



**Contents**

Introduction .....	iv
What You Will Be Learning in this Course .....	iv
Course Aim .....	v
Course Objectives .....	v
Working through this course .....	vi
Course Material .....	vi
Study Units .....	vii
Presentation Schedule .....	viii
Assessment .....	viii
Tutor-Marked Assignment (TMAs) .....	ix
Final Examination and Grading .....	ix
Course Marking Scheme .....	x
Facilitators/Tutors and Tutorials .....	x
Summary .....	x

## **Introduction**

Net-Centric or network centered computing is an ongoing area in the twenty-first century with a great interest among software engineers as it is an enabling technology for modern distributed computing systems and applications. Today, Net-Centric applications have invaded the lives of people in many ways. Net-Centric Computing (NCC) is a distributed environment where applications and data are downloaded from servers and exchanged with peers across a network. Net-centric Computing focuses on large-scale distributed computing systems and applications that communicate through open, wide-area networks like the Internet. General examples of large-scale network-centric systems are the World-Wide Web and Computational Grids. For several years, major changes are being brought to the world by universal networking capabilities, such as the Internet. Today's technology solutions represent the convergence of computing power, networking capability and the information, data or knowledge that forms the content of these solutions. Net-centric computing refers to an emerging technology architecture and an evolutionary stage of client/server computing. It is a common architecture built on open standards that supports in different ways for different people to collaborate and to reach different information sources. The evolutionary nature of net-centric computing links technological capabilities and strategic opportunities, helping people in facing today's new problems and providing the flexibility to meet tomorrow's challenges.

## **What You Will Be Learning in this Course**

This course consists of units and a course guide. This course guide tells you briefly what the course is about, what course material you will be using and how you can work through these materials. In addition, it advocates some general guidelines for the amount of time you are likely to spend on each unit of the course in order to complete it successfully. It gives you guidance in respect of your Tutor-Marked Assignments which will be made available in the assignment file. There will be regular tutorial classes that are related to the course. It is advisable for you to attend these tutorial sessions.

This course teaches the technology on which everything in the world, ranging from education, commerce, communication to even the home, runs which is inter-network.

## Course Aim

The aim of the course is to furnish you with full knowledge on inter-networking. It teaches how systems connect one with the other, communication modes, two or more systems processing, a single but divided large tasks executed together simultaneously, transmission technologies and much more.

## Course Objectives

To achieve the aims set out, the course has a set of objectives. Each unit has specific objectives which are included at the beginning of the unit. You may wish to refer to them during your study to check on your progress. You should always look at the unit objectives after completion of each unit. By doing so, you would know whether you have followed the instruction in the unit.

Below are the comprehensive objectives of the course as a whole. By meeting these objectives, you should have achieved the aims of the course as a whole. In addition to the aims earlier stated, this course sets to achieve some objectives. Thus, after going through the course, you should be able to:

- Identify the configurations of Distributed systems
- Describe the standards of wireless technology
- Implement security schemes or ciphers on the network
- Explain the categories of networks
- Define the concept of Parallel Systems
- Classify parallel Programming Models
- Describe Message Passing Programming
- Explain the concept of:
  - Dependence Analysis
  - Open MP Programming
  - Evaluation of Programs
- Optimizations for Scalar Architectures and Models for Parallel Computing
  - Dependence Analysis, Open MP Programming, Evaluation of Programs, Optimizations for Scalar Architectures and Models for Parallel Computing.
- Explain the concepts of Distributed systems:
  - Characterization of Distributed systems
  - system models
  - distributed objects and
  - remote method invocation
- Implement the concept of Distributed transactions:
  - Explain flat & nested distributed transactions and concurrency

Explain Service-oriented architectures:

Identify the characteristics of SOAs, Hadoop and Spark Define Mobile and wireless computing

○ Enumerate the Technologies for Wireless Communication ○

Explain wireless cellular systems

○ Appreciate wireless network technologies

Discuss Wireless Application Protocols: Mobile IP, WAP, SMS, Bluetooth

Implement the Frameworks for mobile application development (e.g Ionic, React Native, Xamarin, Adobe PhoneGap, J2ME)

Define Cloud computing:

○ Explain the cloud computing technologies, infrastructure, and architecture

Discuss Cloud computing development models (public, private, community and hybrid cloud), service models (SaaS, PaaS, IaaS). Improve their data privacy through hardware protection

### **Working through this course**

To complete this course, you are required to read each study unit, read the textbooks and read other materials which may be provided by the CODEL

Each unit contains self-assessment exercises and at certain point in the course, you would be required to submit assignments for assessment purposes. At the end of the course there is a final examination. The course should take you about a total of 17 weeks to complete. Below, you will find listed all the components of the course, what you have to do and how you should allocate your time to each unit in order to complete the course on time and successfully.

This course entails that you spend a lot time reading. I would advise that you avail yourself the opportunity of comparing your knowledge with that of other learners.

### **Course Material**

The major components of the course are:

Course Guide

Study Units

Presentation Schedule

Tutor-Marked Assignments

References/Further Reading

## Study Units

The study units in this course are as follows:

### **Module 1 Net-Centric Computing Fundamentals**

- Unit 1 Introduction to Distributed Computing
- Unit 2 Mobile and Wireless Computing
- Unit 3 Network Security
- Unit 4 Client/ Server Computing (Using the web)
- Unit 5 Building Web Application

### **Module 2 Parallel Systems**

- Unit 1 Introduction to Parallel Systems
- Unit 2 Parallel Programming Models
- Unit 3 Message Passing Programming
- Unit 4 Dependence Analysis
- Unit 5 Open MP Programming
- Unit 6 Evaluation of Programs

### **Module 3 Distributed Systems**

- Unit 1 Introduction to Distributed Systems
- Unit 2 Systems Models
- Unit 3 Distributed Objects
- Unit 4 Remote Methods Invocation
- Unit 5 Using UML for Component Based Designs

### **Module 4 Distributed Transactions**

- Unit 1 Introduction to Distributed Transactions
- Unit 2 Flat and Nested Distributed Transactions
- Unit 3 Concurrency
- Unit 4 Characteristics of Service Oriented Architecture- Hadoop & Spark

### **Module 5 Mobile & Cloud Computing**

- Unit 1 Introduction to Mobile and Cloud Computing
- Unit 2 Technologies for Wireless Communications
- Unit 3 Wireless Cellular Systems
- Unit 4 Overview of Wireless LAN, IEEE 802.11, Personal Area Network, Bluetooth
- Unit 5 High Speed Wireless Network: HiperLAN

The first module introduces Distributed Computing, dynamic devices and mode of transmission, security of connected and communicating systems

using ciphers, Client and Servers communication and building of web applications.

Module Two explains parallel systems and parallel programming models. Other issues treated are Message passing Programming, Dependence Analysis, Open MP Programming, Program Evaluation using Algorithms, Optimizations for Scalar Architectures and Models for Parallel Computing.

In module Three, we have discussed connected Systems, its models and characteristics, Distributed Objects, Remote Method Invocation and using UML for Component Based Design. Module Four treated Transactions on Connected Systems, Flat and Nested Distributed Transactions, Simultaneity or Concurrency. Module Five introduces Mobile devices and the Internet, Wireless Communications Technologies, Wireless Cellular Systems, Wireless Local Area Networks, Personal Area Networks, IEEE 802.11 and Bluetooth and High-speed Wireless Networks

Each unit consists of one or two weeks' work and include an introduction, objectives, reading materials, exercises, conclusion, summary, tutor-marked assignments (TMAs), references and other resources. The units direct you to work on exercises related to the required reading. In general, these exercises test you on the materials you have just covered or require you to apply it in some way to assist you in evaluating your progress and to reinforce your comprehension of the material. Together with TMAs, these exercises will help you in achieving the stated learning objectives of the individual units and of the course as a whole.

### **Presentation Schedule**

Your course materials have important dates for early, timely completion and submission of your TMAs and attending tutorials. You should remember that you are required to submit all your assignments by the stipulated time and date. You should guide against working behind deadlines.

### **Assessment**

There are three aspects to the assessment of the course. First is made up of self-assessment exercises. Second, consists of the tutor-marked assignments and third is the written examination/end of course examination.

You are advised to do the exercises. In tackling the assignments, you are expected to apply information, knowledge and techniques you have

gathered during the course. The assignments must be submitted to your facilitator for formal assessment in accordance with the deadline stated in the presentation schedule and the assessment file. The work you submit to your tutor for assessment will count for 30% of your total course mark. At the end of the course, you will need to sit for a final or end of course examination of about three hours duration. This examination will count for 70% of your total course mark.

### **Tutor-Marked Assignment (TMAs)**

The TMA is a continuous assessment component of your course. It accounts for 30% of the total score. You will be given four TMAs to answer. Three of these must be answered before you are allowed to sit for end of course examination. The TMAs would be given to you by your facilitator and should be returned after you have done the assignment. Assignment questions for the units in this course are contained in the assignment file. You will be able to complete your assignments from the information and material contained in your reading, references and study units. However, it is desirable in all degree level of education to demonstrate that you have read and researched more into your references, which will give a wider view point and may provide you with a deeper understanding of the subject.

Make sure that each assignment reaches your facilitator on or before the deadline given in the presentation schedule and assignment file. If for any reason you cannot complete your work on time, contact your facilitator before the assignment is due to discuss the possibility of an extension. Extension will not be granted after the due date unless in exceptional circumstances.

### **Final Examination and Grading**

The end of course examination for Net-centric Computing (CIT412) will be for three (3) hours and it has a value of 70% of the total course score. The examination will consist of questions, which will reflect the type of self-testing, practice exercise and tutor-marked assignment problems you have previously encountered. All areas of the course will be assessed.

Use the time between finishing the last unit and sitting for the examination to revise the whole course. You might find it useful to review your self-test, TMAs and comments on them before the examination. The end of course examination covers information from all parts of the course.



### Course Marking Scheme

Assignment	Marks
Assignment 1 – 4	For assignment, best three marks of the four counts at 10% each, i.e., 30% of Course Marks.
End of Course Examination	70% Of the overall Course Marks.
Total	100% of Course Material.

### Facilitators/Tutors and Tutorials

There are 16 hours of tutorials provided in support of this course. You will be notified of the dates, time, and location of these tutorials as well as the name and phone number of your facilitator, as soon as you are allocated to a tutorial group.

Your facilitator will mark and comment on your assignments, keep a close watch on your progress and any difficulties you might face and provide assistance to you during the course. You are expected to mail your Tutor-Marked Assignments to your facilitator before the scheduled date (at least two working days are required). They will be marked by your tutor and returned to you as soon as possible.

Do not delay to contact your facilitator by telephone or e-mail if you need assistance. However, the following might however be the circumstances in which you would find assistance necessary, hence you would have to contact your facilitator if:

You do not understand any part of the study or assigned readings  
You have difficulty with self-tests

You have question or problem with an assignment or with the grading of an assignment.

You should endeavor to attend the tutorials. This is the only chance to have face to face contact with your course facilitator and to ask questions which may be answered instantly. You can raise any problem encountered in the course of your study.

To have more benefits from course tutorials, you are advised to prepare a list of questions before attending them. You will learn a lot from participating actively in discussions.

### Summary

Net-centric Computing is a course that intends to intimate the learner with basic facts on computer networks, network types and categories,

distributed systems, distributed systems models, parallel systems, concurrency, wireless networks and standards, cloud computing and wireless application protocols. Upon completing this course, you would have been equipped with the knowledge of Net-centric computing fundamentals, what network is all about and wireless technologies, cloud computing and service models.

I wish you success in the course and I hope you find it very interesting.

**MAIN  
COURSE**

<b>CONTENTS</b>		<b>PAGE</b>
<b>Module 1</b>	<b>Net-Centric Computing Fundamentals....</b>	<b>1</b>
Unit 1	Introduction to Distributed Computing.....	1
Unit 2	Mobile and Wireless Computing .....	2
Unit 3	Network Security .....	22
Unit 4	Client/ Server Computing (Web Application)...	28
Unit 5	Building Web Application.....	33
<b>Module 2</b>	<b>Parallel Systems .....</b>	<b>49</b>
Unit 1	Introduction to Parallel Systems .....	49
Unit 2	Parallel Programming Models .....	64
Unit 3	Message Passing Programming .....	71
Unit 4	Dependence Analysis .....	77
Unit 5	OpenMP Programming .....	82
Unit 6	Evaluation of Programs .....	89
<b>Module 3</b>	<b>Distributed Systems .....</b>	<b>96</b>
Unit 1	Introduction to Distributed Transactions.....	96
Unit 2	Systems Models.....	104
Unit 3	Distributed Objects.....	116
Unit 4	Remote Method Invocation .....	121
Unit 5	Using UML for Component Based Design ...	131
<b>Module 4</b>	<b>Distributed Transactions .....</b>	<b>140</b>
Unit 1	Introduction to Distributed Transactions.....	140
Unit 2	Flat & Nested Distributed Transactions .....	160
Unit 3	Concurrency.....	169
Unit 4	Characteristics of Service Oriented Architecture (Hadoop & Spark).....	180
<b>Module 5</b>	<b>Mobile &amp; Cloud Computing .....</b>	<b>184</b>
Unit 1	Introduction to Mobile & Cloud Computing ...	184
Unit 2	Technologies for Wireless Communications ...	193
Unit 3	Wireless Cellular Systems .....	199
Unit 4	Overview of Wireless LAN, IEEE 802.11, Personal Area Network & Bluetooth.....	207
Unit 5	High Speed Wireless Networks: HiperLAN ...	218

## **MODULE 1: NET-CENTRIC COMPUTING FUNDAMENTALS**

### **UNIT 1: INTRODUCTION TO DISTRIBUTED COMPUTING**

#### **CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main Content
  - 3.1 Distributed Computing
  - 3.2 Web 2.0 Technologies
  - 3.3 Service Orientation
  - 3.4 Virtualization
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

#### **INTRODUCTION TO MODULE**

Module 1 presents the essentials of Net-centric computing. Here, we are going to discuss the concepts of established networks in different location and job running on each simultaneously (distributed computing) the output of which are to be combined after completion, mobility issues and the security of resources running on different autonomous systems as well as when being transmitted from one point to the other. Others include client/server concepts and web building.

#### **1.0 INTRODUCTION**

A Distributed System is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another from any system in order to appear as a single system to the end-user. The computers that are in a distributed system can be physically together and connected by a local network, or they can be geographically distant and connected by a wide area network. A distributed system can consist of any number of possible components such as mainframes, personal computers, workstations, minicomputers, and so on. Common use cases of a distributed systems are electronic banking systems, massive multiplayer online games, and sensor networks.

## 1.1 Functionality

There are two general ways that distributed systems function:

Each component of the system works to achieve a common goal and the end-user views results as one combined unit.

Each component has its own end-user and the distributed system facilitates sharing resources or communication services.

## 1.2 Architectural models

Distributed systems generally consist of four different basic architectural models:

Client-server — Clients contact the server for data, then format it and display it to the end-user.

Three-tier — Information about the client is stored in a middle tier rather than on the client, to simplify application deployment.

n-tier — Generally used when the server needs to forward requests to additional enterprise services on the network.

Peer-to-peer — There are no additional nodes used to provide services or manage resources. Responsibilities are uniformly distributed among components in the system, known as peers, which can serve as either client or server.

## 2.0 INTENDED LEARNING OUTCOMES (ILOS)

By the end of this unit, you will be able to:

Explain the concept of distributed systems

Describe the Architectural Models of Distributing Computing Explain the term, Service Orientation

Describe the concept, Virtualization

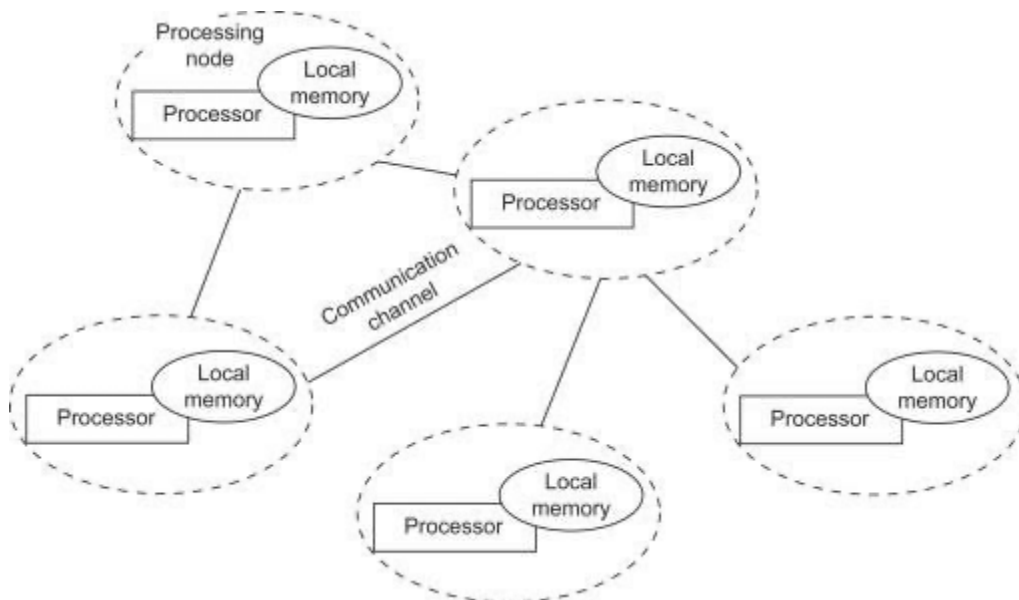
## 3.0 MAIN CONTENT

### 3.1 Distributed computing

Distributed Computing is a much broader technology that has been around for more than three decades now. Distributed computing is computing over distributed autonomous computers that communicate only over a network. Distributed computing systems are usually treated differently from parallel computing systems or shared-memory systems where multiple computers share a common memory pool that is used for communication between the processors. Distributed memory systems use multiple computers to solve a common problem, with computation

distributed among the connected computers (nodes) and using message-passing to communicate between the nodes.

Example of distributed computing is the grid computing where the nodes may belong to different administrative domains. Another example is the network-based storage virtualization solution which uses distributed computing between data and metadata servers.



*Figure 1.1.1: Distributed Computing Systems*

Distributed computing, however, can include heterogeneous computations where some nodes may perform a lot more computation, some perform very little computation and a few others may perform specialized functionality (like processing visual graphics).

One of the main advantages of using distributed computing is that efficient scalable programs can be designed so that independent processes are scheduled on different nodes and they communicate only occasionally to exchange results – as opposed to working out of a shared memory with multiple simultaneous accesses to a common memory.

It is obvious that cloud computing is also a specialized form of distributed computing, where distributed Software as a Service (SaaS) applications utilize thin clients (such as browsers) which offload computation to cloud-hosted servers (and services).

Distributed computing, virtualization, service orientation, and Web 2.0 form the core technologies enabling the provisioning of cloud services from anywhere on the globe.

Distributed computing is a foundational model for cloud computing because cloud systems are distributed systems. Besides administrative tasks mostly connected to the accessibility of resources in the cloud, the

extreme dynamism of cloud systems—where new nodes and services are provisioned on demand—constitutes the major challenge for engineers and developers.

### 3.2 Web 2.0 technologies

Web 2.0 technologies constitute the interface through which cloud computing services are delivered, managed, and provisioned. Besides the interaction with rich interfaces through the Web browser, Web services have become the primary access point to cloud computing systems from a programmatic standpoint.

### 3.3 Service Orientations

Service orientation is the underlying paradigm that defines the architecture of a cloud computing system. Cloud computing is often summarized with the acronym *XaaS meaning, Everything-as-a-Service*—that clearly underlines the central role of service orientation. Infrastructure-as-a-Service solutions provide the capabilities to add and remove resources, but it is up to those who deploy systems on this scalable infrastructure to make use of such opportunities with wisdom and effectiveness.

Platform-as-a-Service solutions embed into their core offering algorithms and rules that control the provisioning process and the lease of resources. These can be either completely transparent to developers or subject to fine control. Integration between cloud resources and existing system deployment is another element of concern.

### 3.4 Virtualization

Virtualization is another element that plays a fundamental role in cloud computing. This technology is a core feature of the infrastructure used by cloud providers. Virtualization concept is more than 40 years old but cloud computing introduces new challenges, especially in the management of virtual environments, whether they are abstractions of virtual hardware or a runtime environment

#### Discussion

Which of the security infrastructure is most critical and why?

### 4.0 Self-Assessment Exercises

Describe two Service Orientations  
What is Virtualization?

Give an advantage of Distributed Computing.

**Answer**

Infrastructure-as-a-Service solutions provide the capabilities to add and remove resources, but it is up to those who deploy systems on this scalable infrastructure to make use of such opportunities with wisdom and effectiveness.

Platform-as-a-Service solutions embed into their core offering algorithms and rules that control the provisioning process and the lease of resources. These can be either completely transparent to developers or subject to fine control. Integration between cloud resources and existing system deployment is another element of concern.

Virtualization is the core feature of the infrastructure used by cloud providers. but cloud computing introduces new challenges, especially in the management of virtual environments. Virtual environments could be abstractions of virtual hardware or a runtime environment.

One of the main advantages of using distributed computing is that efficient scalable programs can be designed so that independent processes are scheduled on different nodes and they communicate only occasionally to exchange results

**5.0 CONCLUSION**

Distributed computing is computing over distributed autonomous computers that communicate only over a network. Distributed computing systems are usually treated differently from parallel computing systems or shared-memory systems, where multiple computers share a common memory pool that is used for communication between the processors

**6.0 SUMMARY**

Virtualization is another element that plays a fundamental role in cloud computing. Platform-as-a-Service solutions embed into their core offering algorithms and rules that control the provisioning process and the lease of resources. Infrastructure-as-a-Service solutions provide the capabilities to add and remove resources, but it is up to those who deploy systems on this scalable infrastructure to make use of such opportunities with wisdom and effectiveness. One of the main advantages of using distributed computing is that efficient scalable programs can be designed so that independent processes are scheduled on different nodes and they communicate only occasionally to exchange results.



## **7.0 REFERENCES/FURTHER READING**

[Moving To The Cloud | ScienceDirect](#)

## **UNIT 2: MOBILE & WIRELESS COMPUTING**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main Content
  - 3.1 Mobile and Wireless Computing
    - 3.1.1 Mobile Computing
  - 3.2 Mobile Communications
  - 3.3 Mobile Hardware
  - 3.4 Mobile Software
  - 3.5 Mobile Classification
  - 3.6 Advantages
  - 3.7 Security Issues
  - 3.8 Current Trends
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

Mobile and wireless computing is a human–computer interaction concept in which a computer could be in motion during normal usage. Mobile and wireless computing involves mobile communication, mobile hardware and mobile software, does involve the use of physical cable but devices are connected through electromagnetic waves. The farther the usage location to the network source the less the intensity of the cloud and speed of connection and vice versa.

### **2.0 INTENDED LEARNING OUTCOMES (ILOS)**

By the end of this unit, you will be able to:

- Define Wireless Communication
- Classify Mobile and Cloud Computing
- Mention 5 mobile hardware

### **3.0 MAIN CONTENT**

#### **3.1 Mobile & Wireless Computing**

Mobile and wireless Computing is a technology that allows transmission of data, voice and video via a computer or any other wireless enabled device without having to be connected to a fixed physical link.

**Computing Technologies** are the technologies that are used to manage, process, and communicate the data. *Wireless* simply means without any wire i.e. connecting with other devices without any physical connection. Wireless computing is transferring the data or information between computers or devices that are not physically connected to each other and having a “wireless network connection”. For example, mobile devices, Wi-Fi, wireless printers and scanners, etc. Mobiles are not physically connected but then too we can transfer data.

Mobile is a computing device that not require any network connection or any connection to transfer data or information between devices. For example laptops, tablets, smartphones, etc. Mobile computing allows transferring of the data/information, audio, video, or any other document without any connection to the base or central network. These computing devices are the most widely used technologies nowadays.

There are some wireless/mobile computing technologies given below:

#### **Global System for Mobile Communications (GSM) :**

GSM is a Current circuit-switched wireless data communication technology. It is established in Europe by ETSI (European Telecommunications Standards Institute) in the mid-1980s. GSM network has 4 different parts that who's functions are different: Mobile Station, BSS (Base Station Subsystem), NSS (Network Switching Subsystem), OSS (Operation and Support Subsystem).

As the name suggests, GSM is widely used for the mobile communication system. It operates in the frequency band 900-MHz, 1800-MHz, and 1900-MHz. GSM is developed using TDMA (Time Division Multiple Access) for better communication using mobile. It is the most widely used mobile communication system and is mostly required nowadays. It can achieve maximum data transmission speed or data transmission rate up to 9.6Kbps (Kilobits per second).

#### **Code-Division Multiple Access (CDMA) :**

CDMA is a type of wireless computing technology. It is developed during World War II. This technology is mostly used as it provides better network quality, more storage capacity for voice and data communications than TDMA, decreases system noise and interference using power control, provides more security by encoding the user transmission data into a unique code.

CDMA does not provide any user with a specific frequency instead utilizes the entire frequency spectrum available for transmission. It

operates in the frequency range of 800 MHz to 1.9 GHz. It uses Soft Handoff that reduces signal breaks.

### **Wireless in Local Loop (WLL) :**

WLL is a widely used technology for wireless communication systems. It is also called a Fixed Wireless Loop. WLL is very easy to develop and less time is required to install, very cost-effective as wireless systems are less expensive because the cost of cable installation is not added.

WLL allows users to connect to the local telephone station using a wireless link and provides advanced features of customer service. It provides high-quality data transmission and a high data rate. Generally, two types of WLL techniques are available: Local Multipoint Distribution Service (LMDS) and Multichannel Multipoint Distribution Service (MMDS).

### **General Packet Radio Service (GPRS) :**

GPRS is a type of Packet-based Wireless communication technology. It is established by ETSI (European Telecommunications Standards Institute). GPRS can achieve a data transfer rate of up to 114Kbps. It is very cost-effective, highly stable, can achieve a maximum data rate of up to 114Kbps (Kilobits per second). It supports Internet Protocol (IP), X.25 (standard protocol for packet-switched data communication), Point-to-Point protocol (PPP), and based on Gaussian minimum-shift keying (GMSK) which is a modulation technique.

The Gateway GPRS Service Node (GGSN) and the Serving GPRS Service Node (SGSN) are the two core modules required to enable GPRS on GSM network or TDMA network.

### **Short Message Service (SMS) :**

SMS is originally created for a phone/mobile that uses GSM Global System for Mobile communication). This service is used to send text messages even without the Internet connection between two or more mobile devices. This technique is very easy, user-friendly, comfortable and the most effective means of wireless communication.

In this service, less time is required for communication. It does not require any Internet connection for sending text messages. It allows the transmission of short messages i.e. up to 160 characters in length. SMS uses standardized communication protocols. SMS is received by Short Message Service Center (SMSC).

Figure 3.1 below shows the Internet and various devices connected to it wirelessly for communication all over the world.



*Figure 1.2.1: Internet with Mobile Devices connected*

### **3.2 Mobile communication**

The mobile communication refers to the infrastructure put in place to ensure that seamless and reliable communication goes on

These would include devices such as protocols, services, bandwidth, and portals necessary to facilitate and support the stated services

The data format is also defined at this stage

This ensures that there is no collision with other existing systems which offer the same service.

the media is unguided/unbounded, the overlaying infrastructure is basically radio wave-oriented

That is, the signals are carried over the air to intended devices that are capable of receiving and sending similar kinds of signals.

### **3.3 Mobile hardware**

mobile devices or device components that receive or access the service of mobility

They would range from portable laptops, smartphones, tablet Pc's, Personal Digital Assistants



Figure 1.2.2: Mobile Hardware

### 3.4 Mobile Software

Mobile software is the actual program that runs on the mobile hardware

It deals with the characteristics and requirements of mobile applications

This is the engine of the mobile device

It is the operating system of the appliance

Its the essential component that operates the mobile device

Since portability is the main factor, this type of computing ensures that users are not tied or pinned to a single physical location, but are able to operate from anywhere. It incorporates all aspects of wireless communications



Figure 1.2.3: Mobile Software

### 3.5 Mobile Classification

Mobile computing is not only limited to mobile phones, but there are various gadgets available in the market that are built on a platform to support mobile computing

They are usually classified in the following categories:

#### 3.5.1. Personal Digital Assistant (PDA)



Figure 1.2.4: Personal Data Assistant

The main purpose of this device is to act as an electronic organizer or day planner that is portable, easy to use and capable of sharing information with your computer systems.

PDA is an extension of the PC, not a replacement

These systems are capable of sharing information with a computer system through a process or service known as synchronization

Both devices will access each other to check for changes or updates in the individual devices

The use of infrared and Bluetooth connections enables these devices to always be synchronized.

With PDA devices, a user can browse the internet, listen to audio clips, watch video clips, edit and modify office documents, and many more services

The device has a stylus and a touch sensitive screen for input and output purposes

#### 3.5.2 Smartphones

It combines the features of a PDA with that of a mobile phone or camera phone

It has a superior edge over other kinds of mobile phones.

Smartphones have the capability to run multiple programs concurrently

These phones include high-resolution touch screens, web browsers that can:

access and properly display standard web pages rather than just mobile-optimized sites

high-speed data access via Wi-Fi and high speed cellular broadband.

The most common mobile Operating Systems (OS) used by modern smartphones include:

Google's Android

Apple's iOS

Nokia's Symbian

RIM's BlackBerry OS

Samsung's Bada

Microsoft's Windows Phone, and embedded Linux distributions such as Maemo and MeeGo. Such operating systems can be installed on different phone models, and typically each device can receive multiple OS software updates over its lifetime.



*Figure 1.2.5: Smart Phones*

### **3.5.3 Tablet PC and iPads**

This mobile device is larger than a mobile phone or a PDA and integrates into a touch screen and is operated using touch sensitive motions on the screen. They are often controlled by a pen or by the touch of a finger.

They are usually in slate form and are light in weight. Examples would include iPads, Galaxy Tabs, BlackBerry Playbooks etc.

They offer the same functionality as portable computers.



They support mobile computing in a far superior way and have enormous processing horsepower

Users can edit and modify document files, access high speed internet, stream video and audio data, receive and send e-mails, attend/give lectures and presentations among its very many other functions

They have excellent screen resolution and clarity



Figure 1.2.6: Ipads & PCs

### 3.6 Advantages of Mobile Computing

#### Location Flexibility

This has enabled users to work from anywhere as long as there is a network connection established

A user can work without being in a fixed position

Their mobility ensures that they are able to carry out numerous tasks at the same time and perform their stated jobs.

#### Saves Time

The time consumed or wasted while travelling from different locations or to the office and back, has been slashed

One can now access all the important documents and files over a secure channel or portal and work as if they were on their computer

It has enhanced telecommuting in many companies

It has also reduced unnecessary incurred expenses

**Enhanced Productivity**

Users can work efficiently and effectively from whichever location they find comfortable

This in turn enhances their productivity level

**Ease of Research**

Research has been made easier, since users earlier were required to go to the field and search for facts and feed them back into the system

It has also made it easier for field officers and researchers to collect and feed data from wherever they are without making unnecessary trips to and from the office to the field

**Entertainment**

Video and audio recordings can now be streamed on-the-go using mobile computing

It's easy to access a wide variety of movies, educational and informative material

With the improvement and availability of high speed data connections at considerable cost, one is able to get all the entertainment they want as they browse the internet for streamed data

One is able to watch news, movies, and documentaries among other entertainment offers over the internet

This was not possible before mobile computing dawned on the computing world.

**Streamlining of Business Processes**

Business processes are now easily available through secured connections

Looking into security issues, adequate measures have been put in place to ensure authentication and authorization of the user accessing the services

Some business functions can be run over secure links and sharing of information between business partners can also take place

Meetings, seminars and other informative services can be conducted using video and voice conferencing

Travel time and expenditure is also considerably reduced

### **3.7 Security Issues**

Mobile computing has its fair share of security concerns as any other technology

Due to its nomadic nature, it's not easy to monitor the proper usage

Users might have different intentions on how to utilize this privilege

Improper and unethical practices such as hacking, industrial espionage, pirating, online fraud and malicious destruction are some but few of the problems experienced by mobile computing

Another big problem plaguing mobile computing is credential verification

As other users share username and passwords, it poses as a major threat to security

This being a very sensitive issue, most companies are very reluctant to implement mobile computing to the dangers of misrepresentation

The problem of identity theft is very difficult to contain or eradicate

Issues with unauthorized access to data and information by hackers, is also an enormous problem

Outsiders gain access to steal vital data from companies, which is a major hindrance in rolling out mobile computing services.

No company wants to lay open their secrets to hackers and other intruders, who will in turn sell the valuable information to their competitors

It's also important to take the necessary precautions to minimize these threats from taking place

Some of those measures include:

Hiring qualified personnel.

Installing security hardware and software

Educating the users on proper mobile computing ethics

Auditing and developing sound, effective policies to govern mobile computing

Enforcing proper access rights and permissions

In the absence of such measures, it's possible for exploits and other unknown threats to infiltrate and cause irrefutable harm

These may be in terms of reputation or financial penalties

In such cases, it's very easy to be misused in different unethical practices.

If these factors aren't properly worked on, it might be an avenue for constant threat

Various threats still exist in implementing this kind of technology

### 3.8 Current Trends

These are the list of the current mobile technologies starting from 5G technologies which is the hottest mobile technology available in the market.

#### 3.8.1. 5G

In telecommunications, **5G** is the fifth generation technology standard for broadband cellular networks. Telecoms company began to deploy it in 2019, and is the planned successor to the 4G networks providing connectivity to most current cellphones. 5G networks are predicted to have more than 1.7 billion subscribers worldwide by 2025. 5G networks cellular networks' service area is divided into small geographical areas called cells. All 5G wireless devices in a cell are connected to the Internet and telephone network by radio waves through a local antenna in the cell. It has an advantage of having greater bandwidth and download speeds up to 10 gigabits per second (Gbit/s). 5G is not only faster than existing networks, 5G can connect more different devices. Due to the increased bandwidth, the networks will increasingly be used as general internet service providers (ISPs) for laptops and desktop computers, competing with existing ISPs such as cable internet, and also will make possible new applications in internet-of-things (IoT) and machine-to-machine areas

#### 3.8.2 4G

- **4G** is the fourth generation of broadband cellular network technology that precedes 5G. A 4G system must provide capabilities defined by ITU in IMT Advanced. Recent applications include amended mobile web access, IP telephony,

gaming services, high-definition mobile TV, video conferencing, and 3D television.

WIMAX standard was first-released commercially and deployed in South Korea in 2006 and has since been deployed in most parts of the world.

Long Term Evolution (LTE) standard was first-released commercially and deployed in Oslo, Norway, and Stockholm, Sweden in 2009, and has since been deployed throughout most parts of the world. It was to be considered as 4G LTE. The 4G wireless cellular standard was defined by the International Telecommunication Union (ITU) and specifies the key characteristics of the standard, including transmission technology and data speeds.

### 3.8.3. 3G or third generation

3G mobile telecommunications is a generation of standards for mobile phones and mobile telecommunication services fulfilling the International Mobile Telecommunications-2000 (IMT-2000) specifications by the International Telecommunication Union. Application services include wide-area wireless voice telephone, mobile Internet access, video calls and mobile TV, all in a mobile environment.

### 3.8.4. Global Positioning System (GPS)

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather, anywhere on or near the Earth, where there is an unobstructed line of sight to four or more GPS satellites

The GPS program empowers the military, civil and commercial users around the world

It (GPS) is the backbone for modernizing the global air traffic system, weather, and location services.

### 3.8.5. Long Term Evolution (LTE)

LTE is a standard for wireless communication of high-speed data for mobile phones and data terminals

It is based on the GSM/EDGE and UMTS/HSPA network technologies, increasing the capacity and speed using new modulation techniques

It is related with the implementation of fourth Generation (4G) technology

### 3.8.6. WiMAX

WiMAX (Worldwide Interoperability for Microwave Access) is a wireless communications standard designed to provide 30 to 40 megabit-per-second data rates, with the latest update providing up to 1 Gbit/s for fixed stations

It is a part of a fourth generation or 4G wireless-communication technology

WiMAX far surpasses the 30-metre wireless range of a conventional Wi-Fi Local Area Network (LAN), offering a metropolitan area network with a signal radius of about 50 km

WiMAX offers data transfer rates that can be superior to conventional cable-modem and DSL connections, however, the bandwidth must be shared among multiple users and thus yields lower speed in practice

### 3.8.7. Near Field Communication

Near Field Communication (NFC) is a set of standards for smartphones and similar devices to establish radio communication with each other by touching them together or bringing them into close proximity, usually no more than a few centimeters

Present and anticipated applications include contactless transactions, data exchange, and simplified setup of more complex communications such as Wi-Fi. Communication is also possible between an NFC device and an unpowered NFC chip, called a "tag"

## 3.9 Conclusion

Today's computing has rapidly grown from being confined to a single location

With mobile computing, people can work from the comfort of any location they wish to as long as the connection and the security concerns are properly factored

In the same light, the presence of high-speed connections has also promoted the use of mobile computing

Being an ever growing and emerging technology, mobile computing will continue to be a core service in computing, and Information and Communications Technology

## Discussion

Why is the Mobile Software important in Mobile and Cloud Computing?

#### 4.0 Self-Assessment Exercise

Define Mobile Computing.

##### Answer

**Mobile computing** is a human–computer interaction in which a computer could be in motion during normal usage.

Explain Near Field Communication as one of the current trends in Mobile Computing

##### Answer

##### Near Field Communication

Near Field Communication (NFC) is a set of standards for smartphones and similar devices to establish radio communication with each other by touching them together or bringing them into close proximity, usually no more than a few centimeters

Present and anticipated applications include contactless transactions, data exchange, and simplified setup of more complex communications such as Wi-Fi. Communication is also possible between an NFC device and an unpowered NFC chip, called a "tag"

#### 5.0 CONCLUSION

Mobile and Wireless Computing has come to stay in every of our life endeavors ranging from homes, commerce, education as well as finance. I doubt if we can recover from it.

#### 6.0 SUMMARY

Mobile and wireless Computing is a technology that allows transmission of data, voice and video via a computer or any other wireless enabled device without having to be connected to a fixed physical link. Being an ever growing and emerging technology, mobile computing will continue to be a core service in computing, and Information and Communications Technology. Mobile classification are usually classified into personal digital assistant (PDA), smartphones and tablet PC and iPads. The advantages of mobile computing are location flexibility, time savings, enhanced productivity, ease of research, entertainment and streamlining of business processes. Mobile computing has its fair share of security concerns as any other technology. Current Trends in mobile technology include 5G, 4G, 3G or third generation, Global Positioning System (GPS), Long Term Evolution (LTE), WiMAX and Near Field Communication.

## 7.0 REFERENCES/FURTHER READING

<file:///C:/Tech-U%20Issues/CSC%20412-%20Netcetric%20Computing/CSC%20412-Content/MOBILE%20AND%20WIRELESS%20COMPUTING.pdf>

<https://www.geeksforgeeks.org/wireless-mobile-computing-technologies/>



## UNIT 3 NETWORK SECURITY

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Fundamentals of Network Security
  - 3.1 Network Security
  - 3.2 Data as the Life-Blood of Business
  - 3.3 Three Keys Focuses of Network Security
  - 3.4 Benefits of Network Security
  - 3.5 Network Security Tools and Techniques
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

The transmission of data from one point A on the network to the other point, B is a great concern and therefore, there is the need to deploy measure that can secure the transmission of data away from unauthorized individuals. Hence, the need for network security.

### 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, students will be able to:

Explain the concept of network

Understand the importance of network security

Identify and explain the network security tools and techniques.

### 3.0 Main Content

#### 3.1 Network Security

Network security is a term that describes the security tools, tactics and security policies designed to monitor, prevent and respond to unauthorized network intrusion, while also protecting digital assets, including network traffic. Network security includes hardware and software technologies (including resources such as savvy security analysts, hunters, and incident responders) and is designed to respond to the full range of potential threats targeting your network.

Network security is the defense you use to protect yourself against ever-increasing cybercrime.

### 3.3 The Three Key Focuses of Network Security

There are three key focuses that should serve as a foundation of any network security strategy: protection, detection and response.

Protection entails any tools or policies designed to prevent network security intrusion.

*Detection* refers to the resources that allow you to analyze network traffic and quickly identify problems before they can do harm.

*Response* is the ability to react to identified network security threats and resolve them as quickly as possible.

### 3.4 Benefits of Network Security

Network security tools and devices enable organizations to protect its sensitive information, overall performance, reputation and its continuity in business. Secure and reliable networks protect not just organizational interests and operations, but also any client or customer who exchanges information with the organization, in addition to the general public. The benefits of network security are:

#### **Builds trust**

Security for large systems translates to security for everyone. Network security boosts client and consumer confidence, and it protects your business from the reputational and legal fallout of a security breach.

#### **Mitigates risk**

The right network security solution will help your business stay compliant with business and government regulations, and it will minimize the business and financial impact of a breach if it does occur.

#### **Protects proprietary information**

Your clients and customers rely on you to protect their sensitive information. Your business relies on that same protection, too. Network security ensures the protection of information and data shared across the network.

#### **Enables a more modern workplace**

From allowing employees to work securely from any location using VPN to encouraging collaboration with secure network access, network security provides options to enable the future of work. Effective network security also provides many levels of security to scale with your growing business.

### 3.5 Network Security Tools and Techniques

Enterprises' network encounter varying degrees of threats, and therefore should be prepared to defend, identify and respond to a full range of attacks. However, the reality is that the biggest danger to most companies are not fly-by-night threat actors, but the attackers that are well-funded and are targeting specific organizations for specific reasons. Hence, network security strategy needs to be able to address the various methods these actors might employ. Here are 14 different network security tools and techniques designed to help you do just that:

**Access control:** If threat actors cannot access your network, the amount of damage they will be able to do will be extremely limited. But in addition to preventing unauthorized access, be aware that even *authorized* users can be potential threats. Access control allows you to increase your network security by limiting user access and resources to only the parts of the network that directly apply to individual users' responsibilities.

**Anti-malware software:** Malware, in the form of viruses, trojans, worms, keyloggers, spyware, etc. are designed to spread through computer systems and infect networks. Anti-malware tools are a kind of network security software designed to identify dangerous programs and prevent them from spreading. Anti-malware and antivirus software may also be able to help resolve malware infections, minimizing the damage to the network.

**Anomaly detection:** It can be difficult to identify anomalies in your network without a baseline understanding of how that network *should* be operating. Network anomaly detection engines (ADE) allow you to analyze your network, so that when breaches occur, you will be alerted to them quickly enough to be able to respond.

**Application security:** For many attackers, applications are a defensive vulnerability that can be exploited. Application security helps establish security parameters for any applications that may be relevant to your network security.

**Data Loss Prevention (DLP):** Often, the weakest link in network security is the human element. DLP technologies and policies help protect staff and other users from misusing and possibly compromising sensitive data or allowing said data out of the network.

**Email security:** As with DLP, email security is focused on shoring up human-related security weaknesses. Via phishing strategies (which are often very complex and convincing), attackers persuade email recipients to share sensitive information via desktop or mobile device, or inadvertently download malware into the targeted network. Email security helps identify dangerous emails and can also be used to block attacks and prevent the sharing of vital data.

**Endpoint security:** The business world is becoming increasingly, “*bring your own device*” (BYOD), to the point where the distinction between personal and business computer devices is almost non-existent. Unfortunately, sometimes the personal devices become targets when users rely on them to access business networks. Endpoint security adds a layer of defense between remote devices and business networks.

**Firewalls:** Firewalls function much like gates that can be used to secure the borders between your network and the internet. Firewalls are used to manage network traffic, allowing authorized traffic through while blocking access to non-authorized traffic.

**Intrusion prevention systems:** Intrusion prevention systems (also called intrusion detection) constantly scan and analyze network traffic/packets, so that different types of attacks can be identified and responded to quickly. These systems often keep a database of known attack methods, so as to be able to recognize threats immediately.

**Network segmentation:** There are many kinds of network traffic, each associated with different security risks. Network segmentation allows you to grant the right access to the right traffic, while restricting traffic from suspicious sources.

**Security information and event management (SIEM):** Sometimes simply pulling together the right information from so many different tools and resources can be prohibitively difficult — particularly when time is an issue. SIEM tools and software give responders the data they need to act quickly.

**Virtual private network (VPN):** VPN tools are used to authenticate communication between secure networks and an endpoint device. Remote-access VPNs generally use IPsec or Secure Sockets Layer (SSL) for authentication, creating an encrypted line to block other parties from eavesdropping.

**Web security:** Including tools, hardware, policies and more, web security is a blanket term to describe the network security measures businesses take to ensure safe web use when connected to an internal network. This helps prevent web-based threats from using browsers as access points to get into the network.

**Wireless security:** Generally speaking, wireless networks are less secure than traditional networks. Thus, strict wireless security measures are necessary to ensure that threat actors are not gaining access.

## Discussion

What tools can be used to secure the network? Discuss

## 4.0 Self-Assessment/Exercise

Identify and explain the benefits of network security?

Answer:

The benefits of network security are the following:

### **Builds trust**

Security for large systems translates to security for everyone. Network security boosts client and consumer confidence, and it protects your business from the reputational and legal fallout of a security breach.

### **Mitigates risk**

The right network security solution will help your business stay compliant with business and government regulations, and it will minimize the business and financial impact of a breach if it does occur.

### **Protects proprietary information**

Your clients and customers rely on you to protect their sensitive information. Your business relies on that same protection, too. Network security ensures the protection of information and data shared across the network.

### **Enables a more modern workplace**

From allowing employees to work securely from any location using VPN to encouraging collaboration with secure network access, network security provides options to enable the future of work. Effective network security also provides many levels of security to scale with your growing business.

## 5.0 CONCLUSION

Network security tools and devices exist to help your organization protect not only its sensitive information, but also its overall performance, reputation and even its ability to stay in business.

## 6.0 SUMMARY

Three key focuses that should serve as a foundation of any network security strategy are protection, detection and response. Protection entails any tools or policies designed to prevent network security intrusion. *Detection* refers to the resources that allow you to analyze network traffic and quickly identify problems before they can do harm. *Response* is the ability to react to identified network security threats and resolve them as quickly as possible. The network security tools and techniques are access control, anti-malware, anomaly detection, application security, data loss prevention (DLP), endpoint security, firewalls and email security. Others are intrusion prevention

systems, network segmentation, Security information and event management (SIEM), virtual private network (VPN), web security and wireless security.

## **7.0 REFERENCES/FURTHER READING**

[Application Intelligence | Application Visibility | Gigamon](#)

## UNIT 4 CLIENT-SERVER COMPUTING

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Fundamentals of Client Server Computing
  - 3.1 Client Server Computing
  - 3.2 Characteristics of Client Server Computing
  - 3.3 Difference Between Client Server and Peer-to-Peer Computing
  - 3.4 Advantages of Client Server Computing
  - 3.5 Disadvantages of Client Server Computing
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

There are two configurations of networks: Client-Server and Peer-to-Peer networks. In client server, the client requests resources while the server serves same. In Peer-to-peer configuration, each node is free to communicate with others or not. The nodes under this configuration are not over-seen by any node or the other, they relate in a workgroup.

### 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, students will able to:

- Explain the concept of Client Server category of networks.
- Describe a client
- Identify the differences between the Client-Server and the Peer-to-peer configuration of networks

### 3.0 MAIN CONTENT

#### 3.1 Client Server Computing

In client-server computing, the client requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network, as pictured in figure 1.4.1, but sometimes they may reside in the same system.

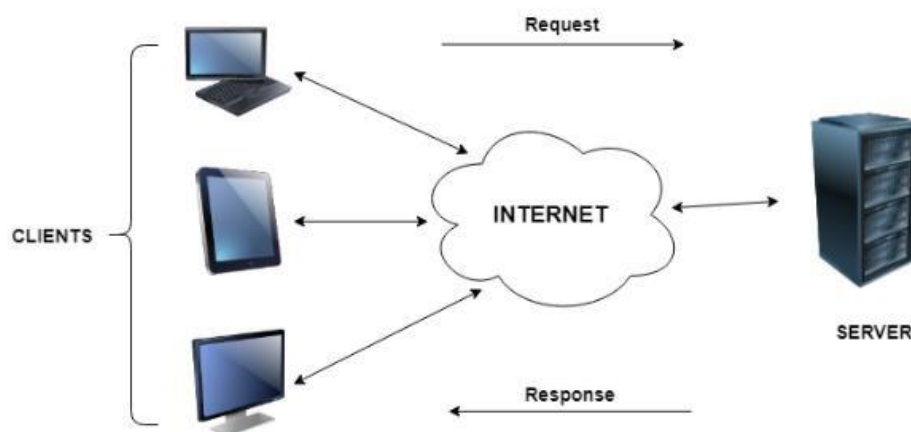


Figure 1.4.1: Client-Server Computing

### 3.2 Characteristics of Client Server Computing

The salient points for client server computing are as follows:

The client server computing works with a system of request and response. The client sends a request to the server and the server responds with the desired information.

The client and server should follow a common communication protocol so they can easily interact with each other. All the communication protocols are available at the application layer.

A server can only accommodate a limited number of client requests at a time. So it uses a system based to priority to respond to the requests.

Denial of Service (DoS) attacks hinders servers' ability to respond to authentic client requests by inundating it with false requests. An example of a client server computing system is a web server. It returns the web pages to the clients that requested them.

### 3.3 Differences between Client-Server and Peer-to-Peer Computing

The major differences between client-server computing and peer-to-peer computing are as follows:

In client server computing, a server is a central node that services many client nodes. On the other hand, in a peer-to-peer system, the nodes collectively use their resources and communicate with each other.



In client server computing, the server is the one that communicates with the other nodes. In peer-to-peer computing, all the nodes are equal and share data with each other directly. Client-Server computing is believed to be a sub-category of the peer-to-peer computing.

### **3.4 Advantages of Client-Server Computing**

All the required data is concentrated in a single place i.e. the server. So it is easy to protect the data and provide authorisation and authentication.

The server need not be located physically close to the clients yet, the data can be accessed efficiently.

It is easy to replace, upgrade or relocate the nodes in the client-server model because all the nodes are independent and request data only from the server.

All the nodes i.e clients and server may not be built on similar platforms yet, they can easily facilitate the transfer of data.

### **3.5 Disadvantages of Client Server Computing**

If all the clients simultaneously request data from the server, it may get overloaded. This may lead to congestion in the network.

If the server fails for any reason, then none of the requests of the clients can be fulfilled. This leads to failure of the client-server network.

The cost of setting and maintaining a client-server model are quite high.

### **Discussion**

What makes the Client Server configuration peculiar from the Peer-to-peer? Discuss

### **4.0 SELF-ASSESSMENT/EXERCISE**

Discuss the advantages of client Server computing

Answer:

#### **Advantages of Client-Server Computing**

All the required data is concentrated in a single place i.e. the server. So it is easy to protect the data and provide authorisation and authentication.

The server need not be located physically close to the clients yet, the data can be accessed efficiently.

It is easy to replace, upgrade or relocate the nodes in the client-server model because all the nodes are independent and request data only from the server.

All the nodes i.e clients and server may not be built on similar platforms yet, they can easily facilitate the transfer of data.

Identify the characteristics of client server computing?

Answer:

### **Characteristics of Client Server Computing**

The characteristics of the client-server computing are as follows:

The client server computing works with a system of request and response. The client sends a request to the server and the server responds with the desired information.

The client and server should follow a common communication protocol so they can easily interact with each other. All the communication protocols are available at the application layer.

A server can only accommodate a limited number of client requests at a time. So it uses a system based to priority to respond to the requests.

Denial of Service (DoS) attacks hinders servers' ability to respond to authentic client requests by inundating it with false requests.

An example of a client server computing system is a web server. It returns the web pages to the clients that requested them.

## **5.0 CONCLUSION**

Client server and peer-to-peer computing are unique one from the other and so, have their merits and demerits. The choice of either is dependent on the intention of creating your network.

## **6.0 SUMMARY**

In client server computing the server is the one that communicates with the other nodes. In peer to peer to computing, all the nodes are equal and share data with each other directly. A server can only accommodate a limited number of client requests at a time. So it uses a system based to priority to respond to the requests. The characteristics of the client-server computing are as follows: the client server computing works with a system of request and response, client and server should follow a common communication protocol so they can easily interact with each other, a server can only accommodate a limited number of client requests at a time and that Denial of Service (DoS) attacks hinders servers' ability to respond to authentic client requests by inundating it with false requests.

## 7.0 REFERENCES/FURTHER READING

Andrew S., T., & David J., W. (2011). *COMPUTER NETWORKS* (M. Horton, H. Michael, D. Tracy, & H. Melinda (eds.); fifth). Pearson Education.

Joseph, M. K. (2007). Computer Network Security and Cyber Ethics (review). In *portal: Libraries and the Academy* (fourth, Vol. 7, Issue 2). McFarland & Company, Inc. <https://doi.org/10.1353/pla.2007.0017>

Pande, J. (2017). *Introduction to Cyber Security (FCS)*. <http://uou.ac.in>

Stewart, J. M., Tittel, E., & Chapple, M. (2011). *CISSP: Certified Information Systems Security Professional Study Guide*. Wiley.

## UNIT 5 BUILDING WEB APPLICATIONS

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Fundamentals Building a Web Applications
  - 3.1 Building a Web Application
    - 3.1.1 A Web app
  - 3.2 Prerequisites for Building a Web Application
  - 3.3 Steps to Building a Web Application
    - 3.3.1 Source an Idea
    - 3.3.2 Do Market Research
    - 3.3.3 Define your Web App Functionality
    - 3.3.4 Sketch Your Web Application
    - 3.3.5 Plan Your Web App Workflow
    - 3.3.6 Wire-framing/ Prototyping Your Web Application
    - 3.3.7 Seek Early Validation
    - 3.3.8 Before Starting the Development Stage
    - 3.3.9 Architect and Build Your Database
    - 3.3.10 Build the Front End
    - 3.3.11 Build Your Back-End
    - 3.3.12 Host Your Web Application
    - 3.3.13 Deploy Your Web Application
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

#### A Web app

An interactive computer program, built with web technologies (HTML, CSS, JS), which stores (Database, Files) and manipulates data (CRUD), and is used by a team or single user to perform tasks over the internet. The HTML and the CSS serves as the front-end to receive data from the user while the database, programming like Javascript and PHP serves as the back-end.

### 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, students will able to:

Explain the concept of an App

Enumerate the prerequisite for building a web application

Describe the steps for building a web application

### 3.0 MAIN CONTENT

#### 3.1 Building a Web Application (app)-Prerequisites for Building a Web Application

To make a data-centric web app from the bottom-up, it is advantageous to understand:

Backend language (e.g. Python, Ruby) - control how your web app works

Web front end (HTML, CSS, Javascript) - for the look and feel of your web app

DevOps (Github, Jenkins) - Deploying / hosting your web app

If you do not have any experience with the points above, you need not worry. You have two options:

Learn the points above - there are lots of resources online to help you. [Codecademy](#) is recommend .

Use a web app builder like Budibase - As a builder, Budibase will remove the need to learn a backend language. On top of that, Budibase will also take care of a lot of your DevOps tasks such as hosting.

#### 3.2 Building a Web Application

##### 3.2.1 Step 1 – Source an idea

Before making a web app, you must first understand what you intend to build and more importantly, the reason for it.

**Your idea should stem from solving someone’s problem. Ideally, your own problem.**

It is important that developer choose an idea which interests him /her.

Ask yourself:

How much time do I have to build this app?

What am I interested in?

What apps do I enjoy using?

What do I like about these apps?

How much time/money will this app save or generate for me (as a user)?

How much will it improve my life

##### 3.2.2 Step 2 – Market Research

Once you have chosen your idea(s), it is important to research the market to see:

If a similar product exists

### If a market exists

The number 1 reason start-ups fail, is the failure to achieve product-market fit. “**Product/market fit** means being in a good market with a product that can satisfy that market.” To quickly find out if a similar web app exists, use the following tools to search for your idea:

Google

Patent and trademark search

Betalist

Product hunt

If a similar product exists, do not worry. This can be a sign a market for your new idea exists. Your future competitors have laid the groundwork, educated the market. It is time for you to swoop in and steal the thunder. If a similar product does not exist, it is a possibility you have struck lucky. On the other hand, it is a possibility someone before has ventured down this path and hit a dead-end.

Nobody wants to experience that, so it is important to dive deep into the market and source the wisdom of:

Your Web App’s target market - Share your web app idea on forums related to your target market. If you know anyone who works within your target market, explain your idea to them. The more you talk and receive validation from your target market, the better.

Google Trends - A quick search of your web app idea will reveal relating trends.

SEO tool – MOZ/Ahrefs is recommended. Google’s keyword planner will suffice. Write a list of keywords relating to your web app. If it is an ‘OKR tool’, use the tools to search ‘OKR tool’, ‘OKR app’, and ‘objectives and key results software’. If the SEO tool indicates there are lots of people searching for your keyword terms, this is a small indicator you have a target market.

Social Media - Jump over to Twitter/Facebook groups and present your idea to your target market.

Events - If there is a local event in your area attracting people from your target market, go to it. Share your idea and record the feedback.

After completing the above steps, you should have enough information to understand if there is a market for your product. If there is a market for your product, and there is also established competition, it is important to research them.

### 3.2.3 Step 3 - Define your web apps functionality

You have got your idea, you have validated the market, it is now time to list everything you want your app to do. A common mistake here is to get carried away. The more functionality you add, the longer it will take to build your web app. Quite often, the longer a web app takes to build, the more frustration you will experience.

Only define functionality which solves your target markets problems. Remember, your web app is a work in progress and the first goal is version

It will still have cool features and delight your users, but you must keep things simple.

For direction, I have included a list of basic functions required for a simple CRM app.

Users can create an account

Users can retrieve lost passwords

Users can change their passwords

Users can create new contacts

Users can upload new contacts

Users can assign a value to contacts

Users can write notes under contacts

Users can label a contact as a lead, customer, or associate

Users can filter contacts by lead, customer, or associate

Users can view the total value of leads, customers and associates

The above list will help you define your features. Once you are done, roll up your sleeves. It is time to get creative! **Moving from the Ideation stage, to design stage.**

### 3.2.4 Step 4 - Sketch your web app

There are multiple stages of designing a web app. The first stage is sketching using a notebook (with no lines) and pen/pencil. After step 1, 2 and 3, you should have an idea of what your web app is, who your users are, and the features it will have. Sketch out the wireframe of your web apps UI (User Interface) - it does not have to be exact - this is just a sketch. When sketching, consider the following:

Navigation

Branding

Forms

Buttons

Any other interactive elements

Sketch different versions of your web app. Consider how your web app's functionality will affect the overall design. Annotate your sketch and

outline how your app should work. Taking notes will help you clarify and understand why you have designed certain elements at a later stage. Overcomplicating the design at this stage will only lead to frustration.

### 3.2.5 Step 5 – Plan your web apps workflow

It is time to put yourself in the shoes of your user. Here, we are going to plan your web apps workflow. Now is the time to go back to step 2 and look at your market research. Take your list of competitors and sign up to their free trials. Have a quick play around with their product. Write notes on what you thought was good and what you thought was bad. Pay particular attention to the workflow.

After you have finished analysing your competitor's web apps, it is time to write down different workflows for your app. Consider the following points:

- How does a user signup
- Do they receive a verification email
- How does a user log in
- How does a user change their password
- How does a user navigate through the app
- How does a user change their user settings
- How does a user pay for the app
- How does a user cancel their subscription

All of a sudden our one-page web app turns into a 10-page web app. Write a list of all the different pages your web application will have. Consider the different states of pages. For example, the homepage will have two states; logged in and logged out. Logged in users will see a different page than logged out users.

### 3.2.6 Step 6 – Wireframing / Prototyping Your Web Application

Ok, it is time to turn those sketches and that new-found understanding of your web application into a wireframe/prototype.

Wireframing is the process of designing a blueprint of your web application while Prototyping is taking wireframing a step further, adding an interactive display.

The decision to wireframe or prototype is up to you. If you have the time, I would have recommended prototyping as it will make it easier to communicate your web app when seeking validation. You can prototype/wireframe using the following tools:

Sketch (macOS)

InVision Studio (macOs)

Adobe XD (macOS, Windows)



Figma (Web, macOS, Windows, Linux)

Balsamiq (macOS, Windows, Web)

I recommend you create a design system / style guide first. You can find inspiration at UXPin. Design systems improve design consistency. But it's not required.

### 3.2.7 Step 7 – Seek early validation

You have now got a beautiful wireframe/prototype which visually describes your web app. It is time to show your beautiful wireframe to the world. At this stage we want constructive feedback.

Simply asking your friends would they use your new web app is not enough. You should start with a small number of representative users. Go to your target market's forums, watering holes, their places of work and verify the problem with them, and present your solution. Try to build a rapport with these representatives as they could become your customers. I like to use this stage to test my sales pitch - the ultimate tokens of validation are pre-launch sales. Takes notes and document all feedback. The learning from these meetings will help direct the development of your MEP (Minimal Excellent Product).

Ok, now you have got great feedback and product validation. It is time to start building your web app.

### 3.2.8 Before Starting the development stage.

Before we make our web app, I would like to share the following tips:

Attempt to get a small section of your app fully working. What we would call a "Complete Vertical".

- Building the smallest possible section will allow you to piece all the bits together, and iron out those creases early.
- You will get great satisfaction early by having something working - great motivation.
- Create things that you know you will throw away later - if it gets you something working now.

At the start - expect things to change a lot as you learn and discover what you have not thought about.

- Have faith that your app will stabilise.
- Do not be afraid to make big changes.

Spend time learning your tools.

- You may feel like you are wasting your time, reading, or experimenting with "hello world". Learning the correct way to do things will have a huge positive, cumulative effect on your productivity over time.

- Where possible, “Go with the grain” of your tools. Realise that as soon as you step out of the normal flow / usage of your toolset, you are on your own and could be in a deep time sink. There are always exceptions to this of course!  
Do not avoid issues that need to be fixed.
  - Face your issues head on - they will never go away and will only grow in stature.
  - However, If things are still likely to change - its best to spend as little time as possible on things... It’s a tricky balance!

### 3.2.9 Step 8 – Architect and build your database

So, we know roughly our web application’s functionality, what it looks like, and the pages required. Now it is time to determine what information we will store in our database.

#### A Database

A database is simply a collection of data! Data can be stored to disk, or in memory on a server, or both. You could create a folder on your hard drive, store a few documents, and call it a database. A Database Management System (DBMS) is a system that provides you with consistent APIs to (most commonly):

- Create databases, update and delete databases

- Read and write data to databases

- Secure access to a database by providing levelled access to different areas and functions

What data you need to store and what your users need to do, will determine the type of database required to run your web app.

#### Database Types

There are many types of database for many different purposes. A web app will most commonly use one of the following:

##### SQL

You should use a SQL database if your data is very relational. Your data is relational if you have multiple, well defined record types that have relationships between them. For example, a “Customer” may have many “Invoices” stored against their record. Typically, you would create a Customer table and an Invoice table - which could be linked together by “Foreign Key” columns. E.g. Customer.Id = Invoice.CustomerId.

SQL databases have an extremely powerful query language that allows you to present your data in all sorts of useful ways. They have been around

for decades, are very well understood, and usually a safe choice. MySQL, Postgresql, Microsoft SQLServer are some of the most common - along with many more modern offerings.

The downside of SQL databases is that you must declare all your tables and columns up front. There can be a lot of overhead to manage. If you have never used one before – you are in for a pretty steep learning curve. However, there are plenty of learning resources available, and it is always a great skill to have.

### **Document Database**

You should use a document database if your data is not very relational. Document databases store “documents”. Each record in your database is simply a big blob of structured data - often in JSON format. If you need to store relationships between your records, you will have to write code to manage this yourself. However, many other aspects of using document databases are much simpler. Your database can be “schemaless” - meaning that you do not have to declare your records’ definitions up front. Generally speaking, the bar to entry to a document database is much lower. They also tend to be much more scalable than SQL databases. They usually offer some querying capabilities, although sometimes not as powerful as SQL. Examples of document databases are: MongoDB, CouchDb, Firebase (serverless), Dynamo Db (AWS).

### **Decide how to segregate your data**

Each of your clients has their own, private dataset. One of the worst things that can happen to your app is for one client’s data to be seen by another client.

Even if there is only a small amount of non-sensitive leaked data, and no damage is done, an event like this will massively erode trust in the security of your app. You must architect a solid strategy for segregating your clients’ data to make sure that this never happens. Broadly speaking, you have two options - Physical Separation and Logical Separation.

### **Physical separation**

Every one of your clients has a separate database (although could share a database server with others). This makes it much more difficult to make a mistake that leads to data leakage.

#### **Pros:**

- Most secure
- More scalable

#### **Cons:**

- Managing, maintaining and upgrading is more complex
- Query all your clients’ data together is more difficult

For example, listing all Invoices in a database will only return Invoices for one of your clients. In order to get another Client's invoices, you need to connect to another database.

Since each of your client's data is in its own database, you can easily spread them all across many database servers, without the need for "sharding". Your app will be much easier to scale this way.

The code you will need to write:

When creating a new client, you need to create a new database and populate with any seed data.

You need to keep a record somewhere of all your clients, and how to connect to each client's database.

If you need to upgrade your database (e.g. add a new table), you need to code to upgrade each separately.

If you need to query all your client's data into one, you need to pull the data out of each and aggregate it.

### **Logical separation**

All of your clients are stored in one giant database. Every time you need to get data for a single client, you must remember to include a filter for the client. E.g. 'select' from customers where customerClientId = "1234"

#### **Pros:**

Easier to get started

Easier to maintain and upgrade

Can easily query all your clients' data with one query

#### **Cons:**

Easy to make a mistake that will result in a data breach  
More difficult to scale

You now only have one database to manage. Setting this up and connecting to your database is easy. Your speed to market increases. When you need to upgrade your database, you can do so with a few clicks, or by typing a few commands. It is very easy to add new features. As you gain more users, your database will grow to millions of rows. Put some effort into how your database handles this extra volume and load. You will have to start tuning your queries.

When you're under pressure, it is so easy to forget to include that "where clientId = 1234" filter. Doing so could result in a business ending data breach.

**Ensure your database is secured.** You should look into best practices for securing your particular database. Some databases come with a default administrator login, which people often forget to change. This could leave your data open to the world.

From the start, you should create a login with “Just Enough” access. If your app only reads and writes data, then it should authenticate to your database using a login with only data reading and writing access.

### 3.2.10 Step 9 - Build the frontend

Note: In reality, you will build your backend and frontend at the same time. But for this post, we’ll keep it simple.

#### A frontend

The Frontend is the visual element of your web application. It defines what you see and interact with. The frontend is developed with HTML, CSS, and JavaScript.

If using server pages, getting started is super easy. Your backend framework is all set up and ready to start building. This is where the huge benefit lies with server pages.

With SPA, it’s a little trickier.

First, you need to set up your development environment. The components of this will be:

- A code editor, such as VS Code, Sublime Text

- A compilation, and packaging framework: Webpack

  - Gulp

  - Grunt

This is also used for serving and “Hot Loading” your application at development time, on a nodejs web server, running on localhost.

- A frontend framework (strictly not necessary, but highly advised unless you are an experienced frontend developer):

  - React

  - Ember

  - Vue

  - Svelte

  - The list is endless!

Configuring your packaging tool to talk to your backend - which is most likely running on a different port on localhost. Usually, this is done using Node’s HTTP proxy. Most packaging solutions have this option built-in, or available as plugins. This point commonly gets people stuck, and may need a diagram. Remember - if you write your backend API in C Sharp (for example) then at development time, you will be running it on a local web server, through your code editor. i.e. your frontend and backend are running on 2 different web servers, in development. However, in production, your frontend should (probably) be running on the

SAME web server as your backend - mainly because you want them to run under the same domain. This means a few things:

At dev (development) time, your frontend should make API requests to its own (Nodejs server - e.g. Webpack dev server). This Nodejs server should then proxy all “/api” request to your backend server.

When building for production, you need to get your compiled frontend files into your backend server - so they can be served as static files. You can copy and paste the files in when you deploy, but you will want to set up some sort of script to do this.

There is always a significant time required to set up your dev (development) environment for a SPA. There are plenty of boilerplate templates out there for your frameworks of choice. However, I have never written an app that has not eventually needed some custom code on top of the boilerplate.

Still, I always choose a SPA.

The end product for a web app is a much more usable application.

When you are up and running with your dev environment, I find SPAs much more productive to work with - which is more likely to do with the capabilities of modern javascript frameworks than anything else.

Writing a SPA is really the only way to make a Progressive Web Application.

You should now have a better idea of how to setup your frontend and define the look and feel of your web app. In most cases, I build the frontend and backend together.

### 3.2.11 Step 10 - Build your backend

The backend is typically what manages your data. This refers to databases, servers, and everything the user cannot see within a web application. Building your backend is one of the toughest parts of web app development. If you feel overwhelmed, a tool like Budibase can take away many of the complexities - including the following tasks. If you feel confident, continue.

When building your web app, you need to choose between:

Server Pages (Multiple Page Application)  
and Single Page Application

“But is not this the frontend?” - I hear you say. Yes! But your choice will affect how you develop your backend.

The primary jobs of the backend will be to:

Provide HTTP endpoints for your frontend, which allow it to operate on your data. E.g. Create, Read, Update and Delete (“CRUD”) records.

Authenticate users (verify they are who they say they are: a.k.a log them in).

Authorization. When a logged in user makes a request, the backend will determine whether they are allowed (authorized) to perform the requested action.

Serve the frontend

If you have chosen Server Pages, your backend will also be generating your frontend and serving it to your user.

With a single page app, the backend will simply serve your static frontend files (i.e. your “Single Page” and its related assets).

When choosing your backend:

Go with what is familiar.

Try Budibase

Server Pages / SPA should inform your decision of framework choices within your chosen language. For example, a SPA will only require an API only framework. Server pages need their own framework.

Django

Express

o Flask

How will users authenticate?

o Username and password?

o Open ID (i.e. sign in as Google, FB, etc)

Be sure to read up on security best practices. I highly recommend: OWASP

What user levels will you create in the system?

Environments. You will usually need to create multiple environments.

For example:

Testing - for all the latest development features.

Beta - to give early releases to clients.

Production - Your live system.

### 3.2.12 Step 11 - Host your web application

Hosting involves running your web app on a particular server. When using Budibase, this step can be automated with Budibase hosting. With Budibase, you are still required to buy a domain. If you are not using Budibase to host your web application, follow these quick steps:\

Buy a domain - [Namecheap](#)  
Buy/Setup an SSL certificate - [Let's Encrypt](#)  
Choose a cloud  
    provider: [Amazon](#)  
[MS Azure](#)  
[Google Cloud Platform](#)

Lower cost: Digital Ocean / Linode - if you are happy managing your own VMs  
Zeit Now, Heroku, Firebase are interesting alternatives that aim to be faster and easier to get things done - you should read about what they offer.

Choosing one of these hosting options will almost certainly provide you with everything you need. They have ample documentation and community supports, and are generally reliable options.

### 3.2.13 Step 12 - Deploy your web app

You have sourced your idea, validated it, designed and developed your web app, and chosen your hosting provider. You are now at the last step. Well done!

The deployment step includes is how your web application gets from your source control on your computer to your cloud hosting from step 11. How does your application get from Source Control / Your computer to your cloud hosting provider?

The following development tools provide continuous integration and will help you with deploying your web app to your cloud hosting:

[GitLab](#)  
[Bitbucket](#)  
[Jenkins](#)

To start with, you can just deploy directly from your machine of course. You have made a web application. Well done. You should take some time to celebrate this achievement. You are the proud owner of a new web app.

### Discussion

How can cybercrime be mitigated? Discuss

## 4.0 SELF-ASSESSMENT/EXERCISE

Mention and explain the Database types.

There are many types of database for many different purposes. A web app will most commonly use one of the following:



## SQL

You should use a SQL database if your data is very relational. Your data is relational if you have multiple, well defined record types that have relationships between them. For example, a “Customer” may have many “Invoices” stored against their record. Typically, you would create a Customer table and an Invoice table - which could be linked together by “Foreign Key” columns. E.g. Customer.Id = Invoice.CustomerId.

SQL databases have an extremely powerful query language that allows you to present your data in all sorts of useful ways. They have been around for decades, are very well understood, and usually a safe choice. MySQL, Postgresql, Microsoft SQLServer are some of the most common - along with many more modern offerings.

The downside of SQL databases is that you must declare all your tables and columns up front. There can be a lot of overhead to manage. If you have never used one before – you are in for a pretty steep learning curve. However, there are plenty of learning resources available, and it is always a great skill to have.

## Document Database

You should use a document database if your data is not very relational. Document databases store “documents”. Each record in your database is simply a big blob of structured data - often in JSON format. If you need to store relationships between your records, you will have to write code to manage this yourself. However, many other aspects of using document databases are much simpler. Your database can be “schemaless” - meaning that you do not have to declare your records’ definitions up front.

Generally speaking, the bar to entry to a document database is much lower. They also tend to be much more scalable than SQL databases. They usually offer some querying capabilities, although sometimes not as powerful as SQL. Examples of document databases are: MongoDB, CouchDb, Firebase (serverless), Dynamo Db (AWS).

What do we mean by the backend, the types and what determine your backend choice? Explain.

The backend is typically what manages your data. This refers to databases, servers, and everything the user cannot see within a web application. Building your backend is one of the toughest parts of web app development. If you feel overwhelmed, a tool

like Budibase can take away many of the complexities - including the following tasks. If you feel confident, continue.

When building your web app, you need to choose between:  
Server Pages (Multiple Page Application)  
and Single Page Application

“But is not this the frontend?” - I hear you say. Yes! But your choice will affect how you develop your backend.

The primary jobs of the backend will be to:

Provide HTTP endpoints for your frontend, which allow it to operate on your data. E.g. Create, Read, Update and Delete (“CRUD”) records.

Authenticate users (verify they are who they say they are: a.k.a log them in).

Authorization. When a logged in user makes a request, the backend will determine whether they are allowed (authorized) to perform the requested action.

Serve the frontend

If you have chosen Server Pages, your backend will also be generating your frontend and serving it to your user.

With a single page app, the backend will simply serve your static frontend files (i.e. your “Single Page” and its related assets).

When choosing your backend:

Go with what is familiar.

○ Try Budibase

○ Server Pages / SPA should inform your decision of framework choices within your chosen language. For example, a SPA will only require an API only framework. Server pages need their own framework.

- Django
- Express
- Flask

## 5.0 CONCLUSION

There are many types of database for many different purposes. A web app will most commonly use one of SQL or Document database. You should look into best practices for securing your particular database. Some databases come with a default administrator login, which people often forget to change. This could leave your data open to the world.

From the start, you should create a login with “Just Enough” access. If your app only reads and writes data, then it should authenticate to your database using a login with only data reading and writing access.

## 6.0 SUMMARY

Building a Web Application steps include: Source an idea, Market Research, Define your web apps functionality, Sketch your web app, Plan your web apps workflow, Wireframing / Prototyping Your Web Application, Seek early validation, Architect and build your database, Build the frontend, Build your backend, Host your web application and Deploy your web app. To make a data-centric web app from the bottom-up, it is advantageous to understand: Backend language (e.g. Python, Ruby) - control how your web app works; Web front end (HTML, CSS, Javascript) - for the look and feel of your web app; and DevOps (Github, Jenkins) - Deploying / hosting your web app. There are many types of database for many different purposes but a web app will most commonly use one of SQL and Document database. The backend is typically what manages your data. This refers to databases, servers, and everything the user cannot see within a web application.

Building your backend is one of the toughest parts of web app development. If you feel overwhelmed, a tool like Budibase can take away many of the complexities - including the follow tasks.

## 7.0 REFERENCES/FURTHER READING

[The web framework for perfectionists with deadlines | Django \(djangoproject.com\)](https://www.djangoproject.com/)  
[Studio | InVision \(invisionapp.com\)](https://invisionapp.com/)

**MODULE 2: PARALLEL SYSTEMS****UNIT 1: INTRODUCTION TO PARALLEL SYSTEMS****CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Parallel Processing Systems
  - 3.2 Flynn's Classification of Parallel Systems
  - 3.3 Relevance of Flynn's Classification to Parallel Systems
  - 3.4 Parallel Computers and Applications
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

**Introduction of Module**

**Parallel computing** is a type of computation in which many calculations or processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time. There are several different forms of parallel computing: bit-level, instruction-level, data, and task parallelism. Parallelism has long been employed in high-performance computing, but has gained broader interest due to the physical constraints preventing frequency scaling. As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multi-core processors.

This module will consist of four units are follows  
Unit 1: Introduction to Parallel Systems  
Unit 2: Parallel Programming Models  
Unit 3: Message Passing Programming  
Unit 4: Dependence Analysis  
Unit 5: OpenMP Programming  
Unit 6: Evaluation of Programs

## 1.0 INTRODUCTION

Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.

## 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, students will be able to:

- Define the concept of Parallel Systems

- Explore the Parallel Systems types

- Differentiate between the two Single Instruction, Multiple Data stream (SIMD) schemes

## 3.0 MAIN CONTENT

### 3.1 Parallel Processing Systems

**Parallel Processing Systems** are designed to speed up the execution of programs by dividing the program into multiple fragments and processing these fragments simultaneously. Such systems are multiprocessor systems also known as tightly coupled systems. Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both. Parallel computing is an evolution of serial computing where the jobs are broken into discrete parts that can be executed concurrently.

Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different CPUs. Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized. Several models for connecting processors and memory modules exist, and each topology requires a different programming model. The three models that are most commonly used in building parallel computers include synchronous processors each with its own memory, asynchronous processors each with its own memory and asynchronous processors with a common, shared memory.

### 3.2 Flynn's Classification of Parallel Systems

Flynn has classified the computer systems based on parallelism in the instructions and in the data streams. These are:

#### Single Instruction, Single Data stream (SISD):

An SISD computing system is a uniprocessor machine capable of executing a single instruction, which operates on a single data stream (see Figure 2.1.1 below). In SISD, machine instructions are processed sequentially; hence computers adopting this model are popularly called *sequential computers*. Most conventional computers are built using the SISD model. 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 by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, Macintosh, and workstations.

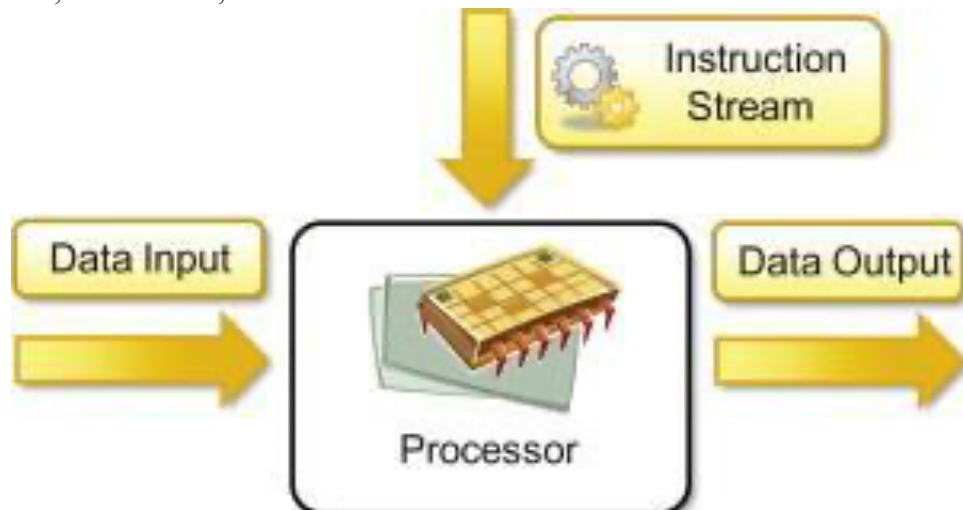


Figure 1.1.1: Single Instruction, Single Data stream (SISD)

#### Single Instruction, Multiple Data stream (SIMD):

SIMD represents single-instruction multiple-data streams. The SIMD model of parallel computing includes two parts such as a front-end computer of the usual von Neumann style, and a processor array as displayed in the figure 2.1.2.

The processor array is a collection of identical synchronized processing elements adequate for simultaneously implementing the same operation on various data. Each processor in the array has a small amount of local memory where the distributed data resides while it is being processed in parallel. The processor array is linked to the memory bus of the front end so that the front end can randomly create the local processor memories as if it were another memory.

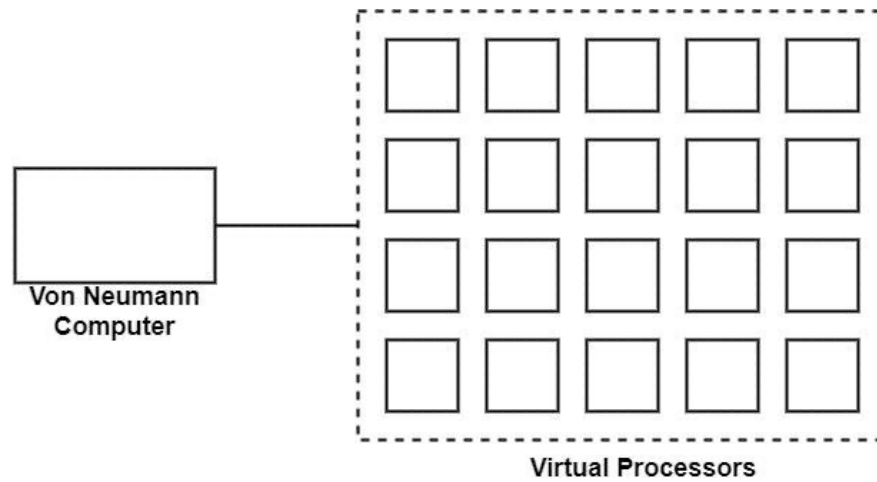


Figure 2.1.2: Single instruction stream, multiple data stream (SIMD)

A program can be developed and performed on the front end using a traditional serial programming language. The application program is performed by the front end in the usual serial method, but problem command to the processor array to carry out SIMD operations in parallel. The similarity between serial and data-parallel programming is one of the valid points of data parallelism. Synchronization is created irrelevant by the lock-step synchronization of the processors. Processors either do nothing or similar operations at the same time.

In SIMD architecture, parallelism is exploited by using simultaneous operations across huge sets of data. This paradigm is most beneficial for solving issues that have several data that require to be upgraded on a wholesale basis. It is dynamically powerful in many regular scientific calculations.

Two main configurations have been applied in SIMD machines. In the first scheme, each processor has its local memory. Processors can interact with each other through the interconnection network. If the interconnection network does not support a direct connection between given groups of processors, then this group can exchange information via an intermediate processor.

In the second SIMD scheme, processors and memory modules communicate with each other via the interconnection network. Two processors can send information between each other via intermediate memory module(s) or possibly via intermediate processor(s). The BSP (Burroughs' Scientific Processor) used the second SIMD scheme.

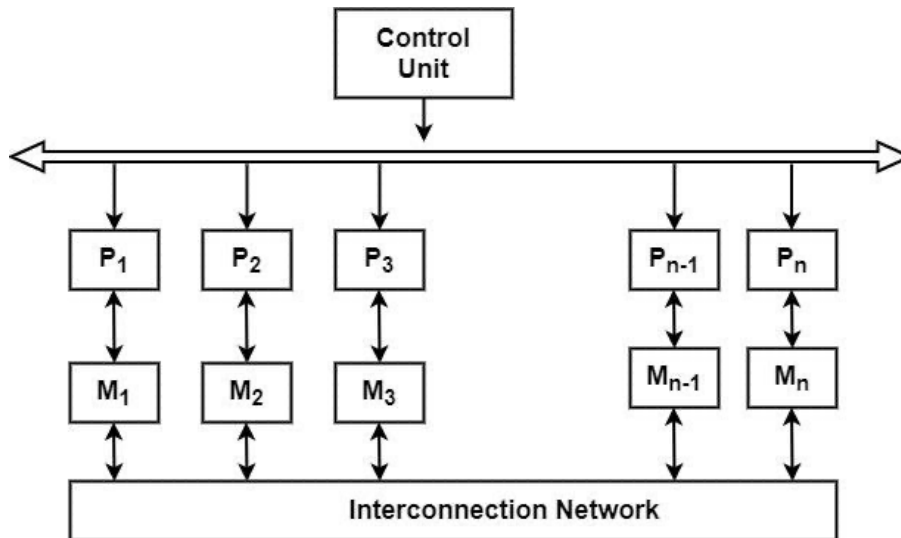
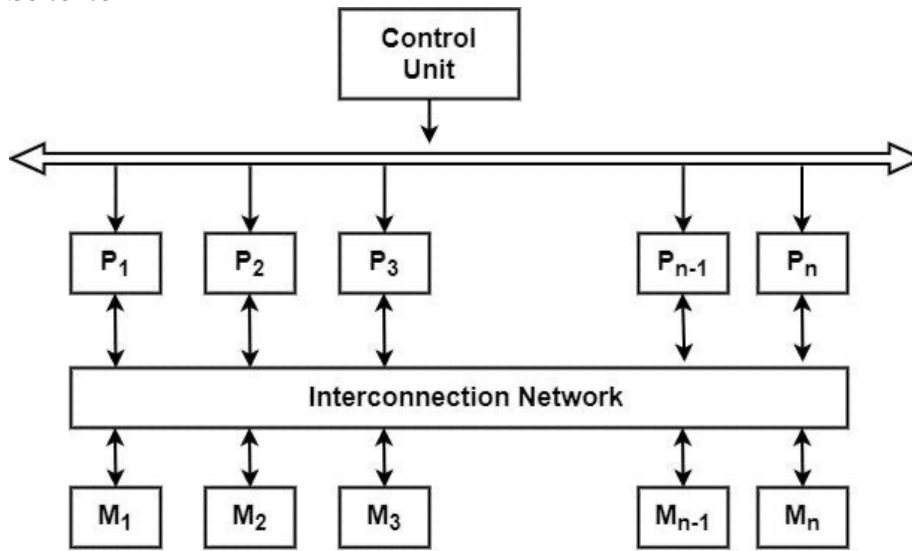


Figure 2.1.2a: Single instruction stream, multiple data stream (SIMD) Scheme-1



**Two SIMD Schemes**

Figure 2.1.2b: Single Instruction, Multiple Data stream (SIMD) Scheme-2

**Multiple Instruction, Single Data stream (MISD).**

In this association, multiple processing elements are structured under the control of multiple control units. Each control unit is handling single instruction stream and processed through its corresponding processing element. But single processing element is processing only a one data stream at a time. Hence, for handling single data stream and multiple instruction streams, multiple processing elements and multiple control units are organised in this classification. All processing elements are relate with the common shared memory for the organisation of one data stream as given in Figure 2.1.3. The only identified instance of a computer capable of MISD operation is the C.mmp built by Carnegie-Mellon University.



This type of computer organisation is denoted as:

$$I_s > 1$$

$$D_s = 1$$

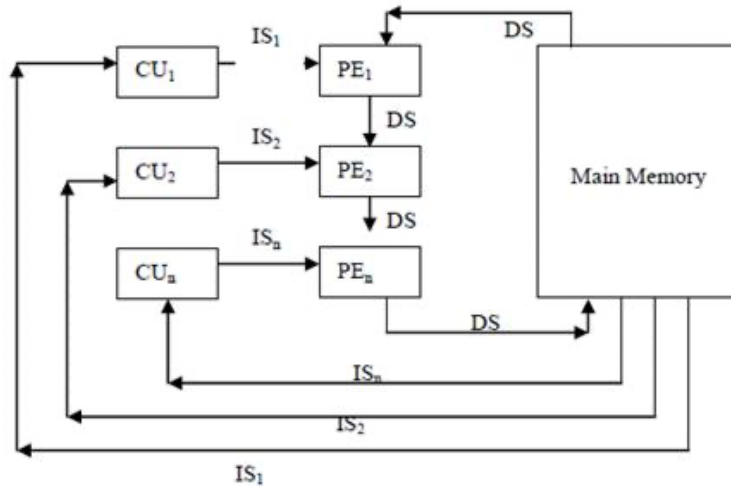


Figure 2.1.3: Multiple-Instruction Single-Data streams (MISD)

This classification is not popular in commercial machines as the thought of single data streams implementing on multiple processors is rarely functional. But for the particular applications, MISD organisation can be very useful. For example, Real time computers need to be fault tolerant where several processors implement the same data for producing the redundant data. This is also called as N- version programming. All these redundant data are measured to as results which should be similar; otherwise faulty unit is returned. Thus MISD machines can be useful to fault tolerant real time computers.

#### Multiple Instruction, Multiple Data stream (MIMD).

MIMD stands for Multiple-instruction multiple-data streams. It includes parallel architectures are made of multiple processors and multiple memory modules linked via some interconnection network. They fall into two broad types including shared memory or message passing.

A shared memory system generally accomplishes interprocessor coordination through a global memory shared by all processors. These are frequently server systems that communicate through a bus and cache memory controller.

The bus/ cache architecture alleviates the need for expensive multi-ported memories and interface circuitry as well as the need to adopt a message-passing paradigm when developing application software. Because access to shared memory is balanced, these systems are also called SMP (symmetric multiprocessor) systems. Each processor has an equal opportunity to read/write to memory, including equal access speed.

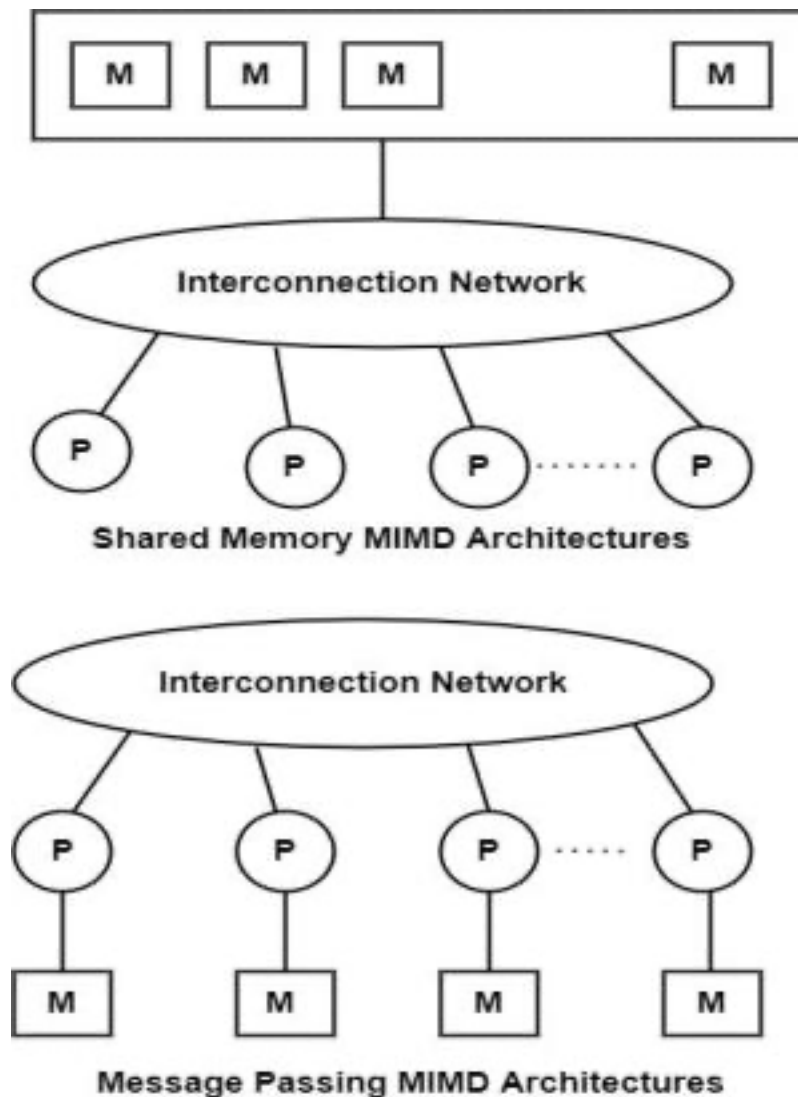


Figure 2.1.4: Multiple-Instruction Multiple-Data streams (MIMD)

The above classification of parallel computing system is focused in terms of two independent factors: the number of data streams that can be simultaneously processed, and the number of instruction streams that can be simultaneously processed. Here, by ‘instruction stream’ we mean an algorithm that instructs the computer what to do whereas ‘data stream’ (i.e. input to an algorithm) means the data that are being operated upon.

### 3.3 Relevance of Flynn’s Classification to Parallel Systems

Even though Flynn has classified the computer ‘systems into four types based on parallelism but only two of them are relevant to parallel computers. These are SIMD and MIMD computers.

SIMD computers are consisting of 'n' processing units receiving a single stream of instruction from a central control unit and each processing unit operates on a different piece of data. Most SIMD computers operate synchronously using a single global clock. The block diagram of SIMD computer is shown below:

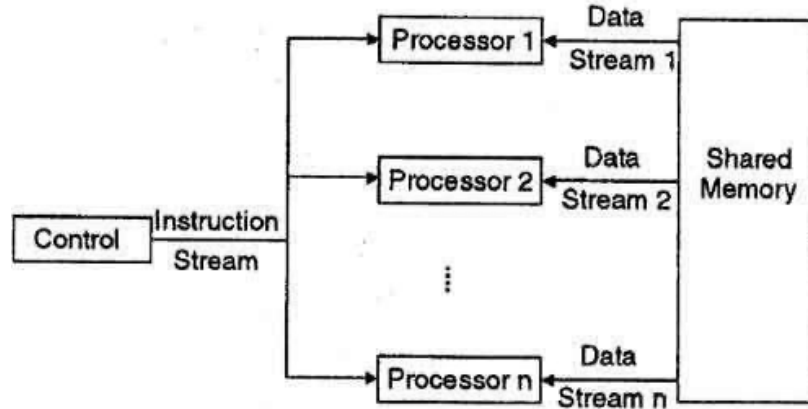


Figure 2.1.5: Single Instruction, Multiple Data stream (SIMD) Block Diagram

MIMD computers are consisting of 'n' processing units; each with its own stream of instruction and each processing unit operate on unit operates on a different piece of data. MIMD is the most powerful computer system that covers the range of multiprocessor systems. The block diagram of MIMD computer is shown.

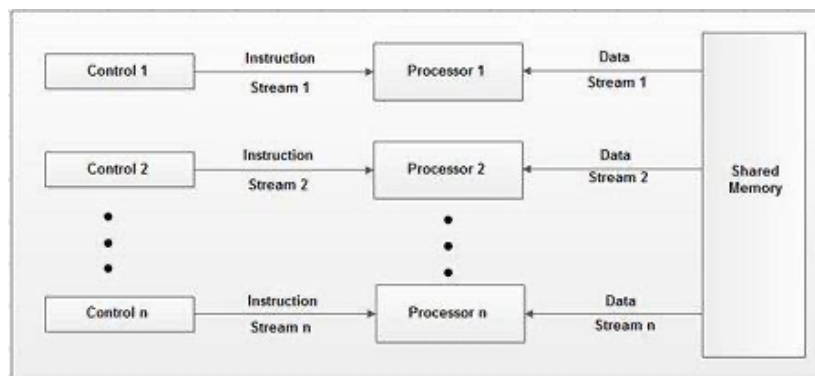


Figure 2.1.6: Multiple Instruction, Multiple Data stream (MIMD) Block Diagram

The SIMD systems are easier to program because it deals with single thread of execution. On the hand, the MIMD machines are more efficient because you can utilize the full machine power.

### 3.4 Parallel Computers and Applications

Parallel operating systems are primarily concerned with managing the resources of parallel machines. A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem. So,

a parallel computer may be a supercomputer with hundreds or thousands of processors or may be a network of workstations.

A few years ago, parallel computers could be found only in research laboratories and they were used mainly for computation intensive applications like numerical simulations of complex systems. Today, there are a lot of parallel computers available in the market; used to execute both data intensive applications in commerce and computation intensive applications in science and engineering.

Today, new applications arise and demand faster computers. Commercial applications are the most used on parallel computers. A computer that runs such an application should be able to process large amount of data in sophisticated ways. These applications include graphics, virtual reality, and decision support, parallel databases, medicine diagnosis and so on. We can say with no doubt that commercial applications will define future parallel computers architecture but scientific applications will remain important users of parallel computing technology.

Concurrency becomes a fundamental requirement for algorithms and programs. A program has to be able to use a variable number of processors and also has to be able to run on multiple processors computer architecture. According to Tanenbaum, a distributed system is a set of independent computers that appear to the user like a single one. So, the computers have to be independent and the software has to hide individual computers to the users. MIMD computers and workstations connected through LAN and WAN are examples of distributed systems. The main difference between parallel systems and distributed systems is the way in which these systems are used. A parallel system uses a set of processing units to solve a single problem A distributed system is used by many users together.

### **Discussion**

What is the difference of firewalls at Application security and internet security?

## **4.0 SELF-ASSESSMENT/EXERCISES**

### **What is Parallel Systems**

#### **Answer:**

**Parallel Processing Systems** are designed to speed up the execution of programs by dividing the program into multiple fragments and processing these fragments simultaneously. Such systems are multiprocessor systems also known as tightly coupled systems. Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers

connected by a network to form a parallel processing cluster or a combination of both. Parallel computing is an evolution of serial computing where the jobs are broken into discrete parts that can be executed concurrently.

### What are the Flynn's Classification of Parallel Systems?

#### Answer:

Single instruction stream, single data stream (SISD):

An SISD computing system is a uniprocessor machine capable of executing a single instruction, which operates on a single data stream (see Figure 2.1.1 below). In SISD, machine instructions are processed sequentially; hence computers adopting this model are popularly called *sequential computers*. Most conventional computers are built using the SISD model. 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 by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, Macintosh, and workstations.

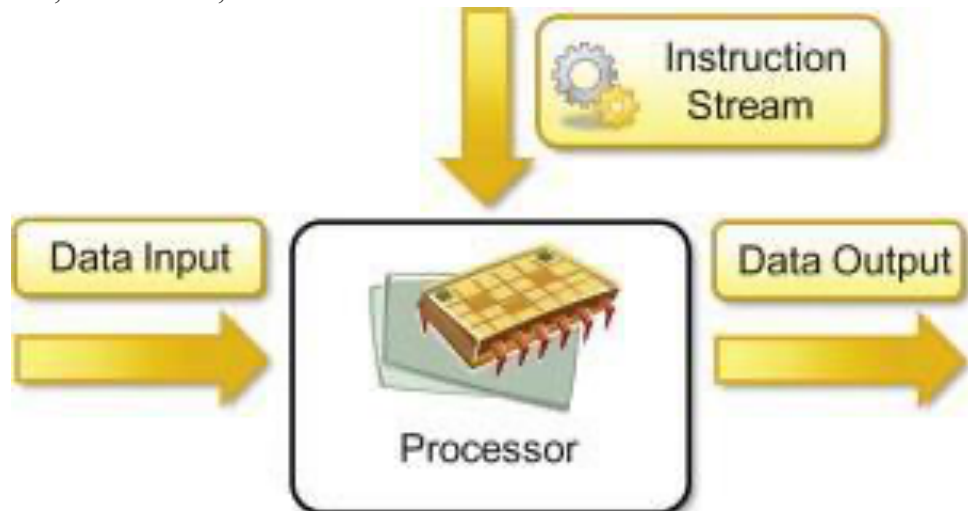


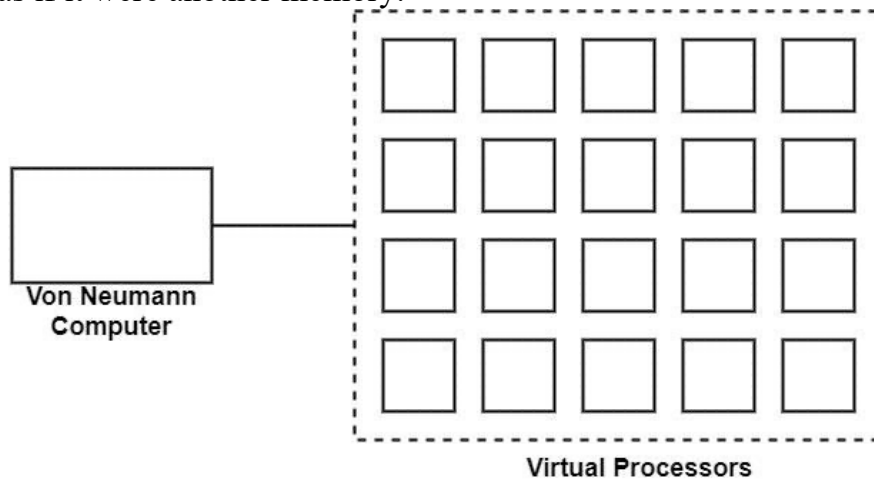
Figure 2.1.1: Single Instruction, Single Data stream (SISD)

Single Instruction, Multiple Data stream (SIMD):

SIMD represents single-instruction multiple-data streams. The SIMD model of parallel computing includes two parts such as a front-end computer of the usual von Neumann style, and a processor array as displayed in the figure 2.1.2.

The processor array is a collection of identical synchronized processing elements adequate for simultaneously implementing the same operation on various data. Each processor in the array has a small amount of local memory where the distributed data resides while it is being processed in parallel. The processor array is linked to the memory bus of the front end

so that the front end can randomly create the local processor memories as if it were another memory.



*Figure 2.1.2: Single instruction stream, multiple data stream (SIMD)*

A program can be developed and performed on the front end using a traditional serial programming language. The application program is performed by the front end in the usual serial method, but problem command to the processor array to carry out SIMD operations in parallel. The similarity between serial and data-parallel programming is one of the valid points of data parallelism. Synchronization is created irrelevant by the lock-step synchronization of the processors. Processors either do nothing or similar operations at the same time.

In SIMD architecture, parallelism is exploited by using simultaneous operations across huge sets of data. This paradigm is most beneficial for solving issues that have several data that require to be upgraded on a wholesale basis. It is dynamically powerful in many regular scientific calculations.

Two main configurations have been applied in SIMD machines. In the first scheme, each processor has its local memory. Processors can interact with each other through the interconnection network. If the interconnection network does not support a direct connection between given groups of processors, then this group can exchange information via an intermediate processor.

In the second SIMD scheme, processors and memory modules communicate with each other via the interconnection network. Two processors can send information between each other via intermediate memory module(s) or possibly via intermediate processor(s). The BSP (Burroughs' Scientific Processor) used the second SIMD scheme.

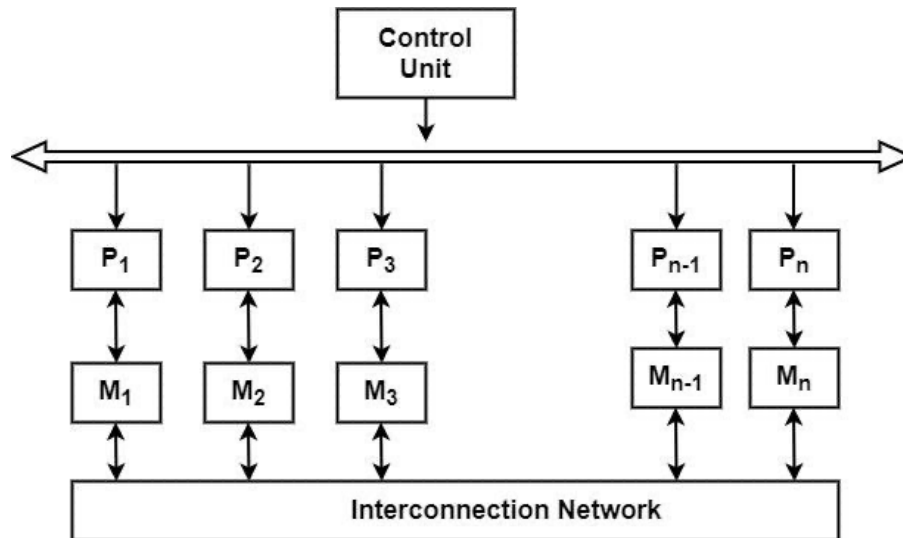


Figure 2.1.2a: Single instruction stream, multiple data stream (SIMD) Scheme-1



Two SIMD Schemes

Figure 2.1.2b: Single Instruction, Multiple Data stream (SIMD) Scheme-2

#### Multiple Instruction, Single Data stream (MISD).

In this association, multiple processing elements are structured under the control of multiple control units. Each control unit is handling single instruction stream and processed through its corresponding processing element. But single processing element is processing only a one data stream at a time. Hence, for handling single data stream and multiple instruction streams, multiple processing elements and multiple control units are organised in this classification. All processing elements are relate with the common shared memory for the organisation of one data stream as given in Figure 2.1.3. The only identified instance of a computer capable of MISD operation is the C.mmp built by Carnegie-Mellon University.

This type of computer organisation is denoted as:

$$I_s > 1$$

$$D_s = 1$$

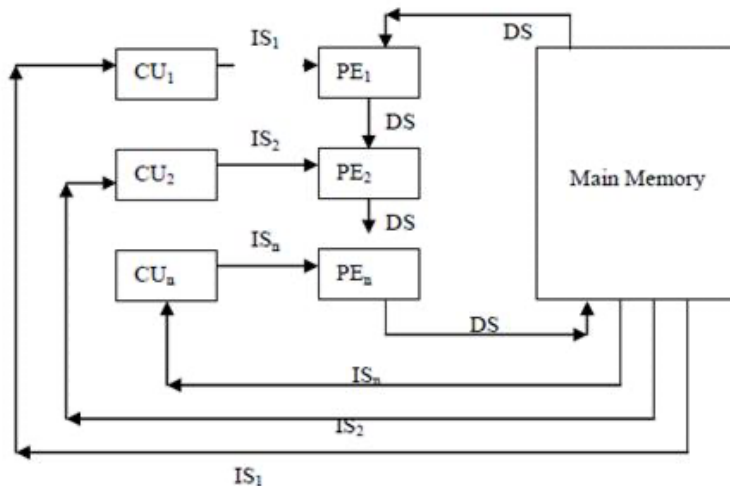


Figure 2.1.3: Multiple-Instruction Single-Data streams (MISD)

This classification is not popular in commercial machines as the thought of single data streams implementing on multiple processors is rarely functional. But for the particular applications, MISD organisation can be very useful. For example, Real time computers need to be fault tolerant where several processors implement the same data for producing the redundant data. This is also called as N- version programming. All these redundant data are measured to as results which should be similar; otherwise faulty unit is returned. Thus MISD machines can be useful to fault tolerant real time computers.

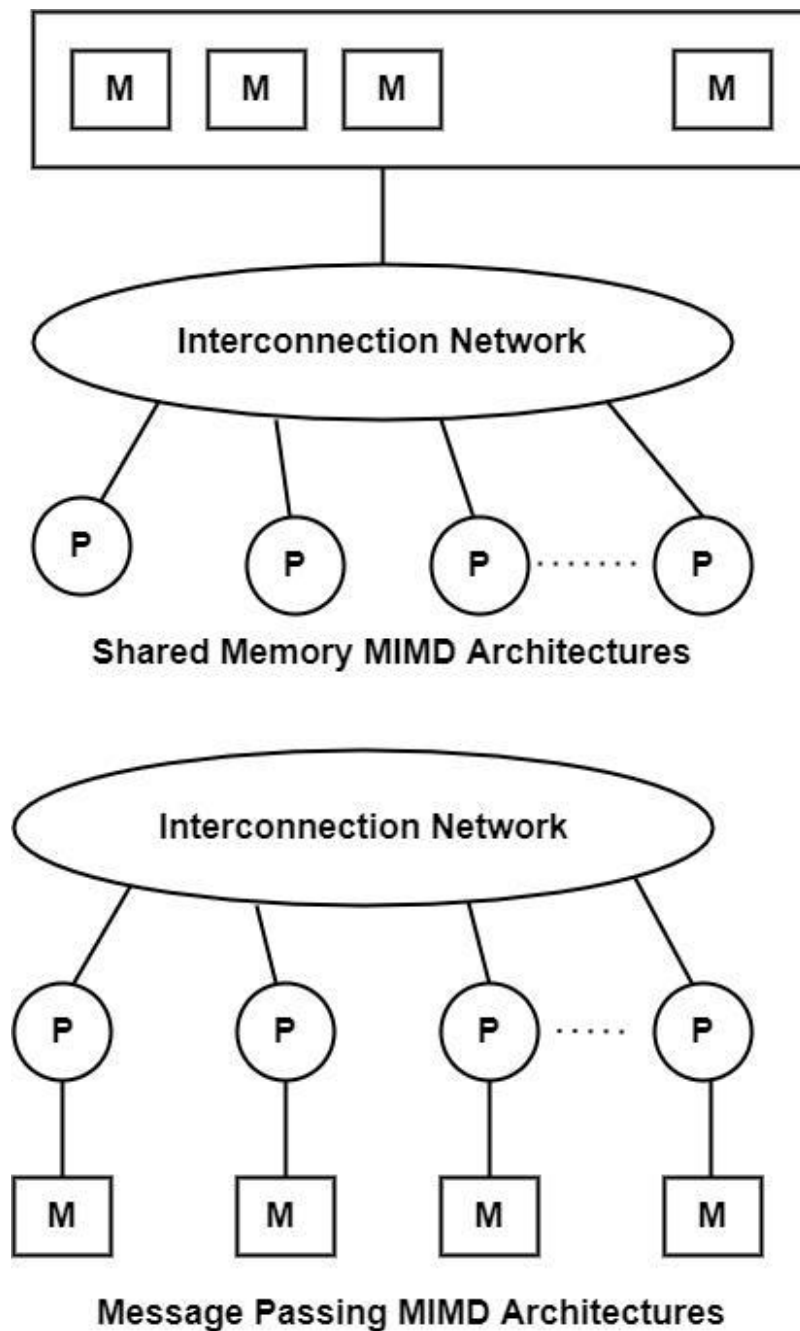
#### Multiple Instruction, Multiple Data stream (MIMD).

MIMD stands for Multiple-instruction multiple-data streams. It includes parallel architectures are made of multiple processors and multiple memory modules linked via some interconnection network. They fall into two broad types including shared memory or message passing.

A shared memory system generally accomplishes interprocessor coordination through a global memory shared by all processors. These are frequently server systems that communicate through a bus and cache memory controller.

The bus/ cache architecture alleviates the need for expensive multi-ported memories and interface circuitry as well as the need to adopt a message-passing paradigm when developing application software. Because access to shared memory is balanced, these systems are also called SMP (symmetric multiprocessor) systems. Each processor has an equal opportunity to read/write to memory, including equal access speed.





**Figure 2.1.4: Multiple-Instruction Multiple-Data streams (MIMD)**

The above classification of parallel computing system is focused in terms of two independent factors: the number of data streams that can be simultaneously processed, and the number of instruction streams that can be simultaneously processed. Here, by 'instruction stream' we mean an algorithm that instructs the computer what to do whereas 'data stream' (i.e. input to an algorithm) means the data that are being operated upon.

## 5.0 CONCLUSION

There seem to be no system that functions serially even, the human system works in parallel. For example, the respiratory system, circulatory system and locomotive system are all functioning simultaneously. The parallel computers run a number of job chunks simultaneously.

## 6.0 SUMMARY

**Parallel Processing Systems** are designed to speed up the execution of programs by dividing the program into multiple fragments and processing these fragments simultaneously. Flynn's Classification of Parallel Systems are Single Instruction, Single Data stream (SISD); Single instruction stream, multiple data stream (SIMD); Multiple-Instruction Single-Data streams (MISD) and Multiple-Instruction Multiple-Data streams (MIMD). The above classification of parallel computing system is focused in terms of two independent factors: the number of data and the number of instruction streams that can be simultaneously processed. Parallel operating systems are primarily concerned with managing the resources of parallel machines. A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem.

## 7.0 REFERENCES/FURTHER READING

<https://ecomputernotes.com/fundamental/disk-operating-system/parallel-processing-systems>  
<https://www.tutorialspoint.com/what-is-mimd-architecture>

## UNIT 2 PARALLEL PROGRAMMING MODELS

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Parallel Programming Models
  - 3.2 MPI
  - 3.3 OpenMP
  - 3.4 MapReduce
  - 3.5 OpenCL
- 3.6 CUDA
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

In computing, a **parallel programming model** is an abstraction of parallel computer architecture, with which it is convenient to express algorithms and their composition in programs. The value of a programming model can be judged on its generality: how well a range of different problems can be expressed for a variety of different architectures, and its performance: how efficiently the compiled programs can execute. The implementation of a parallel programming model can take the form of a library invoked from a sequential language, as an extension to an existing language, or as an entirely new language.

Consensus around a particular programming model is important because it leads to different parallel computers being built with support for the model, thereby facilitating portability of software. In this sense, programming models are referred to as bridging between hardware and software.

### 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, students will able to:

Explain the concept of Parallel programming

Enumerate and explain 4 of the parallel programming models

## 3.0 MAIN CONTENT

### 3.1 Parallel Programming Models

A parallel programming model is a set of program abstractions for fitting parallel activities from the application to the underlying parallel hardware. It spans over different layers: applications, programming languages, compilers, libraries, network communication, and I/O systems. Two widely known parallel programming models are:

*shared memory* and  
*message passing*

There are also different:

combinations of both.

In the shared-memory programming model, tasks share a common address space, which they read and write in an asynchronous manner. The communication between tasks is implicit. If more than one task accesses the same variable, the semaphores or locks can be used for synchronization. By keeping data local to the processor and making private copies, expensive memory accesses are avoided, but some mechanism of coherence maintenance is needed when multiple processors share the same data with the possibility of writing.

In the message-passing programming model, tasks have private memories, and they communicate explicitly via message exchange. To exchange a message, each sends operation needs to have a corresponding receive operation. Tasks are not constrained to exist on the same physical machine.

A suitable combination of two previous models is sometimes appropriate. Processors can directly access memory on another processor. This is achieved via message passing, but what the programmer actually sees is shared-memory model.

**Mainstream parallel programming** environments are based on augmenting traditional sequential programming languages with low-level parallel constructs (library function calls and/or compiler directives).

### 3.1 The Programming Models

#### 3.1.1 Message Passing Interface (MPI)

The MPI is a library of routines with the bindings in Fortran, C, and C++ and it is an example of an explicitly parallel API that implements the message-passing model via library function calls. The set of processes with separate address spaces coordinate the computation by explicitly sending and receiving messages. Each process has a separate address

space, its own program counter, and its own call stack. However, high-level constructs such as synchronization, communication, and mapping data to processes are left to a programmer to implement. MPI supports point-to-point communication between any two processes. It also enables the collective communication operations where a group of processes perform global/collective operations, such as gather, scatter, reduce, and scan.

In an heterogeneous environment, in order to optimize the performance, an MPI implementation may map processes to processors in a particular way. Similarly, an MPI implementation may optimize the way processes communicate during a global operation. For example, in case of `MPI_Reduce`, the communicating nodes do not have to form a tree structure, if an alternative structure brings better performance for the underlying parallel machine.

### 3.2.2. *OpenMP* (Open Multi-Processing)

On the other side, OpenMP is an example of mainly implicit parallel API intended for shared-memory multiprocessors. It exploits parallelism through compiler directives and the library function calls. Unlike MPI, where all threads are spawned at the beginning of the execution and are active until the program terminates, in OpenMP, a single master thread starts execution, and additional threads are active only during the execution of a parallel region. To reduce the overheads, these threads are spawned when the program enters a parallel region for the first time, and they are blocked while the program is executing a nonparallel region.

Sections work-sharing construct breaks work into multiple distinct sections, such that each section is entirely executed by a single thread. It is an example of task parallelism paradigm. Its general form is presented in figure 2.2.1.

The Sections Construct

```
#pragma omp parallel{
  #pragma omp sections{
    #pragma omp section
      block1
    #pragma omp section
      block2
  }
}
```

*Figure 2.2.1: OpenMP Session Construct*

For work-sharing construct splits iterations of a loop among different threads, such that each iteration is entirely executed by a single thread. It is an example of data-parallelism paradigm. Its general form is shown in figure 2.2.2.

The For Construct

```
#pragma omp parallel for
for(i=0; i<n; i++)
  a[i] = b[i] + c[i];
```

*Figure 2.2.2: OpenMP For Construct*

Cilk is a language extension for C programming language with parallel constructs, resembling to OpenMP. Both OpenMP and Cilk can automatically choose parallelism to achieve good performance. Cilk++ brings the same for C++ language.

Nesting OpenMP is unfortunately not fully composable, which can be a serious limitation when compared with the other abstract parallel programming models. Nesting of OpenMP can create explosive numbers of threads in recursive situations, which rapidly exhaust system resources, especially stack space, and require that the program be shut down. To prevent this, the maximum number of levels of parallel nesting that will

be activated when using OpenMP is set to one by default. While this is somewhat limiting (nested parallelism as supported by TBB and Cilk Plus is incredibly useful), it avoids a generally intolerable condition. With the continued popularity of OpenMP being so strong, we can expect additional proposals to refine OpenMP into a better ability to exploit nested parallelism opportunities when they exist. Without such solutions, programs are best to avoid relying on nesting of parallelism in order to get performance if using OpenMP.

### 3.2.3. MapReduce

One of the most widely used parallel programming models today is MapReduce. MapReduce is easy both to learn and use, and is especially useful in analyzing large datasets. While it is not suitable for several classes of scientific computing operations that are better served by message-passing interface or OpenMP, such as numerical linear algebra or finite element and finite difference computations, MapReduce's utility in workflows frequently called “big data” has made it a mainstay in high performance computing. MapReduce programming model and the Hadoop open-source framework supports it.

### 3.2.4. *OpenCL* (Open Computing Language)

OpenCL has some advantages over other parallel programming models. First of all, it is the only one of the “open” standards for which there, actually, are implementations by all major vendors—unlike for OpenMP or OpenACC. The level of vendor support, however, is a different story. OpenCL is a library that can be used with any C/C++ compiler, which makes it independent of additional tools. The kernels are written separately in a C-like language and compiled at runtime for the present hardware. The kernel compiler comes with the OpenCL implementation provided by the hardware vendor. A kernel written in OpenCL will run everywhere, including conventional CPUs, Intel Xeon Phi coprocessors, GPGPUs, some FPGAs, and even mobile devices.

OpenCL programs are divided into host and kernel code. Only the latter is executed on the compute device. In the host program, kernels and memory movements are queued into command queues associated with a device. The kernel language provides features like vector types and additional memory qualifiers. A computation must be mapped to work-groups of work-items that can be executed in parallel on the compute units (CUs) and processing elements (PEs) of a compute device. A work-item is a single instance of a kernel function. For each kernel-call, an NDRange ( $n$ -dimensional range) specifies the dimension, number, and shape of the work-groups. Global synchronization during the execution of a kernel is

unavailable. Work-items inside a work-group can be synchronized. OpenCL provides a complex memory model with a relaxed consistency.

### 3.2.5. The CUDA (Compute Unified Device Architecture) programming model

The CUDA programming model is a parallel programming model that provides an abstract view of how processes can be run on underlying GPU architectures. The evolution of GPU architecture and the CUDA programming language have been quite parallel and interdependent. Although the CUDA programming model has stabilized over time, the architecture is still evolving in its capabilities and functionality. GPU architecture has also grown in terms of the number of transistors and number of computing units over years, while still supporting the CUDA programming model.

Until 2000 GPU architectures supported fixed pipeline functionality tightly coupled with graphics pipeline. Separate silicon real estate was dedicated to each state of the pipeline. Around 2001 programmability for 2D operations (pixel shaders) and 3D operations (vertex shaders) were introduced. Then from approximately 2006 through 2008 all these operations were combined to be executed by a shared and common computational unit using a much higher-level programmable feature. This programmability was introduced as the CUDA programming model. Since then the CUDA programming model has been used to implement many algorithms and applications other than graphics, and this explosion of use and permeability of CUDA with hitherto unknown applications has catapulted the GPU's near ubiquitous use in many domains of science and technology. Since then all the GPUs designed are CUDA-capable. It should be noted that before CUDA was released, there were attempts to create high-level languages and template libraries. But such efforts tapered down with the introduction of CUDA, and more effort was spent on refining CUDA and building libraries using its constructs.

#### Discussion

Explain the peculiarities of the CUDA programming model.

## 4.0 SELF-ASSESSMENT/EXERCISES

Mention and explain two widely known parallel programming models:

#### Answer

*shared memory* and  
*message passing*

In the shared-memory programming model, tasks share a common address space, which they read and write in an asynchronous manner. The communication between tasks is implicit. If more



than one task accesses the same variable, the semaphores or locks can be used for synchronization. By keeping data local to the processor and making private copies, expensive memory accesses are avoided, but some mechanism of coherence maintenance is needed when multiple processors share the same data with the possibility of writing.

In the message-passing programming model, tasks have private memories, and they communicate explicitly via message exchange. To exchange a message, each sends operation needs to have a corresponding receive operation. Tasks are not constrained to exist on the same physical machine.

Define the term, Parallel Programming

Answer:

A parallel programming model is a set of program abstractions for fitting parallel activities from the application to the underlying parallel hardware. It spans over different layers: applications, programming languages, compilers, libraries, network communication, and I/O systems.

## 5.0 CONCLUSION

A suitable combination of two previous parallel programming models is sometimes appropriate. Processors can directly access memory on another processor. This is achieved via message passing, but what the programmer actually sees is shared-memory model

## 6.0 SUMMARY

In computing, a **parallel programming model** is an abstraction of parallel computer architecture, with which it is convenient to express algorithms and their composition in programs. Two widely known parallel programming models are: shared memory and message passing. Some of the Parallel Programming models are Message Passing Interface(MPI), Open Multi-processing (OpenMP), MapReduce, Open Computing Language(OpenCL) and Compute Unified Device Architecture (CUDA)

## 7.0 REFERENCES/FURTHER READING

Linköping University Electronic Press  
<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-40734>  
[High-Performance Computing - an overview | ScienceDirect Topics](#)  
[https://en.wikipedia.org/wiki/Parallel\\_programming\\_model](https://en.wikipedia.org/wiki/Parallel_programming_model)

## UNIT 3 MESSAGE PASSING PROGRAMMING

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Messages and Message-Passing Programming
  - 3.2 Message-Passing Programming Model
  - 3.4 Single-Program-Multiple-Data (SPMD)
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

In computer science, **message passing** is a technique for invoking behavior (i.e., running a program) on a computer. The invoking program sends a message to a process (which may be an actor or object) and relies on that process and its supporting infrastructure to then select and run some appropriate code. Message passing differs from conventional programming where a process, subroutine, or function is directly invoked by name. Message passing is key to some models of concurrency and object-oriented programming.

Message passing is ubiquitous in modern computer software. It is used as a way for the objects that make up a program to work with each other and as a means for objects and systems running on different computers (e.g., the Internet) to interact. Message passing may be implemented by various mechanisms, including channels.

### 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, students will able to:

Define what a Message is

Explain the concept of Message Passing

Itemize and explain the different Message Passing Programming models

### 3.0 MAIN CONTENT

#### 3.1 The message-passing

A message transfer is when data moves from variables in one sub-program to variables in another sub-program. The message consists of the data being sent. The message passing system has no interest in the value of this data. It is only concerned with moving it. In general the following information has to be provided to the message passing system to specify the message transfer.

Which processor is sending the message?

Where is the data on the sending processor?

What kind of data is being sent?

How much data is there?

Which processor(s) are receiving the message?

Where should the data be left on the receiving processor?

How much data is the receiving processor prepared to accept?

In general, the sending and receiving processors will cooperate in providing this information. Some of this information provided by the sending processor will be attached to the message as it travels through the system and the message passing system may make some of this information available to the receiving processor.

As well as delivering data, the message passing system has to provide some information about progress of communications. A receiving processor will be unable to use incoming data if it is unaware of its arrival. Similarly, a sending processor may wish to find out if its message has been delivered. A message transfer therefore provides synchronisation information in addition to the data in the message.

The essence of message passing is communication and many of the important concepts can be understood by analogy with the methods that people use to communicate, phone, fax, letter, radio etc. Just as phones and radio provide different kinds of service different message passing systems can also take very different approaches. For the time being we are only interested in general concepts rather than the details of particular implementations.

#### 3.2 The message-passing programming model

The sequential paradigm for programming is a familiar one. The programmer has a simplified view of the target machine as a single processor which can access a certain amount of memory. He or she

therefore writes a single program to run on that processor. The paradigm may, in fact, be implemented in various ways, perhaps in a time-sharing environment where other processes share the processor and memory, but the programmer wants to remain above such implementation-dependent details, in the sense that the program or the underlying algorithm could in principle be ported to any sequential architecture -- that is after all the point of a paradigm.

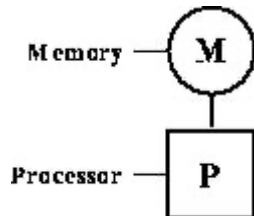


Figure 2.3.1: The sequential programming paradigm

The message-passing paradigm is a development of this idea for the purposes of parallel programming. Several instances of the sequential paradigm are considered together. That is, the programmer imagines several processors, each with its own memory space, and writes a program to run on each processor. So far, so good, Parallel programming by definition requires co-operation between the processors to solve a task, which requires some means of communication. The main point of the message-passing paradigm is that the processes communicate by sending each other messages. Thus the message-passing model has no concept of a shared memory space or of processors accessing each other's memory directly -- anything other than message-passing is out with the scope of the paradigm. As far as the programs running on the individual processors are concerned, the message passing operations are just subroutine calls. Those with experience of using networks of workstations, client-server systems or even object-oriented programs will recognise the message-passing paradigm as nothing novel.

### 3.3 Single Program Multiple Data Streams (SPMD)

SPMD mode is a method of parallel computing, its processors run the same program, but execute different data. SPMD could get better computing performance through increasing the number of processors. This also increased power consumption, and had problems of heat dissipation at high clock speeds. Previously, computing performance was increased through clock speed scaling. Parallel computing allow more instructions to complete in a given time through parallel execution. Nowadays, parallel computing has entered main stream use, following the introduction of multi-core processors.

### 3.3.1 SPMD operation mechanism

The same program code is loaded to all the processors. Data is distributed to each processor. The barrier is like a control signal generated by all processors. It could synchronize the execution of processors at some point.

#### The first example of SPMD: Titanium

Titanium is a Java-based language for writing high performance scientific applications on large scale multiprocessors.

```
public static void main(String[] args) {
System.out.println("Hello from thread " + Ti.thisProc() );
Ti.barrier() ;
if (Ti.thisProc() == 0)
System.out.println("Done."); }
```

Data locality: No communication between processors.

#### The second example of SPMD: MPI

MPI is a standard interface for message passing parallel programs written in C, C++, or Fortran.

```
begin program
x = 0z = 2b = 7
if (rank == 0) then
x = x + 1b=x* 3
send(x)
else
receive(y)
z=b* y (10)
endif
f = reduce(SUM,z)
end program
```

we can see that the variable y will be assigned the constant value 1 due to the send of x and the corresponding receive into y. SPMD has a local view of execution.

### 3.3.2 Advantages of SPMD

#### Locality

Data locality is essential to achieving good performance on large-scale machines, where communication across the network is very expensive.

### Structured Parallelism

The set of threads is fixed throughout computation. It is easier for compilers to reason about SPMD code, resulting in more efficient program analyses than in other models.

### Simple runtime implementation

SPMD belongs to MIMD, it has a local view of execution and parallelism is exposed directly to the user, compilers and runtime systems require less effort to implement than many other MIMD models.

### 3.3.3 Disadvantages of SPMD

SPMD is a flat model, which makes it difficult to write hierarchical code, such as divide-and-conquer algorithms, as well as programs optimized for hierarchical machines.

The second disadvantage may be that it seems hard to get the desired speedup using SPMD.

The advantages of SPMD are very obvious, and SPMD is still a common use on many large-scale machines. Many scientists have done researches to improve SPMD, such as the recursive SPMD, which provides hierarchical teams. So, SPMD will still be a good method for parallel computing in the future.

### Discussion

Discuss Single Program multiple Data (SPMD).

## 4.0 SELF-ASSESSMENT/EXERCISES

### What actually is the interest of a Message-passing System?

#### Answer

The message passing system has no interest in the value of this data. It is only concerned with moving it. In general the following information has to be provided to the message passing system to specify the message transfer. Which processor is sending the message:

Where is the data on the sending processor.

- What kind of data is being sent.
- How much data is there.
- Which processor(s) are receiving the message.

## 5.0 CONCLUSION

The message-passing paradigm is a development of this idea for the purposes of parallel programming. Several instances of the sequential paradigm are considered together. That is, the programmer imagines

several processors, each with its own memory space, and writes a program to run on each processor. So far, so good, but parallel programming by definition requires co-operation between the processors to solve a task, which requires some means of communication

## 6.0 SUMMARY

Message-passing paradigm involves a set of sequential programs, one for each processor. In reality, it is rare for a parallel programmer to make full use of this generality and to write a different executable for each processor. Indeed, for most problems this would be perverse -- usually a problem can naturally be divided into sub-problems each of which is solved in broadly the same way. **Single Program Multiple Data Streams (SPMD) mode** is a method of parallel computing, its processors run the same program, but execute different data. The advantages of SPMD are data locality, structured parallelism and simple runtime implementation while the disadvantages are that SPMD is a flat model, which makes it difficult to write hierarchical code and that it seems hard to get the desired speedup using SPMD.

## 7.0 REFERENCES/FURTHER READING

Neil MacDonald, Elspeth Minty, Tim Harding, Simon Brown, Edinburgh Parallel Computing Centre, The University of Edinburgh. (Course Notes)

[Advances in GPU Research and Practice | ScienceDirect  
https://slideplayer.com/slide/7559656/](https://slideplayer.com/slide/7559656/)

## UNIT 4 DEPENDENCE ANALYSIS

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Dependency Analysis
  - 3.2 How Dependencies are Found
  - 3.3 Why Do We use Dependency Analysis
  - 3.4 How Dependency Analysis Works
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

Dependencies are the relationships that exist between the constituent parts, or entities, of a complete system. A single dependency represents a directional relationship, and therefore a sequence, between a pair of system entities.

For example, if the system were a journey, there might be a walking dependency from a flight to a taxi. If this is applied to a software context, in a codebase, a function may depend on (invoke) another function. As you can infer, dependencies can have types, or behaviour, as well as a direction, indicating how the transition occurs and in what order. Dependency analysis is the process of extracting the set of entities, their dependencies and their types and direction, from the system so that the system structure can be analysed, understood and improved.

### 3.0 MAIN CONTENT

#### 3.1 Dependency analysis

When examining an artifact for re-use you might want to understand what it depends on. Developing a service that has a dependency on a large number of other distinct systems is likely to result in something that has to be revalidated every time each of those dependencies changes (which might therefore be quite often). To undertake a typical dependency analysis, perform the following steps:

- Identify the artefact with dependencies you want to analyze.

- Trace through any relationships defined on that artefact and identify the targets of the relationships. This impact analysis thus



results in a list of "dependencies" that the selected artefact depends on.

If these "dependencies" also depend on other artefacts, then the selected artefact will also have an indirect dependency. The impact analysis must therefore act recursively looking for relationships from any of the "dependencies".

This process continues until a complete graph is obtained starting at the selected artefact and finishing with artefacts that have no further dependencies. The selected artefact might have a dependency on any artefact in the graph. Kindly note that an object can exist multiple times in the graph if it can be reached in different ways.

### 3.2 How dependencies are found

When impact analysis is started, it does not change the direction of processing through the graph. For example, an object, B, has a dependency on object C, and object B is depended on by objects A and D, as shown in [Figure 1](#) below.

If object A is selected for analysis, the results list includes object B and object C. Despite object D also having a dependency on objects B and C, the analysis keeps tracing down through the dependencies and will not find any objects which are backwards in the dependency hierarchy that are not directly linked to the selected object, so object D will not be in the results list.

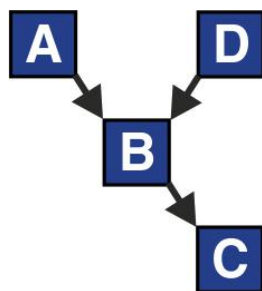


Figure 2.4.1. Dependency analysis graph

### 3.3 Why use dependency analysis

A codebase may have been initially designed with a modular or layered architecture, but over time, that clean structure could have been eroded, introducing unexpected coupling between elements, which can damage or even destroy the architectural intent. This can lead to ongoing development or maintenance of the codebase becoming more complex and time consuming to do, which can result in additional errors and complications creeping into the code. The longer these problems go unchecked, the more the problems can be compounded into the codebase

and therefore more expensive in time and money to address. Beyond dependency analysis tools providing a way for you to address such problems, they may also offer the ability to enforce structural rules dictating how the different modules/layers or external entities such as third party libraries may interact with and within the system respectively. Once in place these rules typically allow the codebase to be automatically checked for adherence and where violations are detected, notifications of said problems can be immediately brought to the attention of team members so the problems can be more quickly identified and addressed before the issues become too embedded.

### **3.3.1 Improve Refactoring**

Software architecture evaluation and refactoring should be a standard activity in any development process because it is a way to reduce risk. It is relatively inexpensive, and it pays for itself in the reduction of costly errors or schedule delays. As software teams grow and/or become more distributed, understanding software architecture becomes even more vital. Dependency analysis allows everyone on the team to have a clear understanding of the architecture (what components depend on other components, etc.).

### **3.3.2 Reduce Technical Debt**

Technical debt is a common concept in modern software development practices that happens when there is no core set of well-defined and enforceable design rules for a code base combined with a culture that does not value creating technical wealth. By establishing and iteratively improving design rules, using dependency analysis, technical debt is paid down and the code base is easier to understand and maintain. Development accelerates and this establishes technical wealth.

### **3.3.3 Understand Impact of Change**

Automatic impact analysis, a feature of dependency analysis, will highlight which parts of the application will be affected by planned codebase changes, such as replacement of modules or third party libraries. Understanding the impact of changes enables teams to quickly and accurately respond to change requests. With impact analysis, teams can be responsive while maintaining control over scope and customer expectations. Impact analysis helps developers calculate the impact of change.

### 3.4 How dependency analysis works

From the perspective of software dependency analysis, there must be an initial parsing phase to gather all the data. There can be different qualities of parsing depending on the approach taken. Some approaches rely on a scan of directories containing the codebase in question whilst other mechanisms exist where the codebase is compiled, resulting in a more complete and precise picture. Once the “database” of information exists, it is loaded into some tool that provides a suitable way of visualising the structure of the codebase, and functionality to assist the user in understanding and improving the architecture.

#### Extract

Dependencies can be extracted using a variety of tools and techniques depending on the language or software that is being analysed. This can be as simple as scanning directories for languages such as C# or Java, to using static analysis tools for languages such as C and C++.

#### Import

Most enterprise class dependency analysis tools have more than one import mechanism for dependencies. This will generally include a number of different supported languages, possibly with the ability for the definition of custom import processes.

#### Interact

Once you have the data imported dependency analysis tools allow you to interact with the dependencies to visualise the implementation against a ‘reference’ architecture. Features such as impact analysis and change logs can be used to help ensure there is no ‘architecture creep’ in the implementation.

#### Discussion

What is dependency Analysis and how are dependencies found?

### 4.0 SELF-ASSESSMENT EXERCISE

#### How dependency analysis works?

From the perspective of software dependency analysis, there must be an initial parsing phase to gather all the data. There can be different qualities of parsing depending on the approach taken. Some approaches rely on a scan of directories containing the codebase in question whilst other mechanisms exist where the codebase is compiled, resulting in a more complete and precise picture. Once the “database” of information exists, it is loaded into some tool that provides a suitable way of visualising the structure of the codebase, and functionality to assist the user in understanding and improving the architecture.

### Extract

Dependencies can be extracted using a variety of tools and techniques depending on the language or software that is being analysed. This can be as simple as scanning directories for languages such as C# or Java, to using static analysis tools for languages such as C and C++.

### Import

Most enterprise class dependency analysis tools have more than one import mechanism for dependencies. This will generally include a number of different supported languages, possibly with the ability for the definition of custom import processes.

### Interact

Once you have the data imported dependency analysis tools allow you to interact with the dependencies to visualise the implementation against a 'reference' architecture. Features such as impact analysis and change logs can be used to help ensure there is no 'architecture creep' in the implementation.

## 5.0 CONCLUSION

Computer systems face a number of security threats. One of the basic threats is data loss, which means that parts of a database can no longer be retrieved. This could be the result of physical damage to the storage medium (like fire or water damage), human error or hardware failures. Another security threat is unauthorized access. Many computer systems contain sensitive information, and it could be very harmful if it were to fall in the wrong hands. Imagine someone getting a hold of your social security number, date of birth, address and bank information. Getting unauthorized access to computer systems is known as cracking.

## 6.0 SUMMARY

To undertake a typical dependency analysis, perform the following steps:

- Identify the artefact with dependencies you want to analyze.

- Trace through any relationships defined on that artefact and identify the targets of the relationships. This impact analysis thus results in a list of "dependencies" that the selected artefact depends on.

- If these "dependencies" also depend on other artefacts, then the selected artefact will also have an indirect dependency. The impact analysis must therefore act recursively looking for relationships from any of the "dependencies".

## 7.0 REFERENCES/FURTHER READING

[WebSphere Service Registry and Repository /8.5.6/](#)

## UNIT 5 OPENMP PROGRAMMING

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Introduction to Open Specification for Multi-Processing (OpenMP)
  - 3.2 Brief History to OpenMP
  - 3.3 A Thread
  - 3.4 A Process
  - 3.5 Differences between Threads and Processes
  - 3.6 OpenMP Programming Model
    - 3.6.1 Explicit Parallelism
    - 3.6.2 Compiler Directive Based
    - 3.6.3 Fork-Join Parallelism
    - 3.6.4 Join
  - 3.7 A Program
  - 3.8 OpenMP/ Hello World
    - 3.8.1 Steps to Create a Parallel Program
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

OpenMP is a library for parallel programming in the SMP (symmetric multi-processors, or shared-memory processors) model. When programming with OpenMP, all threads share memory and data. OpenMP supports C, C++ and Fortran. The OpenMP functions are included in a header file called `omp.h`.

**OpenMP program structure:** An OpenMP program has sections that are sequential and sections that are parallel. In general an OpenMP program starts with a sequential section in which it sets up the environment and initializes the variables.

When run, an OpenMP program will use one thread (in the sequential sections), and several threads (in the parallel sections).

There is one thread that runs from the beginning to the end, and it's called the *master thread*. The parallel sections of the program will cause additional threads to fork. These are called the *slave threads*.

A section of code that is to be executed in parallel is marked by a special directive (`omp pragma`). When the execution reaches a parallel section

(marked by `omp pragma`), this directive will cause slave threads to form. Each thread executes the parallel section of the code independently. When a thread finishes, it joins the master. When all threads finish, the master continues with code following the parallel section.

Each thread has an ID attached to it that can be obtained using a runtime library function (called `omp_get_thread_num()`). The ID of the master thread is 0.

## 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, students will be able to:

Explain the concept of OpenMP programming

Define the concept of thread, process and differentiate between threads and processes

Identify and explain the OpenMP programming models

## 3.0 MAIN CONTENT

### 3.1 Introduction to Open Specification for Multi-Processing (OpenMP)

**Open MP** means Open specifications for MultiProcessing via collaborative work between interested parties from the hardware and software industry, government and academia. It is an Application Program Interface (API) that is used to explicitly direct multi-threaded, shared memory parallelism. API components include Compiler directives, Runtime library routines and Environment variables. Portable because API is specified for C/C++ and Fortran & Implementations on almost all platforms including Unix/Linux and Windows. Standardization is ensured by Jointly defined and endorsed by major computer hardware and software vendors and it is possible to become ANSI standard.

### 3.2 Brief History of OpenMP

In 1991, Parallel Computing Forum (PCF) group invented a set of directives for specifying loop parallelism in Fortran programs. **X3H5**, an ANSI subcommittee developed an ANSI standard based on PCF. In 1997, the first version of OpenMP for Fortran was defined by OpenMP Architecture Review Board. Binding for C/C++ was introduced later. Version 3.1 of it was available since 2011.

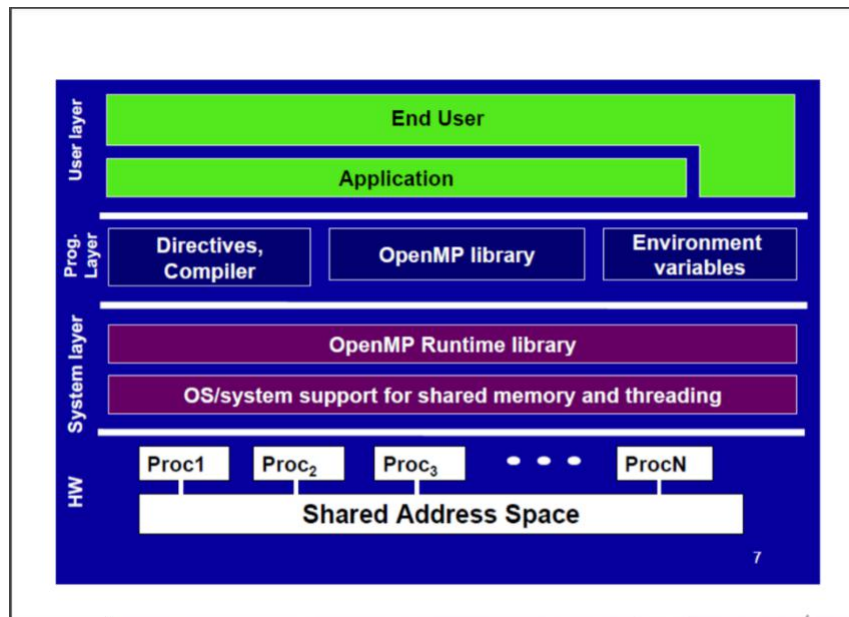


Figure 3: Open Specification for MultiProcessing (OpenMP)

### 3.3 Thread

A process is an instance of a computer program that is being executed. It contains the program code and its current activity. A thread of execution is the smallest unit of a process that can be scheduled by an operating system. Thread model is an extension of the process model where each process consists of multiple independent instruction streams (or threads) that are assigned computer resources by some scheduling procedures. Threads of a process share the address space of this process. Global variables and all dynamically allocated data objects are accessible by all threads of a process. Each thread has its own run-time stack, register, program counter. Threads can communicate by reading/writing variables in the common address space.

### 3.4 A Process

A process contains all the information needed to execute the program.

Process ID

Program code

    Data on run time stack

    Global data

Data on heap

Each process has its own address space. In multitasking, processes are given time slices in a round robin fashion. If computer resources are assigned to another process, the status of the present process has to be saved, in order that the execution of the suspended process can be resumed at a later time.

### 3.5 Differences between threads and processes

A thread is contained inside a process. Multiple threads can exist within the same process and share resources such as memory. The threads of a process share the latter's instructions (code) and its context (values that its variables reference at any given moment). Different processes do not share these resources.

### 3.6 OpenMP Programming Model

OpenMP is based on the existence of multiple threads in the shared memory programming paradigm. A shared memory process consists of multiple threads.

#### 3.6.1 Explicit Parallelism

In Explicit Parallelism, a Programmer has full control over parallelization. OpenMP is not an automatic parallel programming model.

#### 3.6.2 Compiler Directive Based

Most OpenMP parallelism is specified through the use of compiler directives which are embedded in the source code. OpenMP is not necessarily implemented identically by all vendors. It is meant for distributed-memory parallel systems (it is designed for shared address spaced machines) but guaranteed to make the most efficient use of shared memory. Required to check for data dependencies, data conflicts, race conditions, or deadlocks. Required to check for code sequences, meant to cover compiler-generated automatic parallelization and directives to the compiler to assist such parallelization. Designed to guarantee that input or output to the same file is synchronous when executed in parallel.

#### 3.6.3 Fork-Join Parallelism

OpenMP program begin as a single process: the master thread. The master thread executes sequentially until the first parallel region construct is encountered. When a parallel region is encountered, master thread create a group of threads by FORK and becomes the master of this group of threads, and is assigned the thread id 0 within the group. The statement in the program that are enclosed by the parallel region construct are then executed in parallel among these threads.

#### Discussion

Is there a difference among the OpenMP programming models, discuss.



## 4.0 SELF-ASSESSMENT EXERCISES

Enumerate and explain the OpenMP programming models

### OpenMP Programming Model

OpenMP is based on the existence of multiple threads in the shared memory programming paradigm. A shared memory process consists of multiple threads.

### Explicit Parallelism

In Explicit Parallelism, a Programmer has full control over parallelization. OpenMP is not an automatic parallel programming model.

### Compiler Directive Based

Most OpenMP parallelism is specified through the use of compiler directives which are embedded in the source code. OpenMP is not necessarily implemented identically by all vendors. It is meant for distributed-memory parallel systems (it is designed for shared address spaced machines) but guaranteed to make the most efficient use of shared memory. Required to check for data dependencies, data conflicts, race conditions, or deadlocks. Required to check for code sequences, meant to cover compiler-generated automatic parallelization and directives to the compiler to assist such parallelization. Designed to guarantee that input or output to the same file is synchronous when executed in parallel.

### Fork-Join Parallelism.

OpenMP program begin as a single process: the master thread. The master thread executes sequentially until the first parallel region construct is encountered. When a parallel region is encountered, master thread create

group of threads by FORK and becomes the master of this group of threads, and is assigned the thread id 0 within the group. The statement in the program that are enclosed by the parallel region construct are then executed in parallel among these threads.

Explain the concept of a thread, process and explain the differences between the two

### Thread

A process is an instance of a computer program that is being executed. It contains the program code and its current activity. A thread of execution is the smallest unit of a process that can be scheduled by an operating system. Thread model is an extension of the process model where each process consists of multiple independent instruction streams (or threads) that are assigned computer resources by some scheduling procedures. Threads of a process share the address space of this process. Global variables and all dynamically allocated data objects are accessible by all

threads of a process. Each thread has its own run-time stack, register, program counter. Threads can communicate by reading/writing variables in the common address space.

### **A Process**

A process contains all the information needed to execute the program.

Process ID

Program code

Data on run time stack

Global data

Data on heap

Each process has its own address space. In multitasking, processes are given time slices in a round robin fashion. If computer resources are assigned to another process, the status of the present process has to be saved, in order that the execution of the suspended process can be resumed at a later time.

### **Differences between threads and processes**

A thread is contained inside a process. Multiple threads can exist within the same process and share resources such as memory. The threads of a process share the latter's instructions (code) and its context (values that its variables reference at any given moment). Different processes do not share these resources.

## **5.0 CONCLUSION**

More efficient, and lower-level parallel code is possible, however OpenMP hides the low-level details and allows the programmer to describe the parallel code with high-level constructs, which is as simple as it can get.

OpenMP has directives that allow the programmer to:

specify the parallel region

specify whether the variables in the parallel section are private or shared

specify how/if the threads are synchronized

specify how to parallelize loops

specify how the work is divided between threads (scheduling)

## **6.0 SUMMARY**

Open MP means Open specifications for MultiProcessing via collaborative work between interested parties from the hardware and software industry, government and academia. It is an Application

Program Interface (API) that is used to explicitly direct multi-threaded, shared memory parallelism. A thread of execution is the smallest unit of a process that can be scheduled by an operating system. Thread model is an extension of the process model where each process consists of multiple independent instruction streams (or threads) that are assigned computer resources by some scheduling procedures. Threads of a process share the address space of this process. Global variables and all dynamically allocated data objects are accessible by all threads of a process. Each thread has its own run-time stack, register, program counter. Threads can communicate by reading/writing variables in the common address space. A process contains all the information needed to execute the program vis-à-vis the Process ID, Program code, Data on run time stack, Global data and Data on heap.

## 7.0 REFERENCES/FURTHER READING

[OpenMP | Introduction with Installation Guide](#)

In C/C++/Fortran, [parallel programming](#) can be achieved using [OpenMP](#).

[http://en.wikipedia.org/wiki/Process\\_\(computing\)](http://en.wikipedia.org/wiki/Process_(computing))

## UNIT 6 EVALUATION OF PROGRAMS

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Program Evaluation
  - 3.2 Definition of Program Evaluation
    - 3.2.1 Purposes for Program Evaluation
  - 3.3 Barriers
    - 3.3.1 Overcoming Barriers
  - 3.4 Types of Evaluations
    - 3.4.1 Current Evaluation
    - 3.4.2 Formative Evaluation
    - 3.4.3 Process Evaluation
    - 3.4.4 Impact Evaluation
    - 3.4.5 Outcome Evaluation
  - 3.5 Performance or Program Monitoring
  - 3.6 Evaluation Standards and Designs
  - 3.7 Logic Models
  - 3.8 Communicating Evaluation Findings
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

Program evaluation is a systematic method for collecting, analyzing, and using information to answer questions about projects, policies and programs, particularly about their effectiveness and efficiency. In both the public and private sectors, stakeholders often want to know whether the programs they are funding, implementing, voting for, receiving or objecting to are producing the intended effect. While program evaluation first focuses around this definition, important considerations often include how much the program costs per participant, how the program could be improved, whether the program is worthwhile, whether there are better alternatives, if there are unintended outcomes, and whether the program goals are appropriate and useful. Evaluators help to answer these questions, but the best way to answer the questions is for the evaluation to be a joint project between evaluators and stakeholders.

## **2.0 INTENDED LEARNING OUTCOMES (ILOS)**

At the end of this unit, students will be able to:

Define the term, Program Evaluation

identify and explain 5 types of Program evaluation

Identify the barriers to Program evaluation and ways of overcoming such

## **3.0 MAIN CONTENT**

### **3.1 Programs Evaluation**

Evaluation is the systematic application of scientific methods to assess the design, implementation, improvement or outcomes of a program (Rossi & Freeman, 1993; Short, Hennessy, & Campbell, 1996). The term "program" may include any organized action such as media campaigns, service provision, educational services, public policies, research projects, etc. (Center for Disease Control and Prevention [CDC], 1999). The purpose of Programming Evaluation includes:

Demonstrate program effectiveness to funders

Improve the implementation and effectiveness of programs

Better manage limited resources

Document program accomplishments

Justify current program funding

Support the need for increased levels of funding

Satisfy ethical responsibility to clients to demonstrate positive and negative effects of program participation.

Document program development and activities to help ensure successful replication

### **3.3 Barriers**

Program evaluations require funding, time and technical skills: requirements that are often perceived as diverting limited program resources from clients. Program staff are often concerned that evaluation activities will inhibit timely accessibility to services or compromise the safety of clients. Evaluation can necessitate alliances between historically separate community groups (e.g. academia, advocacy groups, service providers. Mutual misperceptions regarding the goals and process of evaluation can result in adverse attitudes.

### **3.3.1 Overcoming Barriers**

Collaboration is the key to successful program evaluation. In evaluation terminology, stakeholders are defined as entities or individuals that are affected by the program and its evaluation. Involvement of these stakeholders is an integral part of program evaluation. Stakeholders include but are not limited to program staff, program clients, decision makers, and evaluators. A participatory approach to evaluation based on respect for one another's roles and equal partnership in the process overcomes barriers to a mutually beneficial evaluation. Identifying an evaluator with the necessary technical skills as well as a collaborative approach to the process is integral. Programs have several options for identifying an evaluator. Health departments, other state agencies, local universities, evaluation associations and other programs can provide recommendations. Additionally, several companies and university departments providing these services can be located on the internet. Selecting an evaluator entails finding an individual who has an understanding of the program and funding requirements for evaluations, demonstrated experience, and knowledge of the issue that the program is targeting.

## **3.4 Types of Evaluation**

Various types of evaluation can be used to assess different aspects or stages of program development. As terminology and definitions of evaluation types are not uniform, an effort has been made to briefly introduce a number of types here.

### **3.4.1 Context Evaluation**

Investigating how the program operates or will operate in a particular social, political, physical and economic environment. This type of evaluation could include a community needs or organizational assessment.

### **3.4.2 Formative Evaluation**

Assessing needs that a new program should fulfill (Short, Hennessy, & Campbell, 1996), examining the early stages of a program's development (Rossi & Freeman, 1993), or testing a program on a small scale before broad dissemination (Coyle, Boruch, & Turner, 1991). Sample question: Who is the intended audience for the program?

### **3.4.3 Process Evaluation**

Examining the implementation and operation of program components. Sample question: Was the program administered as planned?

### **3.4.4 Impact Evaluation**

Investigating the magnitude of both positive and negative changes produced by a program (Rossi & Freeman, 1993). Some evaluators limit these changes to those occurring immediately (Green & Kreuter, 1991). Sample question: Did participant knowledge change after attending the program?

### **3.4.5 Outcome Evaluation**

Assessing the short and long-term results of a program. Sample question: What are the long-term positive effects of program participation?

## **3.5 Performance or Program Monitoring**

Similar to process evaluation, differing only by providing regular updates of evaluation results to stakeholders rather than summarizing results at the evaluation's conclusion (Rossi & Freeman, 1993; Burt, Harrell, Newmark, Aron, & Jacobs, 1997).

## **3.6 Evaluation Standards and Designs**

Evaluation should be incorporated during the initial stages of program development. An initial step of the evaluation process is to describe the program in detail. This collaborative activity can create a mutual understanding of the program, the evaluation process, and program and evaluation terminology. Developing a program description also helps ensure that program activities and objectives are clearly defined and that the objectives can be measured. In general, the evaluation should be feasible, useful, culturally competent, ethical and accurate. Data should be collected over time using multiple instruments that are valid, meaning they measure what they are supposed to measure, and reliable, meaning they produce similar results consistently. The use of qualitative as well as quantitative data can provide a more comprehensive picture of the program. Evaluations of programs aimed at violence prevention should also be particularly sensitive to issues of safety and confidentiality.

Experimental designs are defined by the random assignment of individuals to a group participating in the program or to a control group not receiving the program. These ideal experimental conditions are not

always practical or ethical in "real world" constraints of program delivery. A possible solution to blending the need for a comparison group with feasibility is the quasi-experimental design in which an equivalent group (i.e. individuals receiving standard services) is compared to the group participating in the target program. However, the use of this design may introduce difficulties in attributing the causation of effects to the target program. While non-experimental designs may be easiest to implement in a program setting and provide a large quantity of data, drawing conclusions of program effects are difficult.

### **3.7 Logic Models**

Logic models are flowcharts that depict program components. These models can include any number of program elements, showing the development of a program from theory to activities and outcomes. Infrastructure, inputs, processes, and outputs are often included. The process of developing logic models can serve to clarify program elements and expectations for the stakeholders. By depicting the sequence and logic of inputs, processes and outputs, logic models can help ensure that the necessary data are collected to make credible statements of causality.

### **3.8 Communicating Evaluation Findings**

Preparation, effective communication and timeliness in order to ensure the utility of evaluation findings. Questions that should be answered at the evaluation's inception include: what will be communicated? to whom? by whom? and how? The target audience must be identified and the report written to address their needs including the use of non-technical language and a user-friendly format (National Committee for Injury Prevention and Control, 1989). Policy makers, current and potential funders, the media, current and potential clients, and members of the community at large should be considered as possible audiences. Evaluation reports describe the process as well as findings based on the data

#### **Discussion**

How do you communicate Program evaluation findings.

### **4.0 SELF-ASSESSMENT EXERCISES**

What is Logic Models?

Answer:

Logic models are flowcharts that depict program components. These models can include any number of program elements, showing the development of a program from theory to activities and outcomes. Infrastructure, inputs, processes, and outputs are often included. The



process of developing logic models can serve to clarify program elements and expectations for the stakeholders. By depicting the sequence and logic of inputs, processes and outputs, logic models can help ensure that the necessary data are collected to make credible statements of causality.

### What is Context Evaluation?

Answer:

Investigating how the program operates or will operate in a particular social, political, physical and economic environment. This type of evaluation could include a community needs or organizational assessment

## 5.0 CONCLUSION

Program evaluation is a necessity although there are certain barriers to it which could be surmounted by collaborative efforts from all the stakeholders and appropriate evaluator.

## 6.0 SUMMARY

Program evaluation is a systematic method for collecting, analyzing, and using information to answer questions about projects, policies and programs, particularly about their effectiveness and efficiency. The purpose of Programming Evaluation includes: Demonstrate program effectiveness to funders, Improve the implementation and effectiveness of programs, Better manage limited resources, Document program accomplishments, Justify current program funding, Support the need for increased levels of funding, Satisfy ethical responsibility to clients to demonstrate positive and negative effects of program participation and Document program development and activities to help ensure successful replication. Evaluation should be incorporated during the initial stages of program development. An initial step of the evaluation process is to describe the program in detail. Logic models are flowcharts that depict program components. These models can include any number of program elements, showing the development of a program from theory to activities and outcomes.

## 7.0 REFERENCES/FURTHER READING

Burt, M. R., Harrell, A. V., Newmark, L. C., Aron, L. Y., & Jacobs, L. K. (1997). *Evaluation guidebook: Projects funded by S.T.O.P. formula grants under the Violence Against Women Act*. The Urban Institute. <http://www.urban.org/crime/evalguide.html>

- Centers for Disease Control and Prevention. (1992). *Handbook for evaluating HIV education*. Division of Adolescent and School Health, Atlanta.
- CDC. Framework for program evaluation in public health. *MMWR Recommendations and Reports* 1999;48(RR11):1-40.
- Chalk, R., & King, P. A. (Eds.). (1998). *Violence in Families: Assessing prevention and treatment programs*. Washington DC: National Academy Press.
- Coyle, S. L., Boruch, R. F., & Turner, C. F. (Eds.). (1991). *Evaluating AIDS prevention programs: Expanded edition*. Washington DC: National Academy Press.
- Green, L.W., & Kreuter, M. W. (1991). *Health promotion planning: An educational and environmental approach* (2nd ed.). Mountain View, CA: Mayfield Publishing Company.
- National Committee for Injury Prevention and Control. (1989). Injury prevention: Meeting the challenge. *American Journal of Preventive Medicine*, 5(Suppl. 3).
- Rossi, P. H., & Freeman, H. E. (1993). *Evaluation: A systematic approach* (5th ed.). Newbury Park, CA: Sage Publications, Inc.
- Short, L., Hennessy, M., & Campbell, J. (1996). Tracking the work. In *Family violence: Building a coordinated community response: A guide for communities*.
- Witwer, M. (Ed.) American Medical Association. Chapter 5.  
W.K. Kellogg Foundation. W.K. Kellogg evaluation handbook.  
<http://www.wkkf.org/Publications/evalhdbk/default.htm>  
(<http://www.wkkf.org/Publications/evalhdbk/default.htm>).  
[https://en.wikipedia.org/wiki/Program\\_evaluation](https://en.wikipedia.org/wiki/Program_evaluation)

**MODULE 3: DISTRIBUTED SYSTEMS****UNIT 1: INTRODUCTION TO DISTRIBUTED SYSTEMS****CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main Content
  - 3.1 Distributed Systems
  - 3.2 How a Distributed Systems Works
  - 3.3 Key Characteristics of a Distributed Systems
  - 3.4 Distributed Tracing
  - 3.5 Benefits of Distributed Systems
  - 3.6 Challenges of Distributed Systems
  - 3.7 Risks of Distributed Systems
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

**1.0 INTRODUCTION**

Distributed systems are an important development for IT and computer science as an increasing number of related jobs are so massive and complex that it would be impossible for a single computer to handle them alone. Distributed computing offers additional advantages over traditional computing environments.

**2.0 INTENDED LEARNING OUTCOMES (ILOS)**

By the end of this unit, you will be able to:

- Explain the concept of Distributed Systems
- Describe How a Distributed Systems work
- List and explain the Key Characteristics of a Distributed Systems
- Explain the term, Distributed Tracing
- Mention 5 benefits of Distributed Systems
- Challenges of Distributed Systems
- Risks of Distributed Systems

## 3.0 MAIN CONTENT

### 3.1 Distributed Systems

A Distributed system is a computing environment in which various components are spread across multiple computers (or other computing devices) on a network. These devices split up the work, coordinating their efforts to complete the job more efficiently than if a single device had been responsible for the task. Distributed systems reduce the risks involved with having a single point of failure, bolstering reliability and fault tolerance. Modern distributed systems are generally designed to be scalable in near real-time and additional computing resources can be added on the fly to increasing performance and further reducing time to completion.

Earlier, distributed computing was expensive, complex to configure and difficult to manage

But, Software as a Service (SaaS) platforms has offered expanded functionality, distributed computing has become more streamlined and affordable for businesses, large and small, all types of computing jobs be it database management, video games or Softwares cryptocurrency systems, scientific simulations, blockchain technologies and AI platforms all use Distributed Systems platforms.

### 3.2 How a distributed system works

Distributed systems have evolved over time but today's most common implementations are largely designed to operate via the internet and, more specifically, the cloud

For Example:

A distributed system begins with a task, such as rendering a video to create a finished product ready for release.

The web application, or distributed applications, managing this task — like a video editor on a client computer:

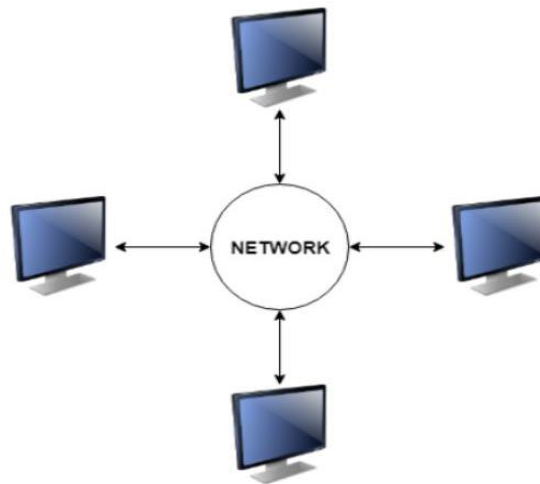
splits the job into pieces

An algorithm gives one frame of the video to each of a dozen different computers (or nodes) to complete the rendering

Once the frame is complete, the managing application gives the node a new frame to work on

This process continues until the video is finished and all the pieces are put back together

Distributed Systems turns a task that might have taken days for a single computer to complete into one that is finished in a matter of minutes.



*Figure 3.1.1: Distributed Operating Systems*

There are many models and architectures of distributed systems in use today.

Client-server systems, the most traditional and simple type of distributed system, involve a multitude of networked computers that interact with a central server for data storage, processing or other common goal. Cell phone networks are an advanced type of distributed system that share workloads among handsets, switching systems and internet-based devices. Peer-to-peer networks, in which workloads are distributed among hundreds or thousands of computers all running the same software, are another example of a distributed system architecture. The most common forms of distributed systems in the enterprise today are those that operate over the web, handing off workloads to dozens of cloud-based virtual server instances that are created as needed, then terminated when the task is complete.

### **3.3 Key Characteristics of a Distributed System**

Distributed systems are commonly defined by the following key characteristics and features:

**Scalability:** The ability to grow as the size of the workload increases is an essential feature of distributed systems, accomplished by adding additional processing units or nodes to the network as needed.

**Concurrency:** Distributed system components run simultaneously. They are also characterized by the lack of a “global clock,” when tasks occur out of sequence and at different rates.

**Availability/fault tolerance:** If one node fails, the remaining nodes can continue to operate without disrupting the overall computation.

**Transparency:** An external programmer or end user sees a distributed system as a single computational unit rather than as its underlying parts, allowing users to interact with a single logical device rather than being concerned with the system's architecture. **Heterogeneity:** In most distributed systems, the nodes and components are often asynchronous, with different hardware, middleware, software and operating systems. This allows the distributed systems to be extended with the addition of new components.

**Replication:** Distributed systems enable shared information and messaging, ensuring consistency between redundant resources, such as software or hardware components, improving fault tolerance, reliability and accessibility.

### 3.4 Distributed Tracing

Distributed tracing, sometimes called distributed request tracing, is a method for monitoring applications — typically those built on a microservices architecture — which are commonly deployed on distributed systems. Distributed tracing is essentially a form of distributed computing in that it is commonly used to monitor the operations of applications running on distributed systems.

In software development and operations, tracing is used to follow the course of a transaction as it travels through an application — an online credit card transaction as it winds its way from a customer's initial purchase to the verification and approval process to the completion of the transaction, for example. A tracing system monitors this process step by step, helping a developer to uncover bugs, bottlenecks, latency or other problems with the application.

Distributed tracing is necessary because of the considerable complexity of modern software architectures. A distributed tracing system is designed to operate on a distributed services infrastructure, where it can track multiple applications and processes simultaneously across numerous concurrent nodes and computing environments. Without distributed tracing, an application built on a microservices architecture and running on a system as large and complex as a globally distributed system environment would be impossible to monitor effectively.

### 3.5 Benefits of Distributed Systems

Distributed systems offer a number of advantages over monolithic, or single, systems, including:

**Greater flexibility:** It is easier to add computing power as the need for services grows. In most cases today, you can add servers to a distributed system on the fly.

**Reliability:** A well-designed distributed system can withstand failures in one or more of its nodes without severely impacting performance. In a monolithic system, the entire application goes down if the server goes down.

**Enhanced speed:** Heavy traffic can bog down single servers when traffic gets heavy, impacting performance for everyone. The scalability of distributed databases and other distributed systems makes them easier to maintain and also sustain high-performance levels.

**Geo-distribution:** Distributed content delivery is both intuitive for any internet user, and vital for global organizations.

### 3.6 What are some challenges of distributed systems?

Distributed systems are considerably more complex than monolithic computing environments, and raise a number of challenges around design, operations and maintenance. These include:

**Increased opportunities for failure:** The more systems added to a computing environment, the more opportunity there is for failure. If a system is not carefully designed and a single node crashes, the entire system can go down. While distributed systems are designed to be fault tolerant, that fault tolerance isn't automatic or foolproof.

**Synchronization process challenges:** Distributed systems work without a global clock, requiring careful programming to ensure that processes are properly synchronized to avoid transmission delays that result in errors and data corruption. In a complex system — such as a multiplayer video game — synchronization can be challenging, especially on a public network that carries data traffic.

**Imperfect scalability:** Doubling the number of nodes in a distributed system doesn't necessarily double performance. Architecting an effective distributed system that maximizes scalability is a complex undertaking that needs to take into account load balancing, bandwidth management and other issues.

**More complex security:** Managing a large number of nodes in a heterogeneous or globally distributed environment creates

numerous security challenges. A single weak link in a file system or larger distributed system network can expose the entire system to attack.

**Increased complexity:** Distributed systems are more complex to design, manage and understand than traditional computing environments.

### 3.7 The risks of distributed systems

The challenges of distributed systems as outlined above create a number of correlating risks. These include:

**Security:** Distributed systems are as vulnerable to attack as any other system, but their distributed nature creates a much larger attack surface that exposes organizations to threats.

**Risk of network failure:** Distributed systems are beholden to public networks in order to transmit and receive data. If one segment of the internet becomes unavailable or overloaded, distributed system performance may decline.

**Governance and control issues:** Distributed systems lack the governability of monolithic, single-server-based systems, creating auditing and adherence issues around global privacy laws such as GDPR. Globally distributed environments can impose barriers to providing certain levels of assurance and impair visibility into where data resides.

**Cost control:** Unlike centralized systems, the scalability of distributed systems allows administrators to easily add additional capacity as needed, which can also increase costs. Pricing for cloud-based distributed computing systems are based on usage (such as the number of memory resources and CPU power consumed over time). If demand suddenly spikes, organizations can face a massive bill.

## 4.0 SELF-ASSESSMENT EXERCISES

### Define Distributed Systems

Answer:

A Distributed system is a computing environment in which various components are spread across multiple computers (or other computing devices) on a network. These devices split up the work, coordinating their efforts to complete the job more efficiently than if a single device had been responsible for the task



## What are Benefits of Distributed Systems

Answer:

Distributed systems offer a number of advantages over monolithic, or single, systems, including:

**Greater flexibility:** It is easier to add computing power as the need for services grows. In most cases today, you can add servers to a distributed system on the fly.

**Reliability:** A well-designed distributed system can withstand failures in one or more of its nodes without severely impacting performance. In a monolithic system, the entire application goes down if the server goes down.

**Enhanced speed:** Heavy traffic can bog down single servers when traffic gets heavy, impacting performance for everyone. The scalability of distributed databases and other distributed systems makes them easier to maintain and also sustain high-performance levels.

**Geo-distribution:** Distributed content delivery is both intuitive for any internet user, and vital for global organizations.

Identify the risks of Distributed Systems

Answer:

The challenges of distributed systems as outlined above create a number of correlating risks. These include:

**Security:** Distributed systems are as vulnerable to attack as any other system, but their distributed nature creates a much larger attack surface that exposes organizations to threats.

**Risk of network failure:** Distributed systems are beholden to public networks in order to transmit and receive data. If one segment of the internet becomes unavailable or overloaded, distributed system performance may decline.

**Governance and control issues:** Distributed systems lack the governability of monolithic, single-server-based systems, creating auditing and adherence issues around global privacy laws such as GDPR. Globally distributed environments can impose barriers to providing certain levels of assurance and impair visibility into where data resides.

**Cost control:** Unlike centralized systems, the scalability of distributed systems allows administrators to easily add additional capacity as needed, which can also increase costs. Pricing for cloud-based distributed computing systems are based on usage (such as the number of memory resources and CPU power consumed over time). If demand suddenly spikes, organizations can face a massive bill.

## 5.0 CONCLUSION

Distributed Systems is integral part of the world today because everything in the world runs on inter-networking ranging from education, e-commerce, entertainments, agriculture and even the kitchen.

## 6.0 SUMMARY

Distributed systems offer a number of advantages over monolithic, or single, systems, including Greater flexibility, Reliability, Enhanced speed and Geo-distribution. Distributed systems raise a number of challenges around design, operations and maintenance. These include Increased opportunities for failure, Synchronization process challenges, Imperfect scalability, More complex security and Increased complexity. The challenges of distributed systems as outlined above create a number of correlating risks. These include Security, Risk of network failure, Governance and control issues and Cost control.

## 7.0 REFERENCES/FURTHER READING

[https://www.splunk.com/en\\_us/data-insider/what-are-distributed-systems.html#elements-of-distributed-systems](https://www.splunk.com/en_us/data-insider/what-are-distributed-systems.html#elements-of-distributed-systems)

## UNIT 2      SYSTEMS MODELS

### CONTENTS

- 1.0 Introduction
- 1.0 Intended Learning Outcomes (ILOs)
- 3.0 Main Content
  - 3.1 A Systems
  - 3.2 A Model and Systems Models
  - 3.3 Systems Model Types
- 4.0 Self-Assessment Exercises
- 4.0 Conclusion
- 5.0 Summary
- 6.0 References/Further Reading

### 1.0 INTRODUCTION

**Systems modeling** is the interdisciplinary study of the use of models to conceptualize and construct systems in business and IT development.

A common type of systems modeling is function modeling, with specific techniques such as the Functional Flow Block Diagram and IDEF0. These models can be extended using functional decomposition, and can be linked to requirements models for further systems partition.

Contrasting the functional modeling, another type of systems modeling is architectural modeling which uses the systems architecture to conceptually model the structure, behavior, and more views of a system.

### 2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- Explain the terms, Systems and Models
- Identify the Systems' different perspective
- Describe the 5 types of UML diagrams

3.1 A **system** is a simplified representation of reality. "System" is a common word, often used with loose meaning. Whereas in the real world, a "system" may seem at times an endless series of connected elements, we refer here to a system as a series of selected, chosen elements with specified boundaries and pre-determined time characteristics.

A 'simple' system could for instance be a nearby coffee shop. This coffee shop has customers who place orders and staff who process them. There may be at times very few customers, whereas at others, the place is very busy (say, because the coffee shop is just nearby the University, and has

free wi-fi, which the students use while enjoying a coffee and chat with their friends). So, for the customers, and the staff too, time is not neutral. It is then useful to look at our coffee-shop-system over a series of sections of time (time steps) that make a day. Perhaps an appropriate time step of one hour is adequate: it is more than enough to encapsulate long hours when little really happens, but is just enough to capture events at peak time. So much, though, may happen in one hour over a cup of coffee, when the place is busy, people meet, many orders are placed, many messages received. Perhaps, a time step of 30 minutes, or even 15 minutes might then be better. So, although many near-empty 15-minute segments might be a waste of computing time, and lead to outputs that may be boring for some parts of the day, these might ensure that important events are not lost at peak time. Yet - so many things may still happen over a period of 15 minutes. Might a time step of 5 minutes be safer? This is obviously not an easy question.

At any rate, a decision must be made, and it is up to the modeler to make it. Each system, such as the coffee-shop-system, has a **time constant**, which we can simply define for the time being as the delay over which the system may strongly change, or, in systems analysis phrasing: over which the **state** of the system may change. One way to empirically choose a time constant is based on experience and knowledge of the system at hand. Note that in the coffee-shop-system, not all the elements are enclosed within the coffee shop itself, which are important for the coffee-shop-system: for instance, it has free wi-fi. We therefore can call it a semi-open system. Biological systems, phytopathological systems in particular, are **semi-open**: they receive and transmit information, components, biomass, or energy from and to their environment.

## 3.2 A model & Systems Models

**3.2.1 A model** is a computer program that describes the mechanics of the considered system. The encoding of a model can be made in many ways.

### 3.2.2 Systems Models

A system is a set of elements that relate to each other in some manner. The elements of a system can be objects, people, organizations, processes, descriptions or even ideas. The relationships between these elements can include different kinds of influence, flows of information, resources, associations, temporal relationships, or origins.

Models of systems therefore try to capture these relationships in a way that gives a perspective on how the system as a whole interacts.

**3.2.3 System modeling** is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML). Models help the analyst to understand the functionality of the system; they are used to communicate with customers.

#### **3.2.4 Models' Different perspectives:**

An **external** perspective, where you model the context or environment of the system.

An **interaction** perspective, where you model the interactions between a system and its environment, or between the components of a system.

A **structural** perspective, where you model the organization of a system or the structure of the data that is processed by the system.

A **behavioral** perspective, where you model the dynamic behavior of the system and how it responds to events.

#### **3.2.5 UML diagrams**

Five types of UML diagrams that are the most useful for system modeling:

**Activity** diagrams, which show the activities involved in a process or in data processing.

**Use case** diagrams, which show the interactions between a system and its environment.

**Sequence** diagrams, which show interactions between actors and the system and between system components.

**Class** diagrams, which show the object classes in the system and the associations between these classes.

**State** diagrams, which show how the system reacts to internal and external events.

Models of both new and existing system are used during **requirements engineering**. Models of the **existing systems** help clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses. These then lead to requirements for the new system. Models of the **new system** are used during requirements engineering to help explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and to document the system for implementation.

### 3.3 Systems Models Types

#### 3.3.1 Context and process models

**Context models** are used to illustrate the operational context of a system

they show what lies outside the system boundaries. Social and organizational concerns may affect the decision on where to position system boundaries. Architectural models show the system and its relationship with other systems.

**System boundaries** are established to define what is inside and what is outside the system. They show other systems that are used or depend on the system being developed. The position of the system boundary has a profound effect on the system requirements. Defining a system boundary is a political judgment since there may be pressures to develop system boundaries that increase/decrease the influence or workload of different parts of an organization.

**Context models** simply show the other systems in the environment, not how the system being developed is used in that environment. **Process models** reveal how the system being developed is used in broader business processes. UML activity diagrams may be used to define business process models.

The example below shows a UML **activity diagram** describing the process of involuntary detention and the role of MHC-PMS (mental healthcare patient management system) in it.

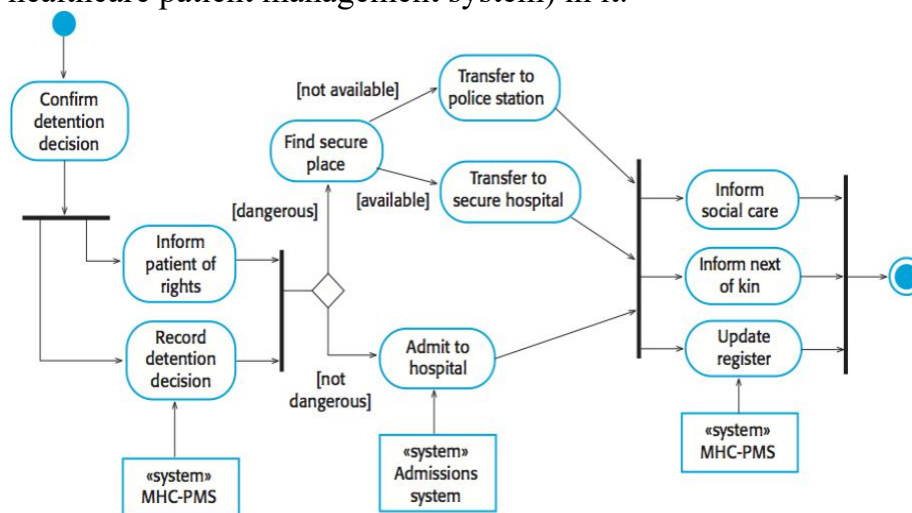


Figure 3.2.1: UML activity diagram for involuntary detention and the role of MHC-PMS

### 3.3.2 Interaction models

Types of interactions that can be represented in a model:

Modeling **user interaction** is important as it helps to identify user requirements.

Modeling **system-to-system interaction** highlights the communication problems that may arise.

Modeling **component interaction** helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.

**Use cases** were developed originally to support requirements elicitation and now incorporated into the UML, . Each use case represents a discrete task (figure 3.3.3) that involves external interaction with a system. Actors in a use case may be people or other systems. Use cases can be represented using a UML use case diagram and in a more detailed textual/tabular format.

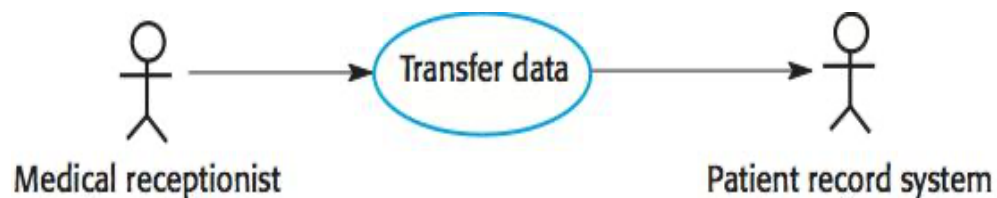


Figure 3.3.3: Simple Sample of Use-cases

Use case description in a tabular format:

Use case title	Transfer data
Description	A receptionist may transfer data from the MHC-PMS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Actor(s)	Medical receptionist, patient records system (PRS)
Preconditions	Patient data has been collected (personal information, treatment summary); The receptionist must have appropriate security permissions to access the patient information and the PRS.
Postconditions	PRS has been updated
Main success scenario	1. Receptionist selects the "Transfer data" option from the menu. 2. PRS verifies the security credentials of the receptionist.

	3. Data is transferred. 4. PRS has been updated.
Extensions	2a. The receptionist does not have the necessary security credentials. 2a.1. An error message is displayed. 2a.2. The receptionist backs out of the use case.

UML **sequence diagrams** are used to model the interactions between the actors and the objects within a system. A sequence diagram shows (figure 3.3.4) the sequence of interactions that take place during a particular use case or use case instance. The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these. Interactions between objects are indicated by annotated arrows.

Medical Receptionist

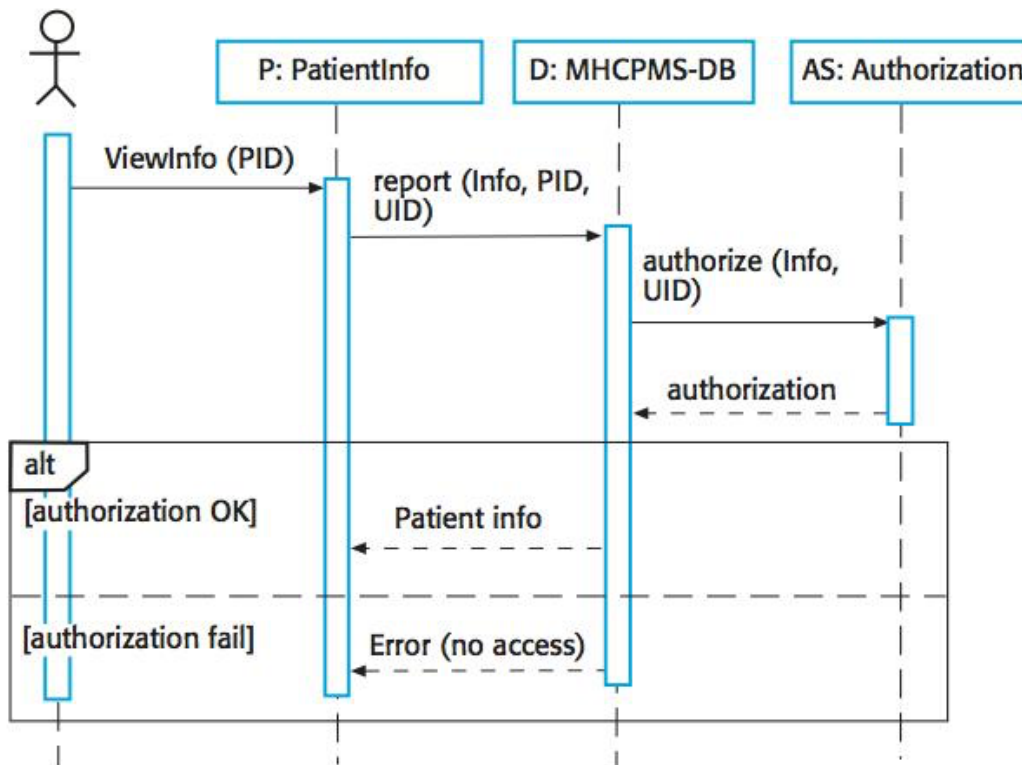


Figure 3.3.4: UML sequence diagrams

### 3.3.3 Structural models

**Structural models** of software display the organization of a system in terms of the components that make up that system and their relationships. Structural models may be **static** models, which show the structure of the system design, or **dynamic** models, which show the organization of the system when it is executing. You create structural models of a system when you are discussing and designing the system architecture.



UML **class diagrams** are used when developing an object-oriented system model to show the classes in a system and the associations between these classes as in figure 3.3.5. An object class can be thought of as a general definition of one kind of system object. An association is a link between classes that indicates that there is some relationship between these classes. When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.

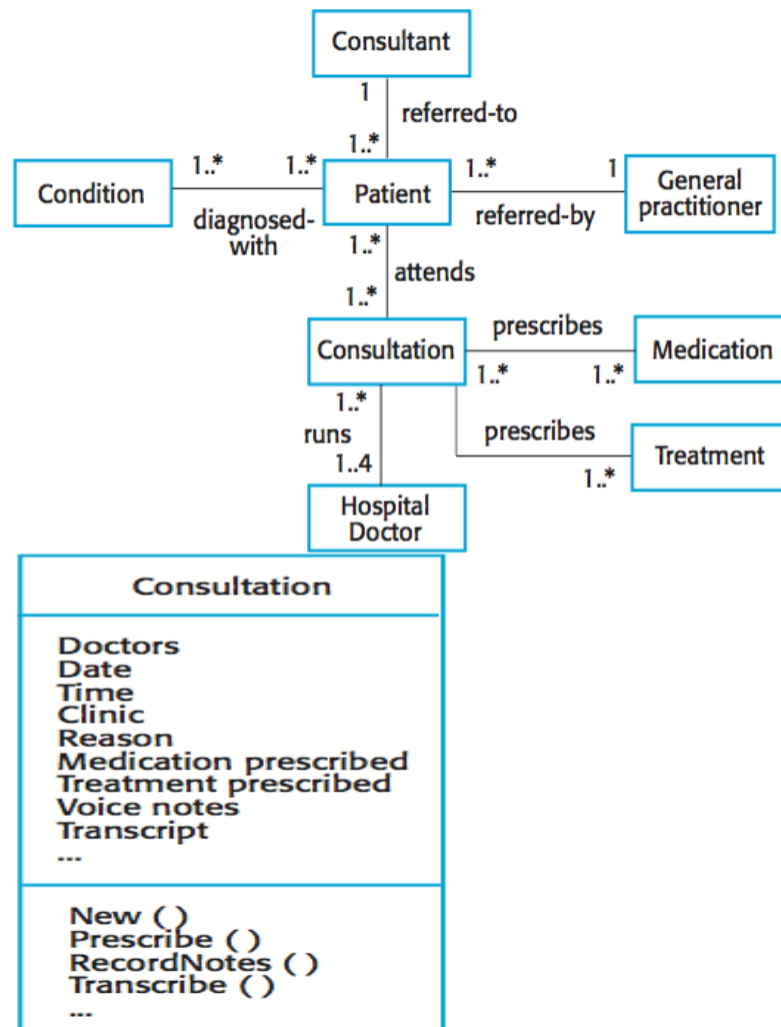


Figure 3.3.5: UML Class Diagrams

**Generalization** is an everyday technique (figure 3.3.6) that we use to manage complexity. In modeling systems, it is often useful to examine the classes in a system to see if there is scope for generalization. In object-oriented languages, such as Java, generalization is implemented using the class **inheritance** mechanisms built into the language. In a generalization, the attributes and operations associated with higher-level classes are also associated with the lower-level

classes. The lower-level classes are subclasses inherit the attributes and operations from their superclasses. These lower-level classes then add more specific attributes and operations.

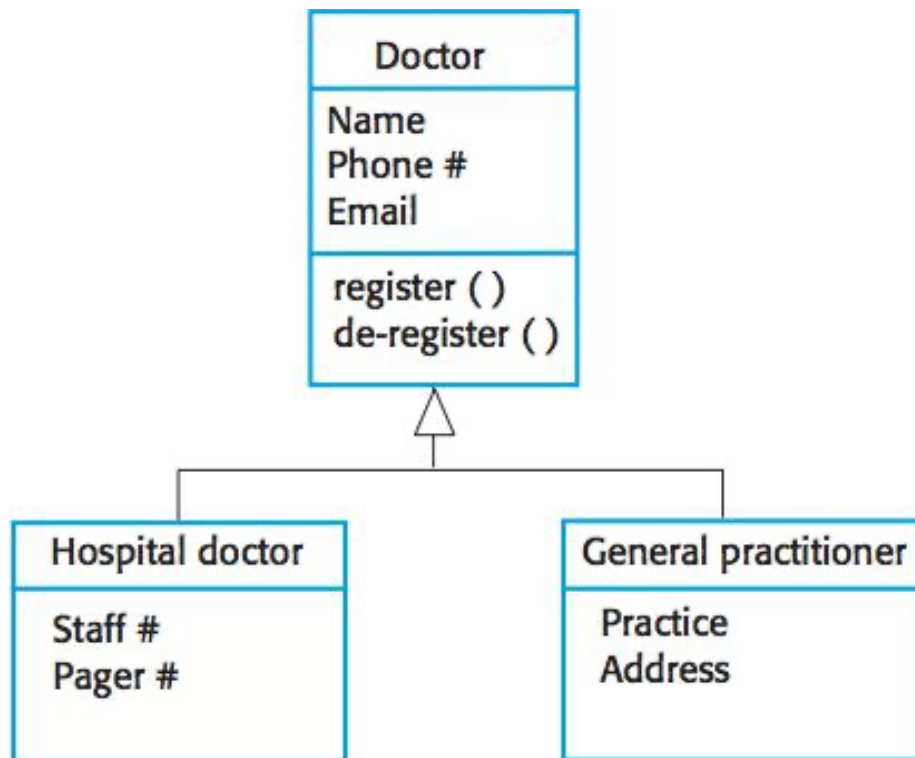


Figure 3.3.6: Generalization implemented using class inheritance mechanisms

An **aggregation** model shows (figure 3.3.7) how classes that are collections are composed of other classes. Aggregation models are similar to the part-of relationship in semantic data models.

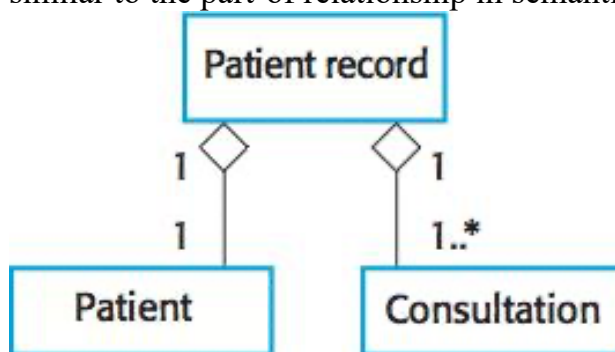


Figure 3.3.7: Aggregation Models

### 3.3.4 Behavioral models

**Behavioral models** are models of the dynamic behavior of a system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus from its environment. Two types of stimuli:

Some **data** arrives that has to be processed by the system

Some **event** happens that triggers system processing. Events may have associated data, although this is not always the case.

Many business systems are data-processing systems that are primarily driven by data. They are controlled by the data input to the system, with relatively little external event processing.

**Data-driven models** show the sequence of actions involved in processing input data and generating an associated output. They are particularly useful during the analysis of requirements as they can be used to show end-to-end processing in a system.

**Data-driven models** can be created using UML **activity diagrams**:

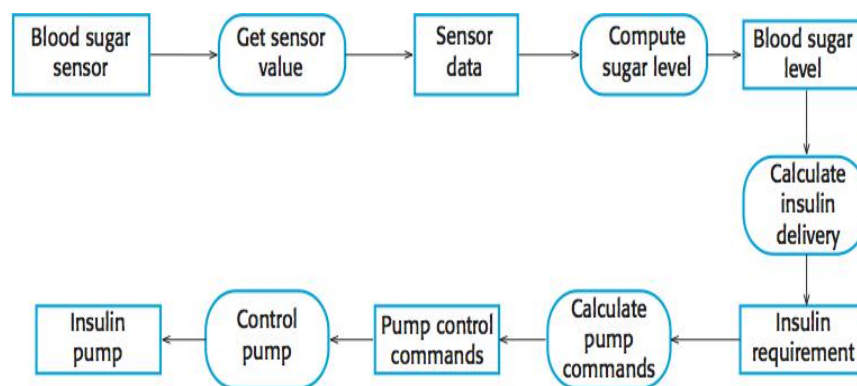


Figure 3.3.8: Data-driven models using UML activity diagrams

Data-driven models can also be created using UML **sequence diagrams**:

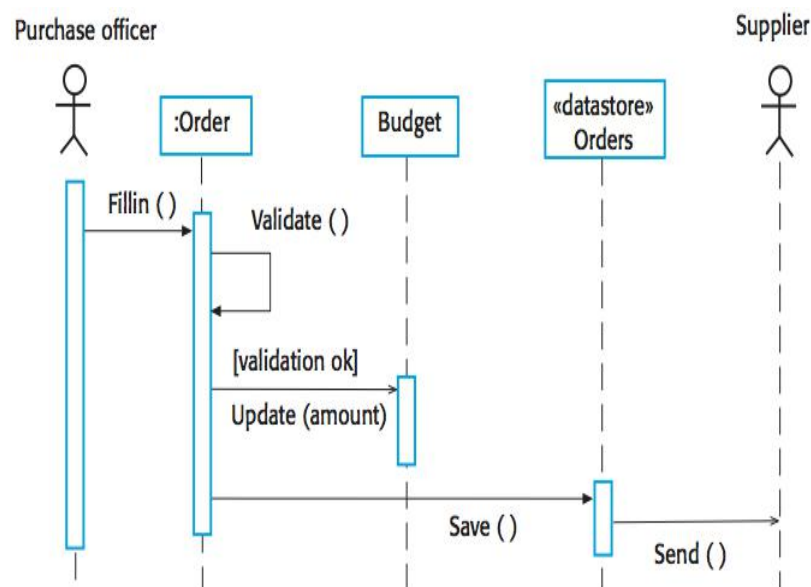


Figure 3.3.9: Data-driven models using UML sequence diagrams

Real-time systems are often event-driven, with minimal data processing. For example, a landline phone switching system responds to events such as 'receiver off hook' by generating a dial tone.

**Event-driven models** shows how a system responds to external and internal events. It is based on the assumption that a system has a finite number of states and that events (stimuli) may cause a transition from one state to another.

Event-driven models can be created using UML **state diagrams**:

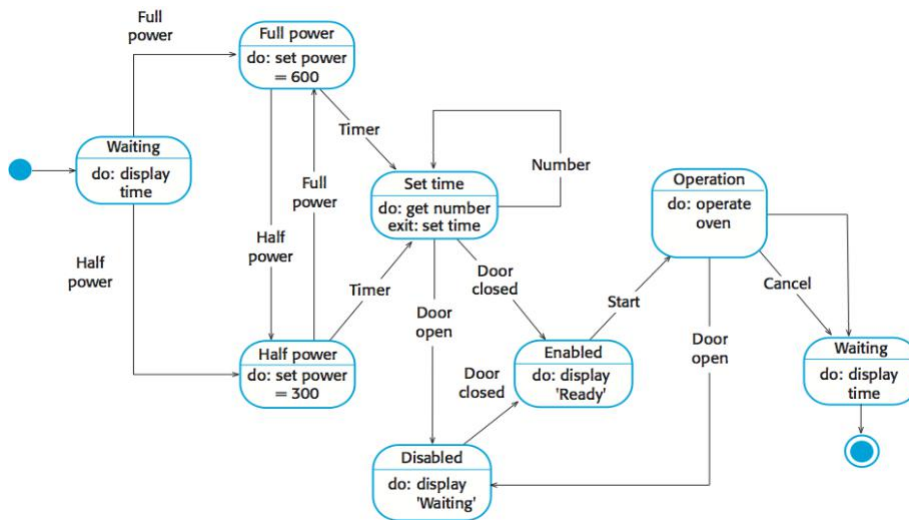


Figure 3.3.10: Event-driven models using UML state diagrams

## 4.0 SELF-ASSESSMENT EXERCISES

Define a Systems, Models and Systems models

Answer:

A **system** is a simplified representation of reality. "System" is a common word, often used with loose meaning. Whereas in the real world, a "system" may seem at times an endless series of connected elements, we refer here to a system as (1) a series of selected, chosen elements (this is a first simplification, and thus an implicit assumption), with (2) specified boundaries (a second simplification and implicit assumption), and (3) pre-determined time characteristics (with a third simplification and implicit assumption).

A **model** is a computer program that describes the mechanics of the considered system. The encoding of a model can be made in many ways. **Systems Models:** A system is a set of elements that relate to each other in some manner. The elements of a system can be objects, people, organizations, processes, descriptions or even ideas. The relationships

between these elements can include different kinds of influence, flows of information, resources, associations, temporal relationships, or origins

State and explain the 5 types of UML diagrams for system modeling.

Answer:

**Activity** diagrams, which show the activities involved in a process or in data processing.

**Use case** diagrams, which show the interactions between a system and its environment.

**Sequence** diagrams, which show interactions between actors and the system and between system components.

**Class** diagrams, which show the object classes in the system and the associations between these classes.

**State** diagrams, which show how the system reacts to internal and external events

Identify the types of interactions that can be represented in a model.

**User interaction:** Modeling user interaction is important as it helps to identify user requirements.

**System-to-system interaction:** Modeling system-to-system interaction highlights the communication problems that may arise.

**Component interaction:** Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.

## 5.0 CONCLUSION

Systems modeling is essential to be able to represent a real life entity. A model enables stakeholders to experience or monitor the elements of a systems and the relationships amongst them. In computing, it is modeling that enables a systems analyst to isolate sub-systems by the single task that each should perform.

## 6.0 SUMMARY

A system is a simplified representation of reality. A model is a computer program that describes the mechanics of the considered system. The encoding of a model can be made in many ways.

Systems Models: A system is a set of elements that relate to each other in some manner. The elements of a system can be objects, people, organizations, processes, descriptions or even ideas. Systems Models types are Context and process models, Interaction models, Structural models and Behavioral models.

## 7.0 REFERENCES/FURTHERREADING

<https://en.wikipedia.org/wiki/Systemsmodeling>

---

## UNIT 3     **DISTRIBUTED OBJECTS**

### CONTENTS

- 2.0 Introduction
- 3.3 Intended Learning Outcomes (ILOs)
- 3.0 Main Content
  - 3.1 Distributed Objects Introduction
  - 3.2 Local Objects Vs. Distributed Objects
  - 3.3 The Distributed Objects Paradigm
  - 3.3 Distributed Objects
  - 3.4 Distributed Objects Systems/ Protocols
  - 3.5 Remote procedure Call & Remote Method Invocation
    - 3.6.1 Remote procedure Call
    - 3.6.2 Remote Procedure Call Model
  - 3.7 Local Procedure Call and Remote Procedure Call
    - 1.7.1 Remote Procedure Calls (RPC)
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

A distributed object is an object that can be accessed remotely. This means that a distributed object can be used like a regular object, but from anywhere on the network. An object is typically considered to encapsulate data and behavior. The location of the distributed object is not critical to the user of the object. A distributed object might provide its user with a set of related capabilities. The application that provides a set of capabilities is often referred to as a *service*. A *Business Object* might be a local object or a distributed object. The term business object refers to an object that performs a set of tasks associated with a particular business process.

### 2.0 INTENDED LEARNING OUTCOMES (ILOS)

By the end of this unit, you will be able to:

- Explain the concept of Distributed Objects
- Differentiate between the Local and Distributed objects
- Identify and explain the distributed object paradigm mechanisms

### 3.1 Distributed Objects

#### The distributed object paradigm

It provides abstractions beyond those of the message-passing model. In object-oriented programming, objects are used to represent an entity significant to an application.

Each object encapsulates:

The **state** or data of the entity: in Java, such data is contained in the instance variables of each object;

The **operations** of the entity, through which the state of the entity can be accessed or updated.

### 3.2 Local Objects vs. Distributed Objects

**Local objects** are those whose methods can only be invoked by a **local process**, a process that runs on the same computer on which the object exists.

A **distributed object** is one whose methods can be invoked by a **remote process**, a process running on a computer connected via a network to the computer on which the object exists.

### 3.3 The Distributed Object Paradigm

In a distributed object paradigm, network resources are represented by distributed objects.

To request service from a network resource, a process invokes one of its operations or methods, passing data as parameters to the method.

The method is executed on the remote host, and the response is sent back to the requesting process as a return value.

**Message-passing paradigm** is **data-oriented** while **Distributed objects paradigm** is **action-oriented**: the focus is on the invocation of the operations, while the data passed takes on a secondary role.

Although less intuitive to human-beings, the distributed-object paradigm is more natural to object-oriented software development.

### 3.4 Distributed Objects

distributed object is provided, or **exported**, by a process called the **object server**. A facility, here called an **object registry**, must be present in the system architecture for the distributed object to be registered. To access a distributed object, a process –an **object client** –



looks up the object registry for a **reference** to the object. This reference is used by the object client to make calls to the methods.

Logically, the object client makes a call directly to a remote method. In **reality**, the call is handled by a software component, called a **client proxy**, which interacts with the software on the client host that provides the runtime support for the distributed object system. The runtime support is responsible for the inter-process communication needed to transmit the call to the remote host, including the marshalling of the argument data that needs to be transmitted to the remote object.

A similar architecture is required on the **server side**, runtime support for the distributed object system handles the receiving of messages and the un-marshalling of data, and forwards the call to a software component called the **server proxy**. The server proxy interfaces with the distributed object to invoke the method call locally, passing in the un-marshalled data for the arguments. The method call results in the performance of some tasks on the server host. The outcome of the execution of the method, including the marshalled data for the return value, is forwarded by the server proxy to the client proxy, via the **runtime support** and **network support** on both sides.

### 3.5 Distributed Object Systems/Protocols

The distributed object paradigm has been widely adopted in distributed applications, for which a large number of mechanisms based on the paradigm are available. Among the most well-known of such mechanisms are:

Java Remote Method Invocation (RMI),

The Common Object Request Broker Architecture (CORBA) systems,

The Distributed Component Object Model (DCOM),

Mechanisms that support the Simple Object Access Protocol (SOAP).

Of these, the **most straightforward is the Java RMI**.

### 3.6 Remote Procedure Call & Remote Method Invocation

#### 3.6.1 Remote Procedure Calls (RPC)

Remote Method Invocation has its origin in a paradigm called Remote Procedure Call

\

### 3.6.2 Remote procedure call model:

A procedure call is made by one process to another, with data passed as arguments.

Upon receiving a call:

1. The actions encoded in the procedure are executed
2. The caller is notified of the completion of the call and
3. A return value, if any, is transmitted from the callee to the caller

## 3.7 Local Procedure Call and Remote Procedure Call

### 3.7.1 Remote Procedure Calls (RPC)

Since its introduction in the early 1980s, the Remote Procedure Call model has been widely in use in network applications.

There are two prevalent APIs for this paradigm.

The *Open Network Computing Remote Procedure Call*, evolved from the RPC API originated from Sun Microsystems in the early 1980s.

The other well-known API is the *Open Group Distributed Computing Environment (DCE) RPC*.

Both APIs provide a tool, *rpcgen*, for transforming remote procedure calls to local procedure calls to the stub.

## 4.0 SELF-ASSESSMENT EXERCISES

### Define Distributed Objects

Answer:

A distributed object is an object that can be accessed remotely. This means that a distributed object can be used like a regular object, but from anywhere on the network. An object is typically considered to encapsulate data and behavior. The location of the distributed object is not critical to the user of the object.

A procedure call is made by one process to another, with data passed as arguments. What are the three things that happen upon the callee receiving a call:

Answer:

1. The actions encoded in the procedure are executed
2. The caller is notified of the completion of the call and  
A return value, if any, is transmitted from the callee to the caller

What are the prevalent Remote Procedure Calls (RPC) APIs widely in use in network applications?

Answer:

There are two prevalent RPC APIs for this paradigm.

The *Open Network Computing Remote Procedure Call*, evolved from the RPC API originated from Sun Microsystems in the early 1980s.

The other well-known API is the *Open Group Distributed Computing Environment* (DCE) RPC.

## 5.0 CONCLUSION

In Object Oriented Programming, we deal with data and functions. There exists communications amongst the objects through parameters and every response is communicated back to the caller. This makes modular programming easily implemented.

## 6.0 SUMMARY

A distributed object is an object that can be accessed remotely. A distributed object is provided, or exported, by a process called the **object server**. A facility, here called an **object registry**, must be present in the system architecture for the distributed object to be registered. To access a distributed object, a process –an **object client** – looks up the object registry for a reference to the object. Among the most well-known distributed object paradigm mechanisms are: Java Remote Method

Invocation (RMI), the Common Object Request Broker Architecture (CORBA) systems, the Distributed Component Object Model (DCOM) and mechanisms that support the Simple Object Access Protocol (SOAP). There are two prevalent RPC APIs for this paradigm vis-à-vis the *Open Network Computing Remote Procedure Call*, evolved from the RPC API originated from Sun Microsystems in the early 1980s and the *Open Group Distributed Computing Environment* (DCE) RPC.

## 7.0 REFERENCES/FURTHER READING

<https://www4.cs.fau.de/~geier/corba-faq/why-distrib-objs.html>

## UNIT 4 REMOTE METHOD INVOCATION

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main Content
  - 3.1 Java Remote Method Invocation
    - 3.1.1 Remote Method Invocation
  - 3.2 The Java RMI Architecture
    - 3.2.1 Object Registry
  - 3.3 The Interaction between the Stub and the Skeleton
  - 3.4 The Remote Interface
    - 3.4.1 A Sample Remote Interface
  - 3.5 The Server-Side Software
  - 3.6 The Remote Interface Implementation
  - 3.7 UML Diagram for the SomeImpl class
    - 3.7.1 Stub and Skeleton Generations
    - 3.7.2 The Stub File for the Object
  - 3.8 The Object Server
  - 3.9 The RMI Registry
  - 3.10 The Client-Side Software
  - 3.11 Looking up the Remote Object
  - 3.12 Invoking the Remote Method
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

In computing, the **Java Remote Method Invocation (Java RMI)** is a Java API that performs remote method invocation, the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage-collection. The original implementation depends on Java Virtual Machine (JVM) class-representation mechanisms and it thus only supports making calls from one JVM to another. The protocol underlying this Java-only implementation is known as Java Remote Method Protocol (JRMP). In order to support code running in a non-JVM context, programmers later developed a CORBA version.

Usage of the term **RMI** may denote solely the programming interface or may signify both the API and JRMP, IIOP, or another implementation, whereas the term RMI-IIOP (read: RMI over IIOP) specifically denotes

the RMI interface delegating most of the functionality to the supporting CORBA implementation.

## 2.0 INTENDED LEARNING OUTCOMES (ILOS)

By the end of this unit, you will be able to:

Explain the concept of Remote Method Invocation

Explain the Interaction between the Stub and the Skeleton

Identify and explain the API for the Java RMI

Describe an Object Server and how it works

## 3.1 Java Remote Method Invocation

### 3.1.1 Remote Method Invocation

Remote Method Invocation (RMI) is an object-oriented implementation of the Remote Procedure Call model. It is an API for Java programs only. Using RMI, an *object server* exports a *remote object* and registers it with a directory service. The object provides remote methods, which can be invoked in client programs.

#### Syntactically:

A remote object is declared with a *remote interface*, an extension of the Java *interface*.

The remote interface is implemented by the object server.

An *object client* accesses the object by **invoking the remote methods** associated with the objects using syntax provided for remote method invocations.

## 3.2 The Java RMI Architecture

### 3.2.1 Object Registry

The RMI API allows a number of directory services to be used for registering a distributed object.

A simple directory service called the RMI registry, *rmiregistry*,

which is provided with the Java Software Development Kit

The RMI Registry is a service whose server, when active, runs on the **object server's host machine**, by convention and by default on the TCP port 1099.

### 3.1 The Interaction between the Stub and the Skeleton

A time-event diagram describing the interaction between the stub and the skeleton:

#### The API for the Java RMI

The Remote Interface

The Server-side Software

    The Remote Interface Implementation

    Stub and Skeleton Generations

The Object Server

The Client-side Software

### 3.4 The Remote Interface

A Java interface is a class that serves as a template for other classes:

It contains declarations or signatures of methods whose — implementations are to be supplied by classes that implements the interface.

A java remote interface is an interface that inherits from the Java *Remote* class, which allows the interface to be implemented using RMI syntax.

Other than the Remote extension and the Remote exception that must be specified with each method signature, a remote interface has the same syntax as a regular or local Java interface.

#### 3.4.1 A sample remote Interface

```

file: SomeInterface.java
to be implemented by a Java RMI server class.

import java.rmi.*
public interface SomeInterface extends Remote {
    signature of first remote method
    public String someMethod1( )
        throws java.rmi.RemoteException;
    signature of second remote method
    public int someMethod2( float throws java.rmi.RemoteException; )
    signature of other remote methods may follow
} // end interface

```

### **A sample remote interface**

The **java.rmi.Remote** Exception must be listed in the *throw* clause of each method signature.

This exception is raised when errors occur during the processing of a remote method call, and the exception is required to be caught in the method caller's program.

Causes of such exceptions include exceptions that may occur during inter-process communications, such as access failures and connection failures, as well as problems unique to remote method invocations, including errors resulting from the object, the stub, or the skeleton not being found.

## **3.5 The Server-side Software**

An object server is an object that provides the methods of and the interface to a distributed object.

Each object server must

- Implement each of the remote methods specified in the interface,
- Register an object which contains the implementation with a directory service.

It is recommended that the two parts be provided as separate classes.

### **3.5.3 The Remote Interface Implementation**

A class which implements the remote interface should be provided.

The syntax is similar to a class that implements a local interface.

```

import java.rmi.*;
import java.rmi.server.*;

/**
 * This class implements the remote interface
 * SomeInterface. */

public class SomeImpl extends UnicastRemoteObject
    implements SomeInterface {
    public SomeImpl() throws RemoteException
        { super();
    }
    public String someMethod1() throws RemoteException
        { // code to be supplied
    }
    public int someMethod2() throws RemoteException {
        // code to be supplied
    }
} // end class

```

### 3.7 UML diagram for the SomeImpl class

#### 3.7.1 Stub and Skeleton Generations

In RMI, each distributed object requires a proxy each for the object server and the object client, known as the object's skeleton and stub, respectively.

These proxies are generated from the implementation of a remote interface using a tool provided with the Java SDK:

the RMI compiler *rmic*.

- **rmic <class name of the remote interface implementation>**

For example:

- **rmic SomeImpl**

As a result of the compilation, two proxy files will be generated, each prefixed with the implementation class name:

**SomeImpl\_skel.class**  
**SomeImpl\_stub.class.**



### 3.7.2 The stub file for the object

The stub file for the object, as well as the remote interface file, must be shared with each object client – these file are required for the client program to compile.

A copy of each file may be provided to the object client by hand.

In addition, the Java RMI has a feature called “stub downloading” which allows a stub file to be obtained by a client dynamically.

### 3.8 The Object Server

The object server class is a class whose code instantiates and exports an object of the remote interface implementation.

A template for the object server class.

```
import java.rmi.*;
.....
public class SomeServer {
    public static void main(String args[]) {
        try{
            code for port number value to be supplied
            SomeImpl exportedObj = new SomeImpl();
            startRegistry(RMIPortNum);
            register the object under the name “some”

            registryURL = "rmi://localhost:" + portNum +
"/some";
            Naming.rebind(registryURL, exportedObj);
            System.out.println("Some Server ready.");
        } // end try
    } // end main
```

This method starts a RMI registry on the local host, if it does not already exists at the specified port number.

```
private static void startRegistry(int RMIPortNum)
    throws
    RemoteException{ try {
        Registry registry=
        LocateRegistry.getRegistry(RMIPortNu m);
        registry.list();
```

The above call will throw an exception  
// if the registry does not already exist

```
}
catch (RemoteException ex) {
    // No valid registry at that port.
```

```

System.out.println(
    "RMI registry cannot be located at port "
+ RMIPortNum);
Registry registry= LocateRegistry.createRegistry(RMIPor
tNum);
System.out.println(
    "RMI registry created at port " + RMIPortNum);
}
} // end startRegistry

```

In our object server template, the code for exporting an object is as follows:

```

register the object under the name "some"
registryURL = "rmi://localhost:" + portNum + "/some";
Naming.rebind(registryURL, exportedObj);

```

The *Naming* class provides methods for storing and obtaining references from the registry.

- In particular, the *rebind* method allow an object reference to be stored in the registry with a URL in the form of:  
**rmi://<host name>:<port number>/<reference name>**
  - The *rebind* method will overwrite any reference in the registry bound with the given reference name.
  - If the overwriting is not desirable, there is also a *bind* method.  
The host name should be the name of the server, or simply "localhost".  
The reference name is a name of your choice, and should be unique in the registry.

When an object server is executed, the exporting of the distributed object causes the server process to begin to listen and wait for clients to connect and request the service of the object. An RMI object server is a concurrent server: each request from an object client is serviced using a separate thread of the server. Note that if a client process invokes multiple remote method calls, these calls will be executed concurrently unless provisions are made in the client process to synchronize the calls.

### 3.9 The RMI Registry

A server exports an object by registering it by a symbolic name with a server known as the RMI registry.

```
// Create an object of the Interface

SomeInterfacel obj = new SomeInterface("Server1");

Register the object; rebind will overwirte existing
registration by same name – bind( ) will not.

Naming.rebind("Server1", obj);
```

A server, called the RMI Registry, is required to run on the host of the server which exports remote objects.

The RMIRegistry is a server located at port 1099 by default It can be invoked dynamically in the server class:

```
import java.rmi.registry.LocateRegistry;
...
LocateRegistry.createRegistry ( 1099 );...
```

Alternatively, an RMI registry can be activated by hand using the **rmiregistry** utility :

**rmiregistry <port number>**

where the port number is a TCP port number.

If no port number is specified, port number 1099 is assumed.

The registry will run continuously until it is shut down (via CTRL-C, for example)

### 3.10 The Client-side Software

The program for the client class is like any other Java class. The syntax needed for RMI involves

- o locating the RMI Registry in the server host, and
- o looking up the remote reference for the server object; the reference can then be cast to the remote interface class and the remote methods invoked.

```
import java.rmi.*;
```

```

....
public class SomeClient {
  public static void main(String args[]) {
    try {
      String registryURL =
        "rmi://localhost:" + portNum + "/some";
      SomeInterface h =
        (SomeInterface)Naming.lookup(registryURL);
      invoke the remote method(s)
      String message = h.method1();
      System.out.println(message);
      method2 can be invoked
    similarly } // end try
    catch (Exception e) {
      System.out.println("Exception in SomeClient: " + e);
    }
  } //end main
  Definition for other methods of the class, if any.
} //end class

```

### 3.11 Looking up the remote object

The *lookup* method of the *Naming* class is used to retrieve the object reference, if any, previously stored in the registry by the object server.

Note that the retrieved reference must be cast to the **remote interface** (**not** its implementation) class.

```

String registryURL =
  "rmi://localhost:" + portNum + "/some";
SomeInterface h =
  (SomeInterface)Naming.lookup(registryURL);

```

### 3.12 Invoking the Remote Method

The remote interface reference can be used to invoke any of the methods in the remote interface, as in the example:

```

String message = h.method1();
System.out.println(message);

```

Note that the syntax for the invocation of the remote methods is the same as for local methods.

It is a common mistake to cast the object retrieved from the registry to the interface implementation class or the server object class.

Instead it should be cast as the interface class.

## 5.0 CONCLUSION

Clearly, an object reference cannot refer to an object on another virtual machine so the invoker of a remote method does not have an actual reference to the remote object. When we define a remote server in Java, the definition must be written as a remote interface and a separate implementation class. A *stub* object on the client's machine implements the remote interface and acts as a proxy for the remote object (the term "stub" comes from CORBA). Clients use the remote interface (not the implementation class) as the type of remote object references, and the client's remote interface reference refers to the stub. When the client sends a message in the remote interface, the receiver is actually the stub, which communicates with the remote object via TCP/IP.

## 6.0 SUMMARY

Remote Method Invocation (RMI) is an object-oriented implementation of the Remote Procedure Call model. It is an API for Java programs only. The RMI API allows a number of directory services to be used for registering a distributed object. A simple directory service called the RMI registry, *rmiregistry*, which is provided with the Java Software Development Kit. The RMI Registry is a service whose server, when active, runs on the **object server's host machine**, by convention and by default on the TCP port 1099. An object server is an object that provides the methods of and the interface to a distributed object which must implement each of the remote methods specified in the interface and register an object which contains the implementation with a directory service. It is recommended that the two parts be provided as separate classes.

## 7.0 REFERENCES/FURTHER READING

[https://en.wikipedia.org/wiki/Java\\_remote\\_method\\_invocation](https://en.wikipedia.org/wiki/Java_remote_method_invocation)  
<https://www.cs.uic.edu/~troy/fall04/cs441/drake/rmi.html>

## **UNIT 5 USING UML FOR COMPONENT-BASED DESIGNS**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main Content
  - 3.1 UML component Diagrams
  - 3.2 Component Diagram at a Glance
  - 3.3 Basic Concepts of Component Diagram
  - 3.4 Interface
    - 3.4.1 Provided Interface
    - 3.4.2 Required Interface
  - 3.5 Subsystems
  - 3.6 Port
  - 3.7 Relationships
  - 3.8 Modelling Source Code
  - 3.9 Modelling an Executable Release
  - 3.10 Modelling a Physical Database
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

Component-based software development (CBD) is a potential breakthrough for software engineering. Unified Modeling Language (UML) can potentially facilitate CBD design and modeling. Although many research projects concentrate on the conceptual interrelation of UML and CBD, few incorporate actual component frameworks into the discussion, which is critical for real-world software system design and modeling

### **2.0 INTENDED LEARNING OUTCOMES (ILOS)**

By the end of this unit, you will be able to:

- Explain basic concepts of UML component diagrams
- identify the UML diagram types
- Differentiate between Provided Interface and Required Interface

### 3.1 UML Component Diagrams

UML Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

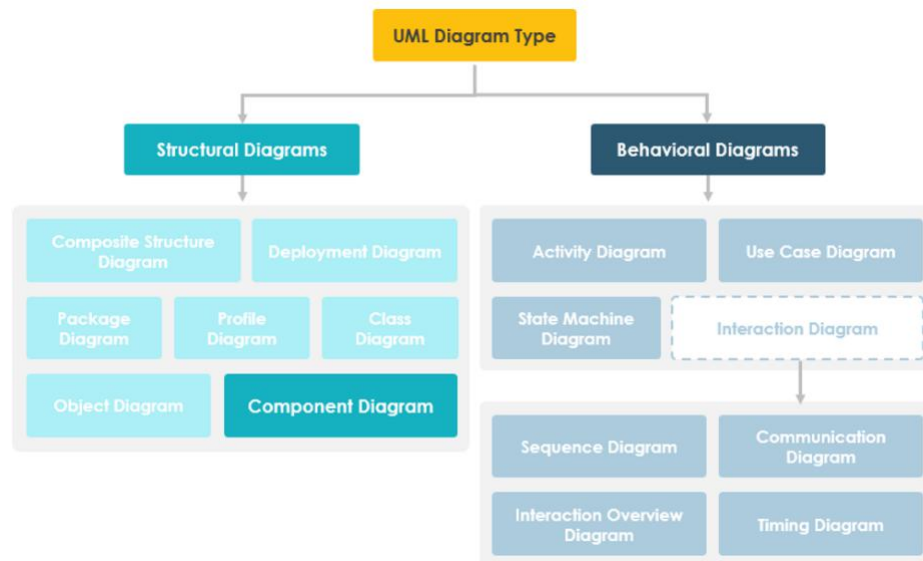


Figure 1: UML Diagram for Systems Designs

### 3.2 Component Diagram at a Glance

A component diagram breaks down the actual system under development into various high levels of functionality. Each component is responsible for one clear aim within the entire system and only interacts with other essential elements on a need-to-know basis.

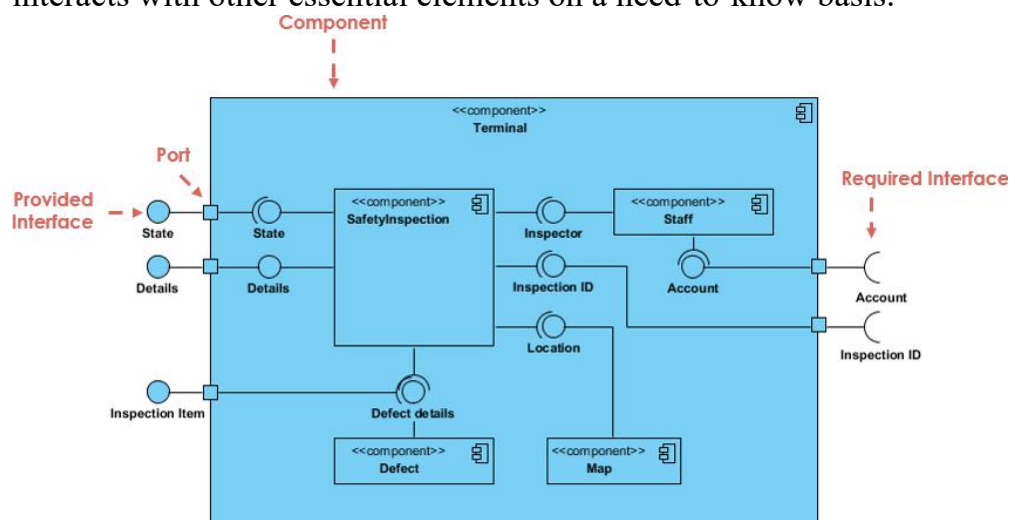


Figure 2: Component Diagram

The example above shows the internal components of a larger component:

The data (account and inspection ID) flows into the component via the port on the right-hand side and is converted into a format the internal components can use. The interfaces on the right are known as required interfaces, which represents the services the component needed in order to carry out its duty.

The data then passes to and through several other components via various connections before it is output at the ports on the left. Those interfaces on the left are known as provided interface, which represents the services to deliver by the exhibiting component.

It is important to note that the internal components are surrounded by a large 'box' which can be the overall system itself (in which case there would not be a component symbol in the top right corner) or a subsystem or component of the overall system (in this case the 'box' is a component itself).

### 3.3 Basic Concepts of Component Diagram

A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. In UML 2, a component is drawn as a rectangle with optional compartments stacked vertically. A high-level, abstracted view of a component in UML 2 can be modeled as:

- A rectangle with the component's name
- A rectangle with the component icon
- A rectangle with the stereotype text and/or icon



Figure 3: A high-level, abstracted view of a component

### 3.4 Interface

In the example below shows two type of component interfaces:

**3.4.1 Provided Interface** symbols with a complete circle at their end represent an interface that the component provides - this "lollipop" symbol is shorthand for a realization relationship of an interface classifier.

**3.4.2 Required Interface** symbols with only a half circle at their end (a.k.a. sockets) represent an interface that the component requires (in both cases, the interface's name is placed near the interface symbol itself).



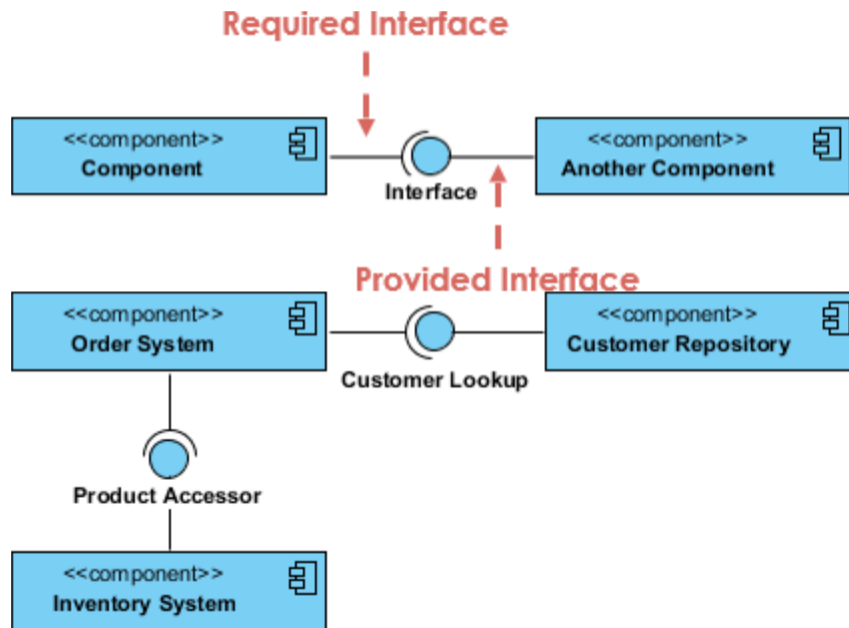


Figure 4: Component Diagram Example - Using Interface (Order System)

### 3.5 Subsystems

The subsystem classifier is a specialized version of a component classifier. Because of this, the subsystem notation element inherits all the same rules as the component notation element. The only difference is that a subsystem notation element has the keyword of subsystem instead of component.

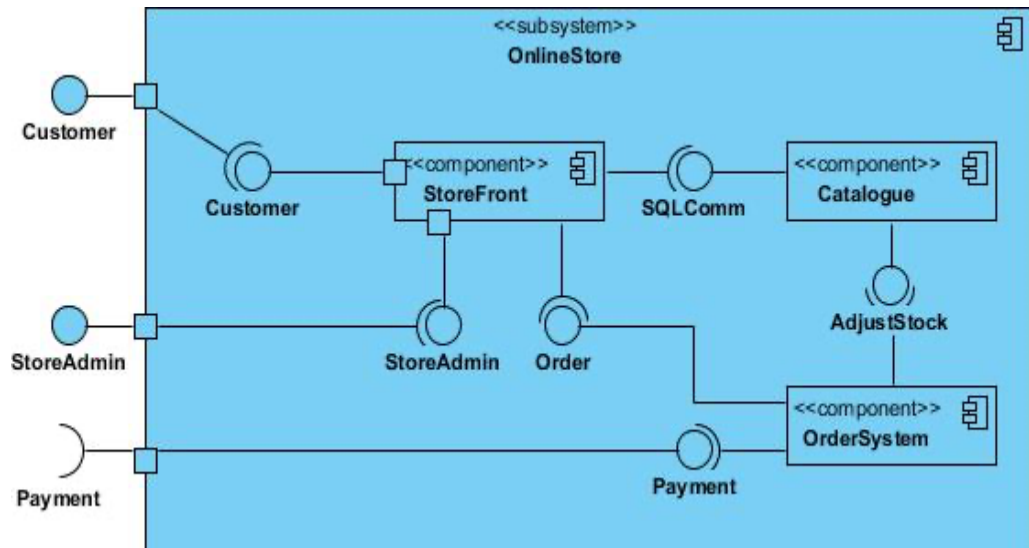


Figure 5: A Sub-system

### 3.6 Port

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.

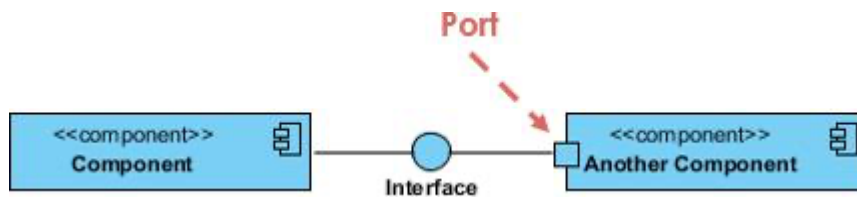



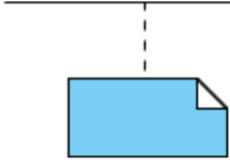
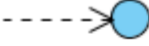



Figure 6: A Port

### 3.7 Relationships

Graphically, a component diagram is a collection of vertices and arcs and commonly contain components, interfaces and dependency, aggregation, constraint, generalization, association, and realization relationships. It may also contain notes and constraints.

Relationships	Notation
<p><b>Association:</b></p> <p>An Association specifies a semantic relationship that can occur between typed instances.</p> <p>It has at least two ends represented by properties, each of which is connected to the type of the end. More than one end of the association may have the same type.</p>	
<p><b>Composition:</b></p> <p>Composite aggregation is a strong form of aggregation that requires a part instance be included in at most one composite at a time.</p> <p>If a composite is deleted, all of its parts are normally deleted with it.</p>	

<p><b>Aggregation</b></p> <p>A kind of association that has one of its end marked shared as kind of aggregation, meaning that it has a shared aggregation.</p>	
<p><b>Constraint</b></p> <p>A condition or restriction expressed in natural language text or in a machine readable language for the purpose of declaring some of the semantics of an element.</p>	
<p><b>Dependency</b></p> <p>A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation.</p> <p>This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s).</p>	
<p><b>Links:</b></p> <p>A generalization is a taxonomic relationship between a more general classifier and a more specific classifier.</p> <p>Each instance of the specific classifier is also an indirect instance of the general classifier.</p> <p>Thus, the specific classifier inherits the features of the more general classifier.</p>	

### 3.8 Modelling Source Code

Either by forward or reverse engineering, identify the set of source code files of interest and model them as components stereotyped as files.

For larger systems, use packages to show groups of source code files.

Consider exposing a tagged value indicating such information as the version number of the source code file, its author, and the date it was last changed. Use tools to manage the value of this tag.

Model the compilation dependencies among these files using dependencies. Again, use tools to help generate and manage these dependencies.

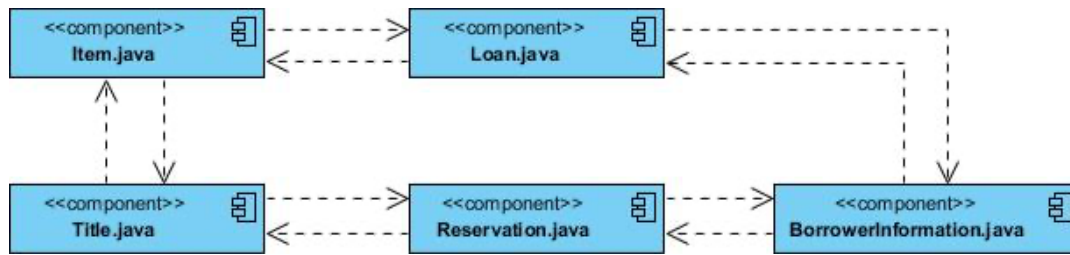
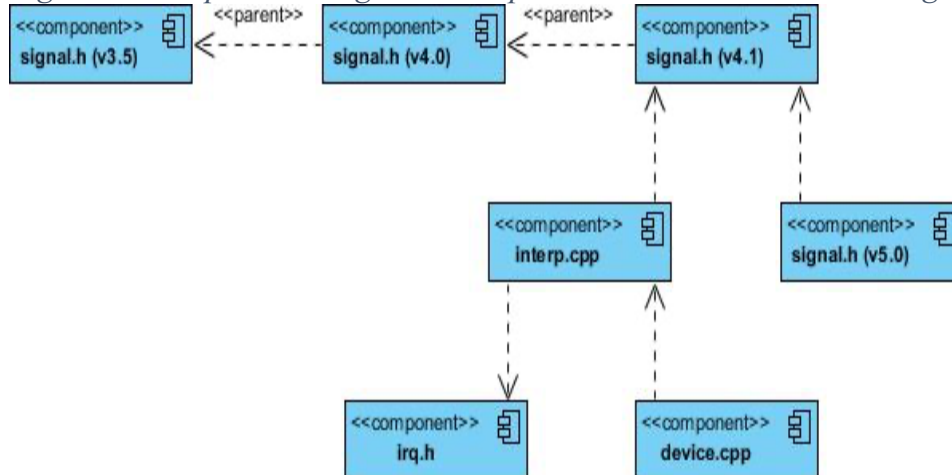


Figure 7: Component Diagram Example - C++ Code with versioning

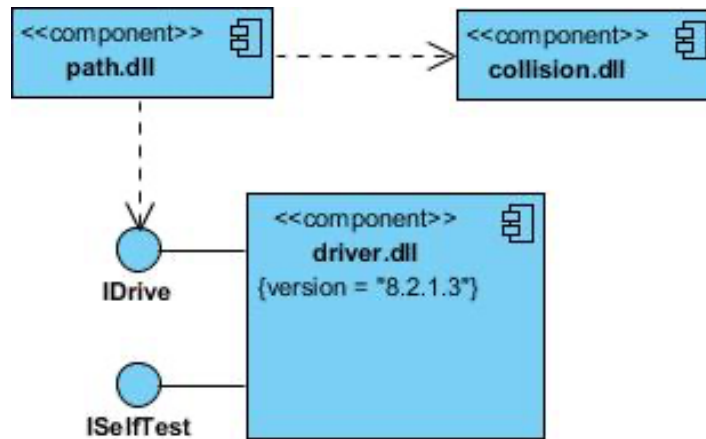


### 3.9 Modelling an Executable Release

Identify the set of components you'd like to model. Typically, this will involve some or all the components that live on one node, or the distribution of these sets of components across all the nodes in the system.

Consider the stereotype of each component in this set. For most systems, you'll find a small number of different kinds of components (such as executables, libraries, tables, files, and documents). You can use the UML's extensibility mechanisms to provide visual cues (clues) for these stereotypes.

For each component in this set, consider its relationship to its neighbors. Most often, this will involve interfaces that are exported (realized) by certain components and then imported (used) by others. If you want to expose the seams in your system, model these interfaces explicitly. If you want your model at a higher level of abstraction, elide these relationships by showing only dependencies among the components.



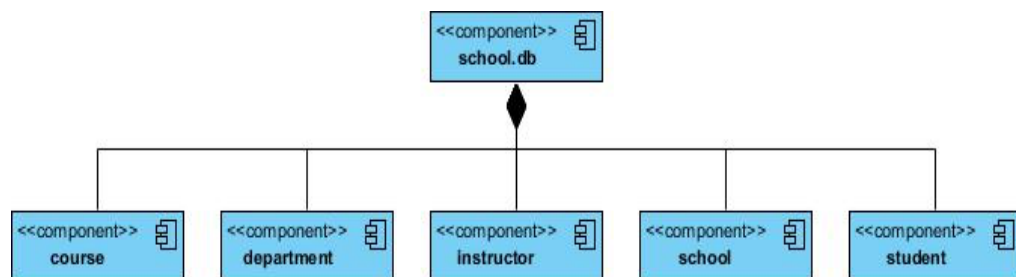
### 3.10 Modelling a Physical Database

Identify the classes in your model that represent your logical database schema.

Select a strategy for mapping these classes to tables. You will also want to consider the physical distribution of your databases. Your mapping strategy will be affected by the location in which you want your data to live on your deployed system.

To visualize, specify, construct, and document your mapping, create a component diagram that contains components stereotyped as tables.

Where possible, use tools to help you transform your logical design into a physical design.



## 5.0 CONCLUSION

**UML** Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering.

## 6.0 SUMMARY

**UML** Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing

executable systems through forward and reverse engineering. A port is often used to help expose required and provided interfaces of a component. Ports are represented using a square along the edge of the system or a component. Graphically, a component diagram is a collection of vertices and arcs and commonly contain components, interfaces and dependency, aggregation, constraint, generalization, association, and realization relationships. It may also contain notes and constraints.

There are two type of component interfaces vis-à-vis, **Provided Interface** and **Required Interface**. **Provided Interface** symbols with a complete circle at their end represent an interface that the component provides - this "lollipop" symbol is shorthand for a realization relationship of an interface classifier. **Required Interface** symbols with only a half circle at their end (a.k.a. sockets) represent an interface that the component requires (in both cases, the interface's name is placed near the interface symbol itself). **UML** Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering.

## 7.0 REFERENCES/FURTHER READING

[What is Component Diagram? \(visual-paradigm.com\)](http://visual-paradigm.com)  
<https://www.semanticscholar.org/paper/Component-Based-Development-Using-UML-Zhao-Siau/509ddc5799765c0c4d71e621768e9d1d48f92dfe>

**MODULE 4: DISTRIBUTED TRANSACTIONS****UNIT 1 DISTRIBUTED TRANSACTIONS****CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Distributed Transactions
  - 3.2 Two Types of Permissible Operations in Distributed Transactions
    - 3.2.1 DML and DDL Transactions
    - 3.2.2 Transactions Control Statements
  - 3.3 Session Trees for Distributed Transactions
  - 3.4 Node Rules
    - 3.4.1 Clients
    - 3.4.2 Database Servers
    - 3.4.3 Local Coordinators
    - 3.4.4 Global Coordinators
    - 3.4.5 Commit Point Site
  - 3.5 How a Distributed Transactions Commits
  - 3.6 Commit Point Strength
  - 3.7 Two-Phase Commit Mechanism
    - 3.7.1 Prepare Phase
    - 3.7.2 Steps in the Prepare Phase
    - 3.7.3 Commit Phase
      - 3.7.3.1 Steps in the Commit Phase
  - 3.8 Guaranteeing Global Database Consistency
  - 3.9 Forget Phase
  - 3.10 In-Doubt Transactions
    - 3.10.1 Automatic Resolution of In-Doubt Transactions
  - 3.11 Failure During the Prepare Phase
  - 3.12 Failure During the Commit Phase
  - 3.13 Manual Resolution of In-Doubt Transactions
  - 3.14 Relevance of Systems Change Numbers for In-Doubt Transactions
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

## 2.0 INTENDED LEARNING OUTCOMES (ILOS)

By the end of this unit, you will be able to:

explain the concept of Distributed Transactions

    identify the two types of permissible operations in Distributed Transactions

state the steps in the Prepare and the Commit phase

### Introduction to Module

A **distributed transaction** is a database transaction in which two or more network hosts are involved. Usually, hosts provide **transactional resources**, while the **transaction manager** is responsible for creating and managing a global transaction that encompasses all operations against such resources. Distributed transactions, as any other transactions, must have all four ACID (atomicity, consistency, isolation, durability) properties, where atomicity guarantees all-or-nothing\_outcomes for the unit of work (operations bundle).

Open Group, a vendor consortium, proposed the X/Open Distributed Transaction Processing (DTP) Model (X/Open XA), which became a de facto standard for behavior of transaction model components.

Databases are common transactional resources and, often, transactions span a couple of such databases. In this case, a distributed transaction can be seen as a database transaction that must be synchronized (or provide ACID properties) among multiple participating databases which are distributed among different physical locations.

## 1.0 INTRODUCTION

A distributed transaction is a type of transaction with two or more engaged network hosts. Generally, hosts provide resources, and a transaction manager is responsible for developing and handling the transaction. Like any other transaction, a distributed transaction should include all four ACID properties (atomicity, consistency, isolation, durability). Given the nature of the work, atomicity is important to ensure an all-or-nothing outcome for the operations bundle (unit of work).

## 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, student will able to:

Explain the term Distributed Transactions

    Identify and explain the two types of Permissible Operations in Distributed Transactions.



### 3.0 MAIN CONTENT

#### 3.1 Distributed Transactions

A **distributed transaction** includes one or more statements that, individually or as a group, update data on two or more distinct nodes of a distributed database.

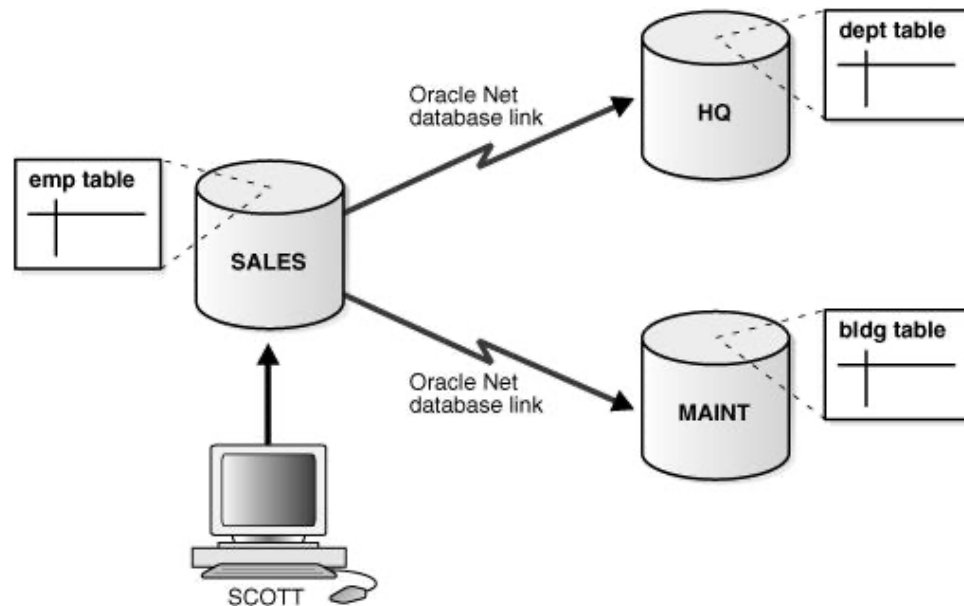


Figure 4.1.1: Sample Database on a Distributed Systems

The following distributed transaction executed by `scott` updates the local sales database, the remote `hq` database, and the remote `maint` database:

```
UPDATE scott.dept@hq.us.acme.com
  SET loc = 'REDWOOD SHORES'
  WHERE deptno = 10;
UPDATE scott.emp
  SET deptno = 11
  WHERE deptno = 10;
UPDATE scott.bldg@maint.us.acme.com
  SET room = 1225
  WHERE room = 1163;
COMMIT;
```

## 3.2 Two Types of Permissible Operations in Distributed Transactions:

DML and DDL Transactions

Transaction Control Statements

### 3.2.1 DML and DDL Transactions

The following are the DML and DDL operations supported in a distributed transaction:

CREATE TABLE AS SELECT  
DELETE  
INSERT (default and direct load)  
LOCK TABLE  
SELECT  
SELECT FOR UPDATE

You can execute DML and DDL statements in parallel, and INSERT direct load statements serially, but note the following restrictions:

All remote operations must be SELECT statements.

These statements must not be clauses in another distributed transaction.

If the table referenced in the *table\_expression\_clause* of an INSERT, UPDATE, or DELETE statement is remote, then execution is serial rather than parallel.

You cannot perform remote operations after issuing parallel DML/DDL or direct load INSERT.

If the transaction begins using XA or OCI, it executes serially.

No loopback operations can be performed on the transaction originating the parallel operation. For example, you cannot reference a remote object that is actually a synonym for a local object.

If you perform a distributed operation other than a SELECT in the transaction, no DML is parallelized.

### 3.2.2 Transaction Control Statements

The following are the supported transaction control statements:

COMMIT  
ROLLBACK  
SAVEPOINT

### 3.3 Session Trees for Distributed Transactions

As the statements in a distributed transaction are issued, the database defines a **session tree** of all nodes participating in the transaction. A session tree is a hierarchical model that describes the relationships among sessions and their roles.

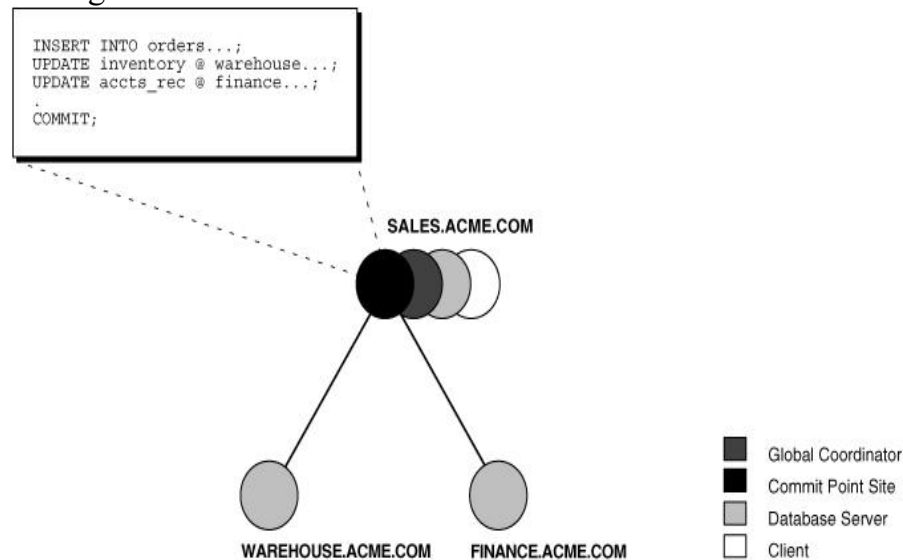


Figure 4.1.2: Example of a Session Tree

All nodes participating in the session tree of a distributed transaction assume one or more of the following roles:

### 3.4 Node Roles

Roles	Description
Client	A node that references information in a database belonging to a different node
Database server	A node that receives a request for information from another node
Global coordinator	The node that originates the distributed transaction
Local coordinator	A node that is forced to reference data on other nodes to complete its part of the transaction
Commit point site	The node that commits or rolls back the transaction as instructed by the global

The role a node plays in a distributed transaction is determined by:  
Whether the transaction is local or remote

The **commit point strength** of the node ("Commit Point Site")

Whether all requested data is available at a node, or whether other nodes need to be referenced to complete the transaction

Whether the node is read-only

### 3.4.1 Clients

A node acts as a client when it references information from a database on another node. The referenced node is a database server. In [Figure 2](#), the node `sales` is a client of the nodes that host the `warehouse` and `finance` databases.

### 3.4.2 Database Servers

A database server is a node that hosts a database from which a client requests data.

In [Figure 2](#), an application at the `sales` node initiates a distributed transaction that accesses data from the `warehouse` and `finance` nodes. Therefore, `sales.acme.com` has the role of client node, and `warehouse` and `finance` are both database servers. In this example, `sales` is a database server *and* a client because the application also modifies data in the `sales` database.

### 3.4.3 Local Coordinators

A node that must reference data on other nodes to complete its part in the distributed transaction is called a local coordinator. In [Figure 2](#), `sales` is a local coordinator because it coordinates the nodes it directly references: `warehouse` and `finance`. The node `sales` also happens to be the global coordinator because it coordinates all the nodes involved in the transaction.

A local coordinator is responsible for coordinating the transaction among the nodes it communicates directly with by:

- Receiving and relaying transaction status information to and from those nodes

- Passing queries to those nodes

- Receiving queries from those nodes and passing them on to other nodes

- Returning the results of queries to the nodes that initiated them

### 3.4.4 Global Coordinator

The node where the distributed transaction originates is called the global coordinator. The database application issuing the distributed transaction is directly connected to the node acting as the global coordinator. For example, in [Figure 2](#), the transaction issued at the node `sales` references information from the database servers `warehouse` and `finance`.

Therefore, `sales.acme.com` is the global coordinator of this distributed transaction.

The global coordinator becomes the parent or root of the session tree. The global coordinator performs the following operations during a distributed transaction:

- Sends all of the distributed transaction SQL statements, remote procedure calls, and so forth to the directly referenced nodes, thus forming the session tree

- Instructs all directly referenced nodes other than the commit point site to prepare the transaction

- Instructs the commit point site to initiate the global commit of the transaction if all nodes prepare successfully

- Instructs all nodes to initiate a global rollback of the transaction if there is an abort response

### 3.4.5 Commit Point Site

The job of the commit point site is to initiate a commit or roll back operation as instructed by the global coordinator. The system administrator always designates one node to be the commit point site in the session tree by assigning all nodes a commit point strength. The node selected as commit point site should be the node that stores the most critical data.

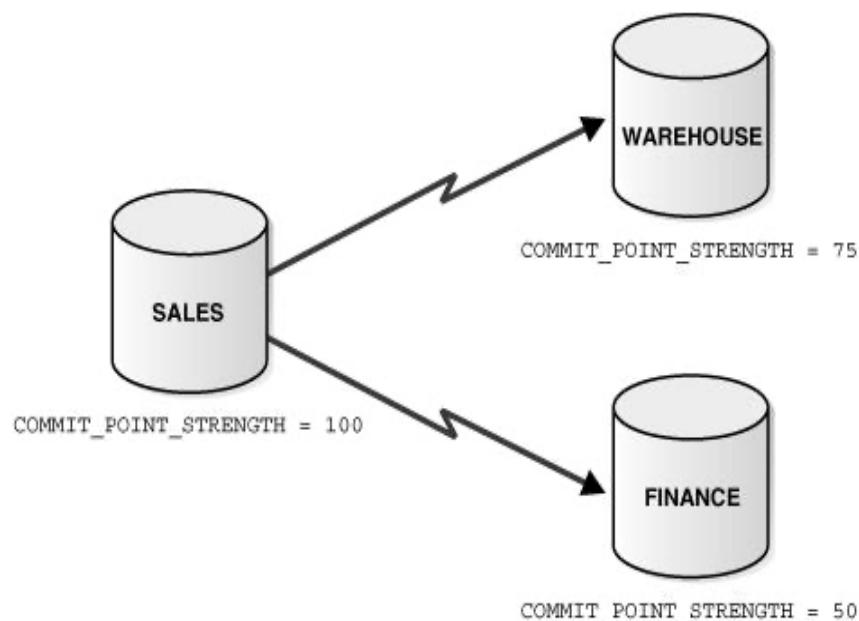


Figure 4.1.3: Commit Point Site

The commit point site is distinct from all other nodes involved in a distributed transaction in these ways:

- The commit point site never enters the prepared state.
- Consequently, if the commit point site stores the most critical data,

this data never remains in-doubt, even if a failure occurs. In failure situations, failed nodes remain in a prepared state, holding necessary locks on data until in-doubt transactions are resolved. The commit point site commits before the other nodes involved in the transaction. In effect, the outcome of a distributed transaction at the commit point site determines whether the transaction at all nodes is committed or rolled back: the other nodes follow the lead of the commit point site. The global coordinator ensures that all nodes complete the transaction in the same manner as the commit point site.

### 3.5 How a Distributed Transaction Commits

A distributed transaction is considered committed after all non-commit-point sites are prepared, and the transaction has been actually committed at the commit point site. The redo log at the commit point site is updated as soon as the distributed transaction is committed at this node.

Because the commit point log contains a record of the commit, the transaction is considered committed even though some participating nodes may still be only in the prepared state and the transaction not yet actually committed at these nodes. In the same way, a distributed transaction is considered *not* committed if the commit has not been logged at the commit point site.

### 3.6 Commit Point Strength

Every database server must be assigned a commit point strength. If a database server is referenced in a distributed transaction, the value of its commit point strength determines which role it plays in the two-phase commit. Specifically, the commit point strength determines whether a given node is the commit point site in the distributed transaction and thus commits before all of the other nodes. This value is specified using the initialization parameter `COMMIT_POINT_STRENGTH`. This section explains how the database determines the commit point site.

The commit point site, which is determined at the beginning of the prepare phase, is selected only from the nodes participating in the transaction. The following sequence of events occurs:

- Of the nodes directly referenced by the global coordinator, the database selects the node with the highest commit point strength as the commit point site.

- The initially-selected node determines if any of the nodes from which it has to obtain information for this transaction has a higher commit point strength.

- Either the node with the highest commit point strength directly referenced in the transaction or one of its servers with a higher commit point strength becomes the commit point site. After the

final commit point site has been determined, the global coordinator sends prepare responses to all nodes participating in the transaction

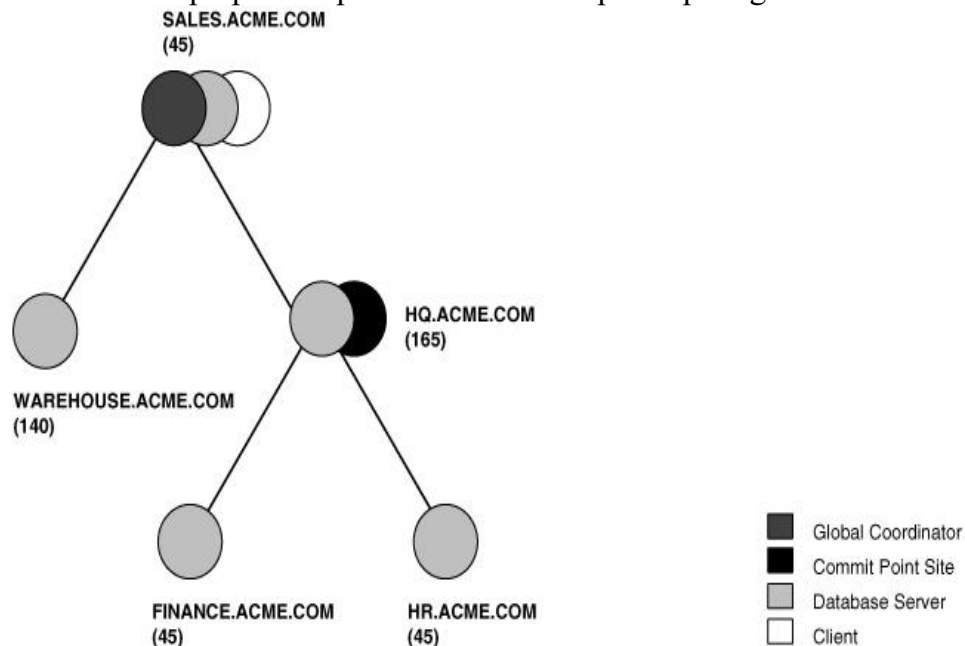


Figure 4.1.4: Commit Point Strengths and Determination of the Commit Point Site

The following conditions apply when determining the commit point site:

A read-only node cannot be the commit point site.

If multiple nodes directly referenced by the global coordinator have the same commit point strength, then the database designates one of these as the commit point site.

If a distributed transaction ends with a rollback, then the prepare and commit phases are not needed. Consequently, the database never determines a commit point site. Instead, the global coordinator sends a **ROLLBACK** statement to all nodes and ends the processing of the distributed transaction.

As [Figure 4](#) illustrates, the commit point site and the global coordinator can be different nodes of the session tree. The commit point strength of each node is communicated to the coordinators when the initial connections are made. The coordinators retain the commit point strengths of each node they are in direct communication with so that commit point sites can be efficiently selected during two-phase commits. Therefore, it is not necessary for the commit point strength to be exchanged between a coordinator and a node each time a commit occurs.

### 3.7 Two-Phase Commit Mechanism

Unlike a transaction on a local database, a distributed transaction involves altering data on multiple databases. Consequently, distributed transaction

processing is more complicated, because the database must coordinate the committing or rolling back of the changes in a transaction as a self-contained unit. In other words, the entire transaction commits, or the entire transaction rolls back.

The database ensures the integrity of data in a distributed transaction using the **two-phase commit mechanism**.

In the **prepare phase**, the initiating node in the transaction asks the other participating nodes to promise to commit or roll back the transaction. During the **commit phase**, the initiating node asks all participating nodes to commit the transaction.

If this outcome is not possible, then all nodes are asked to roll back. All participating nodes in a distributed transaction should perform the same action: they should either all commit or all perform a rollback of the transaction. The database automatically controls and monitors the commit or rollback of a distributed transaction and maintains the integrity of the **global database** (the collection of databases participating in the transaction) using the two-phase commit mechanism. This mechanism is completely transparent, requiring no programming on the part of the user or application developer.

The commit mechanism has the following distinct phases, which the database performs automatically whenever a user commits a distributed transaction:

Phase	Description
Prepare phase	The initiating node, called the <b>global coordinator</b> , asks participating nodes other than the commit point site to promise to commit or roll back the transaction, even if there is a failure. If any node cannot prepare, the transaction is rolled back.
Commit phase	If all participants respond to the coordinator that they are prepared, then the coordinator asks the commit point site to commit. After it commits, the coordinator asks all other nodes to commit the transaction
Forget phase	The global coordinator forgets about the transaction

### 3.7.1 Prepare Phase

The first phase in committing a distributed transaction is the prepare phase. In this phase, the database does not actually commit or roll back the transaction. Instead, all nodes referenced in a distributed transaction (except the commit point site, described in the "Commit Point Site") are told to prepare to commit. By preparing, a node:



Records information in the redo logs so that it can subsequently either commit or roll back the transaction, regardless of intervening failures

Places a distributed lock on modified tables, which prevents reads

When a node responds to the global coordinator that it is prepared to commit, the prepared node *promises* to either commit or roll back the transaction later, but does not make a unilateral decision on whether to commit or roll back the transaction. The promise means that if an instance failure occurs at this point, the node can use the redo records in the online log to recover the database back to the prepare phase.

**Note:**

Queries that start after a node has prepared cannot access the associated locked data until all phases complete. The time is insignificant unless a failure occurs.

### 3.7.1.1 Types of Responses in the Prepare Phase

When a node is told to prepare, it can respond in the following ways:

Response	Meaning
Prepared	Data on the node has been modified by a statement in the distributed transaction, and the node has successfully prepared
Read-only	No data on the node has been, or can be, modified (only queried), so no preparation is necessary
Abort	The node cannot successfully prepare.

#### Prepared Response

When a node has successfully prepared, it issues a **prepared message**. The message indicates that the node has records of the changes in the online log, so it is prepared either to commit or perform a rollback. The message also guarantees that locks held for the transaction can survive a failure.

#### Read-Only Response

When a node is asked to prepare, and the SQL statements affecting the database do not change any data on the node, the node responds with a **read-only message**. The message indicates that the node will not participate in the commit phase

There are three cases in which all or part of a distributed transaction is read-only:

Case	Condition	Consequence
Partially read-only	Any of the following occurs: Only queries are issued at one or more nodes. No data is changed. Changes rolled back due to triggers firing or constraint violations.	The read-only nodes recognize their status when asked to prepare. They give their local coordinators a read-only response. Thus, the commit phase completes faster because the database eliminates read-only nodes from subsequent
Completely read-only with prepare phase	All of following occur: No data changes. Transaction is <i>not</i> started with SET TRANSACTION READ ONLY statement	All nodes recognize that they are read-only during prepare phase, so no commit phase is required. The global coordinator, not knowing whether all nodes are read-only, must still perform the prepare phase.
Completely read-only without two-phase commit	All of following occur: No data changes. Transaction <i>is</i> started with SET TRANSACTION READ ONLY statement.	Only queries are allowed in the transaction, so global coordinator does not have to perform two-phase commit. Changes by other transactions do not degrade global transaction-level read consistency because of global SCN coordination

Note that if a distributed transaction is set to read-only, then it does not use undo segments. If many users connect to the database and their transactions are not set to `READ ONLY`, then they allocate undo space even if they are only performing queries.

### Abort Response

When a node cannot successfully prepare, it performs the following actions:

- Releases resources currently held by the transaction and rolls back the local portion of the transaction.

Responds to the node that referenced it in the distributed transaction with an abort message.

These actions then propagate to the other nodes involved in the distributed transaction so that they can roll back the transaction and guarantee the integrity of the data in the global database. This response enforces the primary rule of a distributed transaction: all nodes involved in the transaction either all commit or all roll back the transaction at the same logical time.

### 3.7.2 Steps in the Prepare Phase

To complete the prepare phase, each node excluding the commit point site performs the following steps:

The node requests that its **descendants**, that is, the nodes subsequently referenced, prepare to commit.

The node checks to see whether the transaction changes data on itself or its descendants. If there is no change to the data, then the node skips the remaining steps and returns a read-only response

The node allocates the resources it needs to commit the transaction if data is changed.

The node saves redo records corresponding to changes made by the transaction to its redo log.

The node guarantees that locks held for the transaction are able to survive a failure.

The node responds to the initiating node with a prepared response or, if its attempt or the attempt of one of its descendants to prepare was unsuccessful, with an abort response.

These actions guarantee that the node can subsequently commit or roll back the transaction on the node. The prepared nodes then wait until a COMMIT or ROLLBACK request is received from the global coordinator.

After the nodes are prepared, the distributed transaction is said to be **in-doubt**. It retains in-doubt status until all changes are either committed or rolled back.

### 3.7.3 Commit Phase

The second phase in committing a distributed transaction is the commit phase. Before this phase occurs, *all* nodes other than the commit point site referenced in the distributed transaction have guaranteed that they are prepared, that is, they have the necessary resources to commit the transaction.

### 3.7.3.1 Steps in the Commit Phase

The commit phase consists of the following steps:

The global coordinator instructs the commit point site to commit.

The commit point site commits.

The commit point site informs the global coordinator that it has committed.

The global and local coordinators send a message to all nodes instructing them to commit the transaction.

At each node, the database commits the local portion of the distributed transaction and releases locks.

At each node, the database records an additional redo entry in the local redo log, indicating that the transaction has committed.

The participating nodes notify the global coordinator that they have committed.

When the commit phase is complete, the data on all nodes of the distributed system is consistent.

Guaranteeing Global Database Consistency

Each committed transaction has an associated system change number (SCN) to uniquely identify the changes made by the SQL statements within that transaction. The SCN functions as an internal timestamp that uniquely identifies a committed version of the database.

In a distributed system, the SCNs of communicating nodes are coordinated when all of the following actions occur:

A connection occurs using the path described by one or more database links

A distributed SQL statement executes

A distributed transaction commits

Among other benefits, the coordination of SCNs among the nodes of a distributed system ensures global read-consistency at both the statement and transaction level. If necessary, global time-based recovery can also be completed.

During the prepare phase, the database determines the highest SCN at all nodes involved in the transaction. The transaction then commits with the high SCN at the commit point site. The commit SCN is then sent to all prepared nodes with the commit decision.

### 3.8 Guaranteeing Global Database Consistency

Each committed transaction has an associated system change number (SCN) to uniquely identify the changes made by the SQL statements within that transaction. The SCN functions as an internal timestamp that uniquely identifies a committed version of the database.

In a distributed system, the SCNs of communicating nodes are coordinated when all of the following actions occur:

- A connection occurs using the path described by one or more database links

- A distributed SQL statement executes

- A distributed transaction commits

Among other benefits, the coordination of SCNs among the nodes of a distributed system ensures global read-consistency at both the statement and transaction level. If necessary, global time-based recovery can also be completed.

During the prepare phase, the database determines the highest SCN at all nodes involved in the transaction. The transaction then commits with the high SCN at the commit point site. The commit SCN is then sent to all prepared nodes with the commit decision.

### 3.9 Forget Phase

After the participating nodes notify the commit point site that they have committed, the commit point site can forget about the transaction. The following steps occur:

- After receiving notice from the global coordinator that all nodes have committed, the commit point site erases status information about this transaction.

- The commit point site informs the global coordinator that it has erased the status information.

- The global coordinator erases its own information about the transaction.

### 3.10 In-Doubt Transactions

The two-phase commit mechanism ensures that all nodes either commit or perform a rollback together. What happens if any of the three phases fails because of a system or network error? The transaction becomes in-doubt.

Distributed transactions can become in-doubt in the following ways:

- A server machine running Oracle Database software crashes

- A network connection between two or more Oracle Databases involved in distributed processing is disconnected

An unhandled software error occurs

The RECO process automatically resolves in-doubt transactions when the machine, network, or software problem is resolved. Until RECO can resolve the transaction, the data is locked for both reads and writes. The database blocks reads because it cannot determine which version of the data to display for a query.

### 3.10.1 Automatic Resolution of In-Doubt Transactions

In the majority of cases, the database resolves the in-doubt transaction automatically. Assume that there are two nodes, local and remote, in the following scenarios. The local node is the commit point site. User **scott** connects to **local** and **executes** and **commits** a distributed transaction that updates **local** and **remote**.

### 3.11 Failure During the Prepare Phase

Figure 5 illustrates the sequence of events when there is a failure during the prepare phase of a distributed transaction:

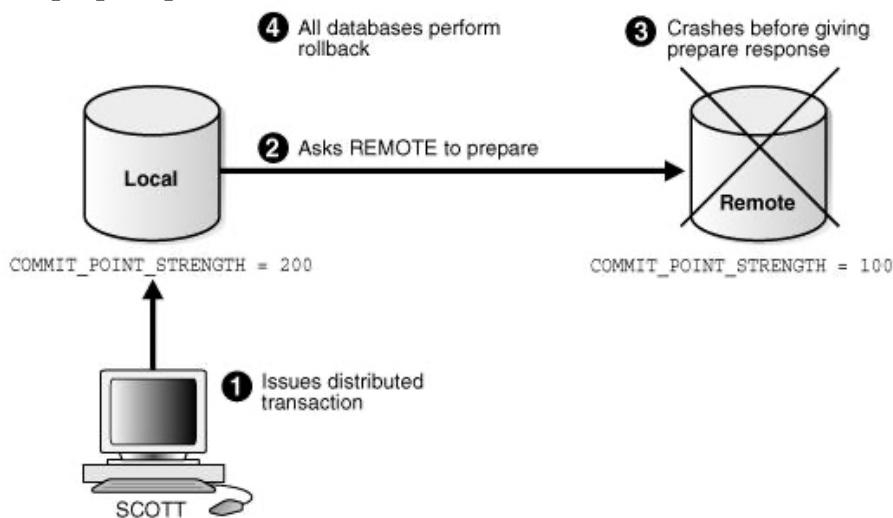


Figure 4.1.1: Failure During Prepare Phase

The following steps occur:

User **SCOTT** connects to **Local** and executes a distributed transaction.

The global coordinator, which in this example is also the commit point site, requests all databases other than the commit point site to promise to commit or roll back when told to do so.

The **remote** database crashes before issuing the prepare response back to **local**.

The transaction is ultimately rolled back on each database by the RECO process when the remote site is restored.

### 3.12 Failure During the Commit Phase

Figure 6 illustrates the sequence of events when there is a failure during the commit phase of a distributed transaction:

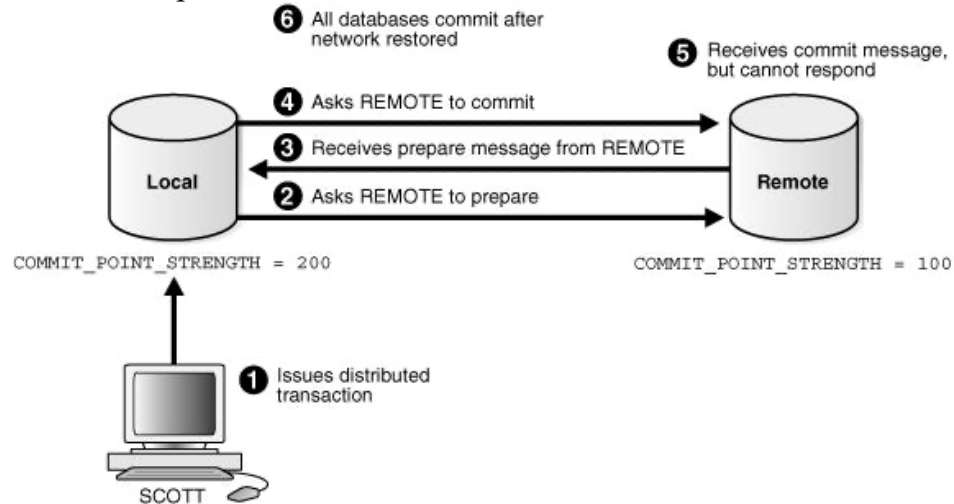


Figure 4.1.6: Failure During the Commit Phase

The following steps occur:

User Scott connects to local and executes a distributed transaction. The global coordinator, which in this case is also the commit point site, requests all databases other than the commit point site to promise to commit or roll back when told to do so.

3. The commit point site receives a prepared message from remote saying that it will commit.

The commit point site commits the transaction locally, then sends a commit message to remote asking it to commit.

The remote database receives the commit message, but cannot respond because of a network failure.

The transaction is ultimately committed on the remote database by the RECO process after the network is restored.

### 3.13 Manual Resolution of In-Doubt Transactions

You should only need to resolve an in-doubt transaction in the following cases:

The in-doubt transaction has locks on critical data or undo segments.

The cause of the machine, network, or software failure cannot be repaired quickly.

Resolution of in-doubt transactions can be complicated. The procedure requires that you do the following:

Identify the transaction identification number for the in-doubt transaction.

**Query**

the `DBA_2PC_PENDING` and `DBA_2PC_NEIGHBORS` views to determine whether the databases involved in the transaction have committed.

If necessary, force a commit using the `COMMIT FORCE` statement or a rollback using the `ROLLBACK FORCE` statement.

### 3.14 Relevance of System Change Numbers for In-Doubt Transactions

A **system change number (SCN)** is an internal timestamp for a committed version of the database. The Oracle Database server uses the SCN clock value to guarantee transaction consistency. For example, when a user commits a transaction, the database records an SCN for this commit in the redo log.

The database uses SCNs to coordinate distributed transactions among different databases. For example, the database uses SCNs in the following way:

An application establishes a connection using a database link.

The distributed transaction commits with the highest global SCN among all the databases involved.

The commit global SCN is sent to all databases involved in the transaction.

SCNs are important for distributed transactions because they function as a synchronized commit timestamp of a transaction, even if the transaction fails. If a transaction becomes in-doubt, an administrator can use this SCN to coordinate changes made to the global database. The global SCN for the transaction commit can also be used to identify the transaction later, for example, in distributed recovery.

**Discussion**

How do you manually resolve an In-Doubt Transactions? Discuss.

## 4.0 SELF-ASSESSMENT/EXERCISES

Explain the sequence of events when there is a failure during the commit phase of a distributed transaction

**Answer**

The following steps occur:

User `Scott` connects to `local` and executes a distributed transaction.

The global coordinator, which in this case is also the commit point site, requests all databases other than the commit point site to promise to commit or roll back when told to do so.



- (iii) The commit point site receives a prepared message from remote saying that it will commit. The commit point site commits the transaction locally, then sends a commit message to remote asking it to commit. The remote database receives the commit message, but cannot respond because of a network failure. The transaction is ultimately committed on the remote database by the RECO process after the network is restored.

In what ways can the Distributed transactions become in-doubt ?

Answer:

A server machine running Oracle Database software crashes

A network connection between two or more Oracle Databases involved in distributed processing is disconnected

An unhandled software error occurs

## 5.0 CONCLUSION

Databases are standard transactional resources, and transactions usually extend to a small number of such databases. In such cases, a distributed transaction may be viewed as a database transaction that should be synchronized between various participating databases allocated between various physical locations. The isolation property presents a unique obstacle for multi-database transactions.

For distributed transactions, each computer features a local transaction manager. If the transaction works at several computers, the transaction managers communicate with various other transaction managers by means of superior or subordinate relationships, which are accurate only for a specific transaction.

## 6.0 SUMMARY

A **distributed transaction** includes one or more statements that, individually or as a group, update data on two or more distinct nodes of a distributed database. Two Types of Permissible Operations in Distributed Transactions are DML and DDL Transactions & Transaction Control Statements. The database ensures the integrity of data in a distributed transaction using the two-phase commit mechanism: the prepare phase and the commit phase. Distributed transactions can become in-doubt in the following ways: either a server machine running Oracle Database software crashes, a network connection between two or more Oracle Databases involved in distributed processing is disconnected or an unhandled software error occurs. When a node cannot successfully prepare, it performs the following actions:

Releases resources currently held by the transaction and rolls back the local portion of the transaction.

Responds to the node that referenced it in the distributed transaction with an abort message.

## **7.0 REFERENCES/FURTHER READING**

[https://en.wikipedia.org/wiki/Distributed\\_transaction](https://en.wikipedia.org/wiki/Distributed_transaction)

<https://www.techopedia.com/definition/29166/distributed-transaction>

## UNIT 2 FLAT AND NESTED DISTRIBUTED TRANSACTIONS

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Flat & Nested Distributed Transactions
  - 3.2 Transactions Commands
  - 3.3 Roles for Running a Transactions Successfully
  - 3.4 Flat & Nested Distributed Transactions
    - 3.4.1 Flat Transactions
      - 3.4.1.1 Limitations of a Flat Transactions
    - 3.4.2 Nested Transactions
      - 3.4.2.1 Advantage
  - 3.5 Role
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

Transactions are inevitable in the smooth running of 21<sup>st</sup> century business life. Almost all daily activities run on network transactions. Applications run processes that have one or more threads that need to be synchronized, in a multithreading environments, enables a concurrency.

### 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, the student will able to:

- Differentiate between the Flat and Nested Transactions
- State the advantage of Nested over Flat transactions
- Learn from existing scenarios of cybercrimes in India

### 3.0 MAIN CONTENT

#### 3.1 Flat & Nested Distributed Transactions

A transaction is a series of object operations that must be done in an ACID-compliant manner. ACID connotes:

Atomicity – The transaction is completed entirely or not at all.

Consistency – It is a term that refers to the transition from one consistent state to another.

Isolation – It is carried out separately from other transactions.

Durability – Once completed, it is long lasting.

### 3.2 **Transactions Commands:**

**Begin** – initiate a new transaction.

**Commit** – End a transaction and the changes made during the transaction are saved. Also, it allows other transactions to see the modifications you've made.

**Abort** – End a transaction and all changes made during the transaction will be undone.

### 3.3 **Roles for Running a Transaction Successfully:**

**Client** – The transactions are issued by the clients.

**Coordinator** – The execution of the entire transaction is controlled by it (handles Begin, commit & abort).

**Server** – Every component that accesses or modifies a resource is subject to transaction control. The coordinator must be known by the transactional server. The transactional server registers its participation in a transaction with the coordinator.

A flat or nested transaction that accesses objects handled by different servers is referred to as a distributed transaction. When a distributed transaction reaches its end, in order to maintain the atomicity property of the transaction, it is mandatory that all of the servers involved in the transaction either commit the transaction or abort it.

To do this, one of the servers takes on the job of coordinator, which entails ensuring that the same outcome is achieved across all servers. The method by which the coordinator accomplishes this is determined by the protocol selected. The most widely used protocol is the 'two-phase commit protocol.' This protocol enables the servers to communicate with one another in order to come to a joint decision on whether to commit or abort the complete transaction.

### 3.4 **Flat & Nested Distributed Transactions**

If a client transaction calls actions on multiple servers, it is said to be distributed. Distributed transactions can be structured in two different ways:

Flat transactions

Nested transactions

### 3.4.1 Flat Transactions:

A flat transaction has a single initiating point (Begin) and a single end point (Commit or abort). They are usually very simple and are generally used for short activities rather than larger ones. A client makes requests to multiple servers in a flat transaction. Transaction T, for example, is a flat transaction that performs operations on objects in servers X, Y, and Z.

Before moving on to the next request, a flat client transaction completes the previous one. As a result, each transaction visits the server object in order. A transaction can only wait for one object at a time when servers utilize locking.

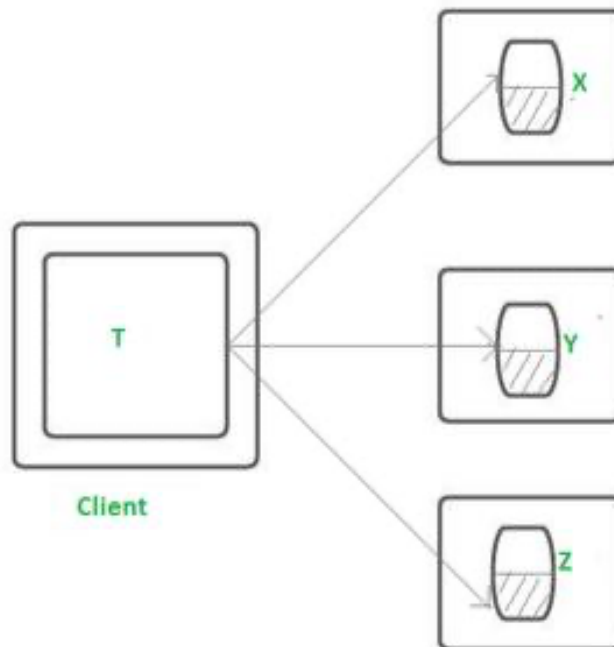


Figure 4.2.1: Flat Transactions

#### 3.4.1.1 Limitations of a Flat Transaction:

- All work is lost in the event of a crash.
- Only one DBMS may be used at a time.
- No partial rollback is possible.

### 3.4.2 Nested Transactions

A transaction that includes other transactions within its initiating point and at the end point are known as nested transactions. So the nesting of the transactions is done in a transaction. The nested transactions here are called sub-transactions. The top-level transaction in a nested transaction

can open sub-transactions, and each sub-transaction can open more sub-transactions down to any depth of nesting. A client's transaction T opens up two sub-transactions, T1 and T2, which access objects on servers X and Y, as shown in the figure 4.2.2 below. T1.1, T1.2, T2.1, and T2.2, which access the objects on the servers M, N and P are opened by the sub-transactions T1 and T2.

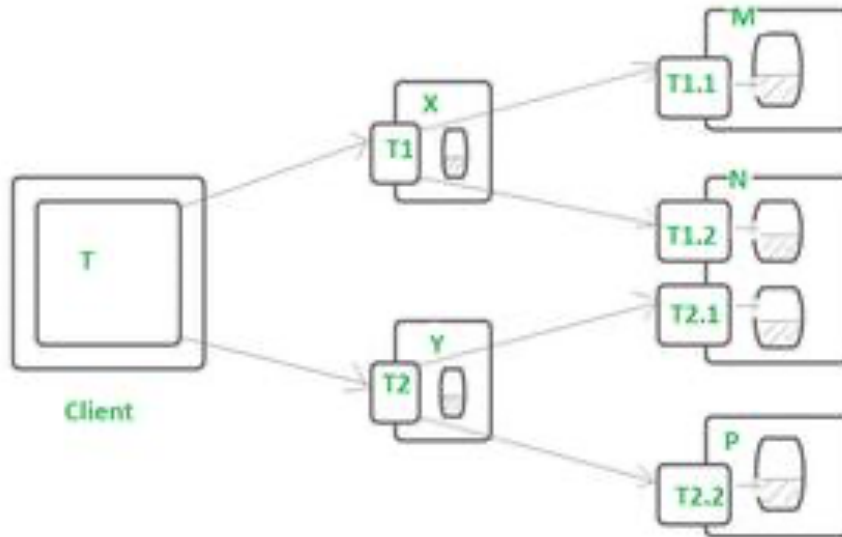


Figure 4.2.2: Nested Transactions

Concurrent Execution of the Sub-transactions is done which are at the same level – in the nested transaction strategy. Here, in the above diagram, T1 and T2 invoke objects on different servers and hence they can run in parallel and are therefore concurrent. T1.1, T1.2, T2.1, and T2.2 are four sub-transactions. These sub-transactions can also run in parallel.

Consider a distributed transaction (T) in which a customer transfers: \$105 from account A to account C and Subsequently, \$205 from account B to account D.

It can be viewed/ thought of as:

Transaction T:

Start

Transfer \$105 from A to C:

Deduct \$105 from A (withdraw from A) & Add \$105 to C (deposit to C)

Transfer \$205 from B to D:

Deduct \$205 from B (withdraw from B) & Add \$205 to D (deposit to

D) End

**Assuming that:**

Account A is on server X

Account B is on server Y, and

Accounts C and D are on server Z.

The transaction T involves four requests – 2 for deposits and 2 for withdrawals. Now they can be treated as sub-transactions (T1, T2, T3, T4) of the transaction T.

As shown in the figure 4.2.3 below, transaction T is designed as a set of four nested transactions: T1, T2, T3 and T4.

### 3.4.2.1 Advantage of Nested Transactions:

The performance is higher than a single transaction in which four operations are invoked one after the other in sequence.

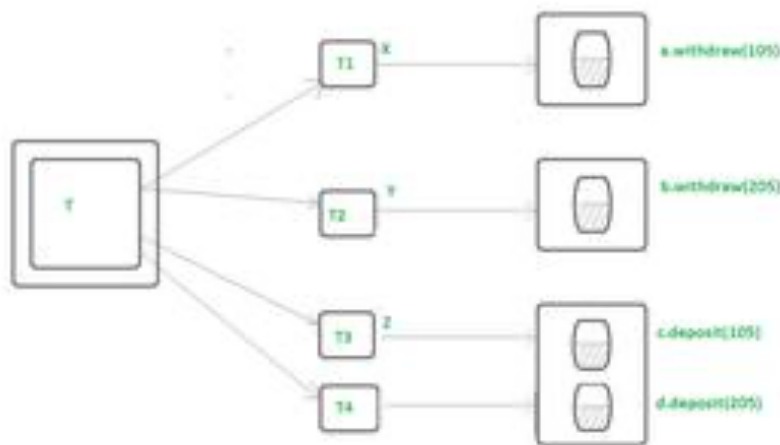


Figure 4.2.2: Nested Transactions

So, the Transaction T may be divided into sub-transactions as:

```
//Start the Transaction
T = open transaction
//T1
openSubtransaction
    a.withdraw(105);
//T2
openSubtransaction
    b.withdraw(205);
//T3
openSubtransaction
    c.deposit(105);
//T4
openSubtransaction
    d.deposit(205);
//End the transaction
close Transaction
```

### 3.5 Role of coordinator

When the Distributed Transaction commits, the servers that are involved in the transaction execution, for proper coordination, must be able to communicate with one another.

When a client initiates a transaction, an “openTransaction” request is sent to any coordinator server. The contacted coordinator carries out the “openTransaction” and returns the transaction identifier to the client. Distributed transaction identifiers must be unique within the distributed system. A simple way is to generate a TID contains two parts – the ‘server identifier’ (example :IP address) of the server that created it and a number unique to the server.

The coordinator who initiated the transaction becomes the distributed transaction’s coordinator and has the responsibility of either aborting it or committing it.

Every server that manages an object accessed by a transaction is a participant in the transaction & provides an object we call the *participant*. The participants are responsible for working together with the coordinator to complete the commit process.

The coordinator every time, records the new participant in the participants list. Each participant knows the coordinator & the coordinator knows all the participants. This enables them to collect the information that will be needed at the time of commit and hence work in coordination.

#### Discussion

Discuss Roles for Running a Transaction Successfully.

### 4.0 SELF-ASSESSMENT/EXERCISES

Explain the role of a Corrdinator

Answer:

When the Distributed Transaction commits, the servers that are involved in the transaction execution, for proper coordination, must be able to communicate with one another.

When a client initiates a transaction, an “openTransaction” request is sent to any coordinator server. The contacted coordinator carries out the “openTransaction” and returns the transaction identifier to the client.



Explain Flat Transactions with the aid of a diagram

Answer:

A flat transaction has a single initiating point (Begin) and a single end point (Commit or abort). They are usually very simple and are generally used for short activities rather than larger ones. A client makes requests to multiple servers in a flat transaction. Transaction T, for example, is a flat transaction that performs operations on objects in servers X, Y, and Z.

Before moving on to the next request, a flat client transaction completes the previous one. As a result, each transaction visits the server object in order. A transaction can only wait for one object at a time when servers utilize locking.

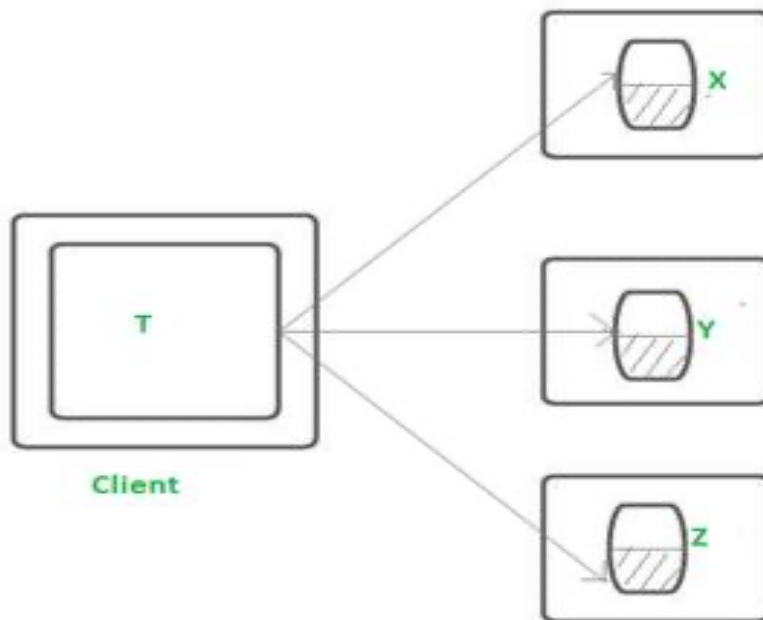


Figure 4.2.1: Flat Transactions

## 5.0 CONCLUSION

A flat transaction has a single initiating point (Begin) and a single end point (Commit or abort). They are usually very simple and are generally used for short activities rather than larger ones. The performance of Nested Transactions is higher than a single transaction in which four operations are invoked one after the other in sequence

## 6.0 SUMMARY

A transaction is a series of object operations that must be done in an ACID (Atomicity, Consistency, Isolation) -compliant manner.

Transactions Commands include Begin, Commit and Abort. Roles for Running a Transaction Successfully include Client, Coordinator and Server. Limitations of a Flat Transaction are that, all work is lost in the event of a crash, only one DBMS may be used at a time and that no partial rollback is possible. Nested Transactions is a transaction that includes other transactions within its initiating point and at the end point. The performance is higher in nested transactions than in a single transaction.

## **7.0 REFERENCES/FURTHER READING**

[Flat & Nested Distributed Transactions - GeeksforGeeks](#)

## UNIT 3 CONCURRENCY

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Concurrency
  - 3.2 Two Models for Concurrent Programming
    - 3.2.1 Shared Memory
    - 3.2.2 Message Passing
  - 3.3 Processes, Threads & Time-Slicing
    - 3.3.1 Process
    - 3.3.2 Thread
    - 3.3.3 Time Slicing
  - 3.4 Shared Memory Example
    - 3.4.1 Interleaving
    - 3.4.2 Race Condition
    - 3.4.3 Reordering
  - 3.5 Message Passing Example
  - 3.6 Concurrent is Hard to Test and Debug
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

Cybercrime is "international" that there are ‘no cyber-borders between countries’ The complexity in types and forms of cybercrime increases the difficulty to fight back, fighting cybercrime calls for international cooperation . Various organizations and governments have already made joint efforts in establishing global standards of legislation and law enforcement both on a regional and on an international scale.

### 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, student will able to:

- Explain the concepts, Concurrency, shared memory
- Identify and explain two models for concurrent programming
  - Describe the following terms: interleaving, reordering, concurrency, race condition and time-slicing.

## 3.0 MAIN CONTENT

### 3.1 Concurrency

Concurrency means multiple computations are happening at the same time. Concurrency is everywhere in modern programming, whether we like it or not:

- Multiple computers in a network

- Multiple applications running on one computer

- Multiple processors in a computer (today, often multiple processor cores on a single chip)

In fact, concurrency is essential in modern programming:

Web sites must handle multiple simultaneous users.

- Mobile apps need to do some of their processing on servers (“in the cloud”).

- Graphical user interfaces almost always require background work that does not interrupt the user. For example, Eclipse compiles your Java code while you’re still editing it.

Being able to program with concurrency will still be important in the future. Processor clock speeds are no longer increasing. Instead, we are getting more cores with each new generation of chips. So in the future, in order to get a computation to run faster, we’ll have to split up a computation into concurrent pieces.

### 3.2 Two Models for Concurrent Programming

There are two common models for concurrent programming:

- Shared memory and

- Message passing.

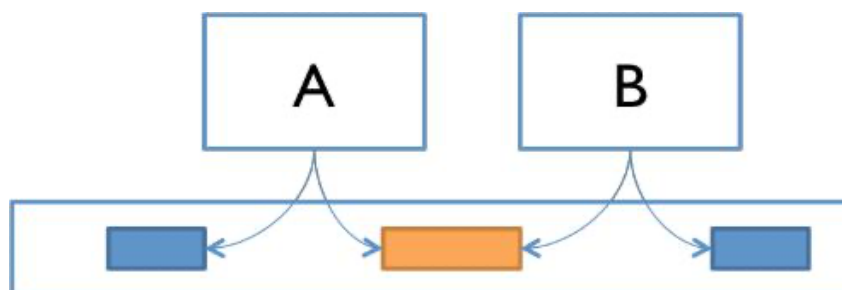


Figure 4.3.1: Shared Memory

### 3.2.1 Shared memory

In the shared memory model of concurrency, concurrent modules interact by reading and writing shared objects in memory. Other examples of the shared-memory model:

A and B might be two processors (or processor cores) in the same computer, sharing the same physical memory.

A and B might be two programs running on the same computer, sharing a common filesystem with files they can read and write.

A and B might be two threads in the same Java program (we will explain what a thread is below), sharing the same Java objects.

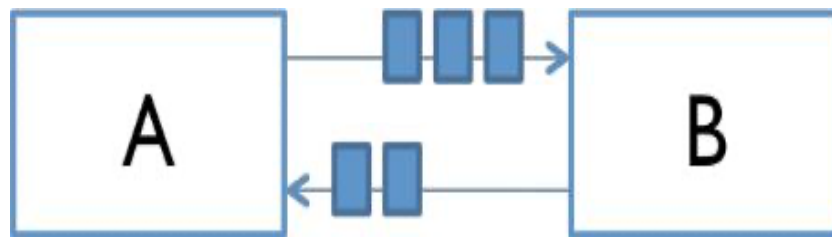


Figure 3.3.2: An example of shared memory

### 3.2.2 Message Passing

In the message-passing model, concurrent modules interact by sending messages to each other through a communication channel. Modules send off messages, and incoming messages to each module are queued up for handling. Examples include:

A and B might be two computers in a network, communicating by network connections.

A and B might be a web browser and a web server – A opens a connection to B, asks for a web page, and B sends the web page data back to A.

A and B might be an instant messaging client and server.

A and B might be two programs running on the same computer whose input and output have been connected by a pipe, like `ls | grep` typed into a command prompt.

## 3.3 Processes, Threads, Time-slicing

The message-passing and shared-memory models are about how concurrent modules communicate. The concurrent modules themselves come in two different kinds: processes and threads.

### 3.3.1 Process

A process is an instance of a running program that is isolated from other processes on the same machine. In particular, it has its own private section of the machine's memory.

The process abstraction is a virtual computer. It makes the program feel like it has the entire machine to itself – like a fresh computer has been created, with fresh memory, just to run that program.

Just like computers connected across a network, processes normally share no memory between them. A process can't access another process's memory or objects at all. Sharing memory between processes is *possible* on most operating system, but it needs special effort. By contrast, a new process is automatically ready for message passing, because it is created with standard input

output streams, which are the System.out and System.in streams you've used in Java.

### 3.3.2 Thread

A thread is a locus of control inside a running program. Think of it as a place in the program that is being run, plus the stack of method calls that led to that place to which it will be necessary to return through.

Just as a process represents a virtual computer, the thread abstraction represents a *virtual processor*. Making a new thread simulates making a fresh processor inside the virtual computer represented by the process. This new virtual processor runs the same program and shares the same memory as other threads in process.

Threads are automatically ready for shared memory, because threads share all the memory in the process. It needs special effort to get "thread-local" memory that's private to a single thread. It's also necessary to set up message-passing explicitly, by creating and using queue data structures. We will talk about how to do that in a future reading.

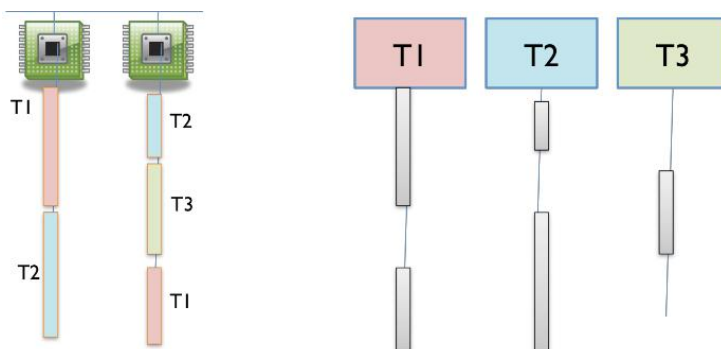


Figure 4.3.3: Many Concurrent Threads, One or Two Processors

### 3.3.3 Time Slicing

When there are more threads than processors, concurrency is simulated by **time slicing**, which means that the processor switches between threads. The figure on the right, above, shows

how three threads T1, T2, and T3 might be time-sliced on a machine that has only two actual processors. In the figure 2, time proceeds downward, so at first one processor is running thread T1 and the other is running thread T2, and then the second processor switches to run thread T3. Thread T2 simply pauses, until its next time slice on the same processor or another processor.

On most systems, time slicing happens unpredictably and non-deterministically, meaning that a thread may be paused or resumed at any time.

### 3.4 Shared Memory Example

Let's look at an example of a shared memory system. The point of this example is to show that concurrent programming is hard, because it can have subtle bugs.

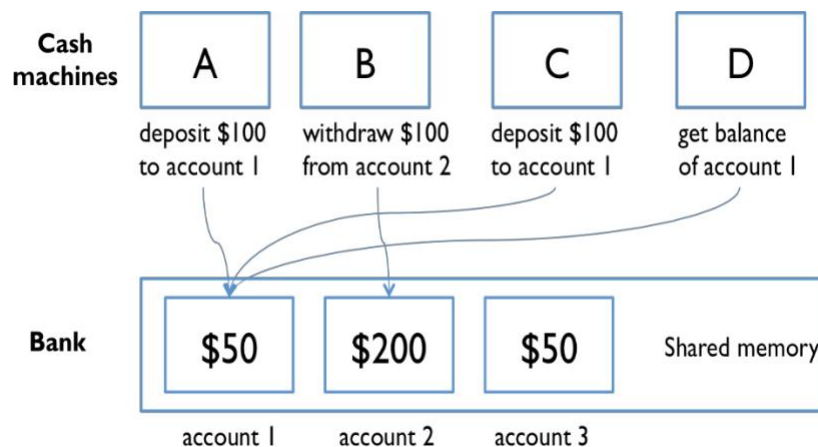


Figure 4.3.4: Shared memory system showing that concurrent programming is hard

Imagine that a bank has cash machines that use a shared memory model, so all the cash machines can read and write the same account objects in memory. To illustrate what can go wrong, let's simplify the bank down to a single account, with a dollar balance stored in the balance variable, and two operations deposit and withdraw that simply add or remove a dollar:

```

suppose all the cash machines share a single bank account
private static int balance = 0;

private static void deposit() {
    balance = balance + 1;
}
private static void withdraw() {
    balance = balance - 1;
}

```

Customers use the cash machines to do transactions like this:

```
deposit(); // put a dollar in
withdraw(); // take it back out
```

In this simple example, every transaction is just a one dollar deposit followed by a one-dollar withdrawal, so it should leave the balance in the account unchanged. Throughout the day, each cash machine in our network is processing a sequence of deposit/withdraw transactions.

```
each ATM does a bunch of transactions that
modify balance, but leave it unchanged afterward
private static void cashMachine() {
    for (int i = 0; i < TRANSACTIONS_PER_MACHINE; ++i)
        { deposit(); // put a dollar in
          withdraw(); // take it back out
        }
}
```

So at the end of the day, regardless of how many cash machines were running, or how many transactions we processed, we should expect the account balance to still be 0.

But if we run this code, we discover frequently that the balance at the end of the day is *not* 0. If more than one `cashMachine()` call is running at the same time – say, on separate processors in the same computer – then balance may not be zero at the end of the day.

### 3.4.1 Interleaving

Here is one thing that can happen. Suppose two cash machines, A and B, are both working on a deposit at the same time. Here is how the `deposit()` step typically breaks down into low-level processor instructions:

```
get balance (balance=0)
add 1
write back the result (balance=1)
```

When A and B are running concurrently, these low-level instructions interleave with each other (some might even be simultaneous in some sense, but let's just worry about interleaving for now):

```
A get balance (balance=0)
A add 1
A write back the result (balance=1)
```



```

B get balance (balance=1)
B add 1
B write back the result (balance=2)

```

This interleaving is fine – we end up with balance 2, so both A and B successfully put in a dollar. But what if the interleaving looked like this:

```

A get balance (balance=0)
      B get balance (balance=0)
A add 1
      B add 1
A write back the result (balance=1)
      B write back the result (balance=1)

```

The balance is now 1 – A’s dollar was lost! A and B both read the balance at the same time, computed separate final balances, and then raced to store back the new balance – which failed to take the other’s deposit into account.

### 3.4.2 Race Condition

A **race condition** means that the correctness of the program (the satisfaction of postconditions and invariants) depends on the relative timing of events in concurrent computations A and B. When this happens, we say “A is in a race with B.”

Some interleavings of events may be OK, in the sense that they are consistent with what a single, nonconcurrent process would produce, but other interleavings produce wrong answers – violating postconditions or invariants.

All these versions of the bank-account code exhibit the same race condition:

```

// version 1
private static void deposit() {
    balance = balance + 1;
}
private static void withdraw() {
    balance = balance - 1;
}
// version 2
private static void deposit() {
    balance += 1;
}

```

```

private static void withdraw() {
    balance -= 1;
}
// version 3
private static void deposit() {
    ++balance;
}
private static void withdraw() {
    --balance;
}

```

You cannot tell just from looking at Java code how the processor is going to execute it. You can't tell what the indivisible operations – the atomic operations – will be. It isn't atomic just because it's one line of Java. It doesn't touch `balance` only once just because the `balance` identifier occurs only once in the line. The Java compiler, and in fact the processor itself, makes no commitments about what low-level operations it will generate from your code. In fact, a typical modern Java compiler produces exactly the same code for all three of these versions!

The key lesson is that you cannot tell by looking at an expression whether it will be safe from race conditions.

### 3.4.3 Reordering

The race condition on the bank account balance can be explained in terms of different interleavings of sequential operations on different processors. But in fact, when you are using multiple variables and multiple processors, you cannot even count on changes to those variables appearing in the same order.

Here's an example:

```

private boolean ready = false;
private int answer = 0;

    computeAnswer runs in one thread
private void computeAnswer() {
    answer = 42;
    ready = true;
}

    useAnswer runs in a different thread
private void useAnswer() {
    while (!ready) {
        Thread.yield();
    }
}

```

```

    }
    if (answer == 0) throw new RuntimeException("answer wasn't
ready!");
}

```

We have two methods that are being run in different threads. `computeAnswer` does a long calculation, finally coming up with the answer 42, which it puts in the `answer` variable. Then it sets the **ready** variable to true, in order to signal to the method running in the other thread, `useAnswer`, that the answer is ready for it to use. Looking at the code, **answer** is set before **ready** is set, so once `useAnswer` sees **ready** as true, then it seems reasonable that it can assume that the **answer** will be 42 but quite not true.

The problem is that modern compilers and processors do a lot of things to make the code fast. One of those things is making temporary copies of variables like `answer` and `ready` in faster storage (registers or caches on a processor), and working with them temporarily before eventually storing them back to their official location in memory. The storeback may occur in a different order than the variables were manipulated in your code. Here is what might be going on under the covers (but expressed in Java syntax to make it clear). The processor is effectively creating two temporary variables, `tmpr` and `tmpa`, to manipulate the fields **ready** and **answer**:

```

private void computeAnswer() {
    boolean tmpr = ready;
    int tmpa = answer;

    tmpa = 42;
    tmpr = true;

    ready = tmpr;
    <-- what happens if useAnswer() interleaves here?
    ready is set, but answer isn't.
    answer = tmpa;
}

```

### 3.5 Message Passing Example

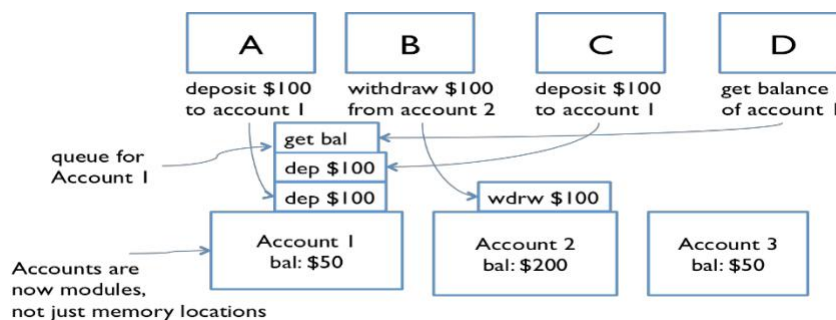


Figure 4.3.5: A message passing example

Now let us look at the message-passing approach to our bank account example.

Now not only are the **cash machine** modules, but the **accounts** are modules, too. Modules interact by sending messages to each other. Incoming requests are placed in a queue to be handled one at a time. The sender does not stop working while waiting for an answer to its request. It handles more requests from its own queue. The reply to its request eventually comes back as another message.

Unfortunately, message passing does not eliminate the possibility of race conditions. Suppose each **account** supports `get-balance` and `withdraw` operations, with corresponding messages. Two users, at **cash machine** A and B, are both trying to withdraw a dollar from the same account. They check the balance first to make sure they never withdraw more than the account holds, because overdrafts trigger big bank penalties:

```
get-balance
if balance >= 1 then withdraw 1
```

The problem is again interleaving, but this time interleaving of the *messages* sent to the bank **account**, rather than the *instructions* executed by A and B. If the **account** starts with a dollar in it, then what interleaving of messages will fool A and B into thinking they can both withdraw a dollar, thereby overdrawing the account?

One lesson here is that you need to carefully choose the operations of a message-passing model. **withdraw-if-sufficient-funds** would be a better operation than just **withdraw**.

### 3.6 Concurrency is Hard to Test and Debug

If we have not persuaded you that concurrency is tricky, here is the worst of it. It is very hard to discover race conditions using testing. And even once a test has found a bug, it may be very hard to localize it to the part of the program causing it.

Concurrency bugs exhibit very poor reproducibility. It is hard to make them happen the same way twice. Interleaving of instructions or messages depends on the relative timing of events that are strongly influenced by the environment. Delays can be caused by other running programs, other network traffic, operating system scheduling decisions, variations in processor clock speed, etc. Each time you run a program containing a race condition, you may get different behavior.

These kinds of bugs are **heisenbugs**, which are nondeterministic and hard to reproduce, as opposed to a “**bohrbug**”, which shows up repeatedly whenever you look at it. Almost all bugs in sequential programming are **bohrbugs**.

A heisenbug may even disappear when you try to look at it with **println** or **debugger**! The reason is that printing and debugging are so much slower than other operations, often 100-1000x slower, that they dramatically change the timing of operations, and the interleaving. So inserting a simple print statement into the `cashMachine()`:

```
private static void cashMachine() {
    for (int i = 0; i < TRANSACTIONS_PER_MACHINE; ++i)
        { deposit(); // put a dollar in
          withdraw(); // take it back out
          System.out.println(balance); // makes the bug disappear!
        }
}
```

...and suddenly the balance is always 0, as desired, and the bug appears to disappear. But it is only masked, not truly fixed. A change in timing somewhere else in the program may suddenly make the bug come back. Concurrency is hard to get right. Part of the point of this reading is to scare you a bit. Over the next several readings, we'll see principled ways to design concurrent programs so that they are safer from these kinds of bugs.

### Discussion

What are **heisenbugs** and **bohrbug** bugs. Discuss.

## 5.0 CONCLUSION

When there are more threads than processors, concurrency is simulated by time slicing, which means that the processor switches between threads. Multithreading abounds in all enterprise developments.

## 6.0 SUMMARY

Concurrency means multiple computations are happening at the same time. Concurrency is everywhere in modern programming. In the shared memory model of concurrency, concurrent modules interact by reading and writing shared objects in memory. In the message-passing model, concurrent modules interact by sending messages to each other through a communication channel. A **race condition** means that the correctness of the program (the satisfaction of postconditions and invariants) depends on the relative timing of events in concurrent computations A and B. Concurrency bugs exhibit very poor reproducibility. It is hard to make

them happen the same way twice. Interleaving of instructions or messages depends on the relative timing of events that are strongly influenced by the environment.

## **7.0 REFERENCES/FURTHER READING**

[Concepts: Concurrency \(uhcl.edu\)](http://uhcl.edu)

[Processes and Threads \(The Java™ Tutorials > Essential Java Classes > Concurrency\) \(oracle.com\)](#)

## **UNIT 4 CHARACTERISTICS OF SERVICE ORIENTED ARCHITECTURE**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Service-Oriented Architecture (SOA)
    - 3.1.1 A Service
  - 3.2 An Example: SOA Apps Provide a Cohesive Platform for Overstock.com (a large Online Retailer)
  - 3.3 The 6 Defining Concepts of SOA
  - 3.4 Understanding SOA: The Transportation Analogy
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

Service-oriented architecture (SOA) is a software development model that allows services to communicate across different platforms and languages to form applications. In SOA, a service is a self-contained unit of software designed to complete a specific task. Service-oriented architecture allows various services to communicate using a loose coupling system to either pass data or coordinate an activity.

### **2.0 Intended Learning Outcomes (ILOs)**

At the end of this unit, students will able to:

- state the characteristics of Service Oriented Architecture (SOA).
- identify the 6 Defining Concepts of SOA.
- employ the understanding of SOA to solve future problems.

### **3.0 MAIN CONTENT**

#### **3.1 Service-Oriented Architecture (SOA)**

Service-Oriented Architecture, can be defined as "services" that provide a platform by which disparate systems can communicate with each other. These services are essentially groups of software components that help a company seamlessly carry out important business processes. SOA

implementation makes interoperability between heterogeneous applications and technologies possible.

The rise of SOA technology and integration in recent years is placing it as one of the most important applications for communicating between different systems — or in this context, **services**.

### 3.1.1 A Service

Services represent building blocks that allow users to organize information in ways that are familiar to them. These building blocks combine information about users and their behavior in a seamless fashion to present a relatively simple interface.

A **service** is commonly characterized by these four properties:

- It logically represents a business activity with a specified outcome.

- It is self-contained

- It is a black box for its consumers

- It may consist of other underlying services

T

o further simplify this concept, an SOA service is the mechanism that satisfies a customer's wants or needs through a negotiated contract. Therefore, **SOA is a collection of different services**.

To better understand what service-oriented architecture is all about, consider this quote from industry expert David Sprott:

### 3.2 An Example: SOA Apps Provide a Cohesive Platform for Overstock.com (a large Online Retailer)

Communication of services can involve something as simple as passing data, or it can involve a coordination of an activity between two or more different SOA services.

One way to illustrate the SOA method is by taking a look at a large online retailer like Overstock.com.

In order for Overstock customers to make a transaction, different programs must work together seamlessly. The various steps in the ordering process can involve various programs developed at different times, each using their own unique platforms and technologies.

For instance, there might be one program that tracks inventory, which is different than the interface (i.e. the Internet) the customer uses to shop. Then, there is likely an entirely different program for their shopping cart and another for processing payment.



SOA services tie all of these various programs together so that an online shopper can quickly find out if what they are looking for is in stock and get it shipped to their doorstep with just a few clicks of their mouse.

### 3.3 The 6 Defining Concepts of SOA

In October of 2009, a manifesto was created about service-oriented architecture. This manifesto states that there are six core values of SOA:

Business value is more important than technical strategy.

Strategic goals are more valuable than project-specific benefits.

Intrinsic interoperability is greater than custom integration.

Shared services over specific-purpose implementations.

Flexibility is given more importance than optimization.

Evolutionary refinement is more important than pursuit of initial perfection.

### 3.4 Understanding SOA: The Transportation Analogy

Another way to think about SOA is through the analogy of transportation. Imagine that you have to travel from your home in Ibadan to a business conference or trade show in Kano. What are the various steps you might take to get there?

First, you will have to drive to the airport, then take a shuttle to the airport terminal. Next, you will board the plane for Kano. After landing, you take another shuttle from the gate to the main terminal, where you have to flag down a taxi or call an Uber to drive you to your hotel. When it is time for the conference to start, you walk to the nearest train stop, hop on, and ride it to the conference center.

All of these various transportation methods worked together to accomplish your end goal of attending the conference — your car, the shuttle bus, airplane, train, and even walking. There were many individual “steps” you had to take to arrive at your final destination on time, and there were likely other ways you could have gone about it.

For instance, instead of driving to the airport, you could have walked to a train station or bus stop and gotten to the airport this way. Or you could have driven completely across the country, thus eliminating your need for any other type of transportation altogether.

However, by combining numerous transportation methods you were able to get to the conference faster and probably cheaper than if you had driven the whole way.

In this analogy of SOA, the various modes of transportation can be viewed as the different “**services**” used to reach an end goal. Just like the cars,

bus, train, and plane all worked together to help you accomplish your goal of attending the conference, combining different units of software applications (services) can help business achieve new milestones in the most efficient manner.

## Discussion

What is Service Oriented Architecture?

## 4.0 SELF-ASSESSMENT/EXERCISES

### Define Services

#### Answer:

Services represent building blocks that allow users to organize information in ways that are familiar to them. These building blocks combine information about users and their behavior in a seamless fashion to present a relatively simple interface.

What are the four properties a **service** is commonly characterized by:

Answer:

It logically represents a business activity with a specified outcome.

It is self-contained

It is a black box for its consumers

It may consist of other underlying services

Explain the concept of SOA using another analogy aside of the Transportation Analogy.

## 5.0 CONCLUSION

In life we realize our different goals via combination of methods. Applying this principle in deploying solutions and even researches enables novelty and the benefits are much more than losses.

## 6.0 SUMMARY

Service-Oriented Architecture (SOA), can be defined as "services" that provide a platform by which disparate systems can communicate with each other. A **Service** represents the building blocks that allow users to organize information in ways that are familiar to them. A **service** is commonly characterized by these four properties: It logically represents a business activity with a specified outcome, it is self-contained, It is a black box for its consumers and It may consist of other underlying services.

## MODULE 5 MOBILE & CLOUD COMPUTING

### UNIT 1 INTRODUCTION TO MOBILE & CLOUD COMPUTING

#### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Mobile and Cloud Computing
  - 3.2 Cloud Computing
  - 3.3 Capabilities of Cloud Computing
  - 3.4 Categories of Cloud Computing Models
    - 3.4.1 Software as a Service (SaaS)
    - 3.4.2 Platform as a Service (PaaS)
    - 3.4.3 Infrastructure as a Service (IaaS)
  - 3.5 Mobile Cloud Computing (MCC)
    - 3.5.1 Advantage of Mobile & Cloud Computing
    - 3.5.2 Disadvantages of Mobile & Cloud Computing
  - 3.6 Mobile & Cloud Computing Security Concerns
  - 3.7 The Top Threats in the Usage of Mobile & Cloud Computing
    - 3.7.1 Data Loss
    - 3.7.2 Untrusted Service Providers
    - 3.7.3 Insecure API
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

#### Introduction to Module

Mobile cloud computing (MCC) is the method of using cloud technology to deliver mobile apps. Complex mobile apps today perform tasks such as authentication, location-aware functions, and providing targeted content and communication for end users. Hence, they require extensive computational resources such as data storage capacity, memory, and processing power. Mobile cloud computing takes the pressure off mobile devices by harnessing the power of cloud infrastructure. Developers build and update rich mobile apps using cloud services and then deploy them for remote access from any device. These cloud-based mobile apps use cloud technology to store and process data so that the app is usable on all types of old and new mobile devices.

Unit 1: Introduction to Mobile & Cloud Computing

Unit 2: Technologies for Wireless Communications

Unit 3: Wireless Cellular Systems

Unit 4: Characteristics of Service Oriented Architecture

## **1.0 INTRODUCTION**

Cloud Computing seems to be the most promising technology of the century we are living. It provides a new manner of sharing distributed resources and services that may be part of different organizations, geographically located in different places and different time zones. Mobile Cloud Computing offers partially the same functionality, with the only additional requirement that, at least, some of the devices are mobile. In this paper, we will try to provide a detailed explanation of Mobile Cloud Computing concept by providing different examples, figures, accessibility, pros and cons and comparison.

## **2.0 INTENDED LEARNING OUTCOMES (ILOS)**

At the end of this unit, student will able to:

- Explain the concept of Cloud computing and its capabilities
- Identify and explain the categories of Cloud computing
- Highlight and explain the top threats in the usage of Mobile and Cloud Computing

## **3.0 MAIN CONTENT**

### **3.1 Mobile & Cloud Computing**

### **3.2 Cloud Computing**

Cloud Computing is the delivery of the computing services, such as servers, databases, storage and networking – over the Internet. These services, usually are offered by so called Cloud Providers, that usually charge based on usage.

Nowadays, everyone that is using a device connected to Internet, might be user of cloud services, even though we might not be aware of it. Almost every online service, including email, document editors or entertaining apps, might be running using cloud services.



Figure 15: Cloud Computing

### 3.3 Capabilities of Cloud Computing

Generally, these are a few of the capabilities of Cloud Computing:

- Create new apps and services
- Store, back up and recover data
- Deliver software
- Analyse data for pattern recognition
- Streaming.

Besides the capabilities that Cloud Computing provides, there are also a lot of benefits that it can offer.

**Cost** – using cloud services lowers the costs that organizations need to spend for buying hardware and software tools for setting up the infrastructure for its needs.

**Speed** – when the organization needs more resources, provisioning additional resources in cloud services can be done in minutes.

**Scaling** – the ability to scale elastically on demand using cloud services appears as their main and most common use case – processing power, storage, bandwidth and whatever the demand is, in less than a minute.

Depending on the type of service a Service Providers provides, there are several categories of Cloud Computing models, as listed:

### 3.4 Categories of Cloud Computing models

#### 3.4.1 Software as a Service (SaaS)

The providers that provide this model of Cloud Computing solutions usually provide a web-based application where the users of the service can operate. In this model, the consumer does not have any control over

the infrastructure in which the service is running, including the network, servers, storage or operating system. It removes the need that several organizations or companies will have to install and run their applications or services on their data centers or company computers. By this, the organizations save a lot of financial resources by saving money on the hardware they would need to run the application, the rent of space where the data center would be located on, or even software license for operating systems and depending software.

### **3.4.2 Platform as a Service (PaaS)**

Platform as a Service is another Cloud Computing model in which the third-party provider provides the necessary hardware and software tools – usually required for development or research – over the Internet. In other words, all the programming languages, libraries, services and other programming tools provided by the provider are deployed in the cloud infrastructure that the provider provides. Similar as in the previous model, SaaS, the end user does not have any control nor have to manage any part of the infrastructure, such as network, operating systems and storage.

### **3.4.3 Infrastructure as a Service (IaaS)**

According to most of the information provided by different surveys, IaaS is the most common cloud-based model provided by the service providers. IaaS refers to the service providers who provide processing capability, storage, network and other fundamental computing resources, to the consumer who wants to run any type of software. Usually these services are made possible by using virtual machines as instances. Xen, Oracle VirtualBox, KVM or Hyper-V are typical examples of providers that offer great possibilities to run these VMs.

## **3.5 Mobile Cloud Computing (MCC)**

In the consumer space, mobility players such as Apple, Google, and Microsoft all offer variants of cloud-based apps and private storage. However, the line between the individual and the professional is increasingly being blurred. Allowing employees access to company resources using private devices makes them expect access to your CRM system on their iPad, with (near) real-time business intelligence reports delivered by the touch of a finger while sharing analysis with their teams on the collaboration platform.

Most of the companies tend to move their apps and services in the cloud. Every company's mission is to grow and evolve. Considering this case, organizations face trouble with new coming employees, which bring their own devices, services and apps. This means that, it requires more efforts

and time to integrate the data to the corporate cloud, in order to ensure support and control over usage of the same. When we add the complex format of making sure that corporate services are up to date, all this process becomes a mess and quite often it becomes a challenging task for the responsible employees.

### **3.5.1 Advantages of Mobile & Cloud Computing**

Mobile Cloud Computing offers a bunch of advantages while using cloud services. Following are listed some of the most important ones:

Flexibility – one of key advantages while using MCC is that the cloud information can be used anywhere, everywhere; all you need is a mobile device of any kind, which is paired or configured with the organization cloud platform.

Real time available data – accessing the data in real time is no longer a challenge while you are out of the office.

No upfront payments – last, but not least – payments. Commonly, cloud applications does not require payment without using it. It is mostly the case pay-for-use, which helps in growing the adoption of the model.

### **3.5.2 Disadvantages of Mobile & Cloud Computing (MCC)**

Whenever there are advantages on any issue, it is sure there would be the disadvantages as well. The following are some listed and most important disadvantages of Mobile and Cloud Computing.

Security – a major concern with Cloud Computing is the security and data integration. When mobile is the subject, the attention must be two times higher: unprotected information can be easily sniffed.

Internet connection – considering the flexibility of MCC, allowing the users to access the data from anywhere, requires Internet connection. Making sure that, when accessing data, the user have access to strong and stable Internet connection, often can cause headache, especially in non-metropolitan areas.

Performance – considering smaller size and lower hardware performance, it is understandable that the performance with MCC will be in a much lower level.

## **3.6 Mobile Cloud Computing Security Concerns**

One of the most significant concerns of Cloud Computing in general and Mobile and Cloud Computing particularly, is data security.

Mobile devices are at the top of the list of the most significant security risks. Confidentiality, integrity and authenticity of information are the most particular threat. Confidentiality is considered a risk when

unauthorized parties manage to intercept data transmission. Allowing such a thing, risks the integrity of the data. The authenticity is risked when these unauthorized parties can use the devices to trigger transactions.

The latest trends of using mobile devices is by using free applications, which can be infected by malicious software. Using open channels over network threatens confidential information. Thus, these applications are often updated or upgraded, trying to provide as much security as possible.

### **3.7 The Top Threats in the usage of Mobile and Cloud Computing**

#### **3.7.1 Data Loss**

Using Cloud Computing is more like outsourcing the data to the service provider.

This means increasing the risk of exposing important data which were not issues in traditional computing. Since more of the service providers provide shared resources, it is more likely for the transactions to crash and data to be lost. Recently, there has been a lot of unintentional deletion of data by the providers. Also, a bad line code can mess up access keys, and the data is lost.

The following solutions can lower the risk:

- Encryption of data while transmission;
- Using access control tools
- Time-to-time back up

#### **3.7.2 Untrusted service providers**

Known as *malicious insiders*, they are the people who have access and authorization to manage data in the care of the service providers, offering cloud services. These people can either be working for other companies or they do it for their personal intentions.

#### **3.7.3 Insecure API**

Usually, the communication between a client (in this case, a mobile device which is handled by the company's employee) and the server (which is somewhere in the cloud) is done by an Application Programming Interface. In order to keep data integration and security in a higher level, the company providing the API should secure the communication channels and the information transmitted. Avoiding insecure APIs can be achieved by using the following techniques:



Applying authentication and access control tools on data transmission channels  
 Implementing the proper security model according to service provider's security protocols

### **Discussion**

What are the advantages and disadvantages of Mobile and cloud computing?

### **4.0 Self-Assessment/Exercises**

Give and explain 3 top threats in the usage of Mobile and Cloud Computing.

#### **Answer**

##### **Data Loss**

Using Cloud Computing is more like outsourcing the data to the service provider.

This means increasing the risk of exposing important data which were not issues in traditional computing. Since more of the service providers provide shared resources, it is more likely for the transactions to crash and data to be lost. Recently, there has been a lot of unintentional deletion of data by the providers. Also, a bad line code can mess up access keys, and the data is lost.

The following solutions can lower the risk:

Encryption of data while transmission;

Using access control tools

Time-to-time back up

##### **Untrusted service providers**

Known as *malicious insiders*, they are the people who have access and authorization to manage data in the care of the service providers, offering cloud services. These people can either be working for other companies or they do it for their personal intentions.

##### **Insecure API**

Usually, the communication between a client (in this case, a mobile device which is handled by the company's employee) and the server (which is somewhere in the cloud) is done by an Application Programming Interface. In order to keep data integration and security in a higher level, the company providing the API should secure the communication channels and the information transmitted. Avoiding insecure APIs can be achieved by using the following techniques:

Applying authentication and access control tools on data transmission channels

Implementing the proper security model according to service provider's security protocols

Whenever there are advantages on any issue, it is sure there would be the disadvantages as well. Produce the disadvantages of Mobile and Cloud computing

The following are some listed and most important disadvantages of Mobile and Cloud Computing.

Security – a major concern with Cloud Computing is the security and data integration. When mobile is the subject, the attention must be two times higher: unprotected information can be easily sniffed.

Internet connection – considering the flexibility of MCC, allowing the users to access the data from anywhere, requires Internet connection. Making sure that, when accessing data, the user have access to strong and stable Internet connection, often can cause headache, especially in non-metropolitan areas.

Performance – considering smaller size and lower hardware performance, it is understandable that the performance with MCC will be in a much lower level.

## **5.0 CONCLUSION**

Nowadays, Cloud Computing is moving in big strides towards becoming the most popular and the used technology, either in the organizational context, or personal domain. Considering the fact that mobile technology provides flexibility, compactness and portability, the big players in the IT industry are really focused on generating, as optimal as possible, solutions that will drive mobile devices.

## **6.0 SUMMARY**

Cloud Computing is the delivery of the computing services, such as servers, databases, storage and networking – over the Internet. Generally, these are a few of the capabilities of Cloud Computing: Create new apps and services, Store, back up and recover data, Deliver software, Analyse data for pattern recognition and Streaming. The following are some listed and most important disadvantages of Mobile and Cloud Computing: Security, Internet connection and Performance. The Top Threats in the usage of Mobile and Cloud Computing are Data Loss, Untrusted service providers and Insecure API.

## **7.0 REFERENCES/FURTHER READING**

Lofstad, S.: Trends in Cloud Computing: The Impact of Mobile Devices.  
Director of Data Center Technologies at Oracle Insight. 2013.  
<http://www.oracle.com/us/corporate/profit/archives/opinion/011813-slofstad-1899122.html>

Bahtovski, A., Gusev, M.: Cloud Computing in Mobile Technologies. The 9<sup>th</sup> Conference for Informatics and Information Technology (CIIT 2012).

Shanklin, M.: Mobile Cloud Computing (A survey paper written under the guidance of Prof.

Raj Jain). <https://www.cse.wustl.edu/~jain/cse574-10/ftp/cloud/>  
Wikipedia: Cloud Computing.  
[https://en.wikipedia.org/wiki/Cloud\\_computing](https://en.wikipedia.org/wiki/Cloud_computing)

Microsoft Azure: What is cloud computing? A beginner's guide.  
<https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/>

Rouse, M.: Definition: Software as a Service (SaaS). May, 2016.  
<http://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service>

Rouse, M.: Definition: Platform as a Service (PaaS). September, 2017.  
<http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS>

Mobile Cloud Computing – Pros and Cons. December, 2014.  
<https://www.getcloudservices.com/blog/mobile-cloud-computing-pros-and-cons/>

Kleiner, C., Disterer, G.: Ensuring mobile device security and compliance at the workplace. Conference on Enterprise Information Systems, HCist 2015, October 7-9, 2015.

Aldossary, S., Allen, W: Data Security, Privacy, Availability and Integrity in Cloud Computing: Issues and Current Solutions. International Journal of Advanced Computer Science and Applications, Vol. 7, No. 4, 2016.

## **UNIT 2    TECHNOLOGIES FOR WIRELESS COMMUNICATIONS**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Technologies for Wireless Communications
    - 3.1.1 Radio
    - 3.1.2 Cellular
    - 3.1.3 Satellite
    - 3.1.4 Wi-fi
  - 3.2 Pros & Cons of Microwave Internet Service
    - 3.2.1 Pros-Lower Initials Costs
    - 3.2.2 Cons-Interference
    - 3.2.3 Pro-mobility
    - 3.2.4 Cons-Shared Bandwidth
  - 3.3 Different Types of Roles
    - 3.3.1 AM and FM
    - 3.3.2 Shortwave Radio
    - 3.3.3 Satellite Radio
    - 3.3.4 Ham Radio
    - 3.3.5 Walkie-Talkie
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

Wireless communication technology defines any method of communication possible without a direct physical connection between the two parties, largely describing systems based on radio waves. The first wireless communication systems came into use at the end of the 19th century, and the technology has matured significantly over the intervening years. Today, many types of devices use wireless communication technology, allowing users to remain in contact even in remote areas.

### **2.0 INTENDED LEARNING OUTCOMES (ILOS)**

At the end of this unit, the student will able to:  
identify and describe some technologies for Wireless Communication

explain the Pros & Cons of Microwave Internet Service identify and explain the types of radios

### **3.0 MAIN CONTENT**

#### **3.1 Technologies for Wireless Communication**

##### **3.1.1 Radio**

Open radio communication was one of the first wireless technologies to find widespread use, and it still serves a purpose today. Portable multichannel radios allow users to communicate over short distances, while citizen's band and maritime radios provide communication services for truckers and sailors. Ham radio enthusiasts share information and serve as emergency communication aids during disasters with their powerful amateur broadcasting equipment, and can even communicate digital data over the radio spectrum.

##### **3.1.2 Cellular**

Cellular networks use encrypted radio links, modulated to allow many different users to communicate across a single frequency band. Because individual handsets lack significant broadcasting power, the system relies on a network of cellular towers, capable of triangulating the source of any signal and handing reception duties off to the most suitable antenna. Data transmission over cellular networks is possible, with at least, modern 3G systems capable of speeds approaching that of wired DSL or cable connections. Cellular providers typically meter their service, charging customers by the minute for voice and by the megabyte for data.

##### **3.1.3 Satellite**

Satellite communication is another wireless technology that has found widespread use in specialized situations. These devices communicate directly with orbiting satellites via a radio signal, allowing users to stay connected virtually anywhere on Earth. Portable satellite phones and modems feature more powerful broadcast and reception hardware than cellular devices due to the increased range, and are correspondingly more expensive. For semi-permanent or permanent installations, such as outfitting a ship for satellite communication, a more traditional communication system may link to a single satellite uplink, allowing multiple users to share the same broadcast equipment.

##### **3.1.4 Wi-Fi**

Wi-Fi is a form of low-power wireless communication used by computers and hand-held electronic devices. In a Wi-Fi setup, a wireless router serves as the communication hub, linking portable devices to a wired internet connection. These networks are extremely limited in range due to the low power of the transmissions, allowing users to connect only within close proximity to a router or signal repeater. Wi-Fi is common in home networking applications, allowing users to link devices without running lengths of cable, and in commercial applications where a business may provide wireless Internet access to their customers. Wi-Fi networks may be free to use, or their owners may secure them with passwords and access restrictions.

### **3.2 Pros & Cons of Microwave Internet Service**

Microwave radio transmission has been used for wireless data transmission since before the terms wireless broadband or WiFi came into common usage. It was primarily used by businesses to connect separate office buildings or locations. Transmission was limited by slower data speeds, line-of-sight connections and bandwidth issues. The development of WiMAX -- Worldwide Interoperability for Microwave Access -- technology has improved upon these shortcomings.

#### **3.2.1 Pro -- Lower Initial Costs**

The costs of installing a microwave tower are significantly less than those of installing traditional buried cable systems, such as DSL or cable. WiMAX technology has a greater range than traditional WiFi and is not limited to line-of sight access, providing for a larger potential customer base per tower. WiMAX operates on frequencies both licensed and non-licensed. The system is governed by IEEE 802.16 standards, which provides a feasible economic model and regulated environment for wireless carriers.

#### **3.2.2 Con -- Interference**

Radio frequency (RF) transmissions can be adversely affected by weather conditions and terrain. Temperature, humidity, precipitation and wind can all cause interference with RF communications. Topographical features like hills and valleys can reflect or block signals. The density and height of nearby trees will also affect reception. Lakes, rivers and other water formations are extremely reflective surfaces in regards to radio transmissions. Large buildings can also create a "shadow" which leaves a dead spot directly behind the structure. These obstacles necessitate proper layout and planning of the wireless networks to minimize signal degradation.

### 3.2.3 Pro -- Mobility

WiMAX was the first of the Fourth Generation, 4G, wireless technologies. Fixed networks can provide service within a 30 mile area. As long as a customer is within that range they are able to access the service. Mobile networks have a range of about 2.5 miles, providing even greater flexibility and availability of connection.

### 3.2.4 Con -- Shared Bandwidth

All connections within range of a tower share the same bandwidth. WiMAX offers speeds up to 70Mbps, but this is attainable only in ideal conditions and with a single user. Connection speeds are significantly reduced as more and more users connect to the network. Slower speeds also result from being farther from the tower.

## 3.3 Different Types of Radios

Radio communication, first developed at the turn of the 20th century, remains a significant part of the technology landscape despite decades of innovation and scientific breakthroughs. Radios work by transmitting and receiving electromagnetic waves that move invisibly at the speed of light, carrying music and speech in a coded form that depends on the type of radio used. Over the decades, radio has evolved into many different types, each of which fulfills different needs.

### 3.3.1 AM and FM

Amplitude modulation, or AM radio, is one of the oldest forms of wireless broadcasting. With AM, an audio signal rapidly modifies the strength of radio waves in a process called **modulation**; an AM receiver decodes the modulation back into sound. With the introduction of the transistor in the 1960s, pocket-sized AM radios became a reality for the first time. Although AM's coding scheme is simple, its sound quality is only fair, and it is vulnerable to electrical noise. FM, which was developed in the 1930s, relies on the modulation of the radio signal's frequency and not its strength. The higher radio frequencies used for FM as well as the modulation scheme give it much better sound quality with less noise than AM.

### **3.3.2 Shortwave Radio**

Shortwave radio lies in a range of frequencies from 1.7 to 30 megahertz, just above the AM radio band in the U.S. Because of the way its frequencies interact with the Earth's ionosphere, shortwave broadcasts can travel thousands of miles -- under some circumstances, listeners can tune in anywhere on Earth. Government and commercial stations broadcast on shortwave frequencies to provide news, information and other content. For example, the U.S. government runs WWV, a station that gives accurate time information, at 2.5, 5, 10, 15 and 20 MHz.

### **3.3.3 Satellite Radio**

One of the newest forms of broadcasting, satellite radio is a commercial, subscription-based service that uses a network of satellites to transmit signals over wide areas. Unlike traditional AM and FM broadcasts, satellite radio is digitally encoded, requiring a special receiver. Even with the receiver, you cannot tune in unless you have a paid subscription; a computer chip in the receiver locks out any channels not paid for. Advantages of satellite radio include good sound quality, nationwide coverage and access to material that sidesteps the Federal Communications Commission's ban on profanity.

### **3.3.4 Ham Radio**

An amateur or "ham" radio operator broadcasts and receives signals over a restricted set of frequencies set aside by the FCC; ham radio requires special training, licensing and equipment. As with shortwave, ham radio broadcasts can travel thousands of miles depending on the time of day and other conditions. For many, ham radio serves as an interesting and entertaining hobby, as operators learn practical radio skills and form friendships with operators in other countries. In times of natural disaster, local communications may be knocked out; ham operators are known to step in to pass along life-saving information.

### **3.3.5 Walkie-Talkie**

A walkie-talkie is a portable, handheld device that sends and receives radio signals, usually within a range of about a mile. Walkie-talkies are used by two or more people to communicate in situations where cell phone service is poor or unavailable, such as in remote locations or in buildings. Because walkie-talkies have low power and short range, you don't need a special license to operate them; they interfere little with other radio signals



**Discussion**

Discuss any two radio types.

**4.0 SELF-ASSESSMENT/EXERCISES****Describe the Wi-Fi technologies**

Answer:

Wi-Fi is a form of low-power wireless communication used by computers and hand-held electronic devices. In a Wi-Fi setup, a wireless router serves as the communication hub, linking portable devices to a wired internet connection. These networks are extremely limited in range due to the low power of the transmissions, allowing users to connect only within close proximity to a router or signal repeater. Wi-Fi is common in home networking applications, allowing users to link devices without running lengths of cable, and in commercial applications where a business may provide wireless Internet access to their customers. Wi-Fi networks may be free to use, or their owners may secure them with passwords and access restrictions

Explain Satellite as a technology for wireless communication  
Satellite communication is another wireless technology that has found widespread use in specialized situations. These devices communicate directly with orbiting satellites via a radio signal, allowing users to stay connected virtually anywhere on Earth. Portable satellite phones and modems feature more powerful broadcast and reception hardware than cellular devices due to the increased range, and are correspondingly more expensive. For semi-permanent or permanent installations, such as outfitting a ship for satellite communication, a more traditional communication system may link to a single satellite uplink, allowing multiple users to share the same broadcast equipment.

**5.0 CONCLUSION**

Wireless communication as technology has come to live with us. Its capability rest most in mobility. It is more comfortable as enables the user to work untethered. Its disadvantage stems from instability as it could be interfered with anytime which does not augur well with mission critical systems.

**6.0 SUMMARY**

Wireless communication technology defines any method of communication possible without a direct physical connection between the two parties, largely describing systems based on radio waves. Different

Types of Radios are AM and FM, Shortwave Radio, Satellite Radio, Ham Radio and Walkie-Talkie. Technologies for Wireless Communication are Radio, Cellular, Satellite and Wi-Fi.

## UNIT 3 WIRELESS CELLULAR SYSTEMS

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Wireless Cellular Systems
    - 3.1.1 Cellular Concepts
    - 3.1.2 Frequency Reuse
      - 3.1.2.1 Interference and Reuse
    - 3.1.3 Multiple Access
      - 3.1.3.1 FDMA
      - 3.1.3.2 TDMA
      - 3.1.3.3 CDMA
    - 3.1.4 Systems Capacity
      - 3.1.4.1 Channel Capacity
      - 3.1.4.2 Cellular Capacity
        - 3.1.4.2.1 Cellular analog Capacity
        - 3.1.4.2.2 TDMA/ EDMA Capacity
        - 3.1.4.2.3 CDMA Capacity
    - 3.1.5 Modulation and Coding
      - 3.1.5.1 Modulations
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

Wireless communications are especially useful for mobile applications, so wireless systems are often designed to cover large areas by splitting them into many smaller cells. This cellular approach introduces many difficulties such as how to avoid interference, or how to hand-over from one cell to another, while maintaining good service quality. Coverage, capacity, interference, and spectrum reuse are important concerns of cellular systems; this chapter reviews these aspects as well as the technologies, tools, and standards used to optimize them.

### 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, student will able to:

Explain some concepts on Wireless Cellular Systems  
Explain the term, Frequency re-use

Identify and explain the techniques for multiple devices to access the wireless system.

### 3.0 MAIN CONTENT

#### 3.1 Wireless Cellular Systems

##### 3.1.1 Cellular Concepts

Providing wireless service over wide areas requires different schemes to efficiently use spectrum in different locations while avoiding interference.

##### 3.1.2 Frequency Reuse

Covering a large geographic area with limited amount of spectrum leads to the reuse of the same frequency in multiple locations; this leads to co-channel interference considerations, meaning interference from different areas (or cells) that use the same frequency channel. Co-channel interference considerations are usually approached by considering the following parameters:

- $S_t$ : total number of RF channels available (given the amount of spectrum and channel width dictated by technology standard),
- $S_0$ : number of channels per cell, which reflects system capacity at a given location,
- $K$ : the reuse factor, the number of cells that is repeated to provide coverage over a large area.

The three quantities are linked by the straightforward relation:

$$S_t = S_0 K \quad \dots \text{equation 5.3.1}$$

The reuse factor  $K$  is therefore an important parameter for capacity. The lowest reuse factor ( $K = 1$ ) maximizes capacity; but this has to be balanced with interference considerations: indeed a higher reuse factor ( $K = 3, 4, 7$ , or higher) provides more distance between cells using the same frequency, which lowers interferences.

##### 3.1.2.1 Interference and Reuse

Spectrum reuse causes interference; quantifying them require us to consider how a signal propagates from one cell to another. Assume a propagation model using a power path loss exponent  $n$ , that is a model where power decays in  $1/R^n$  ( $R$  being the distance separating transmit station from receiver). This means that the ratio of received power to

transmit power may be expressed as  $P_r / P_t = A/R^n$ , (equation 5.3.2) where  $A$  is some constant.

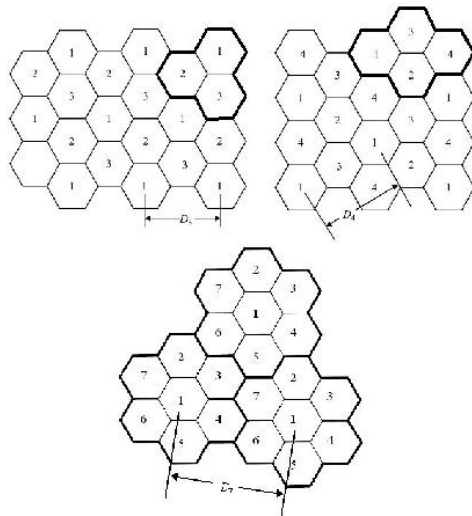


Figure 5.3.1: Frequency reuse patterns

**Figure 5.3.1:** Frequency reuse patterns  $K = 3, 4,$  and  $7,$  on hexagonal cells. Bold contour shows the pattern of cells repeated to provide wide area coverage.  $D$  shows the shortest distance between cells reusing the same frequency.

With this model, signal to interference ratios are estimated as

$$S/I = \frac{R^{-n}}{\sum_{i=1}^{i_0} D_i^{-n}} \quad \dots \text{equation 5.3.3}$$

Where  $i_0$  is the number of co-channel cells nearest to the cell (called first tier or tier one); that number increases with  $K$ . And  $D_i$  is the distance to the tier-one cells reusing the same frequency (as shown in figure 5.3.1). In the case of hexagonal cell approximation the expression simplifies to equation 5.3.1:

$$S/I = \frac{(\sqrt{3K})^n}{i_0} \quad \dots \text{equation 5.3.4}$$

$n$  values vary typically between 2 and 4 with the types of terrain. We will also see that specific wireless technologies require a certain signal to noise and interference ratio (mostly based on data rates); so equation (5.3.1) leads to a minimal acceptable value for  $K$ .

### 3.1.3 Multiple Access

A major requirement of cellular networks is to provide an efficient technique for multiple devices to access the wireless system. These techniques include:

**FDMA:**

Frequency Division Multiple Access, perhaps the most straightforward, in which every user device uses its own frequency channel. This method was used in the first generation analog systems.

**TDMA:**

Time Division Multiple Access, in which a radio channel is divided in time slots, and use devices use their allocated time slots. In fact TDMA systems are often hybrid FDMA as well as multiple channels are used, most 2G systems were TDMA.

**CDMA:**

Code Division Multiple Access, in which orthogonal (or pseudo orthogonal) codes are used to differentiate user devices. CDMA is very spectrum efficient, and was used by 3G standards. There are several approaches to achieve CDMA, such as frequency hopping (FH-CDMA) or direct spreading (DS-SS).

These are the main multiple access techniques, but subtle extensions and combinations can be devised to obtain more efficient schemes.

**3.1.4 System Capacity**

Wireless communications deal with at least two main concerns: coverage and capacity.

**Channel Capacity**

One fundamental concept of information theory is one of channel capacity, or how much information can be transmitted in a communication channel. In the 1940's Claude Shannon invented formal characterization of information theory and derived the well-known Shannon's capacity theorem. That theorem applies to wireless communications.

The Shannon capacity equation gives an upper bound for the capacity in a non-faded channel with added white Gaussian noise:

$$C = W \log_2(1 + S/N) \dots \text{equation 5.3.5}$$

Where  $C$ = capacity (bits/s),  $W$ =bandwidth (Hz),  $S/N$ = signal to noise (and interference) ratio.

That capacity equation assumes one transmitter and one receiver, though multiple antennas can be used in diversity scheme on the receiving side. The equation singles out two fundamentally important aspects: bandwidth and SNR.

Bandwidth reflects how much spectrum a wireless system uses, and explains why the spectrum considerations are so important: they have a direct impact on system capacity. SNR of course reflects the quality of the propagation channel, and will be dealt with in numerous ways: modulation, coding, error correction, and important design choices such as cell sizes and reuse patterns.

### **Cellular Capacity**

Practical capacity of many wireless systems are far from the Shannon's limit (although recent standards are coming close to it); and practical capacity is heavily dependent on implementation and standard choices. Digital standards deal in their own way with how to deploy and optimize capacity. Most systems are limited by channel width, time slots, and voice coding characteristics. CDMA systems are interference limited, and have tradeoffs between capacity, coverage, and other performance metrics (such as dropped call rates or voice quality).

### **Cellular Analog Capacity:**

Fairly straight forward, every voice channel uses a 30 kHz frequency channel, these frequencies may be reused according to a reuse pattern, the system is FDMA. The overall capacity simply comes from the total amount of spectrum, the channel width and the reuse pattern.

### **TDMA/FDMA Capacity:**

In digital FDMA systems, capacity improvements mainly come from the voice coding and elaborate schemes (such as frequency hopping) to decrease reuse factor. The frequency reuse factor hides a lot of complexity; its value depends greatly on the signal to interference levels acceptable to a given cellular system. TDMA systems combine multiple time slots per channels.

### **CDMA Capacity:**

A usual capacity equation for CDMA systems may be fairly easily derived as follows (for the reverse link): first examine a base station with  $N$  mobiles, its noise and interference power spectral density due to all mobiles in that same cell is  $I_{sc} = (N-1)S\alpha$ , where  $S$  is the received power density for each mobile, and  $\alpha$  is the voice activity factor. Other cell interferences  $I_{oc}$  are estimated by a reuse fraction  $\beta$  of the same cell interference level, such that  $I_{oc} = \beta I_{sc}$ ; (usual values of  $\beta$  are around 1/2). The total noise and interference at the base is therefore  $N_t = I_{sc}(1 + \beta)$ . Next assume the mobile signal power density received at the base station is  $S = RE_b/W$ . Eliminating  $I_{sc}$ , we derive:

$$N = 1 + \frac{W}{R} \cdot \frac{1}{E_b/N_t} \cdot \frac{1}{\alpha} \cdot \frac{1}{1 + \beta} \dots \text{equation 5.3.6}$$

Where:

$W$  is the channel bandwidth (in Hz),

$R$  is the user data bit rate (symbol rate in symbol per second),

$E_b/N_t$  is the ratio of energy per bit by total noise (usually given in dB  $E_b/N_t \approx 7\text{dB}$ ),

$\alpha$  is the voice activity factor (for the reverse link), typically 0.5,

and  $\beta$  is the interference reuse fraction, typically around 0.5, and represents the ratio of interference level from the cell in consideration by interferences due to other cells. (The number  $1 + \beta$  is sometimes called reuse factor, and  $1/(1 + \beta)$  reuse efficiency)

This simple equation (5.3.6) gives us a number of voice channels in a CDMA frequency channel.

We can already see some hints of CDMA optimization and investigate certain possible improvement for a 3G system. In particular: improving  $\alpha$  can be achieved with dim and burst capabilities,  $\beta$  with interference mitigation and antenna downtilt considerations,  $R$  with vocoder rate,  $W$  with wider band CDMA,  $E_b/N_t$  with better coding and interference mitigation techniques.

Some aspects however are omitted in this equation and are required to quantify other capacity improvements mainly those due to power control, and softer/soft handoff algorithms.

Of course other limitations come into play for wireless systems, such as base station (and mobile) sensitivity, which may be incorporated into similar formulas; and further considerations come into play such as: forward power limitations, channel element blocking, backhaul capacity, mobility, and handoff.

### 3.1.5 Modulation and Coding

Modulation techniques are a necessary part of any wireless system, without them, no useful information can be transmitted. Coding techniques are almost as important, and combine two important aspects: first to transmit information efficiently, and second to deal with error correction (to avoid retransmissions).

#### Modulation

A continuous wave signal (at a carrier frequency  $f_c$ ) in itself encodes and transmits no information. The bits of information are encoded in the variations of that signal (in phase, amplitude, or a combination thereof). These variations cause the occupied spectrum to increase, thus occupying a bandwidth around  $f_c$ ; and the optimal use of that bandwidth is an important part of a wireless system. Various modulation schemes and coding schemes are used to maximize the use of that spectrum for



different applications (voice or high speed data), and in various conditions of noise, interference, and RF channel resources in general.

**Classic modulation techniques** are well covered in several texts, and we simply recall here a few important aspects of digital modulations (that will be important in link budgets). The main digital modulations used in modern wireless systems are outlined in table 5.3.1.

*Table 5.3.1: The main Digital Modulations in Modern Wireless Systems*

<b>Modulation</b>	<b>Bits encoded by:</b>	<b>Example</b>
Amplitude Shift Keying	Discrete amplitude levels	On/off keying
Frequency Shift Keying	Multiple discrete frequencies	
Phase Shift Keying	Multiple discrete phases	BPSK, QPSK, 8-PSK
Quadrature Ampl. Mod.	Both phase and amplitude	16, 64, 256 QAM

### **Discussion**

Provide the bits they are encoded in and example of the main digital modulations used in modern wireless systems.

## **5.0 CONCLUSION**

Providing wireless service over wide areas requires different schemes to efficiently use spectrum in different locations while avoiding interference.

## **6.0 SUMMARY**

Covering a large geographic area with limited amount of spectrum leads to the reuse of the same frequency in multiple locations; this leads to co-channel interference considerations, meaning interference from different areas (or cells) that use the same frequency channel. Spectrum reuse causes interference; quantifying them require us to consider how a signal propagates from one cell to another. A major requirement of cellular networks is to provide an efficient technique for multiple devices to access the wireless system. These techniques include FDMA, TDMA and CDMA. Wireless communications deal with at least two main concerns: coverage and capacity. The capacities are Channel Capacity and Cellular Capacity. Modulation techniques are a necessary part of any wireless system, without them, no useful information can be transmitted. Coding techniques are almost as important, and combine two important aspects: first to transmit information efficiently, and second to deal with error correction (to avoid retransmissions).

## 7.0 REFERENCES/FURTHER READING

[Contents \(colorado.edu\)](#)

[https://www.tutorialspoint.com/wireless\\_communication/wireless\\_communication\\_cellular\\_networks.htm](https://www.tutorialspoint.com/wireless_communication/wireless_communication_cellular_networks.htm)

<https://www.javatpoint.com/cellular-system-infrastructure>

## **UNIT 4    OVERVIEW OF WIRELESS LAN, IEEE 802.11, PERSONAL AREA NETWORK, BLUETOOTH**

### **CONTENTS**

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 Introduction to Wireless Local Area Network (WLAN)
  - 3.2 WLANs and Access Points
  - 3.3 Emerging WLANs and the Ubiquity of WLANs
  - 3.3 How a WLAN works
  - 3.4 Configuration of WLAN
  - 3.5 How roaming works on a WLAN
  - 3.6 WLAN architecture
  - 3.7 Benefits of a WLAN
  - 3.8 IEEE 802.11 Standard
  - 3.9 Bluetooth Technology
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### **1.0 INTRODUCTION**

A wireless local area network (WLAN) is a wireless distribution method for two or more devices. WLANs use high-frequency radio waves and often include an access point to the Internet. A WLAN allows users to move around the coverage area, often a home or small office, while maintaining a network connection.

A WLAN is sometimes called a local area wireless network (LAWN).

### **2.0 INTENDED LEARNING OUTCOMES (ILOS)**

At the end of this unit, students will able to:

Describe WLAN, IEEE 802.11 standard and Bluetooth technology.

Explain the WLAN Architecture.

Identify and explain the configurations of WLAN

## 3.0 MAIN CONTENT

### 3.1 Wireless Local Area Network (WLAN)

A wireless local-area network (WLAN) is a group of colocated computers or other devices that form a network based on radio transmissions rather than wired connections. A Wi-Fi network is a type of WLAN.

Although some may use the terms “Wi-Fi” and “WLAN” interchangeably but they are not the same. A “Wi-Fi connection” refers to a given wireless connection that a device uses, the WLAN is the network itself. “Wi-Fi” is a superset of the IEEE 802.11 standard and is sometimes used interchangeably with that standard. However, not every Wi-Fi device actually receives Wi-Fi Alliance certification, although Wi-Fi is used by more than 700 million people through about 750,000 Internet connection hot spots. The hot spots themselves also constitute WLANs, of a particular kind.

#### 3.1.1 WLANs and Access Points

Every component that connects to a WLAN is considered a station, and falls into one of two categories: access points (APs) and clients.

**Access points or APs** transmit and receive radio frequency signals with devices that are able to receive transmitted signals; they normally function as routers.

**Clients**, on the other hand, may include a variety of devices, such as desktop computers, workstations, laptop computers, IP phones and other cell phones and smartphone devices.

All stations able to communicate with each other are called basic service sets (BSSs), of which there are two types: independent and infrastructure. Independent BSSs (IBSS) exist when two clients communicate without using APs, but cannot connect to any other BSS. Such WLANs are called a peer-to-peer or an ad-hoc WLANs. The second BSS is called an infrastructure BSS. It may communicate with other stations but only in other BSSs and it must use APs.

#### 3.1.2 Emerging WLANs and its Ubiquity

In the early 1990s, WLANs were very expensive, and were only used when wired connections were strategically impossible.

By the late 1990s, most WLAN solutions and proprietary protocols were replaced by IEEE 802.11 standards in various versions (versions "a" through "n"). WLAN prices also began to decrease significantly.

As technology progressed, WLANs became easier and easier to set up and administrate.

That led to the emergence of the ISP WLAN, where so many small local home networks are mostly coordinated by the Internet Service Provider, and not engineered by the end-user on-site.

In these types of ISP WLAN setups, the ISP's modem is the access point. It's also the router. All that the consumer has to do is plug in the router, use provided security passwords, and connect home devices to the home WLAN.

You could call this “wireless local area network as a service” (WLANaaS) or refer to a “plug-and-play” or abstracted wireless local area network model. In any case, it's ultimately very convenient for the household. Although ISPs don't usually advertise their products as home LANs, that's what they are. Some types of ISP services talk about using the modem as a “gateway” to the Internet, which implies that your WLAN is on the other side of that gateway.

Users of home WLANs are more frequently connecting devices such as phones, televisions, computers and printers to evolved WLAN systems where the ISP will offer some type of dashboard visualization for the WLAN in question.

There's also been some innovation toward peer-to-peer WLANs that work without a defined access point. In other words, all of the devices are independently operated to network together. This challenges the traditional idea that the WLAN was made of access points and clients, as discussed above. At the same time, in the client/server architecture, where a similar approach is used to engineer Internet services, peer-to-peer systems are also challenging that traditional build as well. As the IoT paves the way for advanced connectivity, the WLAN provides that “sub-network” and the convenience of local Wi-Fi operation.

### **3.1.4 How a WLAN works**

Like broadcast media, a WLAN transmits information over radio waves. Data is sent in packets. The packets contain layers with labels and instructions that, along with the unique MAC (Media Access Control) addresses assigned to endpoints, enable routing to intended locations.

### **3.1.5 Configuration of WLAN**

A WLAN can be configured in one of two ways:

## **Infrastructure**

A home or office Wi-Fi network is an example of a WLAN set up in infrastructure mode. The endpoints are all connected and communicate with each other through a base station, which may also provide internet access.

A basic infrastructure WLAN can be set up with just a few parts: a wireless router, which acts as the base station, and endpoints, which can be computers, mobile devices, printers, and other devices. In most cases, the wireless router is also the internet connection.

## **Ad hoc**

In this setup, a WLAN connects endpoints such as computer workstations and mobile devices without the use of a base station. Use of Wi-Fi Direct technology is common for an ad hoc wireless network. An ad hoc WLAN is easy to set up and can provide basic peer-to-peer (P2P) communication. An ad hoc WLAN requires only two or more endpoints with built-in radio transmission, such as computers or mobile devices. After adjusting network settings for ad hoc mode, one user initiates the network and becomes visible to the others.

### **3.1.6 How Roaming works on a WLAN**

For any sized network, access points can extend the area of access. Wi-Fi standards are designed to allow a non-stationary user's connection to jump from one access point to another, though some users and applications may experience brief dropouts. Even with non-overlapping access points, a user's connection is simply paused until connection with the next access point. Additional access points can be wired or wireless. When access points overlap, they can be configured to help optimize the network by sharing and managing loads.

### **3.1.7 WLAN Architecture**

#### **Stations**

Stations are components that connect wirelessly to networks. They are either access points or endpoints, each identified with a unique network address.

#### **Basic Service Set (BSS)**

A BSS is a group of stations that connects to the network. In ad hoc networks, the group of stations is called an Independent BSS (IBSS). A set of connected BSSs, as in a network with multiple access points, is called an Extended Service Set (ESS).

### **Distribution system**

The distribution system connects access points in an ESS. The connections can be wired or wireless. A wireless distribution system (WDS) can use mesh or its own WDS protocol. Fixed wireless is a specialized form of radio transmission for connecting a geographically distant access point.

### **Access point**

The access point is the base station that serves as a hub to which other stations connect. The "access" is that of the stations to the network, but it may also mean internet access, since many routers double as internet modems. In an ESS, access points may be connected with Ethernet cables or wirelessly.

### **Bridge**

The bridge is used to connect a WLAN to a LAN or to an access point.

### **Endpoint**

The endpoint is any end-user station, such as a computer, mobile device, printer, or Internet of Things (IoT) device.

## **3.1.8 Benefits of a WLAN**

**Extended reach:** WLANs enable computing to happen anywhere, even when carrying high data loads and advanced web applications.

**Device flexibility:** A WLAN supports use of a wide range of devices, such as computers, phones, tablets, gaming systems, and IoT devices.

**Easier installation and management:** A WLAN requires less physical equipment than a wired network, which saves money, reduces installation time, and takes up less of a footprint in office settings.

**Scalability:** A WLAN is easy to scale. Adding users is as simple as assigning login credentials.

**Network management:** Nearly all management of a WLAN can be handled virtually. A single software interface can provide visibility, manage users, monitor network health, and collect data.

## **3.2 IEEE 802.11 Standard**

IEEE 802.11 is the set of technical guidelines for implementing Wi-Fi. Selling products under this trademark is overseen by an industry trade association by the name of the Wi-Fi Alliance.

IEEE 802.11 has its roots from a 1985 decision by the U.S. Federal Commission for Communication that opened up the ISM band for unlicensed use. The standard was formally released in 1997. That original standard was called IEEE 802.11-1997 and is now obsolete.

It's common to hear people refer to "802.11 standards" or the "802.11 family of standards." However, to be more precise, there is only one standard (IEEE 802.11-2007) but many amendments. Commonly known amendments include 802.11a, 802.11b, 802.11g, and 802.11n.

### 3.3 Bluetooth Technology

Bluetooth is a short-range wireless communication technology that allows devices such as mobile phones, computers, and peripherals to transmit data or voice wirelessly over a short distance. The purpose of Bluetooth is to replace the cables that normally connect devices, while still keeping the communications between them secure.

The "Bluetooth" name is taken from a 10th-century Danish king named Harald Bluetooth, who was said to unite disparate, warring regional factions. Like its namesake, Bluetooth technology brings together a broad range of devices across many different industries through a unifying communication standard.

Developed in 1994, Bluetooth was intended as a wireless replacement for cables. It uses the same 2.4GHz frequency as some other wireless technologies in the home or office, such as cordless phones and WiFi routers. It creates a 10-meter (33-foot) radius wireless network, called a personal area network (PAN) or piconet, which can network between two and eight devices. This short-range network allows you to send a page to your printer in another room, for example, without having to run an unsightly cable.

Bluetooth uses less power and costs less to implement than Wi-Fi. Its lower power also makes it far less prone to suffering from or causing interference with other wireless devices in the same 2.4GHz radio band. Bluetooth range and transmission speeds are typically lower than Wi-Fi (the wireless local area network that you may have in your home). Bluetooth v3.0 + HS i.e Bluetooth high-speed technology devices, can deliver up to 24 Mbps of data, which is faster than the 802.11b WiFi standard, but slower than wireless-a or wireless-g standards. As technology has evolved, however, Bluetooth speeds have increased.

The Bluetooth 4.0 specification was officially adopted on July 6, 2010. Bluetooth version 4.0 features include low energy consumption, low cost, multivendor interoperability, and enhanced range.



The hallmark feature enhancement to the Bluetooth 4.0 spec is its lower power requirements; devices using Bluetooth v4.0 are optimized for low battery operation and can run off of small coin-cell batteries, opening up new opportunities for wireless technology. Instead of fearing that leaving Bluetooth on will drain your cell phone's battery, for example, you can leave a Bluetooth v4.0 mobile phone connected all the time to your other Bluetooth accessories.

### **3.3.1 Connecting With Bluetooth**

Many mobile devices have Bluetooth radios embedded in them. PCs and some other devices that do not have built-in radios can be Bluetooth-enabled by adding a Bluetooth dongle, for example.

The process of connecting two Bluetooth devices is called "pairing." Generally, devices broadcast their presence to one another, and the user selects the Bluetooth device they want to connect to when its name or ID appears on their device. As Bluetooth-enabled devices proliferate, it becomes important that you know when and to which device you're connecting, so there may be a code to enter that helps ensure you're connecting to the correct device.

This pairing process can vary depending on the devices involved. For example, connecting a Bluetooth device to your iPad can involve different steps from those to pair a Bluetooth device to your car.

### **3.3.2 Bluetooth Limitations**

There are some downsides to Bluetooth. The first is that it can be a drain on battery power for mobile wireless devices like smartphones, though as the technology (and battery technology) has improved, this problem is less significant than it used to be.

Also, the range is fairly limited, usually extending only about 30 feet, and as with all wireless technologies, obstacles such as walls, floors, or ceilings can reduce this range further.

The pairing process may also be difficult, often depending on the devices involved, the manufacturers, and other factors that all can result in frustration when attempting to connect.

### **3.3.3 Security and Bluetooth**

Bluetooth is considered a reasonably secure wireless technology when used with precautions. Connections are encrypted, preventing casual eavesdropping from other devices nearby. Bluetooth devices also shift radio frequencies often while paired, which prevents an easy invasion.

Devices also offer a variety of settings that allow the user to limit Bluetooth connections. The device-level security of "trusting" a Bluetooth device restricts connections to only that specific device. With service-level security settings, you can also restrict the kinds of activities your device is permitted to engage in while on a Bluetooth connection. As with any wireless technology, however, there is always some security risk involved. Hackers have devised a variety of malicious attacks that use Bluetooth networking. For example, "bluesnarfing" refers to a hacker gaining authorized access to information on a device through Bluetooth; "bluebugging" is when an attacker takes over your mobile phone and all its functions.

For the average person, Bluetooth doesn't present a grave security risk when used with safety in mind (e.g., not connecting to unknown Bluetooth devices). For maximum security, while in public and not using Bluetooth, you can disable it completely.

### 3.4 Personal Area Network (PAN)

A personal area network (PAN) is the interconnection of information technology devices within the range of an individual person, typically within a range of 10 meters. For example, a person traveling with a laptop, a personal digital assistant (PDA), and a portable printer could interconnect them without having to plug anything in, using some form of wireless technology. Typically, this kind of personal area network could also be interconnected without wires to the Internet or other networks.

Conceptually, the difference between a PAN and a wireless LAN is that the former tends to be centered around one person while the latter is a local area network (LAN) that is connected without wires and serving multiple users.

In another usage, a personal area network (PAN) is a technology that could enable wearable computer devices to communicate with other nearby computers and exchange digital information using the electrical conductivity of the human body as a data network. For example, two people each wearing business card-size transmitters and receivers conceivably could exchange information by shaking hands. The transference of data through intra-body contact, such as handshakes, is known as **linkup**.

The human body's natural salinity makes it a good conductor of electricity. An electric field passes tiny currents, known as Pico amps, through the body when the two people shake hands. The handshake completes an electric circuit and each person's data, such as e-mail

addresses and phone numbers, are transferred to the other person's laptop computer or a similar device. A person's clothing also could act as a mechanism for transferring this data.

The concept of a PAN first was developed by Thomas Zimmerman and other researchers at M.I.T.'s Media Lab and later supported by IBM's Almaden research lab. In a research paper, Zimmerman explains why the concept might be useful:

As electronic devices become smaller, lower in power requirements, and less expensive, we have begun to adorn our bodies with personal information and communication appliances. Such devices include cellular phones, personal digital assistants (PDAs), pocket video games, and pagers. Currently there is no method for these devices to share data. Networking these devices can reduce functional I/O redundancies and allow new conveniences and services.

### **Discussion**

**Bluetooth is secure.** Discuss.

## **4.0 SELF-ASSESSMENT/EXERCISES**

Identify and explain the benefits of a WLAN

### **Answer**

Extended reach: WLANs enable computing to happen anywhere, even when carrying high data loads and advanced web applications.

Device flexibility: A WLAN supports use of a wide range of devices, such as computers, phones, tablets, gaming systems, and IoT devices.

Easier installation and management: A WLAN requires less physical equipment than a wired network, which saves money, reduces installation time, and takes up less of a footprint in office settings.

Scalability: A WLAN is easy to scale. Adding users is as simple as assigning login credentials.

Network management: Nearly all management of a WLAN can be handled virtually. A single software interface can provide visibility, manage users, monitor network health, and collect data.

A WLAN can be configured in one of two ways. Itemize and explain each of the ways.

**Answer:**

### **Infrastructure**

A home or office Wi-Fi network is an example of a WLAN set up in infrastructure mode. The endpoints are all connected and communicate with each other through a base station, which may also provide internet access.

A basic infrastructure WLAN can be set up with just a few parts: a wireless router, which acts as the base station, and endpoints, which can be computers, mobile devices, printers, and other devices. In most cases, the wireless router is also the internet connection.

### **Ad hoc**

In this setup, a WLAN connects endpoints such as computer workstations and mobile devices without the use of a base station. Use of Wi-Fi Direct technology is common for an ad hoc wireless network. An ad hoc WLAN is easy to set up and can provide basic peer-to-peer (P2P) communication. An ad hoc WLAN requires only two or more endpoints with built-in radio transmission, such as computers or mobile devices. After adjusting network settings for ad hoc mode, one user initiates the network and becomes visible to the others.

## **5.0 CONCLUSION**

Wireless LANs provide high speed data communication in small areas such as building or an office. WLANs allow users to move around in a confined area while they are still connected to the network. In some instances wireless LAN technology is used to save costs and avoid laying cable, while in other cases, it is the only option for providing high-speed internet access to the public. Whatever the reason, wireless solutions are popping up everywhere.

## **6.0 SUMMARY**

A wireless local-area network (WLAN) is a group of colocated computers or other devices that form a network based on radio transmissions rather than wired connections. A Wi-Fi network is a type of WLAN. A WLAN can be configured in one of two ways vis-à-vis as Infrastructure or Ad hoc. WLAN Architecture can take the form of a Station, Basic Service Set (BSS), Distribution system, Access point, Bridge and Endpoint. Many mobile devices have Bluetooth radios embedded in them. There are some downsides to Bluetooth such as being a drain on battery power for mobile wireless devices like smartphones, the range is fairly limited, obstacles such as walls, floors, or ceilings can reduce this range further

and the pairing process may also be difficult, often depending on the devices involved, the manufacturers, and other factors.

A Personal Area Network (PAN) is the interconnection of information technology devices within the range of an individual person, typically within a range of 10 meters. For example, a person traveling with a laptop, a personal digital assistant (PDA), and a portable printer could interconnect them without having to plug anything in, using some form of wireless technology.

## 7.0 REFERENCES/FURTHER READING

<https://www.google.com/search?q=bluetooth&oq=bluetooth&aqs=chrome..69i57j0l5.5255j0j4&sourceid=chrome&ie=UTF-8>

<https://www.techopedia.com/definition/5107/wireless-local-area-network-wlan>

<https://www.cisco.com/c/en/us/products/wireless/wireless-lan.html#~q-a>

<https://www.javatpoint.com/wireless-lan-introduction>

## UNIT 5 HIGH SPEED WIRELESS NETWORK: HIPERLAN

### CONTENTS

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 High Performance Local Area Network (HIPERLAN)
  - 3.2 How HIPERLAN works
  - 3.3 HIPERLAN Protocol Family
  - 3.4 Phases of the HIPERLAN1
  - 3.5 Wireless Asynchronous Mode (WATM)
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading

### 1.0 INTRODUCTION

High Performance Radio Local Area Network (HiperLAN) is one of the wireless networking protocols in Europe. It is an alternative to the Institute of Electrical and Electronics Engineers (**IEEE**) 802.11 standards. The HiperLAN standard was created by the European Telecommunications Standards Institute (ETSI). The original goal of the HiperLAN standard was to create a protocol that featured a higher data transfer rate than the 802.11 standard.

### 2.0 INTENDED LEARNING OUTCOMES (ILOS)

At the end of this unit, students will able to:

- Describe High Performance LAN.
- Explain the HIPERLAN Features.
- Identify and explain the Protocol Family

### 3.0 MAIN CONTENT

**3.1 HIPERLAN** stands for **high performance local area network**. It is a wireless standard derived from traditional LAN environments and can support multimedia and asynchronous data effectively at high data rates of 23.5 Mbps. It is primarily a European standard alternative for the IEEE 802.11 standards and was published in 1996. It is defined by the European Telecommunications Standards Institute (ETSI). It does not necessarily

require any type of access point infrastructure for its operation, although a LAN extension via access points can be implemented.

### 3.2 How HIPERLAN Works

Radio waves are used instead of a cable as a transmission medium to connect stations. Either, the radio transceiver is mounted to the movable station as an add-on and no base station has to be installed separately, or a base station is needed in addition per room. The stations may be moved during operation-pauses or even become mobile. The maximum data rate for the user depends on the distance of the communicating stations. With short distance (<50 m) and asynchronous transmission a data rate of 20 Mbit/s is achieved, with up to 800 m distance a data rate of 1 Mbit/s are provided. For connection-oriented services, e.g. video-telephony, at least 64 kbit/s are offered.

HIPERLAN uses cellular-based data networks to connect to an ATM backbone. The main idea behind HIPERLAN is to provide an infrastructure or ad-hoc wireless with low mobility and a small radius. HIPERLAN supports isochronous traffic with low latency.

### 3.3 HIPERLAN Protocol Family

The HiperLAN standard family has four different versions. The key feature of all four networks is their integration of time-sensitive data transfer services. Over time, names have changed and the former HIPERLANs 2, 3, and 4 are now called HiperLAN2, HIPERACCESS, and HIPERLINK.

	HIPERLAN 1	HIPERLAN 2	HIPERLAN 3	HIPERLAN 4
Application	wireless LAN	access to ATM fixed networks	wireless local loop	point-to-point wireless ATM connections
Frequency	5.1-5.3GHz			17.2-17.3GHz
Topology	decentralized ad-hoc/infrastructure	cellular, centralized	point-to-multipoint	point-to-point
Antenna	omni-directional		directional	
Range	50 m	50-100 m	5000 m	150 m
QoS	statistical	ATM traffic classes (VBR, CBR, ABR, UBR)		
Mobility	<10m/s		stationary	
Interface	conventional LAN	ATM networks		
Data rate	23.5 Mbit/s	>20 Mbit/s		155 Mbit/s
Power conservation	yes		not necessary	

Figure 5.5.1: *HIPERLAN Protocol Family*

### 3.3.1. HIPERLAN 1

Planning for the first version of the standard, called HiperLAN/1, started 1991, when planning of 802.11 was already going on. The goal of the HiperLAN was the high data rate, higher than 802.11. The standard was approved in 1996. The functional specification is EN300652, the rest is in ETS300836.

The standard covers the Physical layer and the Media Access Control part of the Data link layer like 802.11. There is a new sub layer called Channel Access and Control sub layer (CAC). This sub layer deals with the access requests to the channels. The accomplishing of the request is dependent on the usage of the channel and the priority of the request.

CAC layer provides hierarchical independence with Elimination-Yield Non-Preemptive Multiple Access mechanism (EY-NPMA). EY-NPMA codes priority choices and other functions into one variable length radio pulse preceding the packet data. EY-NPMA enables the network to function with few collisions even though there would be a large number of users. Multimedia applications work in HiperLAN because of EY-NPMA priority mechanism. MAC layer defines protocols for routing, security and power saving and provides naturally data transfer to the upper layers.

On the physical layer FSK and GMSK modulations are used in HiperLAN/1. HiperLAN features:

range 50 m

slow mobility (1.4 m/s)

- supports asynchronous and synchronous traffic

- sound 32 kbit/s, 10 ns latency

- video 2 Mbit/s, 100 ns latency

- data 10 Mbit/s

HiperLAN does not conflict with microwave and other kitchen appliances, which are on 2.4 GHz.

#### **Elimination-Yield Non-preemptive Priority Multiple Access (EY-NPMA)**

EY-NPMA is a contention based protocol that has been standardized under ETSI's HIPERLAN, a standard for wireless LANs. Unlike other contention based protocols, EY-NPMA provides excellent support for different classes of traffic regarding quality of service and demonstrates very low collision rates.



EY-NPMA is the medium access mechanism used by HIPERLAN Type 1. It uses active signaling.

**Active signaling** takes advantage of the fact that the current wireless technology enables us to have a slot time very much smaller than the average packet size. Each node that wants to access the medium transmits a non-data preamble pattern consisting of slots. This pattern is made up of alternating idle and busy periods of different lengths (measured in slots). Conflict resolution and collision detection is done during this preamble. The main rule is that if a node detects a signal during one of its listening periods in its pattern, it aborts and defers until the next cycle. Otherwise, the node transmits its packet at the end of the pattern transmission.

With EYNPMA, each station may attempt to access the channel when a condition out of a group of three is met. The three conditions are:

Channel free condition

Synchronized channel condition

Hidden elimination condition

The channel free condition occurs when the channel remains idle for at least a predefined time interval. A station willing to transmit senses the channel for this time interval, the station extends its period of sensing by a random number of slots (backoff). If the channel is still sensed as idle during the backoff period, the station commences transmitting. In both modes of operation unicast transmissions must get positively acknowledged or else the transmission is declared erroneous. Multicast and broadcast packets are not acknowledged.

The synchronized channel condition occurs when the channel is idle in the channel synchronization interval, which starts immediately after the end of the previous channel access cycle.

### 3.4 Phases of the HIPERLAN1 EY-NPMA Access Scheme

The synchronized channel access cycle consists of three distinct phases:

Prioritization

In prioritization, EY-NPMA recognizes five distinct priorities from 0 to 4, with 0 being the highest priority. The cycle begins with each station having data to transmit sensing the channel for as many slots as the priority of the packet in its buffer. All stations that successfully sense the channel as idle for the whole interval proceed to the next phase, the elimination phase.

Contention (Elimination and Yield) Transmission

During the elimination phase, each station transmits an energy burst of random length. These bursts ensure that only the stations having the highest priority data at a time proceed to the elimination phase. The length of the energy burst is a multiple of slots up to a predefined maximum. As

soon as a station finishes bursting, it immediately senses the channel. If the channel is sensed as idle, the station proceeds to the next phase. Otherwise, it leaves the cycle.

During the yield phase, the station that survived the two previous ones, back off for a random number of slots. The station that backs off for the shortest interval eventually gets access of the channel for data transmission. All other station sense the beginning of the transmission and refrain from transmitting.

Important features of the EY-NPMA

No preemption by frames with higher priority after the priority resolution possible.

Hierarchical independence of performance.

Fair contention resolution of frames with the same priority

### **3.5 Wireless Asynchronous Transfer Mode (WATM)**

The concept of WATM was first proposed in 1992 as pointed out in and now it is actively considered as a potential framework for next-generation wireless communication networks capable of supporting integrated, quality-of-service (QoS) based multimedia services. The strength of wireless ATM technology is said to be its ability to provide support for different protocols, such as ISDN1 and Internet protocols. As the volume of wireless traffic is increasing, so is the role of QoS support, which will become very important when multiple services are multiplexed into the same radio access technology. As QoS support is a fundamental property of ATM technology, WATM promises a solution for this requirement. ATM is a very complex system and modifications for wireless communication and mobility management is going to make it more difficult.

#### **Need for WATM**

The area of wireless transmission systems has been increasing rapidly. Mobility raises a new set of questions, techniques, and solutions. This growth will occur in an environment characterized by rapid development of end-user applications and services towards the Internet and broadband multimedia delivery over the evolving fixed-wired infrastructure. Therefore, new developments of wireless networks are needed to enable wireless technologies to interwork with existing wired networks. Therefore, in order for ATM to be successful, it must offer a wireless extension. Otherwise it cannot participate in the rapidly growing field of mobile communications.

As ATM networks scale well from local area networks (LANs) to wide area networks (WANs), and there is a need for mobility in local and wide area applications, a mobile extension of ATM is required in order to have

wireless access in local and wide environments. Many other wireless technologies, such as IEEE 802.11, typically only offer best-effort services or to some extent, time-bounded services. However, these services do not provide as many QoS parameters as ATM networks do. WATM could offer QoS for adequate support of multimedia data streams.

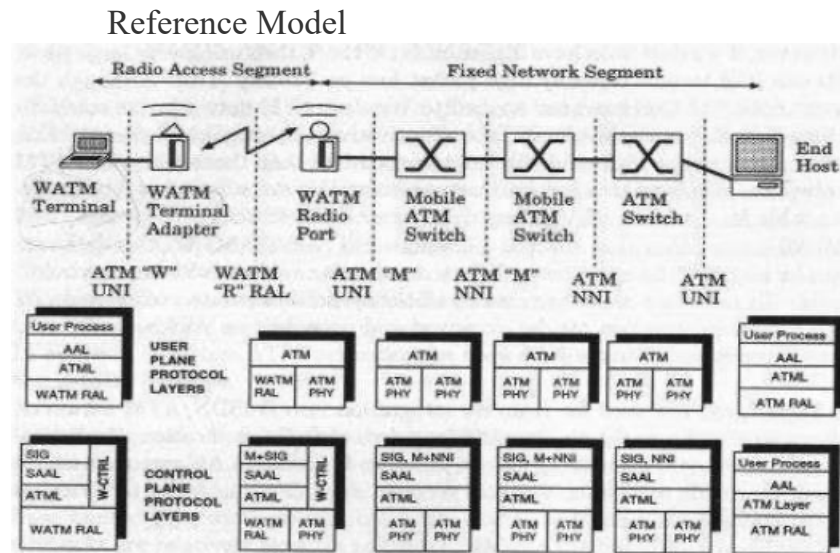


Figure 5.5.2: WATM Reference Model

The WATM system reference model, proposed by ATM Forum Wireless ATM (WATM) group, specifies the signaling interfaces among the mobile terminal, wireless terminal adapter, wireless radio port, mobile ATM switch and non-mobile ATM switch.

It also specifies the user and control planes protocol layering architecture. This model is commonly advocated by many communication companies, such as NEC, Motorola, NTT, Nokia, Symbionics, and ORL.

### Components of WATM

The major components of a Wireless ATM system are:

- WATM terminal
- WATM terminal adapter
- WATM radio port
- mobile ATM switch
- standard ATM network and
- ATM host.

The system reference model consists of a radio access segment and a fixed network segment. The fixed network is defined by "M (mobile ATM)" UNI and NNI interfaces while the wireless segment is defined by "R (Radio)" radio access layer (RAL) interface.

The "W" UNI is concerned with handover signaling, location management, wireless link and QoS control. The "R" RAL governs the signaling exchange between the WATM terminal adapter and the mobile base station. Hence, it concerns channel access, datalink control, meta-signaling, etc. The "M" NNI governs the signaling exchange between the WATM base station and a mobile capable ATM switch. It is also concerned with mobility-related signaling between the mobile capable ATM switches

b) The Broadband Radio Access Networks (**BRAN**)

The broadband radio access networks (BRAN), which have been standardized by the European Telecommunications Standards Institute (ETSI), could have been an RAL for WATM. The main motivation behind BRAN is the deregulation and privatization of the telecommunication sector in Europe. The primary market for BRAN includes private customers and small to medium-sized companies with Internet applications, multi-media conferencing, and virtual private networks. The BRAN standard and IEEE 802.16 have similar goals.

BRAN standardization has a rather large scope including indoor and campus mobility, transfer rates of 25–155 Mbit/s, and a transmission range of 50 m–5 km. Standardization efforts are coordinated with the ATM Forum, the IETF, other groups from ETSI, the IEEE etc. BRAN has specified four different network types:

**HIPERLAN 1:** This high-speed WLAN supports mobility at data rates above 20 Mbit/s. Range is 50 m, connections are multi-point-to-multi-point using ad-hoc or infrastructure networks □

**HIPERLAN/2:** This technology can be used for wireless access to ATM or IP networks and supports up to 25 Mbit/s user data rate in a point-to-multi-point configuration.

**HIPERACCESS:** This technology could be used to cover the last mile to a customer via a fixed radio link, so could be an alternative to cable modems or xDSL technologies.

**HIPERLINK:** To connect different HIPERLAN access points or HIPERACCESS nodes with a high-speed link, HIPERLINK technology can be chosen.

As an access network, BRAN technology is independent from the protocols of the fixed network. BRAN can be used for ATM and TCP/IP networks as illustrated in Figure 5.5.3. Based on possibly different physical layers, the DLC layer of BRAN offers a common interface to higher layers. To cover special characteristics of wireless links and to adapt directly to different higher layer network technologies, BRAN provides a network convergence sub layer. This is the layer which can be used by a wireless ATM network, Ethernet, Fire wire, or an IP network. In the case of BRAN as the RAL for WATM, the core ATM network would use services of the BRAN network convergence sub layer.

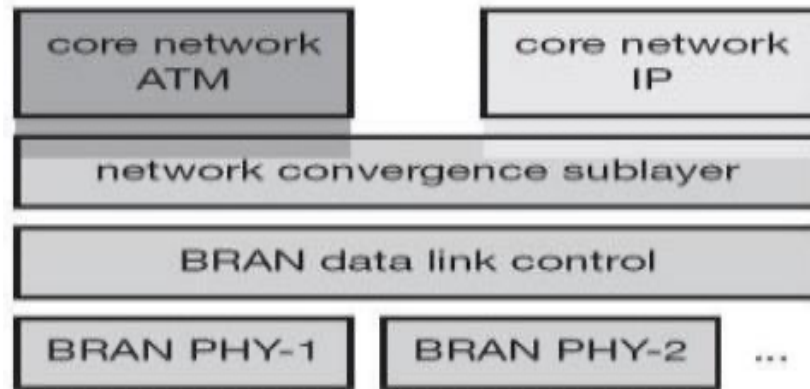


Figure 5.5.3: Layered Model of RAN Wireless Access Network

#### Discussion

BRAN has specified four different network types. Discuss.

### 4.0 SELF-ASSESSMENT/EXERCISES

The major components of a Wireless ATM system are:

Answer:

- a) WATM terminal
- b) WATM terminal adapter
- c) WATM radio port
- d) mobile ATM switch
- e) standard ATM network
- and f) ATM host.

Describe the Broadband Radio Access Networks (**BRAN**)

**Answer:**

The broadband radio access networks (BRAN), which have been standardized by the European Telecommunications Standards Institute (ETSI), could have been an RAL for WATM. The main motivation behind BRAN is the deregulation and privatization of the telecommunication sector in Europe. The primary market for BRAN includes private customers and small to medium-sized companies with Internet applications, multi-media conferencing, and virtual private networks. The BRAN standard and IEEE 802.16 have similar goals.

BRAN standardization has a rather large scope including indoor and campus mobility, transfer rates of 25–155 Mbit/s, and a transmission range of 50 m–5 km. Standardization efforts are coordinated with the ATM Forum, the IETF, other groups from ETSI, the IEEE etc.

## 5.0 CONCLUSION

At high data transmission rates, the packet transmission time of a local area network (LAN) could become comparable to or less than the medium propagation delay. The performance of many LAN schemes degrades rapidly when the packet transmission time becomes small comparative to the medium propagation delay.

## 6.0 SUMMARY

HIPERLAN stands for high performance local area network. It is a wireless standard derived from traditional LAN environments and can support multimedia and asynchronous data effectively at high data rates of 23.5 Mbps. HIPERLAN works using Radio waves instead of a cable as a transmission medium to connect stations. HiperLAN features are range 50 m, slow mobility (1.4 m/s), it supports asynchronous and synchronous traffic, sound 32 kbit/s, 10 ns latency, video 2 Mbit/s, 100 ns latency, data 10 Mbit/s and HiperLAN does not conflict with microwave and other kitchen appliances, which are on 2.4 GHz.

The concept of WATM was first proposed in 1992 as pointed out in and now it is actively considered as a potential framework for next-generation wireless communication networks capable of supporting integrated, quality-of-service (QoS) based multimedia services. The major components of a Wireless ATM system are: WATM terminal, WATM terminal adapter, WATM radio port, mobile ATM switch, standard ATM network and ATM host.

The broadband radio access networks (BRAN), which have been standardized by the European Telecommunications Standards Institute (ETSI), could have been an RAL for WATM. The main motivation behind BRAN is the deregulation and privatization of the telecommunication sector in Europe. The primary market for BRAN includes private customers and small to medium-sized companies with Internet applications, multi-media conferencing, and virtual private networks. The BRAN standard and IEEE 802.16 have similar goals.

BRAN standardization has a rather large scope including indoor and campus mobility, transfer rates of 25–155 Mbit/s, and a transmission range of 50 m–5 km. Standardization efforts are coordinated with the ATM Forum, the IETF, other groups from ETSI, the IEEE etc.

BRAN has specified four different network types: HIPERLAN 1, HIPERLAN/2, HIPERACCESS and HIPERLINK

## 7.0 REFERENCES/FURTHER READING

<https://dl.acm.org/doi/10.1145/103724.103726>

<https://cmd.inp.nsk.su/old/cmd2/manuals/networking/performance/ch01/ch01.htm>