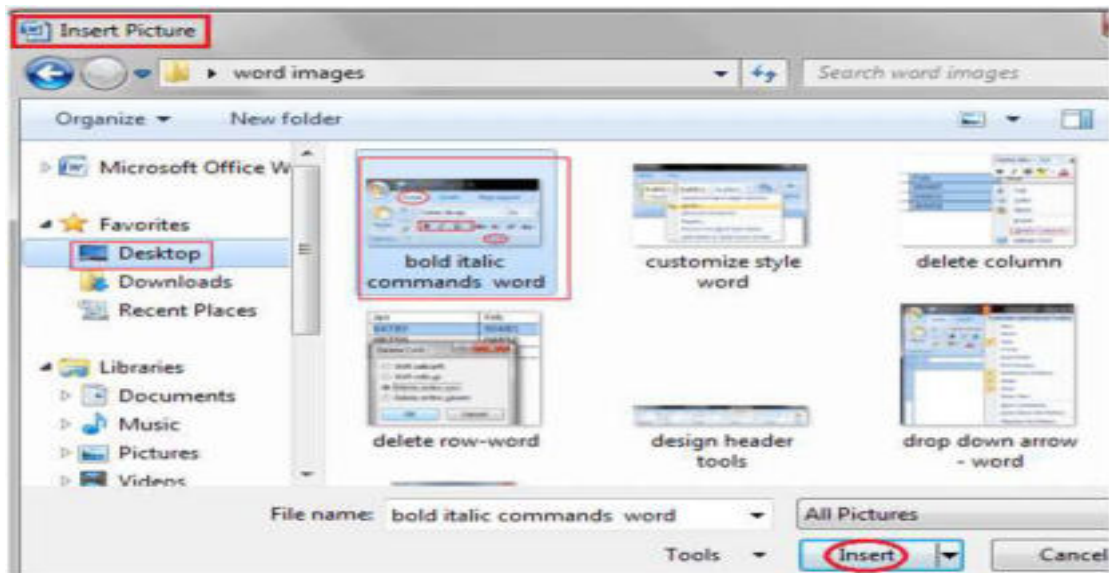
- Place the cursor where you want to insert the picture
- Select the Insert tab on Ribbon
- In Illustrations group click the Picture command



- It displays 'Insert Picture' dialog box

- Select the desired image
- Click Insert to insert the picture

See the image:



How to Insert Clip Art

Clip art refers to a graphic or a picture that you can insert in your document. It comes in different formats and styles. It is used to enhance the appearance of a document. The steps to insert a clip art are given below;

- Place the cursor where you want to insert the clip art
- Select the Insert tab
- In Illustrations group click the Clip Art command



- A task pane appears on the right side of document
- Enter the keyword in 'Search for' field and select the suitable option in 'Search in' and 'Results should be' fields

- Click Go, clip art menu will appear
- Select the desired clip art with a left click

See the image:

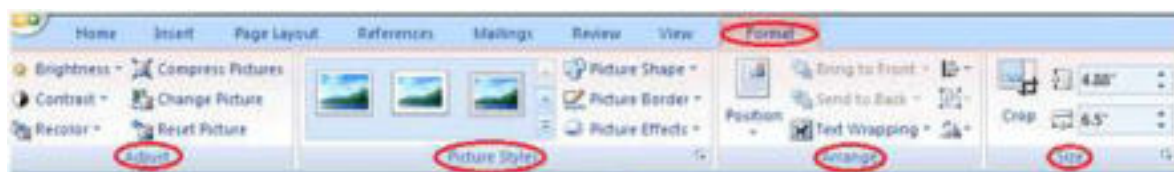


How to Format Picture or Clip Art

You can change the appearance of your picture or clip art to enhance its visual impact. The format tab offers you various options for formatting pictures like contrast, changing color, line style, cropping, etc. The steps to format picture or clip art are given below;

- Select the picture or clip art that you want to format
- Format tab appears in the Ribbon
- Click the Format tab
- It displays four groups of related commands to modify or format picture or clip art

See the image:



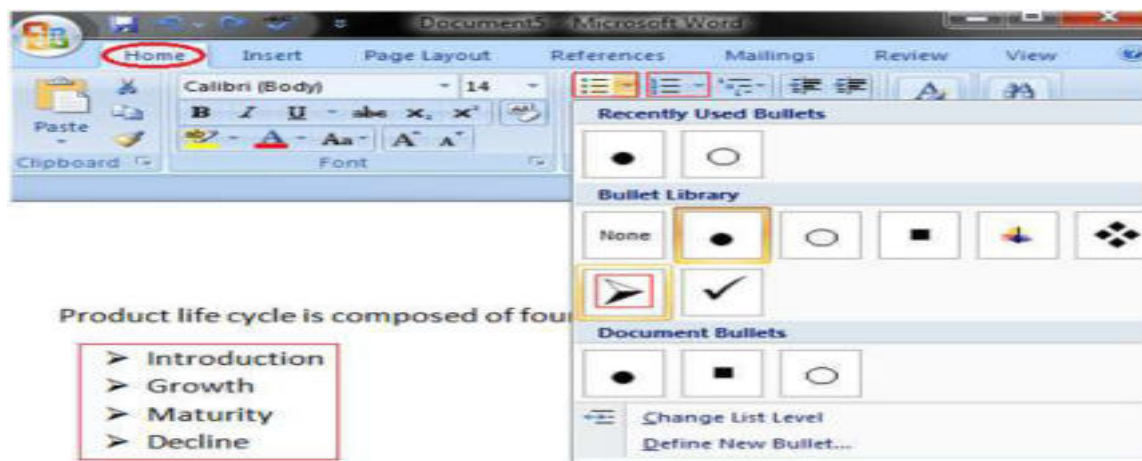
Working with Lists

How to Create Bulleted and Numbered Lists

Bullets and numbers are used to make a list more presentable and readable. A bulleted list attracts more than a simple list. Word offers you various styles of bullets and numbers. The steps to create bulleted lists are given below;

- Select the text you want to convert to bulleted or numbered list
- Select the Home tab
- In Paragraph group click the Bullets or Numbering command
- It displays Bullets or Numbering menu
- With a left click select the desired Bullet or Numbering style
- To increase the list place the cursor at the end of list and press Enter key

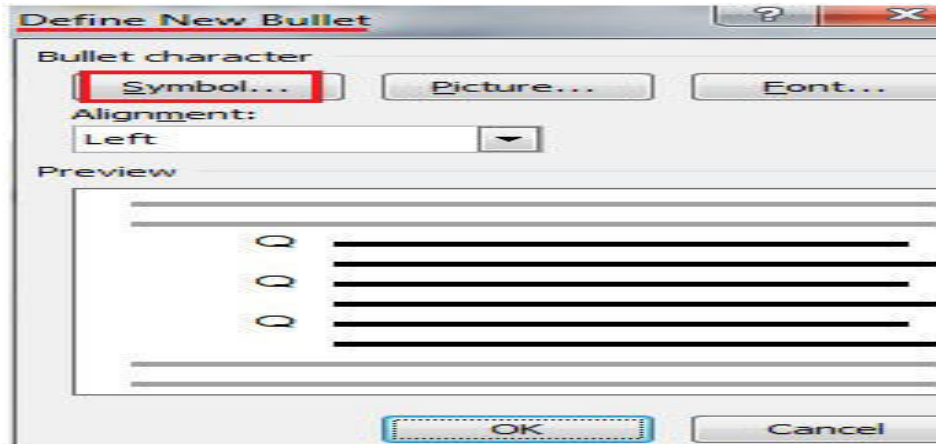
See the image:



How to Use Symbols as Bullets

You can replace the bullets with symbols to make your lists more meaningful and attractive. The steps involved in this process are given below;

- Select the text to convert to bulleted list
- Select the Home tab
- In Paragraph group click the Bullets command
- Click 'Define New Bullet' option
- 'Define New Bullet' dialog box appears



- Click the Symbol command
- It displays 'Symbol' dialog box
- Select the desired symbol
- For more options click the drop down arrow next to Font field

See the image:

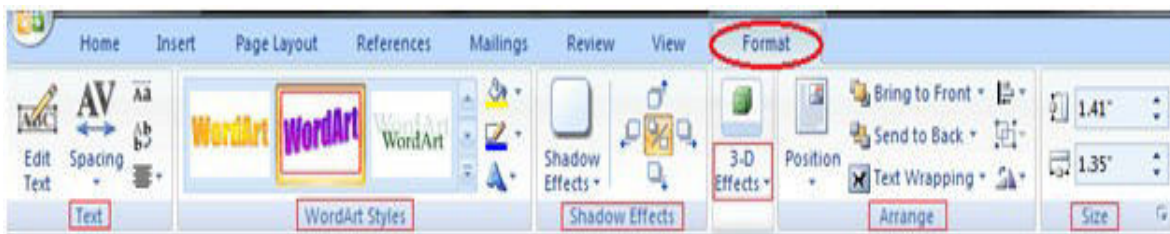


How to Format WordArt

Word also allows you to customize WordArt. You can change WordArt shapes, font face and size, and colors as per your requirement. The steps to format WordArt are given below;

- Select the word art in the document
- Format tab appears in the Ribbon
- It offers five groups of related commands
- Click the suitable command to make desired changes in word art

See the image:



(Ref: www.google.com)

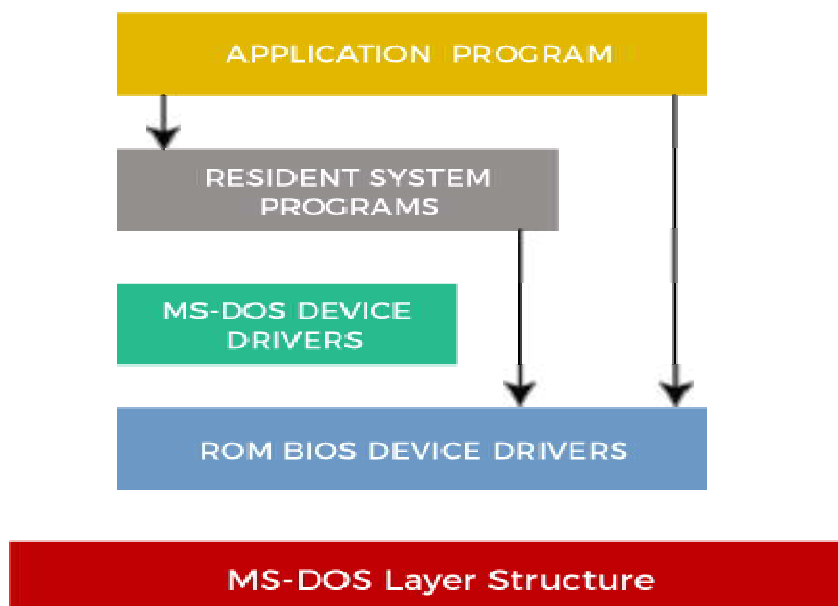
MS-DOS Operating System

A disk operating system (DOS) is an operating system for x86 based personal computers mostly developed by Microsoft. MS-DOS, it's rebranding as IBM PC DOS, and some operating systems attempting to be compatible with MS-DOS. Sometimes it is referred to as "DOS", which is also the generic acronym for disk operating system.

MS-DOS was the main operating system for IBM PC compatible personal computers during the 1980s. It was gradually superseded by operating systems offering a graphical user interface (GUI) in various graphical Microsoft Windows operating system generations.

DOS is also used to describe several similar command-line disk operating systems. Early computers, such as the Commodore 64, Atari 800, and Apple II, all featured a disk operating system, including Commodore Business Machines DOS, Atari DOS, and Apple DOS, respectively. DOS/360 was an OS for IBM mainframes, which first appeared in 1966, but it is unrelated to the 8086-based DOS of the 1980s.

What is MS-DOS ?



Several competing products were released for the x 86 platforms during its lifetime, and MS-DOS went through eight versions until development ceased in 2000. Initially, MS-DOS was targeted at Intel 8086 processors running on computer hardware using floppy disks to store and access the operating system, application software, and user data.

Progressive version releases delivered support for other mass storage media in ever greater sizes and formats and added feature support for newer processors and rapidly evolving

computer architectures. Microsoft's development was the key product from a programming language company to a diverse software development firm, providing essential revenue and marketing resources. It was also the underlying basic operating system on which early versions of Windows ran as a GUI.

How DOS works?

When a computer is powered on, it goes through various steps called the **boot process**. For a computer running a disk operating system in the following six steps, such as:

1. The read-only memory (ROM) bootstrap loader reads the Master Boot Record and passes control over to it.
2. The boot record loads the disk operating system into memory, and it takes control of the machine.
3. The computer transfers data stored on a magnetic disk to its main memory, the random access memory.
4. It also transfers data to external devices attached to the computer, such as a computer screen or printer.
5. The computer provides various applications programming interfaces for programs like character input/output, memory management, program loading, termination, and handling input from the user through a keyboard.
6. The OS also provides file management that organizes, reads, and writes files on storage. The files are organized in a hierarchical structure of directories, subdirectories, and files.

A disk operating system doesn't have a graphical user interface (GUI). Its interface is character-based, so users must type commands in the command line to indicate what actions they want.

Features of DOS

Here are some of the distinguishing features of a disk operating system, such as:

- MS-DOS does not offer GUI (Graphical User Interface) and doesn't accept mouse inputs. It is a character-based interface system where all commands are entered in the text at the command-line prompt.
- A disk operating system manages files, folders and allows program loading and execution. It can control hardware devices such as disk, memory and allocate resources.
- MS-DOS offers a file system to organize, read and write files to the disk storage.
- It is a single-user operating system and performs various tasks to ensure the proper operation of systems.

- It uses a 16-bit file allocation table (FAT16), and a 16-bit interface is used to define the location of the memory of each file uniquely. These identifiers are stored in a tabular format with the name File Allocation table.
- MS-DOS does not support a multiuser operating system, and it is less secure and does not have a concept of user roles. It is very lightweight due to its basic interface and limited features.

Limitations of MS-DOS

Here are the following limitations of the disk operating system, such as:

- **Built-in security:** DOS does not have built-in security, such as file ownership and permissions.
- **No multiuser or multitasking:** It also does not support multiuser or multitasking. It can only run one program at a time, but it provides direct access to the basic I/O system and underlying hardware.
- **Challenging interface:** A user must type in commands and remember commands to run programs and other OS tasks. For example, typing the command **cd** *\directory_name* changes the current working directory to the named directory, and typing the command dir lists the files in the current directory. This approach makes it difficult for beginners to use.

Types of MS-DOS Commands

An instruction given to a computer to perform a specific task is known as a command. The MS-DOS has many commands to perform each task, and these commands are stored in the DOS directory of the disk. The MS-DOS commands are of two types, internal command, and external command.

1. **Internal Command:** Internal commands are built-in commands of MS-DOS, stored in the command interpreter file (COMMAND.COM). These commands reside in memory if the system is at prompt (C:\>) level. Some of the internal commands are DATE, TIME, DIR, VER, etc.
2. **External Command:** External commands are separate program (.com) files that reside in the DOS directory.

| Commands | Description |
|----------|-------------|
|----------|-------------|

| | |
|---------------------|--|
| Dir | List all files of specific directories or subdirectories. |
| CD or CHDIR | Navigate or move to a specific directory. |
| RD or RMDIR | Remove directory. |
| TREE | Display all directory paths. |
| PATH | Set sequential search path for executable files. |
| SUBST | Substitutes a string alias for the pathname. |
| FORMAT | Formats a disk for DOS files. |
| COPY | Copies one or more files to another location. |
| XCOPY | Copy files and directories and their subdirectories. |
| Del | Delete files. |
| Ren
or
rename | Rename the name of a file or directory. |
| ATTRIB | Set or show file attributes. |
| BACKUP | Backup files and directories. |
| PROMPT | Customize DOS prompt. |
| Deltree | Deletes all files and subdirectories from a computer. |
| Help | Lists the available commands or more information about a specific command. |

| | |
|-------------|---|
| mkdir or md | Creates a new subdirectory. |
| Move | Moves files or directories from one directory to another or from one drive to another. |
| Type | Displays the contents of a file on the screen. |
| * | A wildcard character that represents one or more characters a group of files has in common. |
| ? | A wildcard character that represents a single character a group of files has in common. |

MS-DOS Files and Filenames

One of the primary functions of the OS is to handle disk files. A file can contain only data, or it can contain a set of instructions, called a **program**, telling the computer how to perform a particular task. Every file has associated with it a unique **filename** which is used to identify it on the disk. A filename in MS-DOS has two parts; the name and an extension.

The **name** can contain *up to eight characters*. Each filename can have a *three-character extension*. The extension is separated from the name by a period. The period serves as a **delimiter**, indicating where one portion of the filename ends, and the next begins. An extension is usually used to identify files that are related in some way. MS-DOS allows the following characters to be used in a filename and extension:

- Uppercase and lowercase case letters A through Z
- Numbers 0 through 9
- Special characters \$ # & @ () ! ^ ` ~ { }

Any other character used in a filename, including spaces, will cause the name to be terminated at that character. It is generally good practice not to use special characters in filenames because some programs may use them as delimiters or other special purposes.

In addition, the extensions BAT, COM, EXE, and SYS have special meanings in MS-DOS and should therefore normally not be used with your data files. Shown below are examples of both valid and unacceptable filenames.

Valid MS-DOS Filenames:

- BEERSLAW
- DAT
- 1
- EXP

Invalid MS-DOS Filenames:

- EXP 1.DAT (space is not allowed)
- BEERSLAWDATA (too many characters in the name)
- EXP1 (too many characters in the extension)
- HG>HE (> is an invalid character)

UNIT- II

Cloud Computing Architecture-Introduction – Internet as a Platform, The cloud reference model - Types of clouds - Economics of the cloud, Computing platforms and technologies, Cloud computing economics, Cloud infrastructure - Economics of private clouds - Software productivity in the cloud - Economies of scale: public vs. private clouds.

Cloud Computing Architecture

As we know, cloud computing technology is used by both small and large organizations to **store the information** in cloud and **access** it from anywhere at anytime using the internet connection.

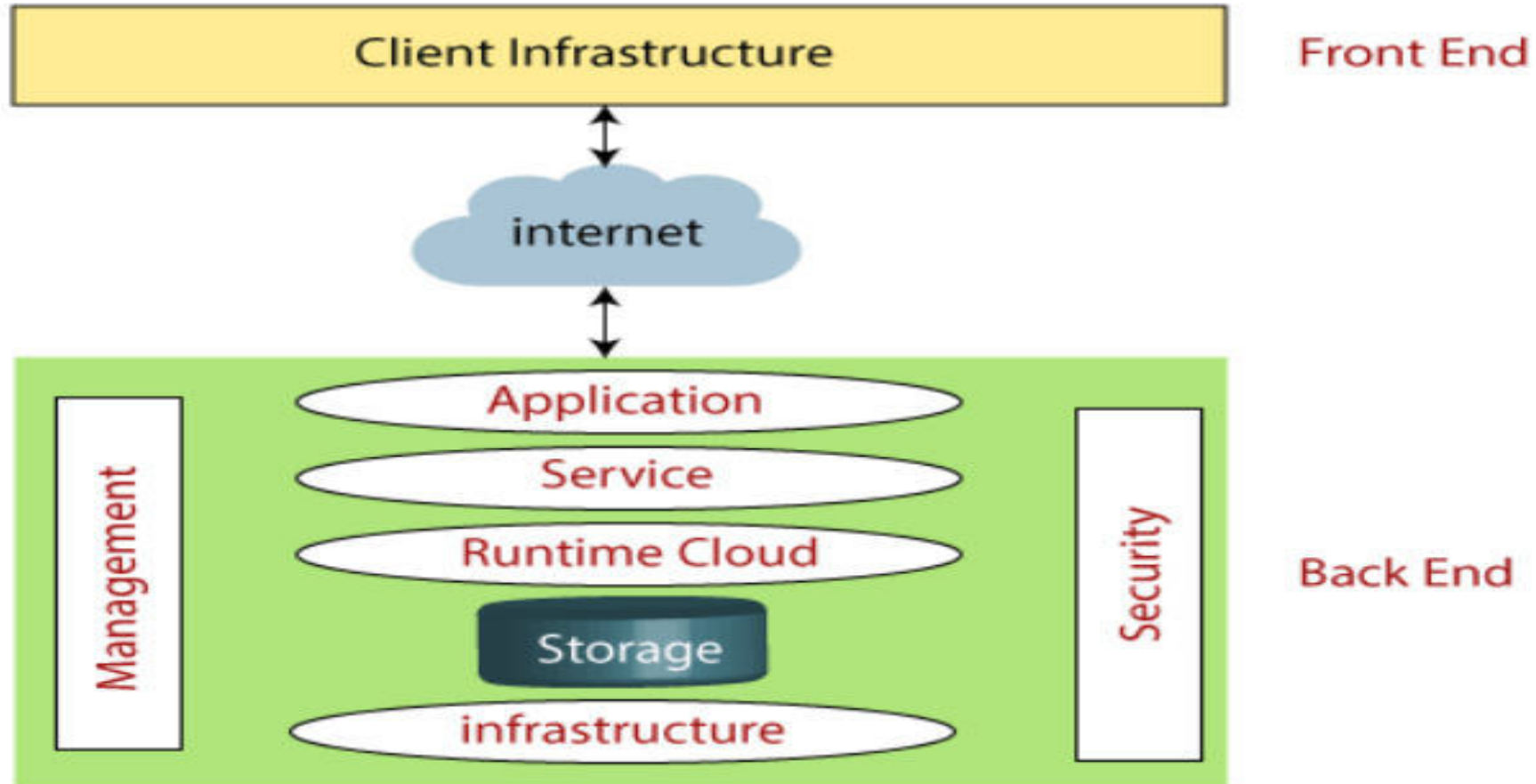
Cloud computing architecture is a combination of **service-oriented architecture** and **event-driven architecture**.

Cloud computing architecture is divided into the following two parts -

- Front End
- Back End

The below diagram shows the architecture of cloud computing -

Architecture of Cloud Computing



Front End

The front end is used by the client. It contains client-side interfaces and applications that are required to access the cloud computing platforms. The front end includes web servers (including Chrome, Firefox, internet explorer, etc.), thin & fat clients, tablets, and mobile devices.

Back End

The back end is used by the service provider. It manages all the resources that are required to provide cloud computing services. It includes a huge amount of data storage, security mechanism, virtual machines, deploying models, servers, traffic control mechanisms, etc.

Components of Cloud Computing Architecture

There are the following components of cloud computing architecture -

1. Client Infrastructure

Client Infrastructure is a Front end component. It provides GUI (Graphical User Interface) to interact with the cloud.

2. Application

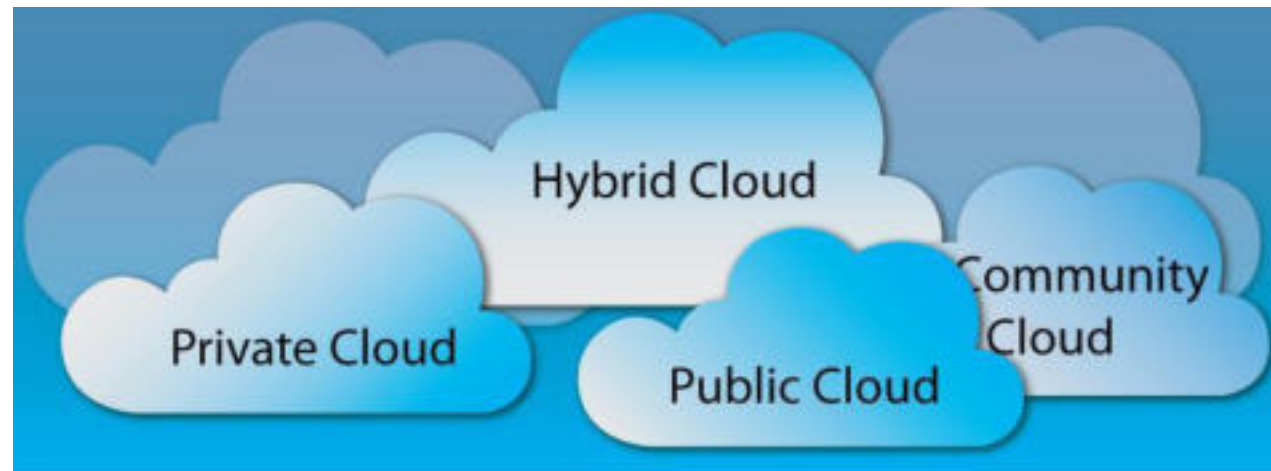
The application may be any software or platform that a client wants to access.

3. Service

A Cloud Services manages that which type of service you access according to the client's requirement.

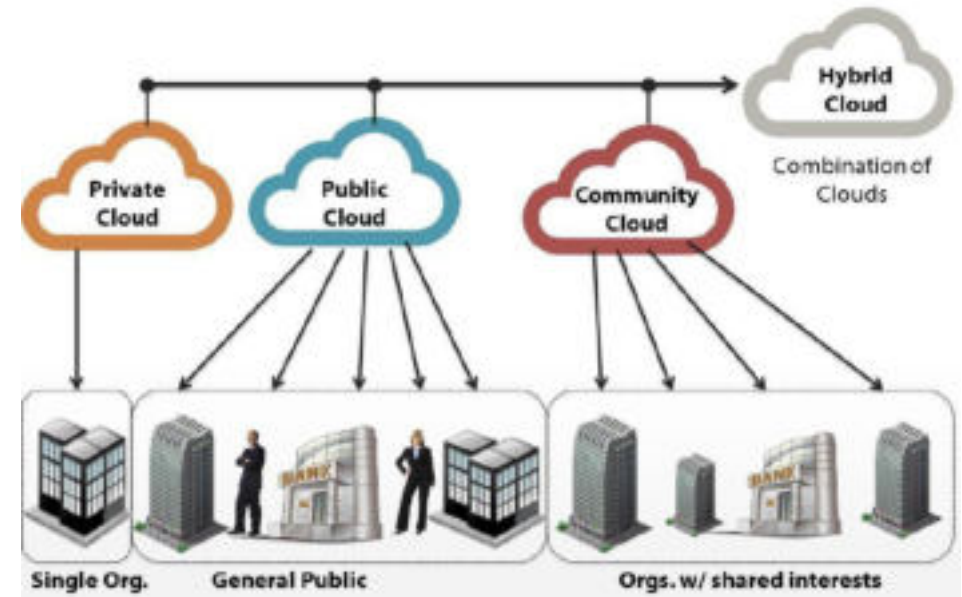
Cloud Deployment Models:

- National Institution of Standards and Technology (NIST) operate under the US Department of Commerce and has defined many of the key concepts used in cloud computing.
- NIST Special Publication 800-145 identifies four models for cloud deployments. These are:
 1. Public Cloud
 2. Private Cloud
 3. Community Cloud
 4. Hybrid Cloud



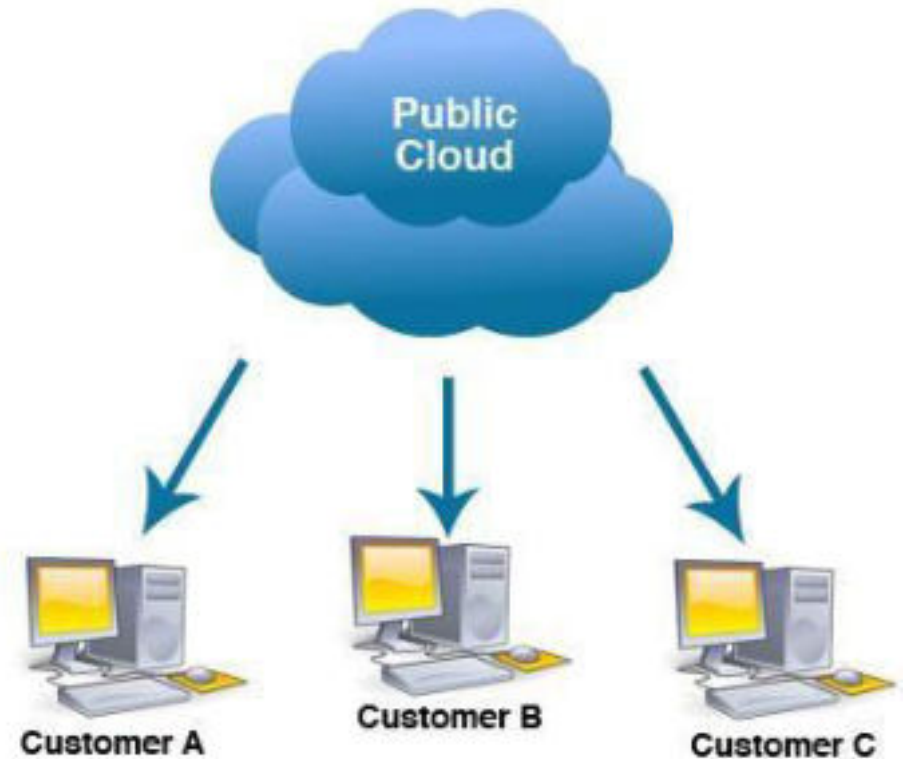
Cloud Deployment Models:

- Private cloud – owned by a single organization
- Public cloud – accessible to the public
- Community cloud – shared by a group of organizations
- Hybrid cloud – a combination of the above three clouds



Public Cloud

- Public clouds are external publicly available environments accessible to multiple tenants.
- The cloud provider is responsible for the creation and the on-going maintenance of the public cloud and its IT resources.
- Public clouds are ideal for individuals, startups and other organizations with financial restrictions.
- Some of the Public cloud examples include Amazon Web Services, Microsoft Azure, IBM Cloud and Google Cloud Platform.



Public Cloud
Providers



Google Cloud Platform



The Case of Public Cloud

Consider this,

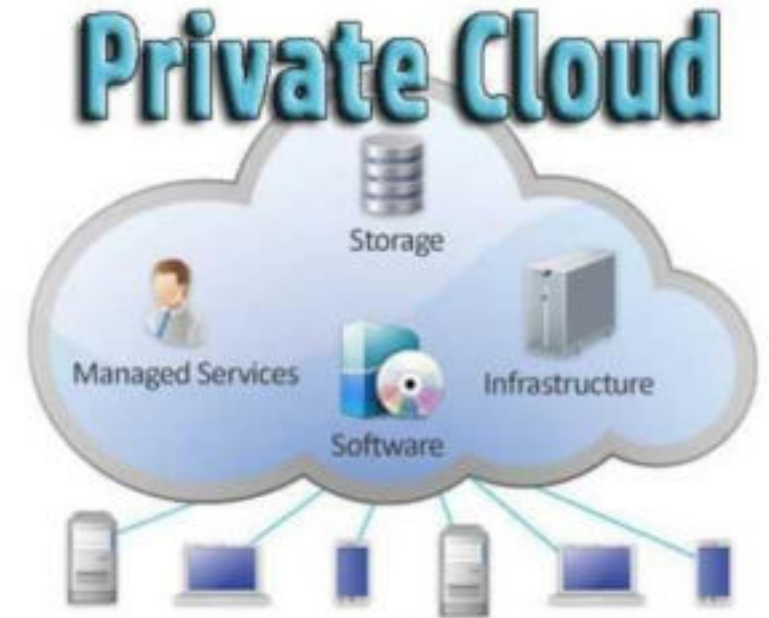
Connected Solutions is a startup venture started by four classmates who are interested in developing IoT (Internet of Things) solutions. However, when it comes to investing in infrastructure like storage and networking, finance is an issue. What would Connected Solutions do?

- If you are limited in financial resources but still require a good amount of space to store your data, using a public cloud is the best option.
- There are good public cloud pricing plans that offer good amount of storage.
- Additionally, public clouds tend to use the latest technological advancements (hardware, software and so on), have better utilization rates, provide greater elasticity and continually test your applications and networks.



Private Cloud

- A Private Cloud is owned by a single organization.
- Private cloud helps an organization in centralizing the access of IT resources by its various locations and departments.
- Private clouds are ideal especially for organizations that have heavily invested in infrastructure.



The Case of Private Cloud

Consider this as an example,

You are a financial organization with a big infrastructure. You deal with sensitive information, hence, security is of utmost importance. You want your infrastructure to be centralized, secure and to have a greater control over it. What is your best option?

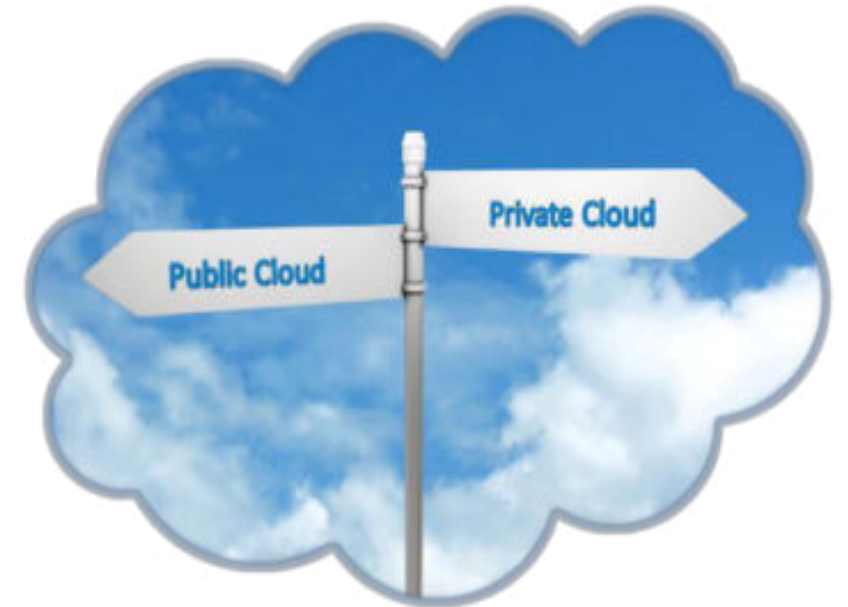
- A private cloud though expensive, comes with greater security and control.
- A private cloud is custom designed by you for your business.
- It allows for rapid scaling of business operations.
- And most importantly, when issues arise you can prioritize and handle them unlike a public cloud where the delays in resolving an issue may pile up.



Public Cloud vs. Private Cloud

Private clouds combine the security of keeping resources inhouse with the scalability of the cloud by storing resources privately. Therefore, they are an ideal solution for companies with compliance requirements who cannot host their resources on a public cloud.

However, public clouds are best when facing financial restrictions and are not dealing with sensitive information.





Public Cloud vs. Private Cloud



VS






 Publically Shared
Virtualised Resources

 Privately Shared
Virtualised Resources

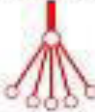

Supports multiple
customers 

Cluster of dedicated
customers 

 Supports connectivity
over the internet

 Connectivity over
internet, fibre and private network 

Suited for less
confidential information 

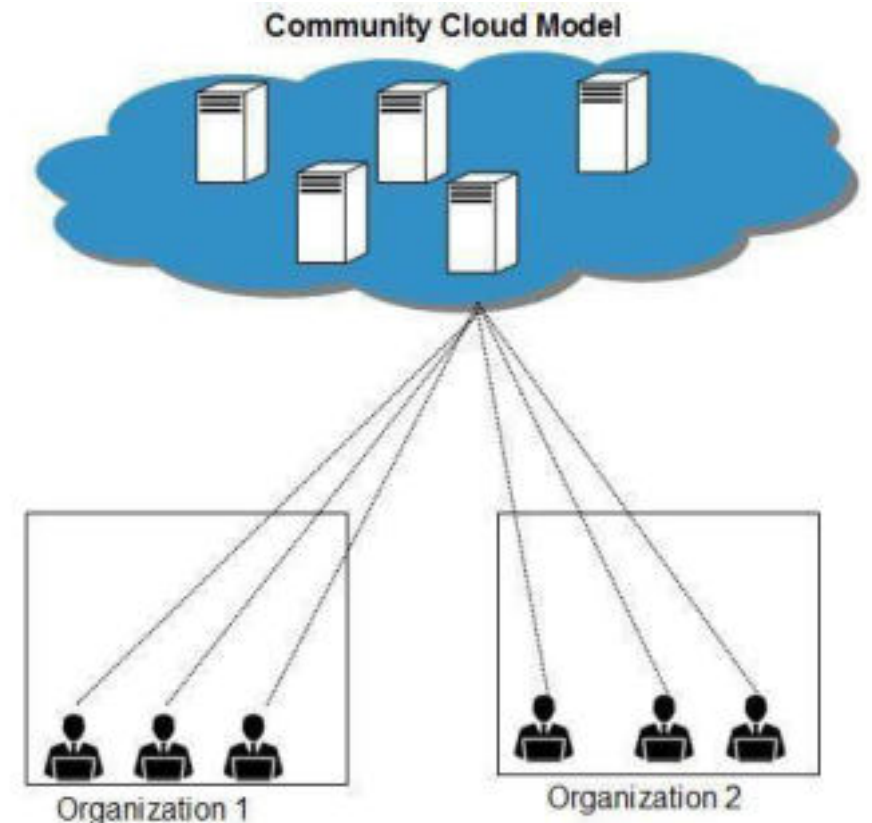
 Suited for secured
confidential information
& core systems 

Community Cloud

- Community cloud model is shared by a group of organizations with similar requirements such as security, compliance and IT policies.
- The community cloud may be jointly owned by the community members or by a third-party cloud provider that provisions a public cloud with limited access.

The Case of Community Cloud

- For example, the Department of Defense and some intelligence agencies have launched data centre improvement initiatives using a community cloud. This enables them to easily share data with other agencies in the community cloud, while still keeping it secure and controlled.

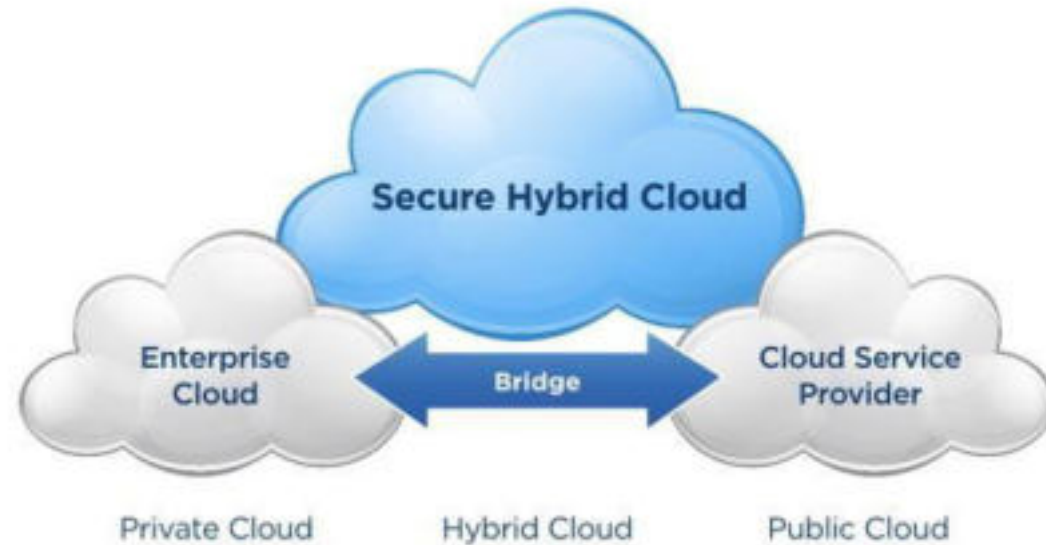


- Another industry taking advantage of community clouds is the healthcare industry, where everyone in the community can support patients and exchange data in a controlled way.
- Uses of community cloud:
 1. Allows for easy sharing and collaboration
 2. Lowers costs



Hybrid Cloud

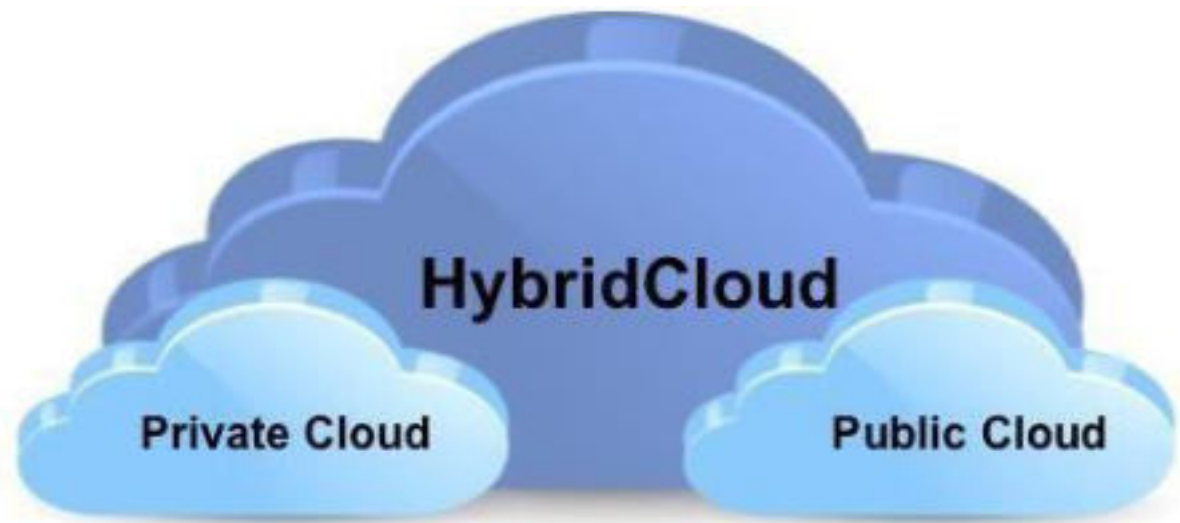
- Hybrid cloud is a combination of two or more models, private cloud, public cloud or community cloud.
- For example, a cloud consumer may choose to deploy cloud services processing sensitive data to a private cloud and other less sensitive cloud services to a public cloud. The result of this combination is a hybrid deployment model.



The Case of Hybrid Cloud

Consider a business with dynamic work load that experiences significant hike in business during holiday season. They require more resources to handle the peak season. However, these resources are not required for the rest of the year. What would be the best option?

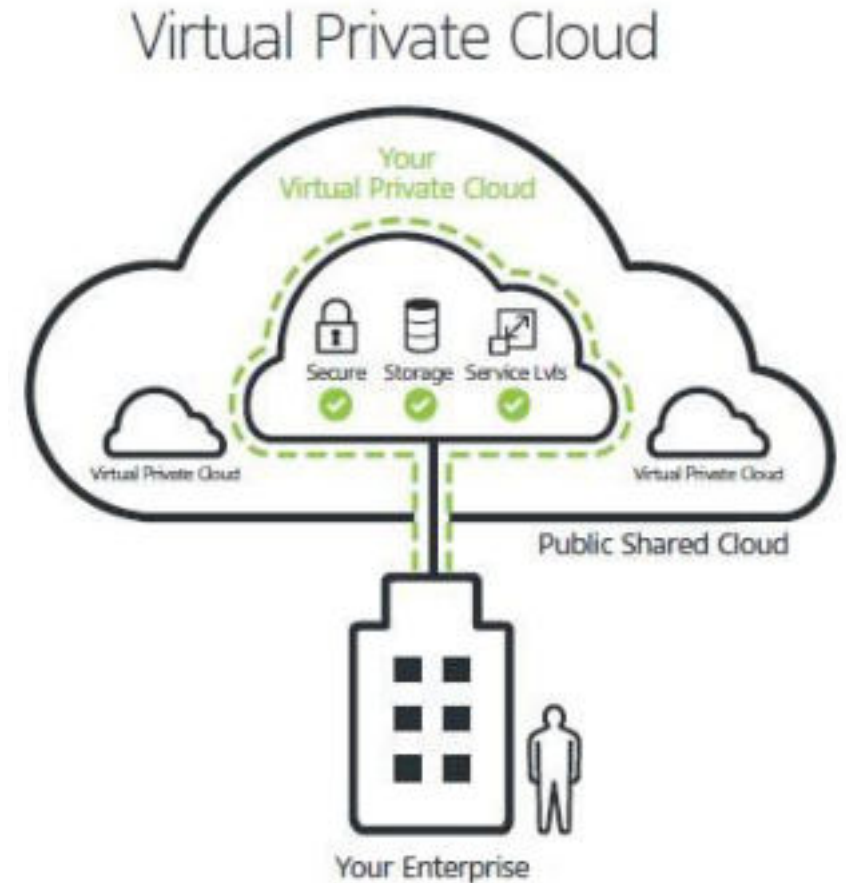
A hybrid cloud - A private cloud for off season operations and public cloud services for the holiday season is an ideal combination for such a business.



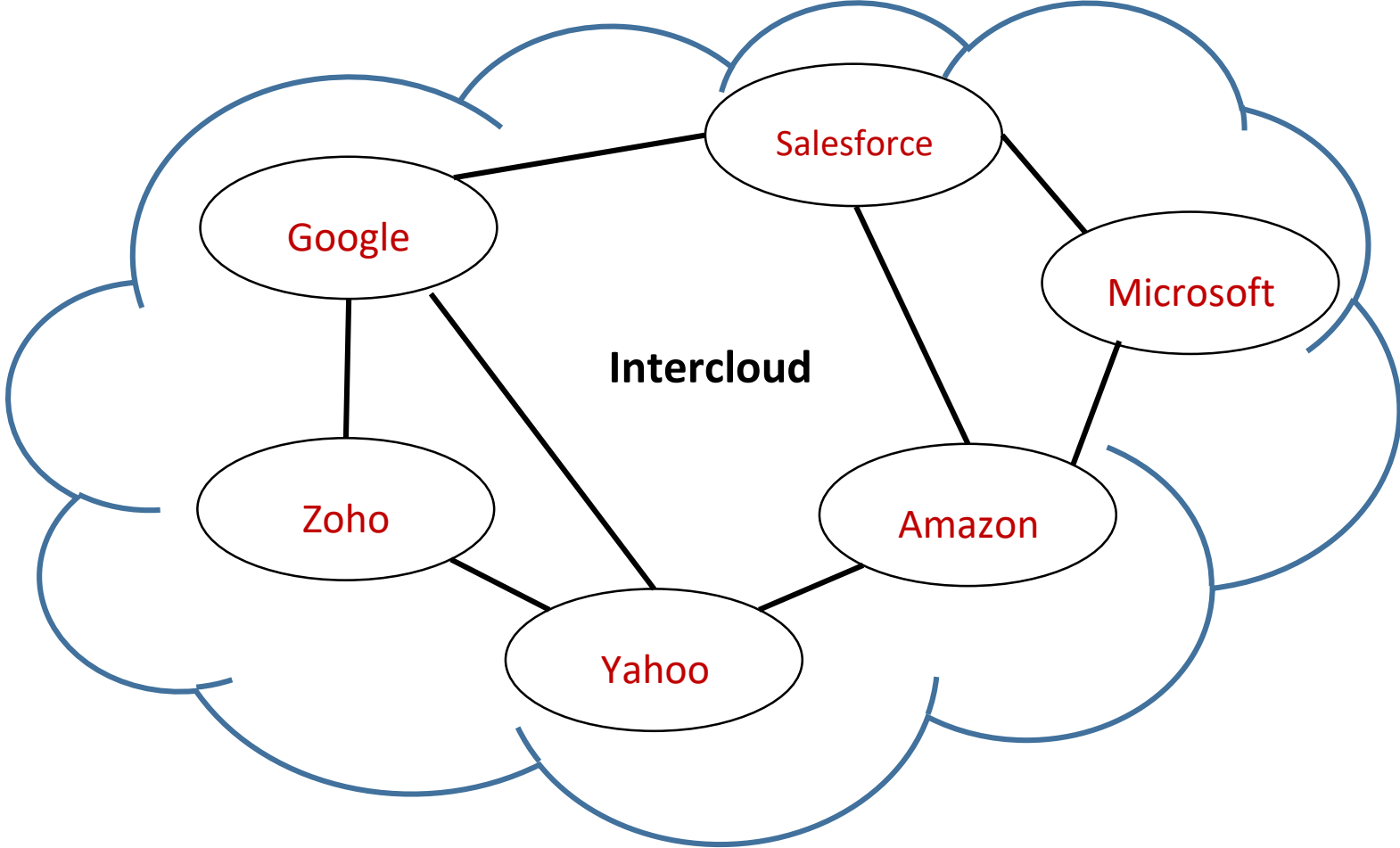
Other Deployment Models:

There are other variations of the basic cloud deployment models such as:

Virtual Private Cloud: also called ‘dedicated cloud’ or ‘hosted cloud’ is a self-contained cloud environment hosted and managed by a public cloud provider.



Inter-Cloud: comprised of two or more interconnected clouds.



TYPES OF CLOUD COMPUTING



PUBLIC

- ✓ Scalable
- ✓ Reliable
- ✓ Inexpensive
- ✓ Location Independent

PRIVATE

- ✓ Scalable
- ✓ Secure
- ✓ Flexible
- ✓ Greater control

HYBRID

- ✓ Scalable
- ✓ Secure
- ✓ Flexible
- ✓ Cost effective

Advantages of Cloud Computing

1. **Cost Efficient** – the pay-as-you-go model significantly minimizes the organization's costs.
2. **Almost Unlimited Storage** – Using cloud storage means unlimited storage capability. No running out of storage or no need of investing in storage devices.
3. **Backup and Recovery** – all the data are backed up on to the cloud. Hence, backup and recovery becomes more easier.
4. **Automatic Software Integration** – the changes to the software made by different developers are tested and integrated several times in a day. This is automatically done when using cloud.
5. **Easy Access to Information** – once registered, information can be accessed from any location and from any device.
6. **Quick Deployment** – using cloud you can get your entire system fully functioning in just a couple of minutes.

Disadvantages of Cloud Computing:

1. Technical issues:

Though cloud enables you to access information from anywhere and on any device, however, the system can sometimes malfunction and besides that you also need a strong internet connection.

2. Security in Cloud:

You should always be careful to choose the most reliable service provider since the company's sensitive information is being handed over to a third party for storage.

3. Prone to Attack:

Storing information in the cloud makes it accessible to hackers.

UNIT- III

Principles of Parallel and Distributed Computing: Parallel vs. distributed computing - Elements of parallel computing - Hardware architectures for parallel processing, Approaches to parallel programming - Laws of caution.

Parallel Computing

- Parallel computing is the process of performing [computational tasks](#) across multiple processors at once to improve computing speed and efficiency. It divides tasks into sub-tasks and executes them simultaneously through different processors.
- There are three main types, or “levels,” of parallel computing: bit, instruction, and task.
- **Bit-level parallelism:** Uses larger “words,” which is a fixed-sized piece of data handled as a unit by the instruction set or the hardware of the processor, to reduce the number of instructions the processor needs to perform an operation.
- **Instruction-level parallelism:** Employs a stream of instructions to allow processors to execute more than one instruction per clock cycle (the oscillation between high and low states within a digital circuit).
- **Task-level parallelism:** Runs computer code across multiple processors to run multiple tasks at the same time on the same data.

Distributed Computing

- Distributed computing is the process of connecting multiple computers via a local network or wide area network so that they can act together as a single ultra-powerful computer capable of performing computations that no single computer within the network would be able to perform on its own.

Distributed computers offer two key advantages:

- **Easy scalability:** Just add more computers to expand the system.
- **Redundancy:** Since many different machines are providing the same service, that service can keep running even if one (or more) of the computers goes down.

• **Key Differences Between Parallel Computing and Distributed Computing**

- While parallel and distributed computers are both important technologies, there are several key differences between them.
- **Difference #1: Number of Computers Required**
- Parallel computing typically requires one computer with multiple processors. Distributed computing, on the other hand, involves several autonomous (and often geographically separate and/or distant) computer systems working on divided tasks.
- **Difference #2: Scalability**
- Parallel computing systems are less scalable than distributed computing systems because the memory of a single computer can only handle so many processors at once. A distributed computing system can always scale with additional computers.
- **Difference #3: Memory**
- In parallel computing, all processors share the same memory and the processors communicate with each other with the help of this shared memory. Distributed computing systems, on the other hand, have their own memory and processors.
- **Difference #4: Synchronization**
- In parallel computing, all processors share a single master clock for synchronization, while distributed computing systems use synchronization algorithms.
- **Difference #5: Usage**
- Parallel computing is used to increase computer performance and for scientific computing, while distributed computing is used to share resources and improve scalability.

Hardware Architectures For Parallel Processing

- Flynn's taxonomy is a classification scheme for computer architectures proposed by Michael Flynn in 1966. The taxonomy is based on the number of instruction streams and data streams that can be processed simultaneously by a computer architecture.

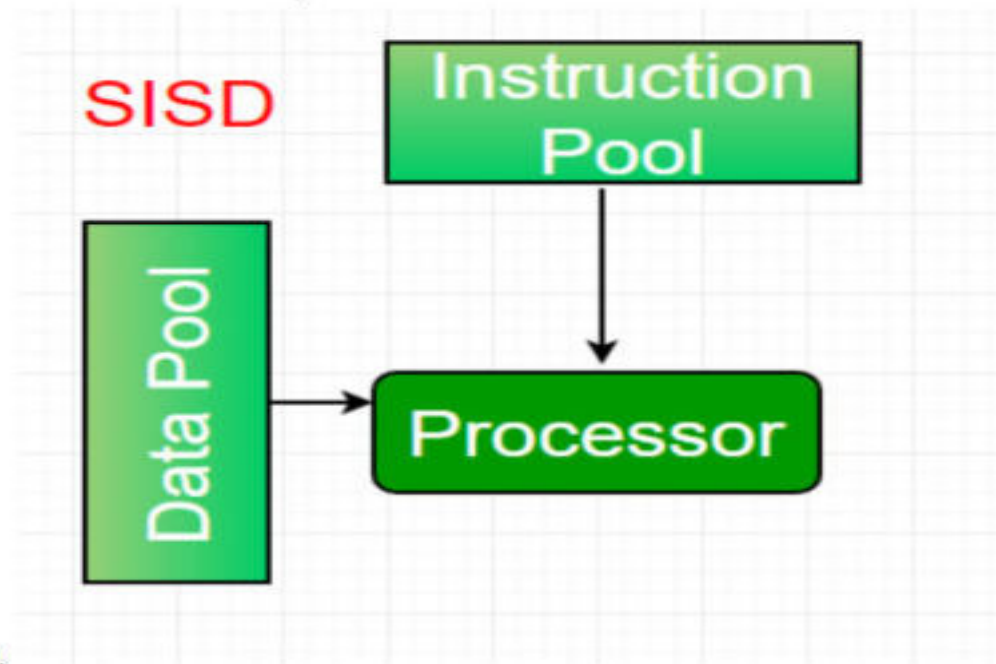
There are four categories in Flynn's taxonomy:

- **Single Instruction Single Data (SISD)**: In a SISD architecture, there is a single processor that executes a single instruction stream and operates on a single data stream. This is the simplest type of computer architecture and is used in most traditional computers.
- **Single Instruction Multiple Data (SIMD)**: In a SIMD architecture, there is a single processor that executes the same instruction on multiple data streams in parallel. This type of architecture is used in applications such as image and signal processing.
- **Multiple Instruction Single Data (MISD)**: In a MISD architecture, multiple processors execute different instructions on the same data stream. This type of architecture is not commonly used in practice, as it is difficult to find applications that can be decomposed into independent instruction streams.
- **Multiple Instruction Multiple Data (MIMD)**: In a MIMD architecture, multiple processors execute different instructions on different data streams. This type of architecture is used in distributed computing, parallel processing, and other high-performance computing applications.

- Flynn's taxonomy is a useful tool for understanding different types of computer architectures and their strengths and weaknesses. The taxonomy highlights the importance of parallelism in modern computing and shows how different types of parallelism can be exploited to improve performance.
- Systems are classified into four major categories:

| | | Instruction Streams | |
|--------------|------|---|---|
| | | one | many |
| Data Streams | one | SISD
traditional von
Neumann single
CPU computer | MISD
May be pipelined
Computers |
| | many | SIMD
Vector processors
fine grained data
Parallel computers | MIMD
Multi computers
Multiprocessors |

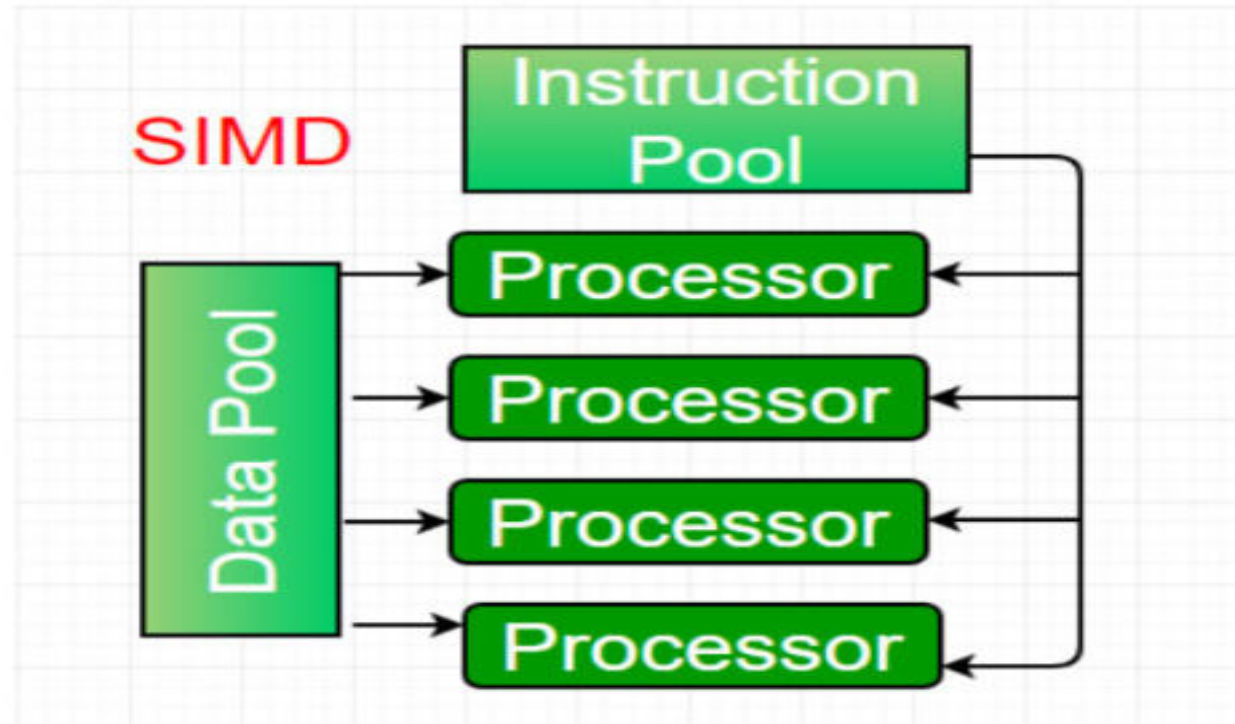
1. **Single-instruction, single-data (SISD) systems** – An SISD computing system is a uniprocessor machine that is capable of executing a single instruction, operating on a single data stream. In SISD, machine instructions are processed in a sequential manner and computers adopting this model are popularly called sequential computers. Most conventional computers have SISD architecture. All the instructions and data to be processed have to be stored in



primary memory.

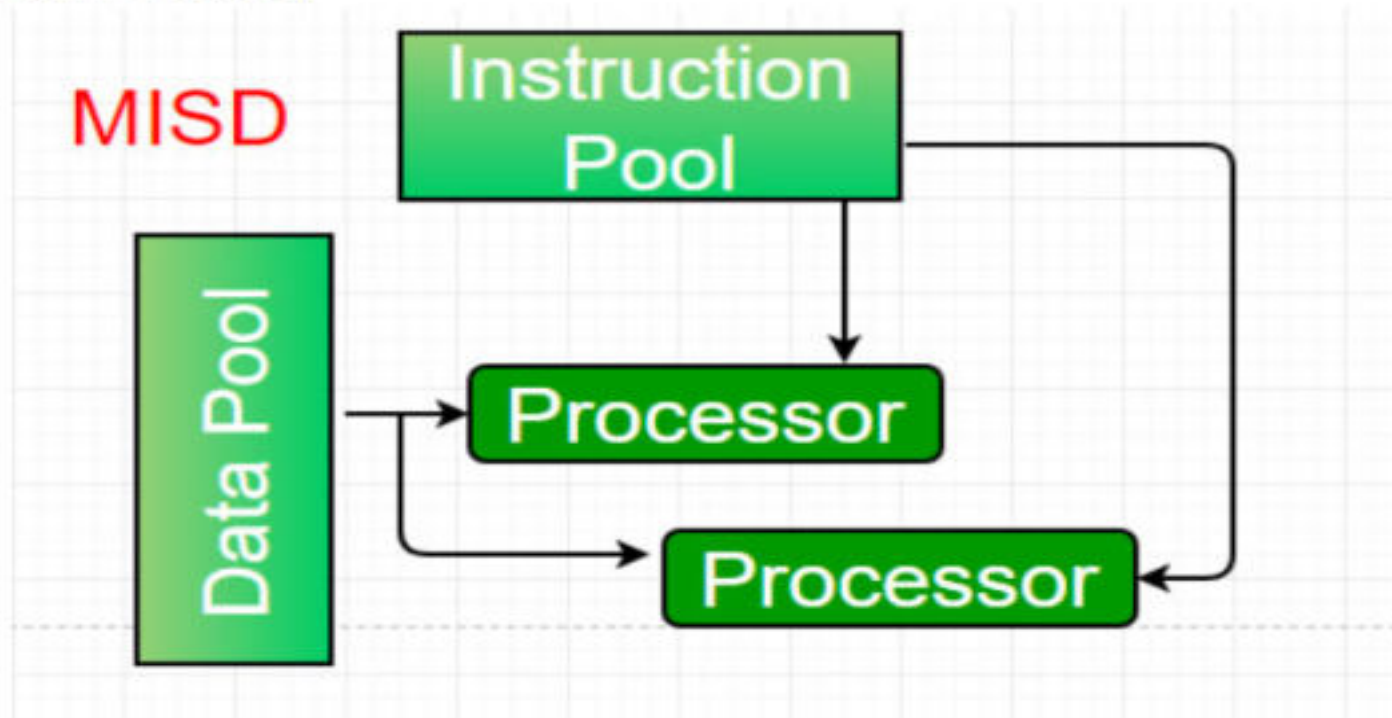
The speed of the processing element in the SISD model is limited(dependent) by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, and workstations.

2. **Single-instruction, multiple-data (SIMD) systems** – An SIMD system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams. Machines based on a SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations. So that the information can be passed to all the processing elements (PEs) organized data elements of vectors can be divided into multiple sets (N-sets for N PE systems) and each PE can process one data set.



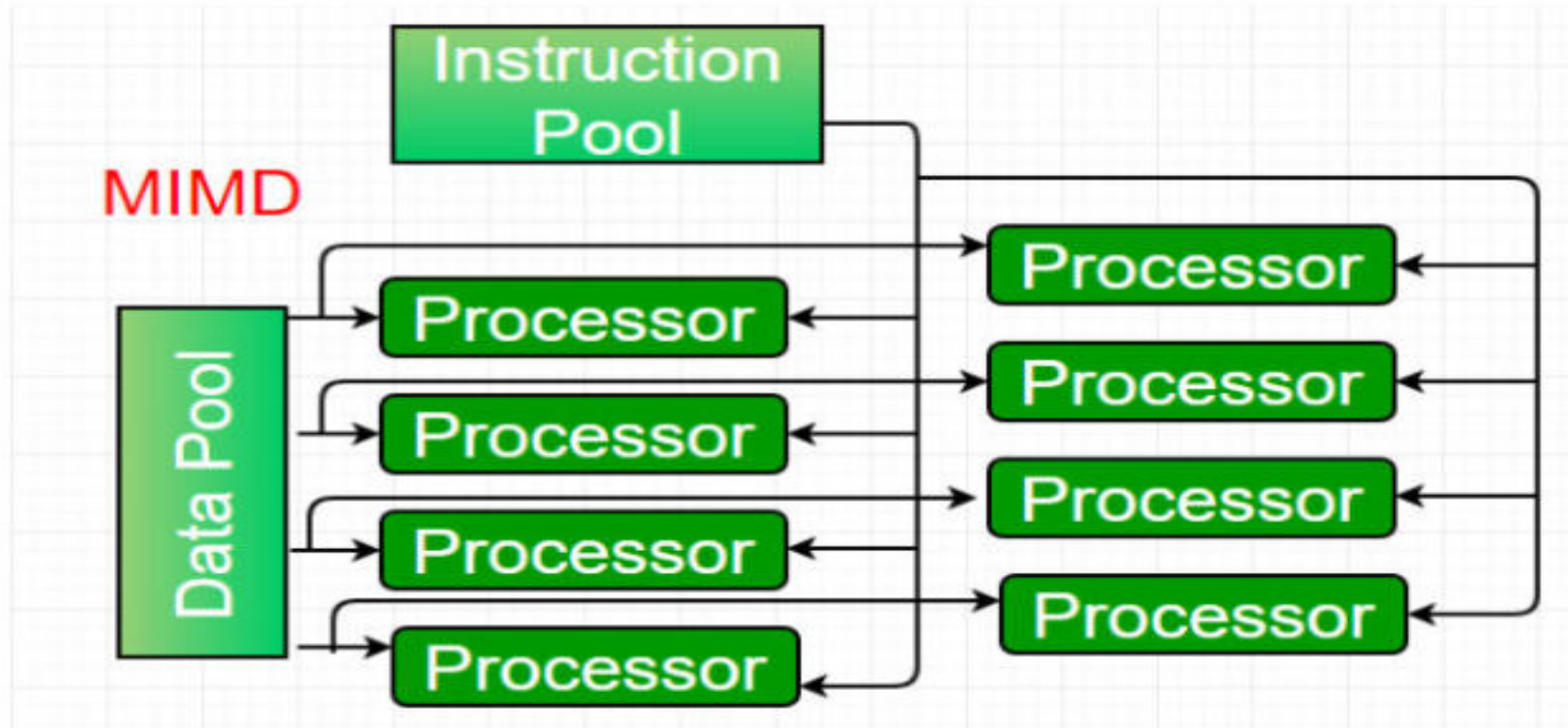
Dominant representative SIMD systems are Cray's vector processing machines.

3. **Multiple-instruction, single-data (MISD) systems** – An MISD computing system is a multiprocessor machine capable of executing different instructions on different PEs but all of them operate on the same dataset.



Example $Z = \sin(x) + \cos(x) + \tan(x)$ The system performs different operations on the same data set. Machines built using the MISD model are not useful in most applications, a few machines are built, but none of them are available commercially.

4. **Multiple-instruction, multiple-data (MIMD) systems** – An MIMD system is a multiprocessor machine that is capable of executing multiple instructions on multiple data sets. Each PE in the MIMD model has separate instruction and data streams; therefore machines built using this model are capable of any application. Unlike SIMD and MISD machines, PEs in MIMD machines work asynchronously.

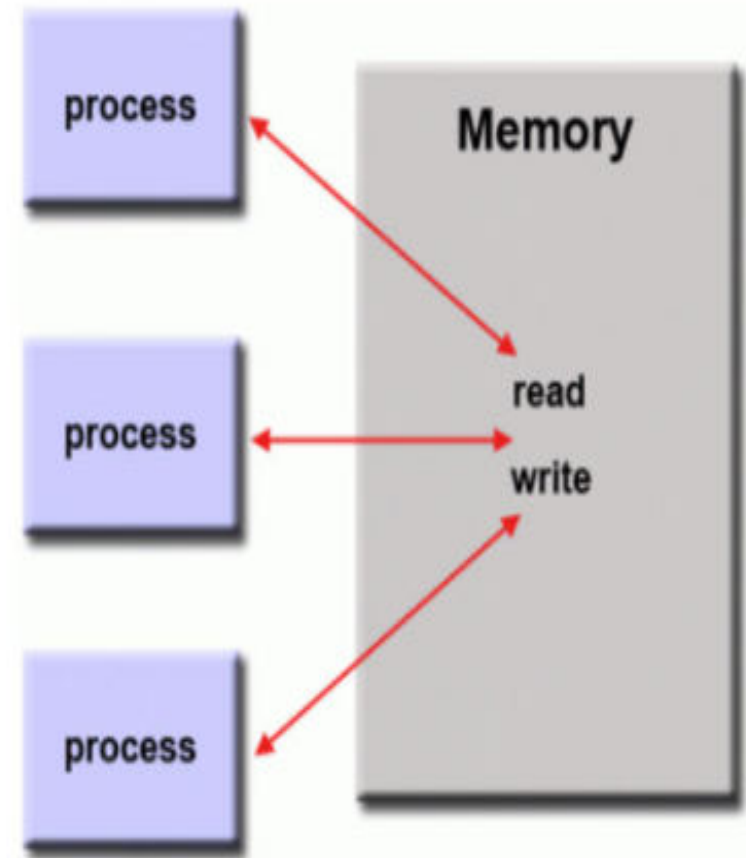


- **Parallel Programming Models**
- There are several parallel programming models in common use:
 - Shared Memory (without threads)
 - Threads
 - Distributed Memory / Message Passing
 - Data Parallel
 - Hybrid
 - Single Program Multiple Data (SPMD)
 - Multiple Program Multiple Data (MPMD)

Parallel programming models exist as an abstraction above hardware and memory architectures.

Shared Memory Model (without threads)

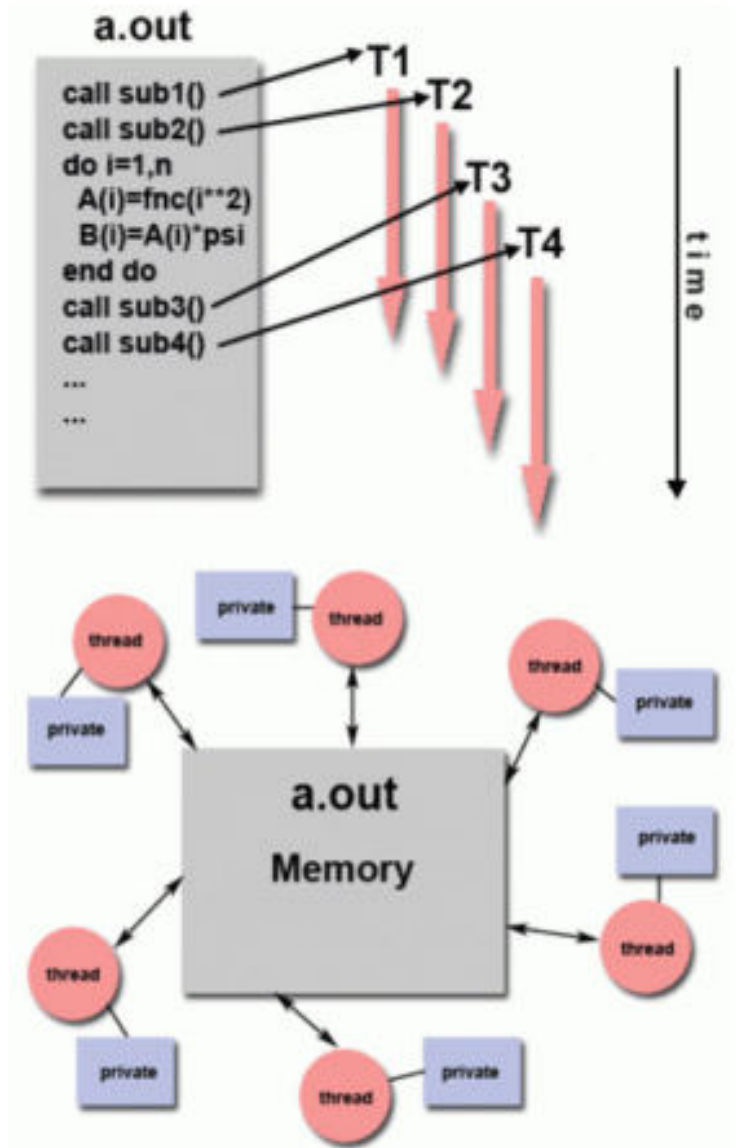
- In this programming model, processes/tasks share a common address space, which they read and write to asynchronously.
- Various mechanisms such as locks / semaphores are used to control access to the shared memory, resolve contentions and to prevent race conditions and deadlocks.
- This is perhaps the simplest parallel programming model.
- An advantage of this model from the programmer's point of view is that the notion of data "ownership" is lacking, so there is no need to specify explicitly the communication of data between tasks. All processes see and have equal access to shared memory. Program development can often be simplified.
- An important disadvantage in terms of performance is that it becomes more difficult to understand and manage *data locality*:
 - Keeping data local to the process that works on it conserves memory accesses, cache refreshes and bus traffic that occurs when multiple processes use the same data.
 - Unfortunately, controlling data locality is hard to understand and may be beyond the control of the average user.



| Shared memory model

Threads Model

- This programming model is a type of shared memory programming.
- In the threads model of parallel programming, a single "heavy weight" process can have multiple "light weight", concurrent execution paths.
- For example:
 - The main program **a.out** is scheduled to run by the native operating system. **a.out** loads and acquires all of the necessary system and user resources to run. This is the "heavy weight" process.
 - **a.out** performs some serial work, and then creates a number of tasks (threads) that can be scheduled and run by the operating system concurrently.
 - Each thread has local data, but also, shares the entire resources of **a.out**. This saves the overhead associated with replicating a program's resources for each thread ("light weight"). Each thread also benefits from a global memory view because it shares the memory space of **a.out**.
 - A thread's work may best be described as a subroutine within the main program. Any thread can execute any subroutine at the same time as other threads.
 - Threads communicate with each other through global memory (updating address locations). This requires synchronization constructs to ensure that more than one thread is not updating the same global address at any time.
 - Threads can come and go, but **a.out** remains present to provide the necessary shared resources until the application has completed.

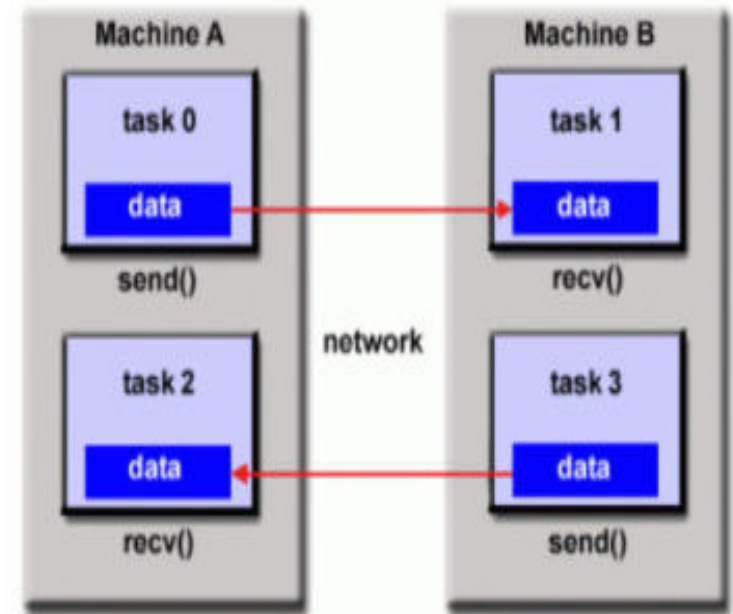


Distributed Memory / Message Passing Model

- This model demonstrates the following characteristics:
 - A set of tasks that use their own local memory during computation. Multiple tasks can reside on the same physical machine and/or across an arbitrary number of machines.
 - Tasks exchange data through communications by sending and receiving messages.
 - Data transfer usually requires cooperative operations to be performed by each process. For example, a send operation must have a matching receive operation.

Implementations:

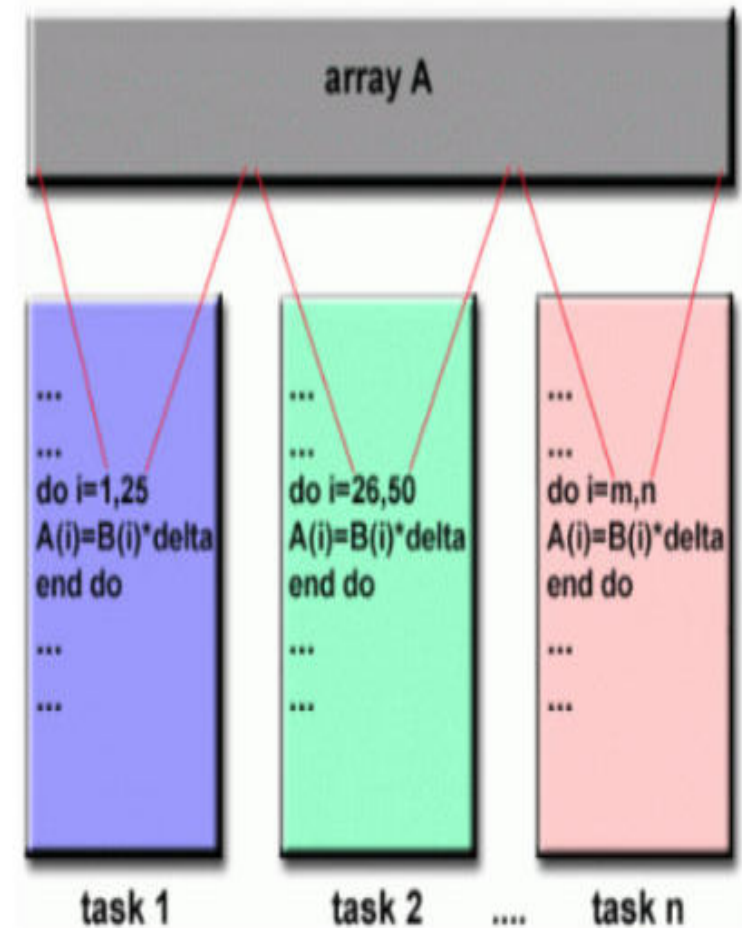
- From a programming perspective, message passing implementations usually comprise a library of subroutines. Calls to these subroutines are imbedded in source code. The programmer is responsible for determining all parallelism.
- Historically, a variety of message passing libraries have been available since the 1980s. These implementations differed substantially from each other making it difficult for programmers to develop portable applications.
- In 1992, the MPI Forum was formed with the primary goal of establishing a standard interface for message passing implementations.



Distributed memory model

Data Parallel Model

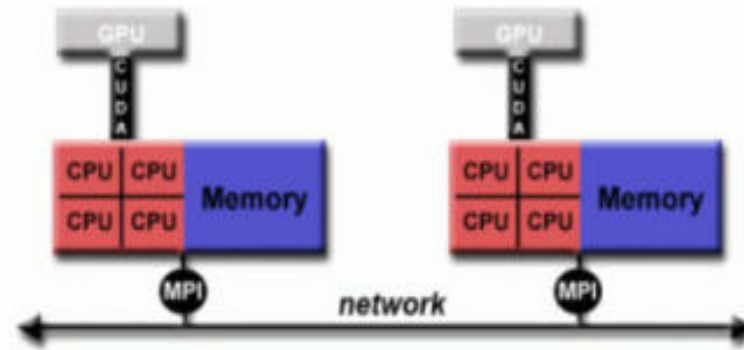
- May also be referred to as the **Partitioned Global Address Space (PGAS)** model.
- The data parallel model demonstrates the following characteristics:
 - Address space is treated globally
 - Most of the parallel work focuses on performing operations on a data set. The data set is typically organized into a common structure, such as an array or cube.
 - A set of tasks work collectively on the same data structure, however, each task works on a different partition of the same data structure.
 - Tasks perform the same operation on their partition of work, for example, "add 4 to every array element".
- On shared memory architectures, all tasks may have access to the data structure through global memory.
- On distributed memory architectures, the global data structure can be split up logically and/or physically across tasks.



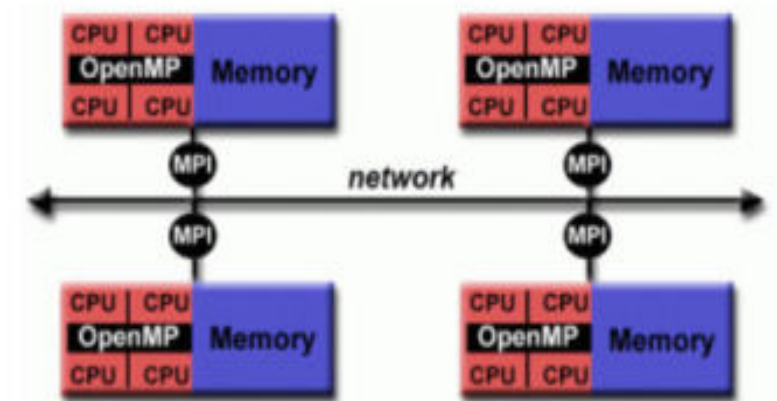
| Data parallel model

Hybrid Model

- A hybrid model combines more than one of the previously described programming models.
- Currently, a common example of a hybrid model is the combination of the message passing model (MPI) with the threads model (OpenMP).



Hybrid model with MPI and CUDA

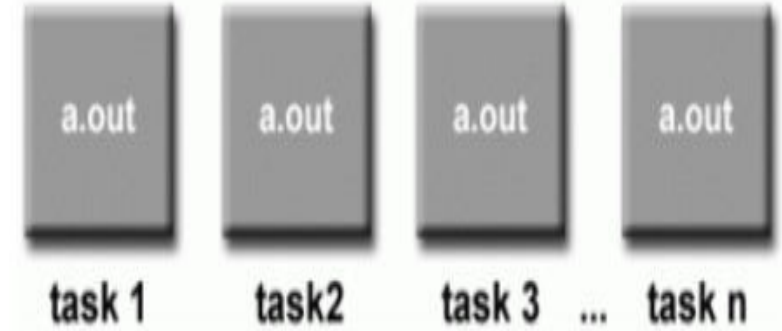


Hybrid model with MPI and OpenMP

- Threads perform computationally intensive kernels using local, on-node data
- Communications between processes on different nodes occurs over the network using MPI
- This hybrid model lends itself well to the most popular (currently) hardware environment of clustered multi/many-core machines.
- Another similar and increasingly popular example of a hybrid model is using MPI with CPU-GPU (graphics processing unit) programming.
 - MPI tasks run on CPUs using local memory and communicating with each other over a network.
 - Computationally intensive kernels are off-loaded to GPUs on-node.
 - Data exchange between node-local memory and GPUs uses CUDA (or something equivalent).
- Other hybrid models are common:
 - MPI with Pthreads
 - MPI with non-GPU accelerators

Single Program Multiple Data (SPMD)

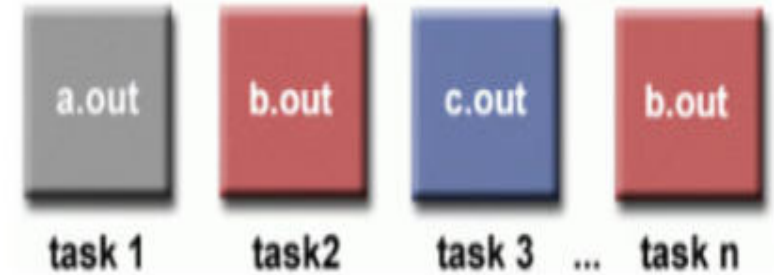
- SPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.
- SINGLE PROGRAM: All tasks execute their copy of the same program simultaneously. This program can be threads, message passing, data parallel or hybrid.
- MULTIPLE DATA: All tasks may use different data
- SPMD programs usually have the necessary logic programmed into them to allow different tasks to branch or conditionally execute only those parts of the program they are designed to execute. That is, tasks do not necessarily have to execute the entire program - perhaps only a portion of it.
- The SPMD model, using message passing or hybrid programming, is probably the most commonly used parallel programming model for multi-node clusters.



| SPMD model

Multiple Program Multiple Data (MPMD)

- Like SPMD, MPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.
- MULTIPLE PROGRAM: Tasks may execute different programs simultaneously. The programs can be threads, message passing, data parallel or hybrid.
- MULTIPLE DATA: All tasks may use different data
- MPMD applications are not as common as SPMD applications, but may be better suited for certain types of problems, particularly those that lend themselves better to functional decomposition than domain decomposition (discussed later under Partitioning).



| *MPMD model*

2.3.3 Approaches to parallel programming

A sequential program is one that runs on a single processor and has a single line of control. To make many processors collectively work on a single program, the program must be divided into smaller independent chunks so that each processor can work on separate chunks of the problem. The program decomposed in this way is a parallel program.

A wide variety of parallel programming approaches are available. The most prominent among them are the following:

- Data parallelism
- Process parallelism
- Farmer-and-worker model

These three models are all suitable for task-level parallelism. In the case of data parallelism, the divide-and-conquer technique is used to split data into multiple sets, and each data set is processed on different PEs using the same instruction. This approach is highly suitable to processing on machines based on the SIMD model. In the case of process parallelism, a given operation has multiple (but distinct) activities that can be processed on multiple processors. In the case of the farmer-and-worker model, a job distribution approach is used: one processor is configured as master and all other remaining PEs are designated as slaves; the master assigns jobs to slave PEs and, on completion, they inform the master, which in turn collects results. These approaches can be utilized in different levels of parallelism.

2.3.4 Levels of parallelism

Levels of parallelism are decided based on the lumps of code (grain size) that can be a potential candidate for parallelism. [Table 2.1](#) lists categories of code granularity for parallelism. All these approaches have a common goal: to boost processor efficiency by hiding latency. To conceal latency, there must be another thread ready to run whenever a lengthy operation occurs. The idea is to execute concurrently two or more single-threaded applications, such as compiling, text formatting, database searching, and device simulation.

As shown in the table and depicted in [Figure 2.7](#), parallelism within an application can be detected at several levels:

- Large grain (or task level)
- Medium grain (or control level)
- Fine grain (data level)
- Very fine grain (multiple-instruction issue)

Table 2.1 Levels of Parallelism

| Grain Size | Code Item | Parallelized By |
|------------|----------------------------------|------------------------|
| Large | Separate and heavyweight process | Programmer |
| Medium | Function or procedure | Programmer |
| Fine | Loop or instruction block | Parallelizing compiler |
| Very fine | Instruction | Processor |

2.3.5 Laws of caution

Now that we have introduced some general aspects of parallel computing in terms of architectures and models, we can make some considerations that have been drawn from experience designing and implementing such systems. These considerations are guidelines that can help us understand how much benefit an application or a software system can gain from parallelism. In particular, what we need to keep in mind is that parallelism is used to perform multiple activities together so that the system can increase its throughput or its speed. But the relations that control the increment of speed are not linear. For example, for a given n processors, the user expects speed to be increased by n times. This is an ideal situation, but it rarely happens because of the communication overhead.

Here are two important guidelines to take into account:

- Speed of computation is proportional to the square root of system cost; they never increase linearly. Therefore, the faster a system becomes, the more expensive it is to increase its speed (Figure 2.8).
- Speed by a parallel computer increases as the logarithm of the number of processors (i.e., $y = k \cdot \log(N)$). This concept is shown in Figure 2.9.

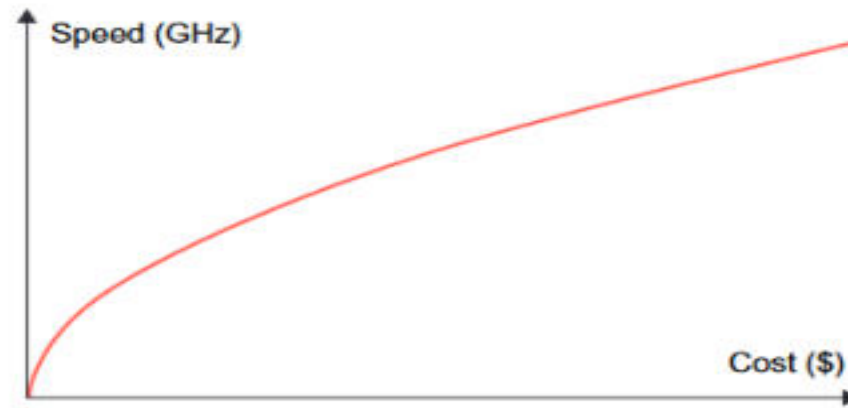


FIGURE 2.8

Cost versus speed.

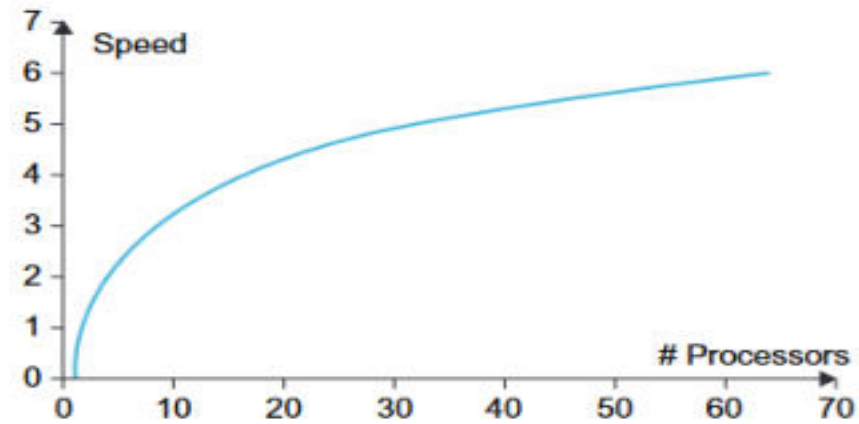


FIGURE 2.9

Number processors versus speed.

- The very fast development in parallel processing and related areas has blurred conceptual boundaries, causing a lot of terminological confusion.
- Even well-defined distinctions such as shared memory and distributed memory are merging due to new advances in technology.
- There are no strict delimiters for contributors to the area of parallel processing. Hence, computer architects, OS designers, language designers, and computer network designers all have a role to play.

UNIT- IV

- Virtualization: Introduction
- Characteristics of virtualized environments
- Taxonomy of virtualization techniques - Virtualization and cloud computing
- Pros and cons of virtualization
- Technology example: VMware: full virtualization
- Types of hardware virtualization: Full virtualization - partial virtualization - para virtualization.

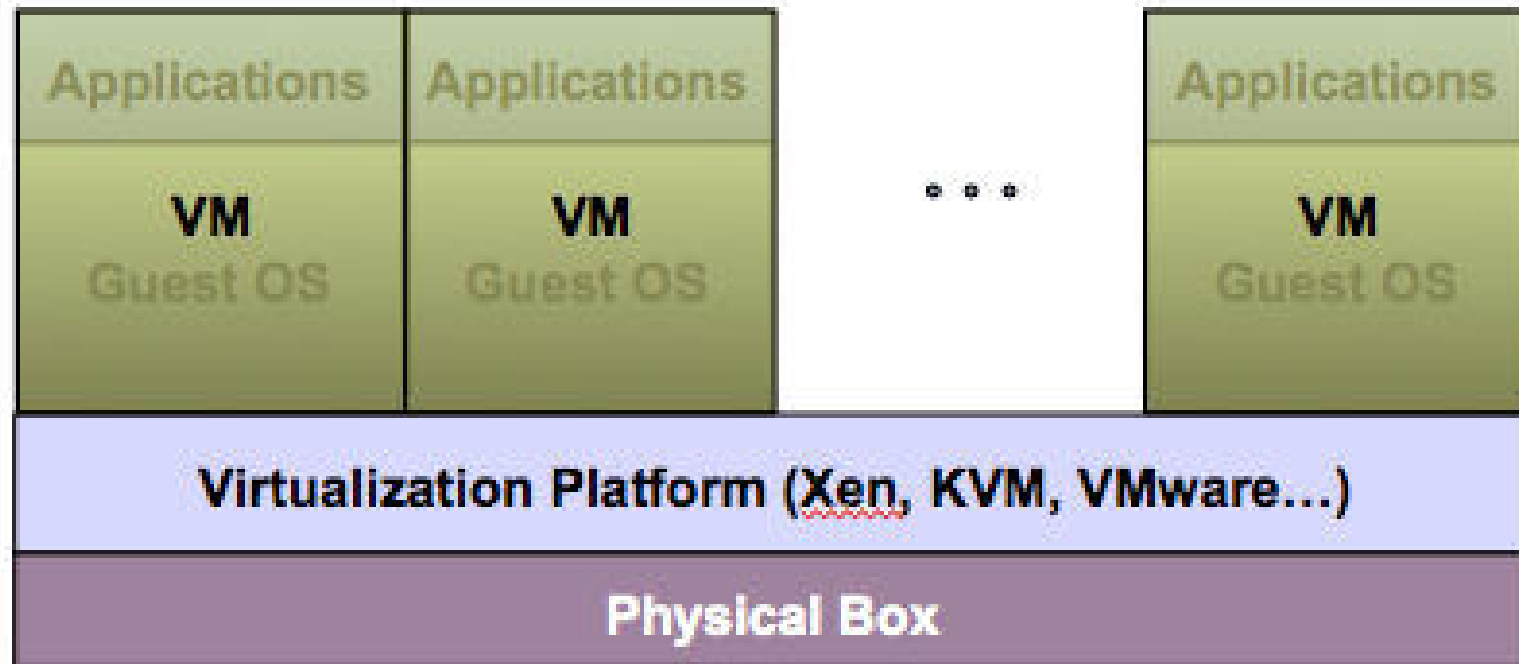
Definition

- **Virtualization** is the ability to run multiple operating systems on a single physical system and share the underlying hardware resources*
- It is the process by which one computer hosts the appearance of many computers.
- Virtualization is used to improve IT throughput and costs by using physical resources as a pool from which virtual resources can be allocated.

*VMWare white paper, *Virtualization Overview*

Virtualization Architecture

- A Virtual machine (VM) is an isolated runtime environment (guest OS and applications)
- Multiple virtual systems (VMs) can run on a single physical system



Hypervisor

- A **hypervisor**, a.k.a. a virtual machine manager/monitor (VMM), or virtualization manager, is a program that allows multiple operating systems to share a single hardware host.
- Each guest operating system appears to have the host's processor, memory, and other resources all to itself. However, the hypervisor is actually controlling the host processor and resources, allocating what is needed to each operating system in turn and making sure that the guest operating systems (called virtual machines) cannot disrupt each other.

Benefits of Virtualization

- Sharing of resources helps cost reduction
- Isolation: Virtual machines are isolated from each other as if they are physically separated
- Encapsulation: Virtual machines encapsulate a complete computing environment
- Hardware Independence: Virtual machines run independently of underlying hardware
- Portability: Virtual machines can be migrated between different hosts.

Virtualization in Cloud Computing

Cloud computing takes virtualization one step further:

- You don't need to own the hardware
- Resources are rented as needed from a cloud
- Various providers allow creating virtual servers:
 - Choose the OS and software each instance will have
 - The chosen OS will run on a large server farm
 - Can instantiate more virtual servers or shut down existing ones within minutes
- You get billed only for what you used

Characteristics Of Virtualized Environments

- **Increased security**

The ability to control the execution of a guest in a completely transparent manner opens new possibilities for delivering a secure, controlled execution environment. The virtual machine represents an emulated environment in which the guest is executed. All the operations of the guest are generally performed against the virtual machine, which then translates and applies them to the host. This level of indirection allows the virtual machine manager to control and filter the activity of the guest, thus preventing some harmful operations from being performed. Resources exposed by the host can then be hidden or simply protected from the guest.

- **Managed execution**

Virtualization of the execution environment not only allows increased security, but a wider range of features also can be implemented. In particular, sharing, aggregation, emulation, and isolation are the most relevant features.

- **Portability**

- The concept of portability applies in different ways according to the specific type of virtualization considered. In the case of a hardware virtualization solution, the guest is packaged into a virtual image that, in most cases, can be safely moved and executed on top of different virtual machines. Except for the file size, this happens with the same simplicity with which we can display a picture image in different computers.
- In the case of programming-level virtualization, as implemented by the JVM or the .NET runtime, the binary code representing application components (jars or assemblies) can be run without any recompilation on any implementation of the corresponding virtual machine.

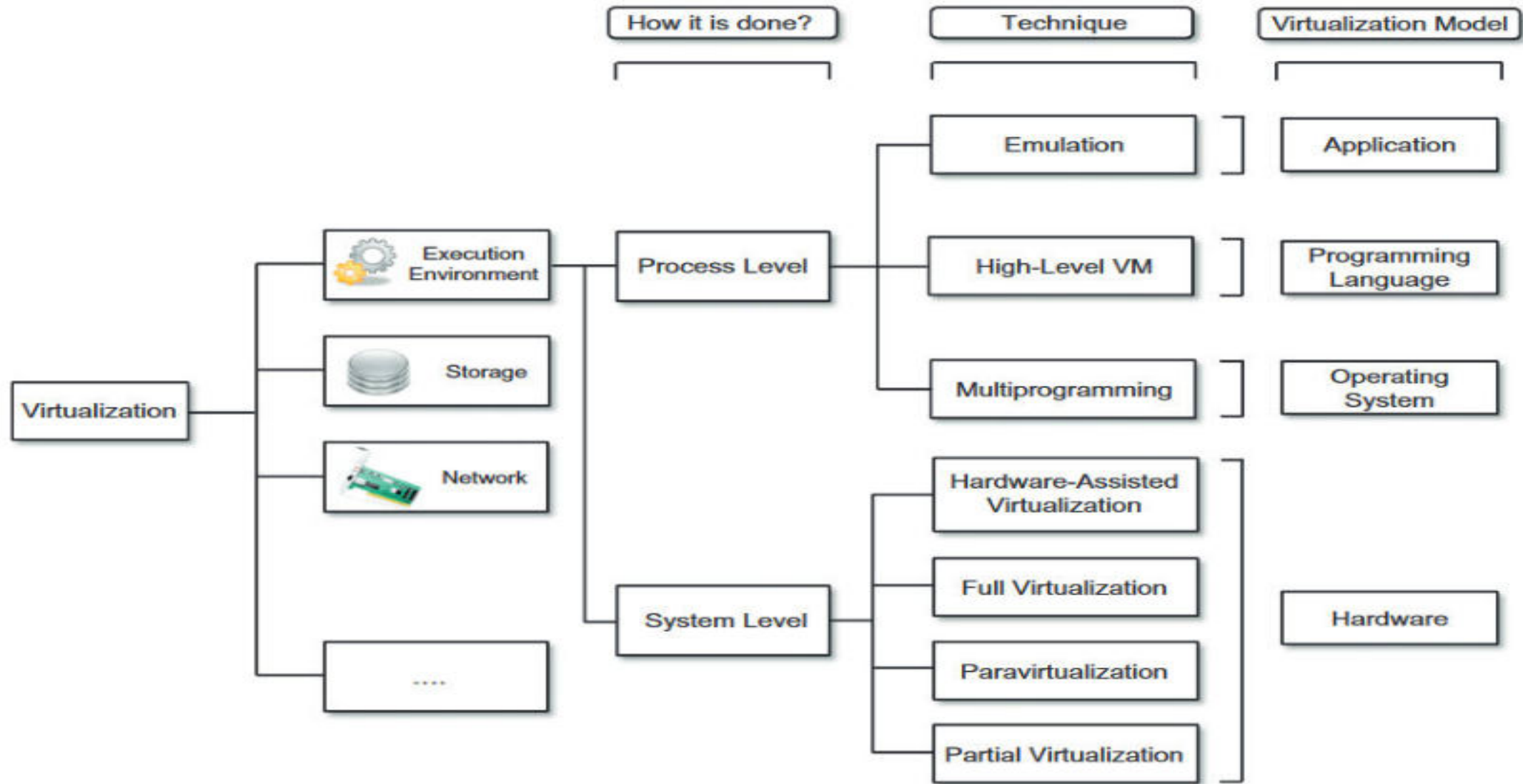


FIGURE 3.3

A taxonomy of virtualization techniques.

Virtualization And Cloud Computing

- Virtualization plays an important role in cloud computing since it allows for the appropriate degree of customization, security, isolation, and manageability that are fundamental for delivering IT services on demand. Virtualization technologies are primarily used to offer configurable computing environments and storage.
- Particularly important is the role of virtual computing environment and execution virtualization techniques. Among these, hardware and programming language virtualization are the techniques adopted in cloud computing systems.
- Hardware virtualization is an enabling factor for solutions in the Infrastructure-as-a-Service (IaaS) market segment, while programming language virtualization is a technology leveraged in Platform-as-a-Service (PaaS) offerings.
- In both cases, the capability of offering a customizable and sandboxed environment constituted an attractive business opportunity for companies featuring a large computing infrastructure that was able to sustain and process huge workloads. Moreover, virtualization also allows isolation and a finer control, thus simplifying the leasing of services and their accountability on the vendor side.

Advantages of Virtualization

- Managed execution and isolation
- Portability
- Efficient use of resources

Disadvantages of Virtualization

- Performance degradation
- Inefficiency and degraded user experience
- Security holes and new threats

Technology Example

VMware: full virtualization

VMware's technology is based on the concept of full virtualization, where the underlying hardware is replicated and made available to the guest operating system, which runs unaware of such abstraction layers and does not need to be modified. VMware implements full virtualization either in the desktop environment, by means of Type II hypervisors, or in the server environment, by means of Type I hypervisors. In both cases, full virtualization is made possible by means of direct execution (for nonsensitive instructions) and binary translation (for sensitive instructions), thus allowing the virtualization of architecture such as x86.

Besides these two core solutions, VMware provides additional tools and software that simplify the use of virtualization technology either in a desktop environment, with tools enhancing the integration of virtual guests with the host, or in a server environment, with solutions for building and managing virtual computing infrastructures.

Hardware Virtualization Techniques

- **Full Virtualization**

- Full virtualization refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware.
- The principal advantage of full virtualization is complete isolation, which leads to enhanced security, ease of emulation of different architectures, and coexistence of different systems on the same platform.
- A key challenge is the interception of privileged instructions such as I/O instructions: Since they change the state of the resources exposed by the host, they have to be contained within the virtual machine manager.
- A simple solution to achieve full virtualization is to provide a virtual environment for all the instructions, thus posing some limits on performance. A successful and efficient implementation of full virtualization is obtained with a combination of hardware and software, not allowing potentially harmful instructions to be executed directly on the host.

- **Partial Virtualization**

- Partial virtualization provides a partial emulation of the underlying hardware, thus not allowing the complete execution of the guest operating system in complete isolation.
- Partial virtualization allows many applications to run transparently, but not all the features of the operating system can be supported, as happens with full virtualization.
- An example of partial virtualization is address space virtualization used in time-sharing systems; this allows multiple applications and users to run concurrently in a separate memory space, but they still share the same hardware resources (disk, processor, and network).
- Historically, partial virtualization has been an important milestone for achieving full virtualization, and it was implemented on the experimental IBM M44/44X. Address space virtualization is a common feature of contemporary operating systems.

- **Paravirtualization**

- This is a not-transparent virtualization solution that allows implementing thin virtual machine managers.
- Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified.
- The aim of paravirtualization is to provide the capability to demand the execution of performance-critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution.

UNIT-I

Cloud Computing Overview – Services of Internet, Origins of Cloud computing – Cloud components – Essential characteristics – On-demand self-service, The vision of cloud computing – Characteristics, benefits, and Challenges ahead.

Why Cloud?

- Cloud provides a host of benefits which make it so popular.
- We can not only store large amounts of data securely on the cloud, but it is also possible to rent the latest hi-tech software and even hardware.
- For example:
 - Consider that you are using a traditional method of computing in your office, now if your company recruits some more employees then you need to do all the hardware as well as software setup for all those employees again which will inevitably increase your expenses.
 - However, using cloud computing you get the platform on rent and simply provide the employees a terminal and credentials to logon to their virtual system. Hence, in this case you rent the processing time, memory and the software setup.
 - Whenever an employee quits, his resources are freed and there is no need to pay for those rentals.

Why Cloud?

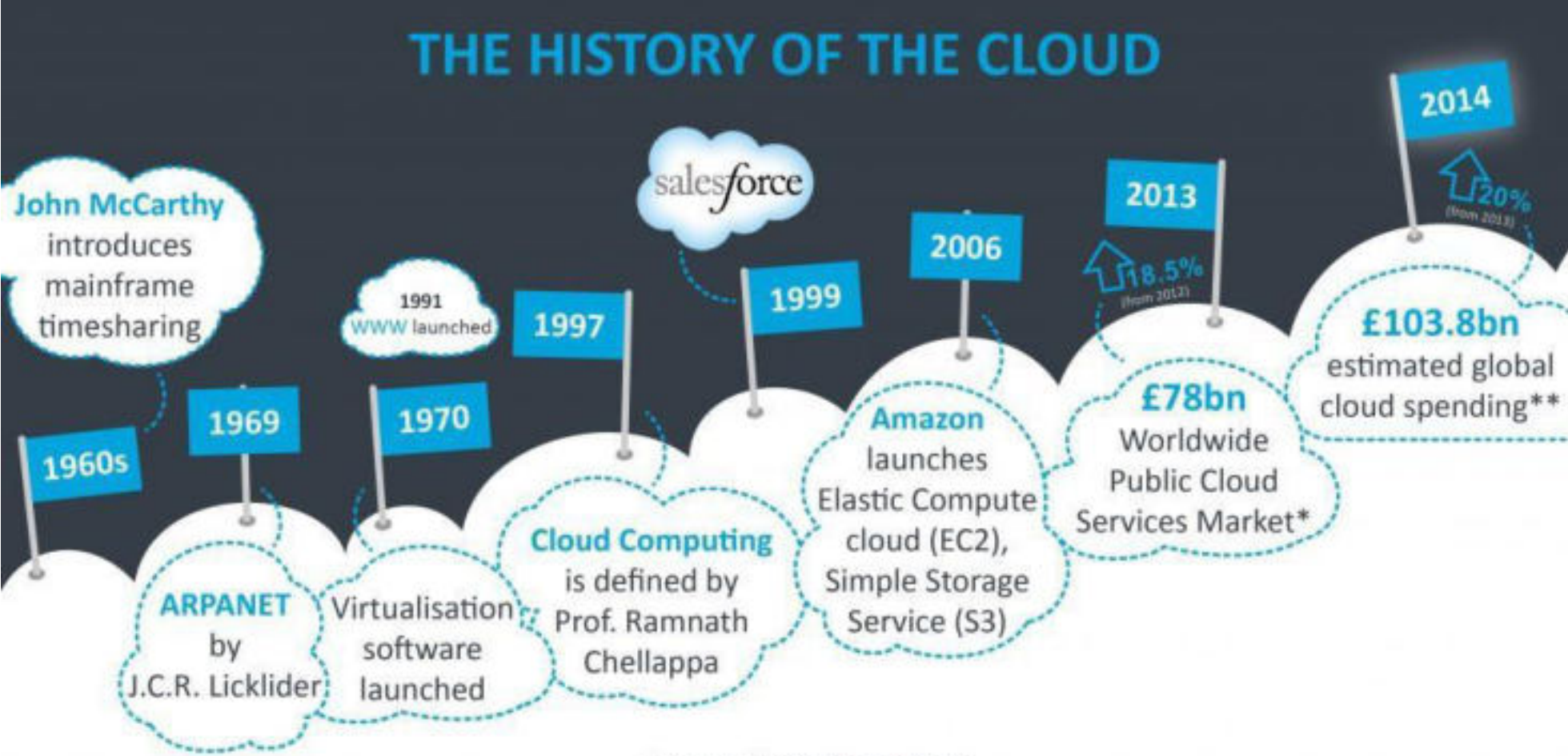
The cloud itself is a set of hardware, networks, storage, services and interfaces that enable the delivery of computing as a service.

Cloud Computing – Definition:

Cloud Computing is the on-demand delivery of computer power, infrastructure, applications, storage and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing.



History of Cloud Computing



* Gartner, ** Constellation Research

History of Cloud Computing

- The origin of cloud can be traced as far back as **1950s** when John McCarthy, a computer scientist, introduced ‘theory of time sharing’.
- In the **1960s**, J.C.R. Licklider developed the ARPANET (Advanced Research Projects Agency Network) which formed the basis of today’s internet.
- The development of virtual machines in **1970s** enabled users to run multiple operating systems on a single machine.
- In **1991**, the World Wide Web emerged.



History of Cloud Computing

- The **1990s** also saw a shift from point-to-point data circuits to virtual private network services being offered by telecommunication companies. This resulted in increased bandwidth at a lower cost.
- In **1997**, the first definition of cloud computing was given by Professor Ramnath Chellapa of Emory University. He defined Cloud Computing as,

‘Computing paradigm where the boundaries of computing will be determined by economic rationale rather than technical limits alone.’



Major Milestones:

- **1999: Salesforce.com**
 - ✓ Established the ability to use a simple website on the Internet to deliver enterprise-level applications.
- **2002: Amazon Web Services**
 - ✓ Featured several cloud-based retail services which included data storage and computation.
- **2006: Amazon's Elastic Compute Cloud (EC2) – the first commercial cloud**
 - ✓ Enabled small companies to rent computers that would host and run their own applications.
- **2007: Dropbox**
 - ✓ MIT student created this file-hosting service that offers file storage and synchronization.

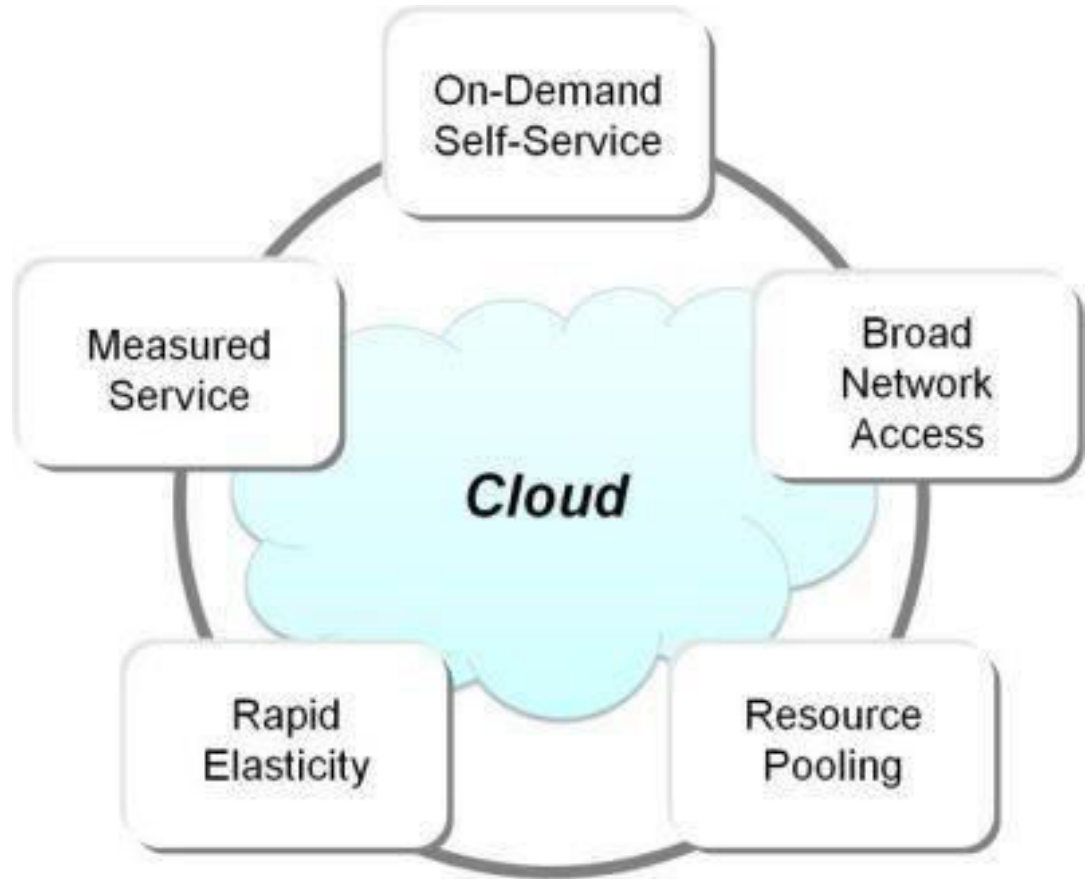


- **2009:**
 - ✓ Google Apps – example of browser-based enterprise applications
 - ✓ Windows Azure – Microsoft’s cloud computing platform



Characteristics of Cloud Computing

1. **On-demand Service**
2. **Broad Network Service**
3. **Resource Pooling**
4. **Rapid Elasticity**
5. **Measured Service**



Characteristics of Cloud Computing

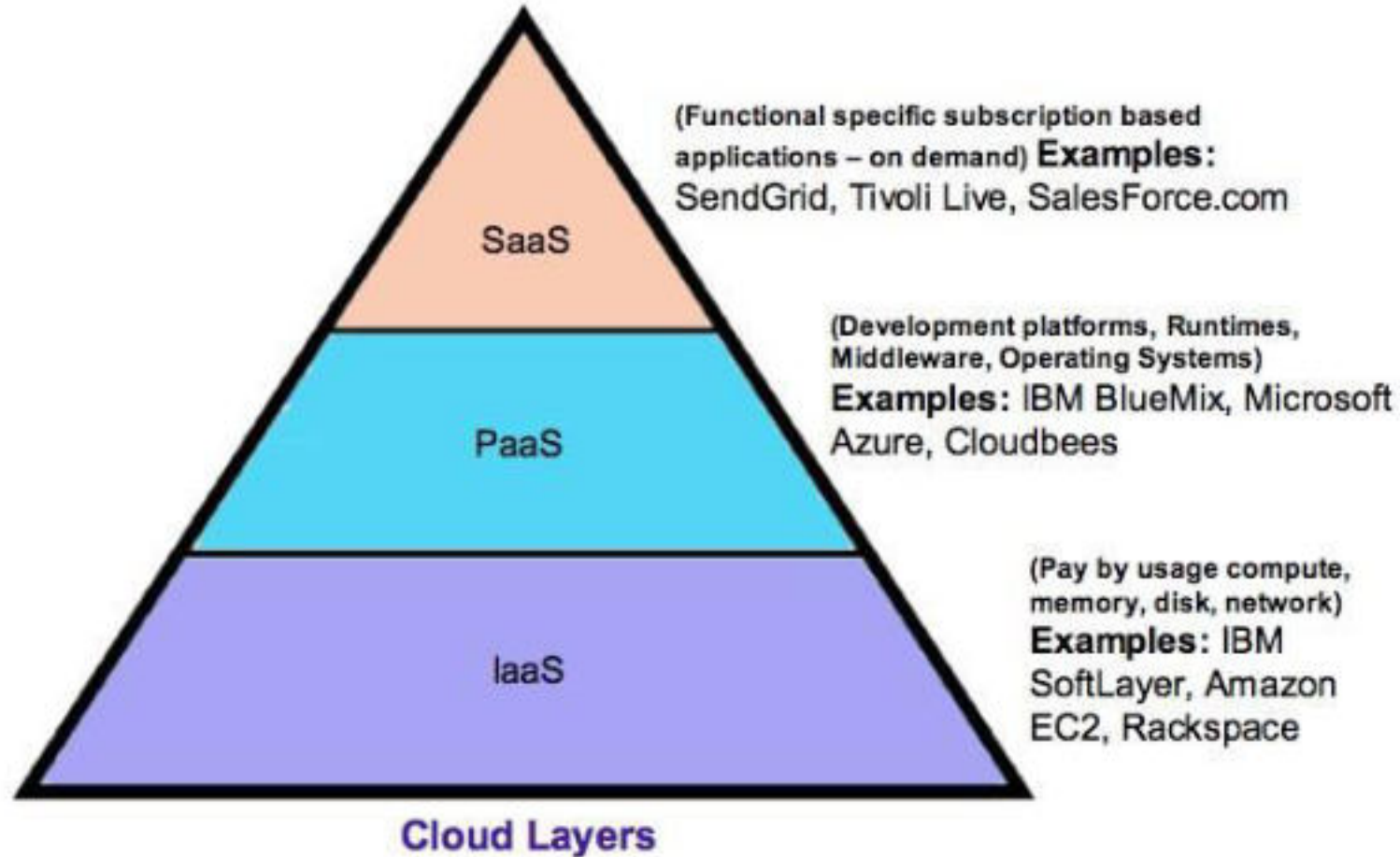
1. **On-demand Service:** The user can provide the required resources automatically without human intervention.
2. **Broad Network Access:** Accessible from any of the consumer's devices (such as Mobile, tablet, laptop and Desktop).
3. **Resource Pooling:** Multiple consumers can use multiple physical and virtual resources dynamically assigned and reassigned according to their demand.
4. **Rapid Elasticity:** Capabilities can be provisioned elastically. They can be scaled up and down based on the consumer's demand.
5. **Measured Service:** The service is measured using the pay-as-you-go pricing model.

Types of Cloud Services:

The major three categories are IaaS, PaaS and SaaS.

1. Infrastructure as a service (IaaS): The cloud provider provides IT infrastructure like the servers, virtual machines, storage, networks, and the operating systems on pay-as-you-go basis.
2. Platform as a service (PaaS): The developers may use the cloud computing services on demand for creating a web or mobile application. They need not worry about setting up the development environment.
3. Software as a service (SaaS): The users just connect to the Internet through phone, PC or the tablet and use the application hosted on cloud. The cloud service providers host and manage the software application, infrastructure, handle maintenance and upgrades including patching.

The Three Major Cloud Services:



Types of Cloud Services:

Other Cloud services:

1. **Data as a service (DaaS):** data are stored on cloud and made available to users on demand regardless of their geographic location.
2. **Desktop as a service** provides a virtual desktop.
3. **Storage as a service (SaaS)** provides data storage infrastructure.
4. **Test environment as a service:** we can rent a test setup to quickly test our application.
5. **Security as a service (SECaaS):** we can allow the cloud provider to take care of the security.
6. **API as a service** is a SaaS exposed as an API (Application Programming Interface). It allows users to access web services such as Google Maps, credit card processing and payroll processing.

Benefits of Cloud Computing

- **Scalability:** To accommodate variable workloads, cloud infrastructure scales on demand.
- **Storage Options:** Users may select between public, private, or hybrid storage choices, based on their security needs and other factors.
- **Control Options:** As-a-service options allow organizations to choose their amount of control.
- **Security Structure:** Virtual private cloud, encryption, and API keys are all security elements that assist keep data safe.
- **Data security:** In the case of hardware failure, networked backups avoid data loss.
- **Savings on Equipment:** Cloud computing makes use of distant resources, which saves businesses money on servers and other hardware.
- **Payments:** Users only pay for the resources they utilize when they employ a “utility” pay structure.
- **Regular Updates:** Service providers update their products regularly to ensure that consumers have access to the most up-to-date technologies.
- **Collaboration:** Because of global access, teams may cooperate from all over the world.
- **Competitive Advantage:** Businesses can move more quickly than competitors who must spend IT resources on infrastructure management.

Challenges

- **Privacy and Security:** The fundamental problem with cloud computing is security and privacy. Security apps, encrypted file systems, and data loss tools can all help to mitigate these issues.
- **Interoperability:** One platform's application should be able to use services from the other platform. Interoperability is the term for this. Web services are making this possible but developing such web services is difficult.
- **Portability:** Applications operating on one cloud platform may be migrated to a different cloud platform and should continue to work without any design or code modifications. Because each cloud provider employs a distinct standard language for their platform, portability is not achievable.
- **Quality of Service:** The providers' Service-Level Agreements (SLAs) are insufficient to ensure availability and scalability. Businesses are hesitant to move to the cloud unless there is a solid service quality assurance.
- **Computing Performance:** For data-intensive cloud applications, high network bandwidth is required, which results in a high cost. Low bandwidth does not match the computational performance requirements of cloud computing.
- **Availability and Dependability:** Because the majority of organizations rely on third-party services, cloud solutions must be dependable and stable.