

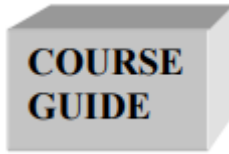


**NATIONAL OPEN UNIVERSITY OF NIGERIA**

**FACULTY OF SCIENCES  
DEPARTMENT OF COMPUTER SCIENCE**

**COURSE CODE: CIT421**

**COURSE TITLE: Net-Centric Computing**



# **CIT421**

## **NET-CENTRIC COMPUTING**

Course Team Oguntunde T (Ph.D) (Developer/Writer) - UI



## NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria Headquarters

University Village, Plot 91 Cadastral Zone, Nnamdi Azikiwe Expressway Jabi, Abuja

### LAGOS OFFICE

14/16 Ahmadu Bello Way

Victoria Island , Lagos

e-mail: [centralinfo@nou.edu.ng](mailto:centralinfo@nou.edu.ng)

URL: [www.nou.edu.ng](http://www.nou.edu.ng)

Published By:

National Open University of Nigeria

First Printed 2021

ISBN:

All Rights Reserved

## Table of Contents

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

## **Introduction**

### **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, 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 National Open University of Nigeria.

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:

1. Course Guide
2. Study Units
3. Presentation Schedule
4. Tutor-Marked Assignments
5. References/Further Reading

## Study Units

The study units in this course are as follows:

<b>Module 1</b>	<b>Net-Centric Computing Fundamentals</b>
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
<b>Module 2</b>	<b>Parallel Systems</b>
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
Unit 7	Optimization for Scalar Architectures
Unit 8	Modules for Parallel Computing
<b>Module 3</b>	<b>Distributed Systems</b>
Unit 1	Introduction to Distributed Systems
Unit 2	Characterization of Distributed Systems
Unit 3	Systems Models
Unit 4	Distributed Objects
Unit 5	Remote Methods Invocation



Unit 6                      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 the 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 and thereby assist you to evaluate 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 the early and 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 falling behind in your work.

### **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 (2) 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**

<b>Assignment</b>	<b>Marks</b>
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 schedule 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.

The following might be 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.

<b>CONTENTS</b>	<b>PAGE</b>
<b>Module 1    Net-Centric Computing Fundamentals.....</b>	<b>1</b>
Unit 1    Introduction to Distributed Computing.....	2
Unit 2    Mobile and Wireless Computing .....	9
Unit 3    Network Security .....	15
Unit 4    Client/ Server Computing (Web Application) .....	
Unit 5    Building Web Application	
 <b>Module 2    Parallel Systems .....</b>	 <b>25</b>
Unit 1    Introduction to Parallel Systems .....	26
Unit 2    Parallel Programming Models .....	
Unit 3    Message Passing Programming .....	37
Unit 4    Dependence Analysis .....	46
Unit 5    Open MP Programming .....	55
Unit 6    Evaluation of Programs .....	
Unit 7    Optimizations for Scalar Architecture .....	
Unit 8    Models for Parallel Computing .....	
 <b>Module 3    Distributed Systems .....</b>	 <b>62</b>
Unit 1    Introduction to Distributed Transactions.....	63
Unit 2    Characterization of Distributed Systems.....	72
Unit 3    Systems Models.....	84
Unit 4    Distributed Objects.....	94
Unit 5    Remote Method Invocation .....	
Unit 6    Using UML for Component Based Design .....	
 <b>Module 4    Distributed Transactions .....</b>	 <b>102</b>
Unit 1    Introduction to Distributed Transactions.....	103

Unit 2	Flat & Nested Distributed Transactions .....	110
Unit 3	Concurrency.....	121
Unit 4	Characteristics of Service Oriented Architecture (Hadoop & Spark).....	128

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

Unit 1	Introduction to Mobile & 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 Networks: HiperLAN	
	.....	

---

## Module 1: Net-Centric Computing Fundamentals

---

### Introduction of Module

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. At the center these solutions net-centric computing lies. 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.

## **Unit 1: Introduction to Distributed Computing**

- 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





## 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:

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

- a. Client-server — Clients contact the server for data, then format it and display it to the end-user.
- b. Three-tier — Information about the client is stored in a middle tier rather than on the client, to simplify application deployment.
- c. n-tier — Generally used when the server needs to forward requests to additional enterprise services on the network.
- d. 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 used distributed computing between data and metadata servers.

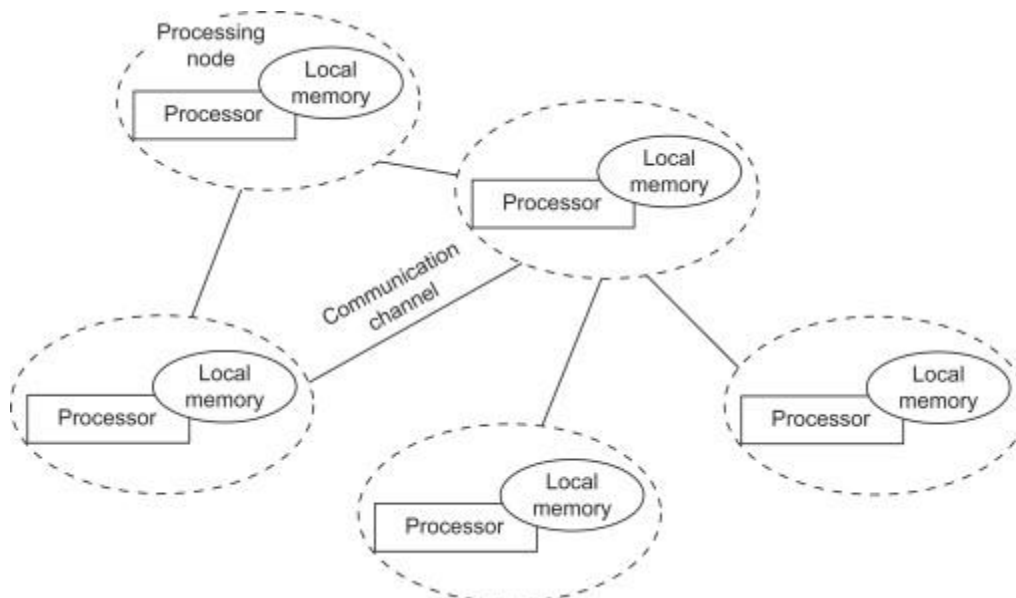


Figure 1: Distributed Computing Systems

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

1. Describe two Service Orientations

### 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.



### 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.



## **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 computing is human–computer interaction by which a computer is expected to be transported during normal usage. Mobile computing involves mobile communication, mobile hardware, and mobile software.



### 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

##### 3.1.1 Mobile Computing

Mobile 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

Mobile computing involves human-computer interaction by which a computer is expected to be transported during normal usage



Figure 2: Internet with Mobile Devices

#### 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 3: 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



Figure 4: Mobile Software

- 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

### 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:

- **Personal Digital Assistant (PDA)**



- 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.

Figure 5: Personal Data Assistant

- 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

- **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:
    - a. Google's Android
    - b. Apple's iOS
    - c. Nokia's Symbian
    - d. RIM's BlackBerry OS
    - e. Samsung's Bada
    - f. 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 6: Smart Phones

- **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 7: Ipads & PCs

### 3.6 Advantages

- **Location Flexibility**
  - This has enabled users to work from anywhere as long as there is a 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.
- **5G**
  - In telecommunications, **5G** is the fifth generation technology standard for broadband cellular networks, which cellular phone companies began deploying worldwide in 2019, and is the planned successor to the 4G networks which provide connectivity to most current cellphones. 5G networks are predicted to have more than 1.7 billion subscribers worldwide by 2025, according to the GSM Association.<sup>[1]</sup> Like its predecessors, 5G networks are cellular networks, in which the 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. The main advantage of the new networks is that they will have greater bandwidth, giving higher download speeds, eventually up to 10 gigabits per second (Gbit/s).<sup>[2]</sup> In addition to 5G being faster than existing networks, 5G can connect more different devices, and even if people are in crowded areas, the servers will be more unified, improving the quality of Internet services.<sup>[3]</sup> Due

to the increased bandwidth, it is expected 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

- **4G**
  - **4G** is the fourth generation of broadband cellular network technology, succeeding 3G, and preceding 5G. A 4G system must provide capabilities defined by ITU in IMT Advanced. Potential and current applications include amended mobile web access, IP telephony, gaming services, high-definition mobile TV, video conferencing, and 3D television.
  - The first-release WIMAX standard was commercially deployed in South Korea in 2006 and has since been deployed in most parts of the world.
  - The first-release Long Term Evolution (LTE) standard was commercially deployed in Oslo, Norway, and Stockholm, Sweden in 2009, and has since been deployed throughout most parts of the world. It has, however, been debated whether first-release versions should be considered 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.
- **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.
- **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 provides critical capabilities to military, civil and commercial users around the world
  - In addition, GPS is the backbone for modernizing the global air traffic system, weather, and location services.
- **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
- **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
- **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

1. Define Mobile Computing.

### Answer

**Mobile computing** is human–computer interaction by which a computer is expected to be transported during normal usage.

2. 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

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

.





## **7.0 References/Further Reading**

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

## 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 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.

It is predicted that by 2021, cybercrime damages will amount to an annual total world cost of \$6 *trillion*, even outpacing the yearly cost of damages attributed to natural disasters. And on an individual company level these damages can be just as catastrophic; the average cost of a [cyberattack](#) is currently \$1.67 million,<sup>2</sup> with operational and productivity loss, along with negative customer experience, being the primary consequences of suffering an attack.

### 3.2 Data as the Life-Blood of Businesses

Data is the lifeblood of any business. It supports business growth, carries vital resources and helps the organization stay healthy. And if data is blood, then the network is the beating heart that pumps it through the business system. But modern cyber threats are like vampires, doing everything possible to get at the blood that keeps any business going.

In order to defend against these vampiric threats and save your business from potentially millions of dollars in data loss, you need more than just a stake and some cloves of garlic; you need effective, robust network security and [network visibility](#).

### 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.

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

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

3.3.3 *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 exist to help organizations protect, not only its sensitive information, but also its overall performance, reputation and even its ability to stay in business. Continued operational ability and an intact reputation are two key benefits of effective network security.

Companies that fall prey to cyberattacks often find themselves crippled from the inside out, unable to deliver services or effectively address customer needs. Similarly, networks play a major role in internal company processes, and when they come under attack, those processes may grind to a halt, further hampering an organization's ability to conduct business or even resume standard operations. But perhaps even more damaging is the detrimental effect that a network breach can have on your business's *reputation*.

Given the rising tide of identity theft and other dangers related to the theft of personal information, many customers are already hesitant when it comes to sharing data with businesses. And if a cyberattack should occur, many of these customers are likely to withdraw in favor of more secure alternatives. The loss or corruption of valuable data, along with a significant disruption to customer services and internal process, topped off with reputational injury that may persist long after other damages have been repaired — it's not hard to see what's at stake when it comes to network security. In fact, it's been suggested that 66 percent of SMBs would have to shut down (either temporarily or permanently) after experiencing a data breach. And even larger, more established businesses may be unable to reclaim their former standing.

On the other hand, reliable tools in network security software and hardware, coupled with the right policies and strategies, can help ensure that when cyberattacks occur, their impact will be minimal.

### 3.5 Network Security Tools and Techniques

Enterprises' network face threats of all shapes and sizes, and thus should be prepared to defend, identify and respond to a full range of attacks. But the reality is that the biggest danger to most companies are not fly-by-night threat actors, but rather attackers that are well-funded and are targeting specific organizations for specific reasons. For that reason, your 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:

1. **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 also 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.

## 2. **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.

## 3. **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'll be alerted to them quickly enough to be able to respond.

## 4. **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.

## 5. **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.

## 6. **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.

## 7. **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.

**8. 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.

**9. 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.

**10. 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.

**11. 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.

**12. 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.

**13. 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.

**14. Wireless security**

Generally speaking, wireless networks are less secure than traditional networks. Thus, strict wireless security measures are necessary to ensure that threat actors aren't 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?



## 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

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.



## **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 clients 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 but sometimes they may reside in the same system.

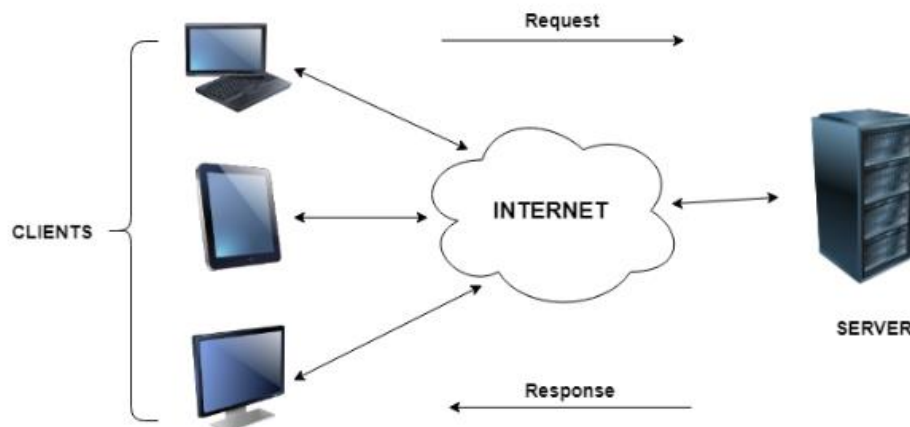


Figure 8: Client-Server Computing

### 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 Difference 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 subcategory of the peer to peer computing.

### 3.4 Advantages of Client Server Computing

The different advantages of client server computing are –

- 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 build on similar platforms yet they can easily facilitate the transfer of data.

### 3.5 Disadvantages of Client Server Computing

The different disadvantages of client server computing are –

- 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 of 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

1. Discuss the advantages of client Server computing
2. Identify the characteristics of client server computing?



#### 5.0 Conclusion

Client server and peer-to-peer computing are unique one from the other and so, have their advantages and disadvantages. 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.



#### 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
    - 3.3.9.1    A Database
    - 3.3.9.2    Database Types
  - 3.3.10    Build the Front End
    - 3.3.10.1            A 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)

#### 3.1.1 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.

#### 3.2 Prerequisites for Building a Web Application

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

1. Backend language (e.g. Python, Ruby) - control how your web app works
2. Web front end (HTML, CSS, Javascript) - for the look and feel of your web app
3. 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:

1. Learn the points above - there are lots of resources online to help you. I'd recommend [Codecademy](https://www.codecademy.com/) .

2. 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.

### 3.3 Building a Web Application

#### 3.3.1 Step 1 – Source an idea

Before making a web app, you must first understand what you intend on building, 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

#### 3.3.2 Step 2 – Market Research

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

1. If a similar product exists
2. 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:

1. [Google](#)
2. [Patent and trademark search](#)
3. [Betalist](#)
4. [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's 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:

1. 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.
2. [Google Trends](#) - A quick search of your web app idea will reveal relating trends.
3. SEO tool – I had recommend MOZ/Ahrefs. Google's keyword planner will suffice. Write a list of keywords relating to your web app. If it's 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.
4. Social Media - Jump over to Twitter/Facebook groups and present your idea to your target market.
5. 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's a market for your product. If there is a market for your product, and there's also established competition, it's important to research them.

### **3.3.3 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 1. 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're done, roll up your sleeves.

It's time to get creative! **Moving from the Ideation stage, to design stage.**

### 3.3.4 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.

### 3.3.5 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 login
- 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.

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

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

#### What is wireframing / prototyping

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

The decision is 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.

### 3.3.7 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.

### **3.3.8 Before Starting the development stage.**

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

1. 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.
2. 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.
3. 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!
4. 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!

### 3.3.10 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.

#### 3.3.10.1 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.

#### 3.3.10.2 Database types

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

##### a. 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.

## **b. 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's 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.

### 3.3.11 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.

#### 3.3.11.1 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:

1. A code editor, such as VS Code, Sublime Text
2. A compilation, and packaging framework:
  1. [Webpack](#)
  2. [Gulp](#)
  3. [Grunt](#)

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

3. A frontend framework (strictly not necessary, but highly advised unless you are an experienced frontend developer):
  1. [React](#)
  2. [Ember](#)
  3. [Vue](#)
  4. [Svelte](#)

The list is endless!

4. 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 dev 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 dev. 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

1. At dev 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.
2. 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 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.

### 3.3.12 Step 10 - Build your backend

#### What do we mean by the backend?

The backend is typically what manages your data. This refers to databases, servers, and everything the user can't 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.

If you feel confident, continue on.



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

1. Server Pages (Multiple Page Application)
2. Single Page Application

“But isn’t 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: aka 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 it’s related assets).

When choosing your backend:

- Go with what’s 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](#)

Login/User & Session Management

- How will users authenticate?
  - Username and password?
  - 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.

### 3.3.13 Step 11 - Host your web application

#### What is hosting

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: \

1. Buy a domain - [Namecheap](#)
2. Buy/Setup an SSL certificate - Let's Encrypt
3. Choose a cloud provider:
  1. Amazon
  2. MS Azure
  3. Google Cloud Platform
  4. Lower cost: Digital Ocean / Linode - if you are happy managing your own VMs
  5. 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 support, and are generally reliable options.

### 3.3.14 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're 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:

1. GitLab
2. Bitbucket
3. Jenkins

There are many of course.

To start with, you can just deploy directly from your machine of course.

And that's it. You have made a web application. Well done. You should take some time to celebrate this achievement. You're the proud owner of a new web app.



## Discussion

How can cybercrime be mitigated? Discuss

## 4.0 Self-Assessment/Exercise

1. Mention and explain the Database types.
2. What do we mean by the backend? Explain.



## 5.0 Conclusion

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

The backend is typically what manages your data. This refers to databases, servers, and everything the user can't 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 \(djangoproject.com\)](#)

[Studio | InVision \(invisionapp.com\)](#)

---

## Module 2: Parallel Systems

---

### Introduction of Module

In network security, threat prevention refers to policies and tools that protect your corporate network.

In the past, threat prevention primarily focused on the perimeter. With an increasing array of threats such as malware and ransomware arriving via email spam and phishing attacks, advanced threat prevention requires an integrated, multilayered approach to security. This may include tools for intrusion threat detection and prevention, advanced malware protection, and additional endpoint security threat prevention.

This module will consist of four units are follows

Unit 1: Firewalls

Unit 2: Virtual Private Networks (VPN)

Unit 3: Security Control Management

Unit 4: Hardware and Software Prevention

## UNIT 1 – Introduction to Parallel Systems

### Contents

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main content
  - 3.1 What is firewall?
    - 3.1.1 Characteristics of Firewall
    - 3.1.2 Needs for Firewall
    - 3.1.3 Limitation of Firewalls
  - 3.2 Type of Firewalls
  - 3.3 How firewall work
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary
- 7.0 References/Further Reading



## 1.0 Introduction

A firewall forms a barrier through which the traffic going in each direction must pass. A firewall security policy dictates which traffic is authorized to pass in each direction. Firewall may be designed to operate as a filter at the level of IP packets, or may operate at a higher protocol layer.



## 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, students will able to

- Demonstrate the concept of firewall
- Explore the importance of firewalls



## 3.0 Main Content



## Discussion

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

## 4.0 Self-Assessment/Exercises

### 1. What is Personal Firewall

#### Answer

A personal firewall controls the traffic between a personal computer or workstation on one side and the Internet or enterprise network on the other side. Personal firewall functionality can be used in the home environment and on corporate intranets. Typically, the personal firewall is a

software module on the personal computer. In a home environment with multiple computers connected to the Internet, firewall functionality can also be housed in a router that connects all of the home computers to a DSL, cable modem, or other Internet interface.

### **1. What are the benefits of host-based firewall?**

#### **Answer**

A host-based firewall is a software module used to secure an individual host. Such modules are available in many operating systems or can be provided as an add-on package. Like conventional stand-alone firewalls, host-resident firewalls filter and restrict the flow of packets. A common location for such firewalls is a server.

There are several benefits to the use of a server-based or workstationbased firewall:

- Filtering rules can be tailored to the host environment. Specific corporate security policies for servers can be implemented, with different filters for servers used for different application.
- Protection is provided independent of topology. Thus both internal and external attacks must pass through the firewall.
- Used in conjunction with stand-alone firewalls, the host-based firewall provides an additional layer of protection.

A new type of server can be added to the network, with its own firewall, without the necessity of altering the network firewall configuration.



## **5.0 Conclusion**

Internet connectivity is no longer optional for organizations. The information and services available are essential to the organization. Moreover, individual users within the organization want and need Internet access, and if this is not provided via their LAN, they will use dial-up capability from their PC to an Internet service provider (ISP). However, while Internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates a threat to the organization.





## **6.0 Summary**

Firewalls can be an effective means of protecting a local system or network of systems from network-based security threats while at the same time affording access to the outside world via wide area networks and the Internet.



## 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 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

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.



## 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, students will able to

- Explain the 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*

but there are also different combinations of both.

**Data-parallel programming model** is also among the most important ones as it was revived again with increasing popularity of MapReduce and GPGPU (General-Purpose computing on Graphics Processing Units).

- 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.

- b. In the message-passing programming model, tasks have private memories, and they communicate explicitly via message exchange. To exchange a message, each send 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).

### 3.2 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 a 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.

### 3.3 OpenMP

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 Table 5.4.

The Sections Construct

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

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 Table 5.5.

The For Construct

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

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.

### 3.4 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.

### 3.5 OpenCL

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.

### 3.6 The CUDA 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 such as Glift [2] and Scout [3]. 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

- a. *shared memory* and
- b. *message passing*

a. 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.

b. 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.



## 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

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:

- a. *shared memory* and
- b. *message passing*



## 7.0 References/Further Reading

1. Linköping University Electronic Press <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-40734>
2. [High-Performance Computing - an overview | ScienceDirect Topics](#)

## **UNIT 3**                      **Message Passing Programming**

### **Contents**

- 1.0    Introduction
- 2.0    Intended Learning Outcomes (ILOs)
- 3.0    Main content
  - 3.1    Message-Passing Programming
  - 3.2    Messages
  - 3.3    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

As long as you are carrying an access card or ID badge, it means that your office uses an access system. How does it really work? It's difficult since most people have never seen an access system. Most people believe it is just a card reader on the wall. Of course there is a little bit more to it in reality. It's not very difficult though, there are just a few parts behind the scenes that make the magic of easily unlocking a door every time.

This unit will give you a full and comprehensive understanding how access control systems, how it work, control list and AAA framework.



## 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, students will able to

- Explain the concept access control
- Manage access control



## 3.0 Main Content

### 3.1 The message-passing programming

### 3.2 Messages

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.

### 3.3 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.

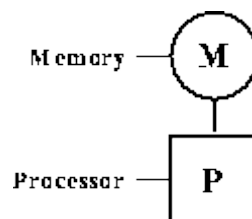


Figure 1: The sequential programming paradigm

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, but 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.

### 3.4 Single-Program-Multiple-Data (SPMD )

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. An example is the transformation or iterative update of a regular grid (perhaps in image processing or the numerical modelling of a problem from physics). The same operation is to be applied at every grid point. A typical parallel algorithm divides the grid into sub-grids and each sub-grid is processed in the same way.

Typically then, a programmer will want to write one program which is to be replicated across multiple processors (probably as many as possible!), often with a one-off controller process and possibly with other one-off processes like a name-server etc.

The acronym "SPMD" stands for single-program-multiple-data and refers to a restriction of the message-passing model which requires that all processes run the same executable. Some vendors provide parallel environments which only support SPMD parallel programs. In practice this is not usually a problem to the programmer, who can incorporate all the different types of process he or she requires into one overall executable. For example, here a controller process performs a different task (e.g. reading, checking and distributing initial data) to a worker process:

```
main(int argc, char **argv)
    if(process is to become a controller process)
        Controller( /* Arguments */);
```

```

else
    Worker( /* Arguments */ );
or in Fortran,
PROGRAM
IF (process is to become a controller process) THEN
    CALL CONTROLLER( /* Arguments */ )
ELSE
    CALL WORKER( /* Arguments */ )
ENDIF
END

```

Often, for related reasons of efficiency, some vendors do not allow time-sharing i.e. multiple processes per processor (some authorities understand the term "SPMD" to include this further restriction). The programmer should bear in mind that in a SPMD environment in which multiple processes per processor are not allowed, having special lightly-loaded one-off processes such as "controllers" or name-servers may be inefficient because a whole processor will be taken up with that process.

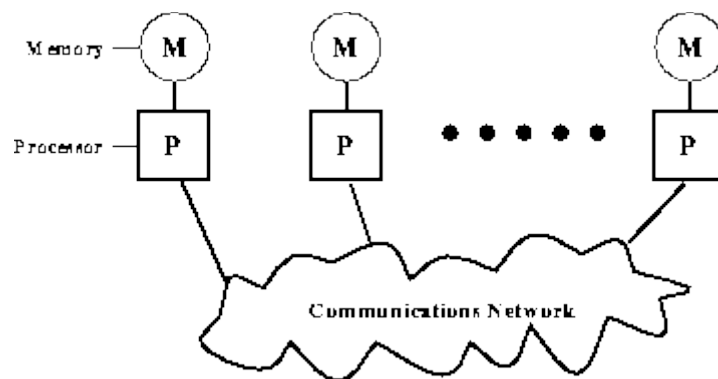


Figure 2: The message-passing programming paradigm.

The message-passing paradigm has become increasingly popular in recent times. One reason for this is the wide number of platforms which can support a message-passing model. Programs written in a message-passing style can run on distributed or shared-memory multi-processors, networks of workstations, or even uni-processor systems. The point of having the paradigm, just as in the sequential case, is that the programmer knows that his or her algorithms should in principle be portable to any architecture that supports a message-passing model. Message-passing is popular, not because it is particularly easy, but because it is so general.



## Discussion

Discuss Single Program multiple Data (SPMD).

## 4.0 Self-Assessment/Exercises

### 1. 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



## **7.0 References/Further Reading**

1. Neil MacDonald, Elspeth Minty, Tim Harding, Simon Brown, Edinburgh Parallel Computing Centre, The University of Edinburgh. (Course Notes)
2. [Advances in GPU Research and Practice | ScienceDirect](#)

## 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
- 4.0     Self-Assessment Exercises
- 5.0     Conclusion
- 6.0     Summary
- 7.0     References/Further Reading



### **1.0 Introduction**

Despite all security measures discussed above, an organization is prone to security breach if its employees lack security caution and awareness on their working computers. These computers contain sensitive organization details and information and therefore need to implement security measures to protect their data. Threats such as unauthorized access, internet fraudsters, viruses and spyware can cause a lot of damages to organization through employees' computer. This

unit will address measures on hardware and software based to prevent potential attack or damage of organization data.



## **2.0 Intended Learning Outcomes (ILOs)**

At the end of this unit, students will able to

- Improve their data privacy through hardware protection
- Protect their selves from software and internet attacks



## **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:

1. Identify the artefact with dependencies you want to analyze.
2. 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.
3. 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.

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.

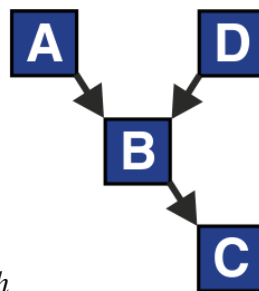


Figure 1. Dependency analysis graph



### Discussion

What is dependence Analysis and how are dependencies found?



### 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:

1. Identify the artefact with dependencies you want to analyze.
2. 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.
3. 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

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

**UNIT 5                      Open MP 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

Despite all security measures discussed above, an organization is prone to security breach if its employees lack security caution and awareness on their working computers. These computers contain sensitive organization details and information and therefore need to implement security measures to protect their data. Threats such as unauthorized access, internet fraudsters, viruses and spyware can cause a lot of damages to organization through employees' computer. This unit will address measures on hardware and software based to prevent potential attack or damage of organization data.



## 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, students will able to

- Improve their data privacy through hardware protection
- Protect their selves from software and internet attacks



## 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.

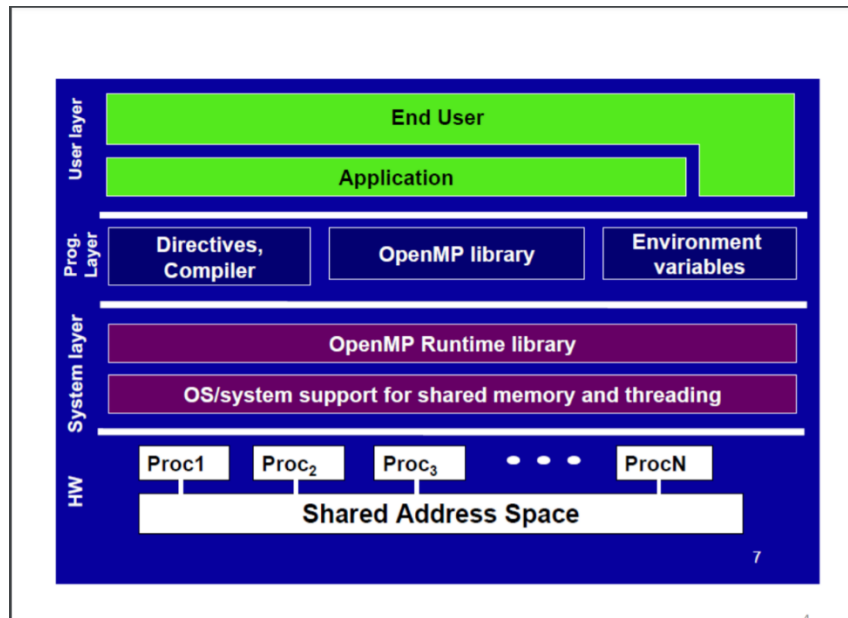


Figure 9: Open Specification for MultiProcessing (OpenMP)

### 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 processing that can be scheduled by an operating system. Thread model is an extension of the process model. Each process consists of multiple independent instruction streams (or threads) that are assigned computer resources by some scheduling procedure. 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**

Shared memory, thread-based parallelism. 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. Meant for distributed-memory parallel systems (it is designed for shared address spaced machines). 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.

### 3.6.4 JOIN

When the threads complete executing the statement in the parallel region construct, they synchronize and terminate, leaving only the master thread. Master thread is shown in red. 10 I/O • OpenMP does not specify parallel I/O. • It is up to the programmer to ensure that I/O is conducted correctly within the context of a multithreaded program. Memory Model • Threads can “cache” their data and are not required to maintain exact consistency with real memory all of the time. • When it is critical that all threads view a shared variable identically, the programmer is responsible for insuring that the variable is updated by all threads as needed.

### 3.7 A “Pragma”

It stands for “pragmatic information”. A pragma is a way to communicate the information to the compiler. The information is non-essential in the sense that the compiler may ignore the information and still produce correct object program.

## 3.8 OpenMP | Hello World program

**Prerequisite:** [OpenMP](#) | [Introduction](#) with [Installation](#) [Guide](#)

In this section, we will learn how to create a parallel **Hello World rogram** using [OpenMP](#).

### 3.8.1 Steps to Create a Parallel Program

1. **Include the header file:** We have to include the OpenMP header for our program along with the standard header files.

```
//OpenMP header
```

```
#include <omp.h>
```

2. **Specify the parallel region:**

In OpenMP, we need to mention the region which we are going to make it as parallel using the keyword **pragma omp parallel**. The **pragma omp parallel** is used to fork additional threads to carry out the work enclosed in the parallel. **The original thread will be denoted as the master thread with thread ID 0.** Code for creating a parallel region would be,

```
#pragma omp parallel
```

```
{
//Parallel region code
}
```

So, here we include

```
#pragma omp parallel
{
printf("Hello World... from thread = %d\n",
omp_get_thread_num());
}
```

**3. Set the number of threads:**  
we can set the number of threads to execute the program using the external variable.

```
export OMP_NUM_THREADS=5
```

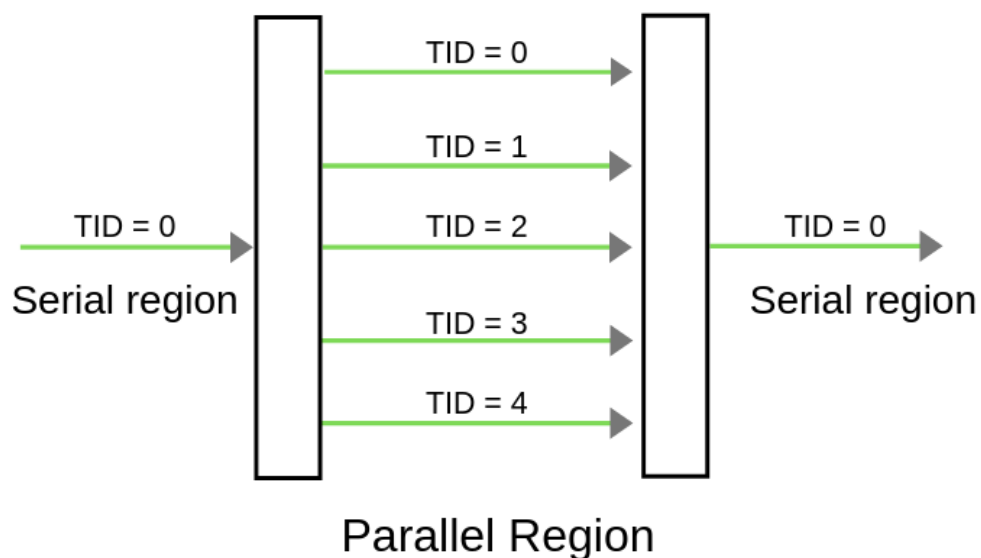


Figure 10: Diagram of parallel region

As per the above figure, once the compiler encounters the parallel regions code, the master thread (*thread which has thread id 0*) will fork into the specified number of threads. Here it will get forked into 5 threads because we will initialise the number of threads to be executed as 5, using the command `export OMP_NUM_THREADS=5`. Entire code within the parallel region will be executed by all threads concurrently.

Once the parallel region ended, all threads will get merged into the master thread.

#### 4. Compile and Run:

##### 4.1 Compile:

```
gcc -o hello -fopenmp hello.c
```

##### 4.2 Execute:

**./hello**

Below is the complete program with the output of the above approach:

**Program:** Since we specified the number of threads to be executed as 5, 5 threads will execute the same print statement at the same point of time. Here we cannot assure the order of execution of threads, i.e **Order of statement execution in the parallel region would not be the same for all executions**. In the below picture, while executing the program for first-time, thread-1 gets completed first whereas, in the second run, thread-0 completed first. `omp_get_thread_num()` will return the thread number associated with the thread.

##### 4.2.1 OpenMP Hello World program

```
// OpenMP program to print Hello World
// using C language
// OpenMP header
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    // Beginning of parallel region
    #pragma omp parallel
    {
        printf("Hello World... from thread = %d\n",
               omp_get_thread_num());
    }
    // Ending of parallel region
}
```

**Output:**

When run for 1st time:

```
akbar@ubuntu: ~/Desktop
File Edit View Search Terminal Help
akbar@ubuntu:~/Desktop$ export OMP_NUM_THREADS=5
akbar@ubuntu:~/Desktop$ gcc -o hello -fopenmp hello.c
akbar@ubuntu:~/Desktop$ ./hello
Hello World... from thread = 1
Hello World... from thread = 0
Hello World... from thread = 4
Hello World... from thread = 3
Hello World... from thread = 2
akbar@ubuntu:~/Desktop$
```

- **When run for multiple time:** Order of execution of threads changes every time.

```
akbar@ubuntu: ~/Desktop
File Edit View Search Terminal Help
akbar@ubuntu:~/Desktop$ ./hello
Hello World... from thread = 1
Hello World... from thread = 0
Hello World... from thread = 4
Hello World... from thread = 3
Hello World... from thread = 2
akbar@ubuntu:~/Desktop$ ./hello
Hello World... from thread = 0
Hello World... from thread = 4
Hello World... from thread = 3
Hello World... from thread = 2
Hello World... from thread = 1
akbar@ubuntu:~/Desktop$
```

**Discussion**

Is it possible to implement One Time Password on system logon security?



## 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

The objective of system security is the protection of information and property from theft, corruption and other types of damage, while allowing the information and property to remain accessible and productive. System security includes the development and implementation of security countermeasures. There are a number of different approaches to computer system security, including the use of a firewall, data encryption, passwords and biometrics.



## 7.0 References/Further Reading

1. [OpenMP | Introduction with Installation Guide](#)
2. In C/C++/Fortran, [parallel programming](#) can be achieved using [OpenMP](#).
3. [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**

Despite all security measures discussed above, an organization is prone to security breach if its employees lack security caution and awareness on their working computers. These computers contain sensitive organization details and information and therefore need to implement security measures to protect their data. Threats such as unauthorized access, internet fraudsters, viruses and spyware can cause a lot of damages to organization through employees' computer. This unit will address measures on hardware and software based to prevent potential attack or damage of organization data.



## **2.0 Intended Learning Outcomes (ILOs)**

At the end of this unit, students will able to

- Improve their data privacy through hardware protection
- Protect their selves from software and internet attacks



## **3.0 Main Content**

### **3.1 Programs Evaluation**

### **3.2 Definition of Program 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).



### 3.2.1 Purposes for Program Evaluation

- ❖ 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 (Short, Hennessy, & Campbell, 1996).
- ❖ 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; Short, Hennessy, & Campbell, 1996). Mutual misperceptions regarding the goals and process of evaluation can result in adverse attitudes (CDC, 1999; Chalk & King, 1998).

#### 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 (Rossi & Freeman, 1993; CDC, 1999). 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 (Burt, Harrell, Newmark, Aron, & Jacobs, 1997; Chalk & King, 1998). 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 (CDC, 1992).

### **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 (<http://www.wkkf.org/Publications/evalhdbk/default.htm>).

Sample question: What are the environmental barriers to accessing program services?

#### **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 (CDC, 1999). 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 (Rossi & Freeman, 1993). 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 (CDC, 1999).

### 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

Is it possible to implement One Time Password on system logon security?



## 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

The objective of system security is the protection of information and property from theft, corruption and other types of damage, while allowing the information and property to remain accessible and productive. System security includes the development and implementation of security countermeasures. There are a number of different approaches to computer system security, including the use of a firewall, data encryption, passwords and biometrics.



## 7.0 References/Further Reading

### References

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>).

**UNIT 7                      Optimization for Scalar Architectures****Contents**

- 1.0    Introduction
- 2.0    Intended Learning Outcomes (ILOs)
- 3.0    Main content
  - 3.1    Hardware Protection Mechanism
    - 3.1.1 CPU Protection
    - 3.1.2 Memory Protection
    - 3.1.3 I/O Protection
  - 3.2    Software and OS security
    - 3.2.1 Authentication
    - 3.2.2 One Time Password
    - 3.2.3 Program Threat
    - 3.2.4 System Threat
  - 3.3    Case/Example
- 4.0    Self-Assessment Exercises
- 5.0    Conclusion
- 6.0    Summary
- 7.0    References/Further Reading



## 1.0 Introduction

Despite all security measures discussed above, an organization is prone to security breach if its employees lack security caution and awareness on their working computers. These computers contain sensitive organization details and information and therefore need to implement security measures to protect their data. Threats such as unauthorized access, internet fraudsters, viruses and spyware can cause a lot of damages to organization through employees' computer. This unit will address measures on hardware and software based to prevent potential attack or damage of organization data.



## 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, students will able to

- Improve their data privacy through hardware protection
- Protect their selves from software and internet attacks



## 3.0 Main Content

### 3.1 Programs Evaluation

Contributed  
Robin Puett, MP

by

### 3.2 Definition of Program 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).

#### 3.2.1 Purposes for Program Evaluation



- ❖ 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 (Short, Hennessy, & Campbell, 1996).
- ❖ 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; Short, Hennessy, & Campbell, 1996). Mutual misperceptions regarding the goals and process of evaluation can result in adverse attitudes (CDC, 1999; Chalk & King, 1998).

#### 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 (Rossi & Freeman, 1993; CDC, 1999). 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 (Burt, Harrell, Newmark, Aron, & Jacobs, 1997; Chalk & King, 1998). 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 (CDC, 1992).

### **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 (<http://www.wkkf.org/Publications/evalhdbk/default.htm>).

Sample question: What are the environmental barriers to accessing program services?

#### **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 (CDC, 1999). 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 (Rossi & Freeman, 1993). 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 (CDC, 1999).

### 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

Is it possible to implement One Time Password on system logon security?



## 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

The objective of system security is the protection of information and property from theft, corruption and other types of damage, while allowing the information and property to remain accessible and productive. System security includes the development and implementation of security countermeasures. There are a number of different approaches to computer system security, including the use of a firewall, data encryption, passwords and biometrics.



## 7.0 References/Further Reading

### References

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>).

**UNIT 8                      Models for Parallel Computing****Contents**

- 1.0     Introduction
- 2.0     Intended Learning Outcomes (ILOs)
- 3.0     Main content
  - 3.1     Models for Parallel Computing
    - 3.1.2 Shared Memory Model
    - 3.1.3 Thread Model
    - 3.1.4 Message Passing Model
    - 3.1.5 Data Parallel Model
    - 3.1.6 Hybrid Model
    - 3.1.7 Single Program Multiple Data (SPMD)
    - 3.1.8 Multiple Program Multiple Data (MPMD)
- 4.0     Self-Assessment Exercises
- 5.0     Conclusion
- 6.0     Summary
- 7.0     References/Further Reading



## 1.0 Introduction

Despite all security measures discussed above, an organization is prone to security breach if its employees lack security caution and awareness on their working computers. These computers contain sensitive organization details and information and therefore need to implement security measures to protect their data. Threats such as unauthorized access, internet fraudsters, viruses and spyware can cause a lot of damages to organization through employees' computer. This unit will address measures on hardware and software based to prevent potential attack or damage of organization data.



## 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, students will able to

- Improve their data privacy through hardware protection
- Protect their selves from software and internet attacks



## 3.0 Main Content

### 3.1 Models for Parallel Computing

A Model for Parallel Programming is an abstraction and is machine architecture independent. A model can be implemented on various hardware and memory architectures. There are several parallel programming models like Shared Memory model, Threads model, Message Passing model, Data Parallel model and Hybrid model etc. As these models are hardware independent, the models can (theoretically) be implemented on a number of different underlying types of hardware. The decision about which model to use in an application is often a combination of a number of factors including the available resources and the nature of the application. There is no universally best implementation of a model, though there are certainly some implementations of a model better than others. Next, we discuss briefly some popular models.

#### 3.1.2 Shared Memory Model

Recent Trends in Parallel Computing in the shared-memory programming model, tasks share a common address space, which they read and write asynchronously. Various mechanisms



such as locks / semaphores may be used to control access to the shared memory. An advantage of this model from the programmer's point of view is that program development can often be simplified. An important disadvantage of this model (in terms of performance) is that for this model, data management is difficult.

### **3.1.3 Threads Model**

In this model a single process can have multiple, concurrent execution paths. The main program is scheduled to run by the native operating system. It loads and acquires all the necessary softwares and user resources to activate the process. A thread's work may best be described as a subroutine within the main program. Any thread can execute any one subroutine and at the same time it can execute other subroutine. Threads communicate with each other through global memory. This requires Synchronization constructs to insure that more than one thread is not updating the same global address at any time. Threads can be created and destroyed, but the main program remains live to provide the necessary shared resources until the application has completed. Threads are commonly associated with shared memory architectures and operating systems.

### **3.1.4 Message Passing Model**

In the message-passing model, there exists a set of tasks that use their own local memories during computation. Multiple tasks can reside on the same physical machine as well across an arbitrary number of machines. Tasks exchange data by sending and receiving messages. In this model, data transfer usually requires cooperation among the operations that are performed by each process. For example, a send operation must have a matching receive operation.

### **3.1.5 Data Parallel Model**

In the data parallel model, most of the parallel work focuses on performing operations on a data set. The data set is typically organised into a common structure, such as an array or a cube. A set of tasks work collectively on the same data structure with each task working on a different portion of the same data structure. Tasks perform the same operation on their partition of work, for example, "add 3 to every array element" can be one task. In shared memory architectures, all tasks may have access to the data structure through the global memory. In the distributed memory architectures, the data structure is split up and data resides as "chunks" in the local memory of each task.

### 3.1.6 Hybrid model

The hybrid models are generally tailor-made models suiting to specific applications. Actually, these fall in the category of mixed models. Such type of application-oriented models keep cropping up.

Other parallel programming models also exist, and will continue to evolve corresponding to new applications. In this types of models, any two or more parallel programming models are combined. Currently, a common example of a hybrid model is the combination of the message passing model (MPI) with either the threads model (POSIX threads) or the shared memory model (OpenMP). This hybrid model lends itself well to the increasingly common hardware environment of networked SMP machines. Another common example of a hybrid model is combining data parallel model with message passing model. As mentioned earlier in the data parallel model, data parallel implementations (F90, HPF) on distributed memory architectures actually use message passing to transmit data transparently between tasks and the programmer.

### 3.1.7 Single Program Multiple Data (SPMD)

SPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models. A single program is executed by all tasks simultaneously. SPMD programs usually have the necessary logic programmed into them to allow different tasks to branch or conditionally execute only those parts of the program they are designed to execute. That is, tasks do not necessarily have to execute the entire program, they may execute only a portion of it. In this model, different tasks may use different data.

### 3.1.8 Multiple Program Multiple Data (MPMD)

Like SPMD, MPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models. MPMD applications typically have multiple executable object files (programs). While the application is being run in parallel, each task can be executed on the same or different program. In this model, all tasks may use different data.



## Discussion

Is it possible to implement One Time Password on system logon security?



## 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

The objective of system security is the protection of information and property from theft, corruption and other types of damage, while allowing the information and property to remain accessible and productive. System security includes the development and implementation of security countermeasures. There are a number of different approaches to computer system security, including the use of a firewall, data encryption, passwords and biometrics.



## 7.0 References/Further Reading

### *References*

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>).

---

**Module 3: Distributed Systems**

---

**Introduction of Module**

Digital devices such as cell phones, tablets, gaming consoles, laptop and desktop computers have become indispensable part of the modern society. With the proliferation of these devices in our everyday lives, there is the tendency to use information derived from them for criminal activities. Crimes such as fraud, drug trafficking, homicide, hacking, forgery, and terrorism often involve computers. To fight computer crimes, digital forensics (DF) originated in law enforcement, computer security, and national defense. Law enforcement agencies, financial institutions, and investment firms are incorporating digital forensics into their infrastructure. Digital forensics is used to help investigate cybercrime or identify direct evidence of a computer-assisted crime. The concept of digital forensics dates back to late 1990s and early 2000s when it was considered as computer forensics. The legal profession, law enforcement, policy makers, the business community, education, and government all have a vested interest in DF. Digital forensics is often used in both criminal law and private investigation. It has been traditionally associated with criminal law. It requires rigorous standards to stand up to cross-examination in court.

This module will consist of four units as follows

Unit 1: Computer Forensics

Unit 2: Network, Disk, Malware and Database Forensics

Unit 3: Email, Memory and Mobile Forensics

Unit 4: Malware Analysis

## **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    Elements of Distributed Systems
- 4.0    Self-Assessment Exercises
- 5.0    Conclusion
- 6.0    Summary
- 7.0    References/Further Reading



## **1.0 Introduction**

Computer Forensics is a scientific method of investigation and analysis in order to gather evidence from the digital devices or computer networks and components which is suitable for presentation in a court of law or legal body. It involves performing a structured investigation while maintaining a documented chain of evidence to find out exactly what happened on a computer and who was responsible for it. Crimes committed within electronic or digital domains, particularly within cyberspace, have become extremely common these days. Criminals are using technology to a great extent in committing various digital offences and creating new challenges for law enforcement agents, attorneys, judges, military, and security professionals. Digital forensics has become an incredibly useful and invaluable tool in the detection of criminal activities, identifying and solving computer-based and computer-assisted crimes.



## **2.0 Intended Learning Outcomes (ILOs)**

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

- Explain the concept of computer forensics
- Explain the characteristics of digital forensics
- Explain digital forensics procedure
- Explain the advantages of computer forensics
- Disadvantages of computer forensics



## 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 have 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 Software's cryptocurrency systems, scientific simulations, blockchain technologies and AI platforms all use Distributed Systems platforms.

### 3.2 Elements of Distributed Systems

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.



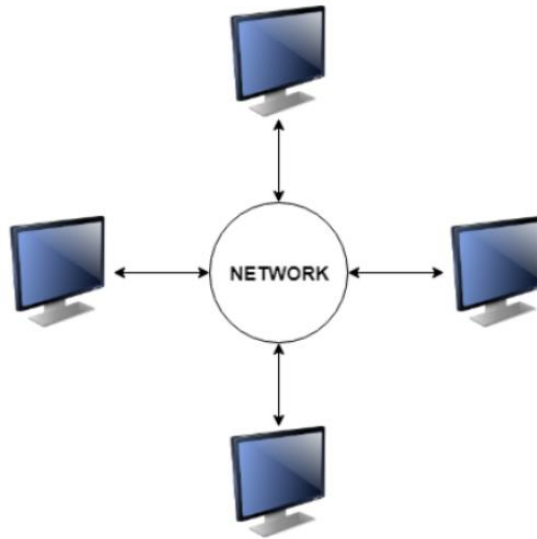


Figure 11: Distributed Operating Systems

## 4.0 Self-Assessment Exercises

2. Define Computer Forensics and what are the Characteristics of Digital Forensics

### Answer

Computer forensics is the process of extracting data and information from computer systems to function as digital evidence for civic purposes, or in most cases to prove and legally impeach cybercrime.

### Characteristics of Digital Forensics

- Identification:
- Preservation
- Analysis
- Documentation
- Presentation



## 5.0 Conclusion

Digital forensics involves the process of identifying, collecting, acquiring, preserving, analysing, and presenting of digital evidence. Digital evidence must be authenticated to ensure

its admissibility in a court of law. Ultimately, the forensic artefacts and forensic methods used (e.g., static or live acquisition) depend on the device, its operating system, and its security features.



## 6.0 Summary

In this unit, we have been able to outline computer forensics history, characteristics of digital forensics, digital forensics procedure, advantages of computer forensics and disadvantages of computer forensics



## 7.0 References/Further Reading

<https://www.techtarget.com/searchsecurity/definition/computer-forensics>

Årnes, A. (Ed.). (2017). *Digital forensics*. John Wiley & Sons.

Kävrestad, J. (2020). *Fundamentals of Digital Forensics*. Springer International Publishing.

Easttom, C. (2021). *Digital Forensics, Investigation, and Response*. Jones & Bartlett Learning.

Nelson, B., Phillips, A., & Steuart, C. (2019). Guide to Computer Forensics and Investigations, 2019. *structure*, 10, 26.

Dafoulas, G. A., & Neilson, D. (2019, October). An overview of digital forensics education. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)* (pp. 1-7). IEEE.

Pachghare, V. K. (2019). *Cryptography and information security*. PHI Learning Pvt. Ltd..

Lin, X., Lin, X., & Lagerstrom-Fife. (2018). *Introductory Computer Forensics*. Springer International Publishing.

Whitman, M. E., & Mattord, H. J. (2021). *Principles of information security*. Cengage learning.

## Unit 2                      Characterization of Distributed Systems

### Contents

- 1.0 Introduction
- 2.0 Intended Learning Outcomes (ILOs)
- 3.0 Main Content
  - 3.1 Patterns in a Distributed Systems
  - 3.2 Benefits of Distributed Systems
  - 3.3 Challenges of Distributed Systems
    - 3.3.1 Increased Opportunities for Failures
    - 3.3.2 Synchronization Process Challenges
    - 3.3.3 Imperfect Scalability
    - 3.3.4 More Complex Security
    - 3.3.5 Increased Complexity
  - 3.4 Risks of Distributed Systems
    - 3.4.1 Security
    - 3.4.2 Risks of Network Failures
    - 3.4.3 Governance and Control Issues
  - 3.5 Access Control in Distributed Systems
  - 3.6 Application of Distributed Systems
  - 3.7 Types of Distributed Systems
  - 3.8 Factors Determining the Need for a Distributed Systems
- 4.0 Self-Assessment Exercises
- 5.0 Conclusion
- 6.0 Summary

## 7.0 References/Further Reading

**1.0 Introduction**

Digital Forensics is defined as the process of preservation, identification, extraction, and documentation of computer evidence which can be used by the court of law. It is a science of finding evidence from digital media like a computer, mobile phone, server, or network. It provides the forensic team with the best techniques and tools to solve complicated digital-related cases. Digital Forensics helps the forensic team to analyze, inspect, identify, and preserve the digital evidence residing on various types of electronic devices.

**2.0 Intended Learning Outcomes (ILOs)**

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

- Explain the concept of Disk Forensics
- Explain the process of Disk Forensics
- Explain Network Forensics procedure
- Explain Network Forensics
- Explain Examinations of Network Forensics
- Explain Malware Forensics

**3.0 Main Content****3.1 Patterns in a Distributed System**

A **Software Design Pattern** is a programming language defined as an ideal solution to a contextualized programming problem. Patterns are reusable solutions to common problems that represent the best practices available at the time. Patterns do not provide finished code, they provide replication capabilities but offer guidance on how to solve a certain issue or implement a needed feature. When thinking about the challenges of a distributed computing platform, the trick is to:

- break it down into a series of interconnected patterns
- simplifying the system into smaller, more manageable and more easily understood components which helps abstract a complicated architecture

Patterns are commonly used to describe distributed systems, such as:

- Command and Query Responsibility Segregation (CQRS) and
- two-phase commit (2PC)

Different combinations of patterns are used to design distributed systems, and each approach has unique benefits and drawbacks.

### 3.2 Benefits of Distributed Systems

- **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

### 3.3 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 vis-à-vis:

#### 3.3.1 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
- Distributed systems are designed to be fault tolerant, that fault tolerance isn't automatic or fool-proof

### **3.3.2 Synchronization process challenges:**

- Distributed systems work without a global clock
- It requires 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

### **3.3.3 Imperfect scalability:**

- Doubling the number of nodes in a distributed system does not 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.

### **3.3.4 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.

### **3.3.5 Increased complexity:**

- Distributed systems are more complex to design, manage and understand than traditional computing environments.

## **3.4 Risks of Distributed Systems**

The challenges of distributed systems as outlined above create a number of correlating risks such as:

### **3.4.1 Security:**

- Distributed systems are as vulnerable to attack as any other system
- their distributed nature creates a much larger attack surface that exposes organizations to threats.

### **3.4.2 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.

### **3.4.3 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.

### **3.4.4 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.

## **3.5 Access Control in Distributed Systems**

Administrators use a variety of approaches to manage access control in distributed computing environments, ranging from:

- traditional access control lists (ACLs) to
- role-based access control (RBAC)
- One of the most promising access control mechanisms for distributed systems is attribute-based access control (ABAC), which controls access to objects and processes using rules that include:
  - information about the user
  - the action requested and
  - the environment of that request.
- Administrators can also refine these types of roles to restrict access to certain times of day or certain locations.

## **3.6 Application of distributed systems**



- Distributed systems are used when a workload is too great for a single computer or device to handle
- They are also helpful in situations when the workload is subject to change, such as e-commerce traffic on Cyber Monday
- Distributed system is used virtually for every internet-connected web application that exists is built on top of some form of.
- Some of the most common examples of distributed systems:
  - Telecommunications networks (including cellular networks and the fabric of the internet)
  - Graphical and video-rendering systems
  - Scientific computing, such as protein folding and genetic research
  - Airline and hotel reservation systems
  - Multiuser video conferencing systems
  - Cryptocurrency processing systems (e.g. Bitcoin)
  - Peer-to-peer file-sharing systems (e.g. BitTorrent)
  - Distributed community compute systems (e.g. Folding@Home)
  - Multiplayer video games
  - Global, distributed retailers and supply chain management (e.g. Amazon)

### 3.7 Types of Distributed Deployments

- Distributed deployments can range from tiny:
  - single department deployments on local area networks to
  - large-scale, global deployments
- In addition to their size and overall complexity, organizations can consider deployments based on:
  - the size and capacity of their computer network
  - the amount of data they'll consume
  - how frequently they run processes
  - whether they'll be scheduled or ad hoc
  - the number of users accessing the system
  - capacity of their data center and the necessary data fidelity and availability requirements.

Based on these considerations, distributed deployments are categorized as:

- Departmental, small enterprise

- Medium enterprise or large enterprise
- Distributed systems can also evolve over time, transitioning from departmental to small enterprise as the enterprise grows and expands.

### **3.8 Factors Determining the need for a Distributed Systems**

- They're essential to the operations of wireless networks, cloud computing services and the internet
- Distributed systems provide scalability and improved performance in ways that monolithic systems cannot
- Distributed systems can offer features that would be difficult or impossible to develop on a single system
- if the Master Catalog does not see the segment bits it needs for a restore, it can ask the other off-site node or nodes to send the segments
- Virtually everything you do now with a computing device takes advantage of the power of distributed systems
  - sending an email
  - playing a game or
  - reading this article on the web.

### **4.0 Self-Assessment Exercises**

1. Explain the following
  - i. Disk Forensics
  - ii. Network Forensics
  - iii. Methods of Network Forensic
  - iv. Database Forensics
  - v. Malware Forensics
3. Explain the symptoms of infected systems
4. Explain different ways malware can get into system



## 5.0 Conclusion

The forensic examination of electronic systems has undoubtedly been a huge success in the identification of cyber and computer-assisted crime. Organisations are placing an increasing importance on the need to be equipped with appropriate incident management capabilities to handle misuse of systems. Computer forensics is an invaluable tool in the process. The domain of computer forensics has grown considerably in the last decade. Driven by industry, focus was initially placed upon developing tools and techniques to assist in the practical application of the technology



## 6.0 Summary

- Digital Forensics is the preservation, identification, extraction, and documentation of computer evidence which can be used in the court of law
- Process of Digital forensics includes 1) Identification, 2) Preservation, 3) Analysis, 4) Documentation and, 5) Presentation
- Different types of Digital Forensics are Disk Forensics, Network Forensics, Wireless Forensics, Database Forensics, Malware Forensics, Email Forensics, Memory Forensics, etc.

Digital forensic Science can be used for cases like 1) Intellectual Property theft, 2) Industrial espionage 3) Employment disputes, 4) Fraud investigations



## 7.0 References/Further Reading

<https://www.techtarget.com/searchsecurity/definition/computer-forensics>

Kävrestad, J. (2020). *Fundamentals of Digital Forensics*. Springer International Publishing.

Easttom, C. (2021). *Digital Forensics, Investigation, and Response*. Jones & Bartlett Learning.

Nelson, B., Phillips, A., & Steuart, C. (2019). Guide to Computer Forensics and Investigations, 2019. *structure*, 10, 26.

Dafoulas, G. A., & Neilson, D. (2019, October). An overview of digital forensics education. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)* (pp. 1-7). IEEE.

Pachghare, V. K. (2019). *Cryptography and information security*. PHI Learning Pvt. Ltd..

Lin, X., Lin, X., & Lagerstrom-Fife. (2018). *Introductory Computer Forensics*. Springer International Publishing.

Whitman, M. E., & Mattord, H. J. (2021). *Principles of information security*. Cengage learning.

## Unit 3                      Systems Models

### Contents

- 1.0     Introduction
- 2.0     Intended Learning Outcomes (ILOs)
- 3.0     Main Content
  - 3.1 Models and Architectures of Distributed Systems
  - 3.2 Characteristics of a Distributed Systems
  - 3.3 Distributed tracing
- 4.0     Self-Assessment Exercises
- 5.0     Conclusion
- 6.0     Summary
- 7.0     References/Further Reading



### 1.0     Introduction

The Internet is a very easy way to reach any system. If confidential data is not properly protected, then it becomes opens to vulnerable access and misuse. Cyber-crime can cause varying degrees of damage by hackers. So, detailed forensic analysis is required to come to a conclusion about an incident and to prove or disprove someone's guilt. Some criminal activities like child pornography, hacking, and identity theft can be traced and the criminals can be punished if proper evidence is found against them. Email communication is also on target. Because it is one of the most popular and commonalty used means of online communication, for both prospects individuals and businesses, emails are normally used by organizations to exchange most simple information, such as meeting schedules, document distribution and some sensitive information



### 2.0 Intended Learning Outcomes (ILOs)

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

- Explain the concept of Email Forensics
- Explain the concept of Memory Forensics

- Explain the concept of Mobile Phone Forensics
- Explain ddigital Forensic Examination Process



### **3.0 Main Content**

#### **3.1 Models and Architectures of Distributed Systems**

There are two Models and Architecture of distributed systems:

- 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
  - 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 and so they are a part of distributed systems.
  - 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:
  - workloads are distributed among hundreds or thousands of computers all running the same software
  - The peer to peer systems contains nodes that are equal participants in data sharing
  - All the tasks are equally divided between all the nodes
  - The nodes interact with each other as required as they share resources
  - This is done with the help of a network.

The most common forms of distributed systems in the enterprise today are those that operate over the web. They hand off workloads to dozens of cloud-based virtual server instances that are created as needed, then terminated when the task is complete.

#### **3.2 Characteristics of a Distributed system**

- **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're 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.

### 3.3 Distributed Tracing

Distributed tracing is a method for monitoring applications — typically those built on a micro-services architecture — which are commonly deployed on distributed systems. Distributed tracing is essentially a form of distributed computing in that 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

- For example, 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 instance, a tracing system monitors this process step by step, helping a developer to uncover bugs, bottlenecks, latency or other problems with the application. 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 micro-services architecture

and running on a system as large and complex as a globally distributed system environment would be impossible to monitor effectively.

## 4.0 Self-Assessment Exercises

### 1. Explain email forensics

Email forensics is the analysis of source and content of the email message, identification of sender and receiver, date and time of email and the analysis of all the entities involved. Email forensics also reforms to the forensics of client or server systems suspected in an email forgery.

### 2. What is the purpose of email header analysis

Email header analysis helps in identifying most of the email related crimes like spear phishing, spamming, email spoofing etc. Spoofing is a technique using which one can pretend to be someone else, and a normal user would think for a moment that it's his friend or some person he already knows

### 3. List the common techniques used in email forensic investigation

- Header Analysis
- Server investigation
- Network Device Investigation
- Sender Mailer Fingerprints
- Software Embedded Identifiers



## 5.0 Conclusion

Email evidence often plays a pivotal role in digital forensics investigations and eDiscovery. When preserving emails from the cloud, forensics experts have to consider issues such as multi-factor authentication, running-in-place searches on the server before the acquisition, handling server errors and throttling, privacy issues, and time constraints.





## 6.0 Summary

In this unit, we have been able to outline email forensics, email header analysis, mobile forensics and mobile device forensics examination process.



## 7.0 References/Further Reading

<https://www.techtarget.com/searchsecurity/definition/computer-forensics>

Årnes, A. (Ed.). (2017). *Digital forensics*. John Wiley & Sons.

Kävrestad, J. (2020). *Fundamentals of Digital Forensics*. Springer International Publishing.

Easttom, C. (2021). *Digital Forensics, Investigation, and Response*. Jones & Bartlett Learning.

Nelson, B., Phillips, A., & Steuart, C. (2019). Guide to Computer Forensics and Investigations, 2019. *structure*, 10, 26.

Dafoulas, G. A., & Neilson, D. (2019, October). An overview of digital forensics education. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)* (pp. 1-7). IEEE.

Pachghare, V. K. (2019). *Cryptography and information security*. PHI Learning Pvt. Ltd..

Lin, X., Lin, X., & Lagerstrom-Fife. (2018). *Introductory Computer Forensics*. Springer International Publishing.

Whitman, M. E., & Mattord, H. J. (2021). *Principles of information security*. Cengage learning.

## Unit 4                      Distributed Objects

### Contents

- 1.0     Introduction
- 1.0     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
  - 3.4     Distributed Objects**
  - 3.5     Distributed Objects Systems/ Protocols**
  - 3.6     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
    - 3.7.1     Remote Procedure Calls (RPC)
- 4.0     Self-Assessment Exercises
- 5.0     Conclusion
- 6.0     Summary
- 7.0     References/Further Reading



## **1.0 Introduction**

We live in a very technologically advanced society. Technology and the use computers have become a part of our everyday life. Because of the increased knowledge and abundance of computer use, viruses have become a huge problem for users. Viruses are destructive programs that attack the computer and interfere with the operations of the computer. A virus can easily corrupt or delete data from your computer, which can become very costly to the owner of the computer. It is important that we learn about how viruses work so that we can avoid them at all cost.

Malware is any piece of software which is intended to cause harm to your system or network. Malware is different from normal programs in a way that they most of them have the ability to spread itself in the network, remain undetectable, cause changes/damage to the infected system or network, persistence. They have the ability to bring down the machine's performance to knees and can cause a destruction of the network. Consider the case when the computer becomes infected and is no longer usable, the data inside becomes unavailable – these are some of the malware damage scenarios. Malware attacks can be traced back to the time, even before the internet became widespread.



## **2.0 Intended Learning Outcomes (ILOs)**

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

- Explain Malware Analysis
- Explain the Types of Malwares
- Explain the Types of Malware Analysis

### **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

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. 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

### i. Define Malware Analysis

Malware analysis is the process of determining the purpose and functionality of a piece of malware. This process will reveal what type of harmful program has infected your network, the damage it's capable of causing, and most importantly how to remove it.

### ii. List and explain Types of Malwares

- **Virus:** Viruses are pieces of malware that require human intervention to propagate to other machines.
- **Worm:** Unlike Viruses, Worms do not need the help of humans to move to other machines. They can spread easily and can infect a high number of machines in a short amount of time.
- **Trojan:** These appear to be normal programs that have a legitimate function, like a game or a utility program. But underneath the innocent looking user interface, a Trojan performs malicious tasks without the user being aware.
- **Spyware:** Spyware is software that gathers personal or confidential information from user systems without their knowledge.
- **Keylogger:** This is a special type of spyware. It is specialized in recording the keystrokes made by the user.
- **Ransomware:** Ransomware is a form of malware that encrypts a victim's files. The attacker then demands a ransom from the victim to restore access to the data upon payment.

iii. Explain Static and Dynamic Analysis

Dynamic analysis also called malware behavior analysis runs the malware program to examine its behavior, while Static analysis examines a malware file without actually running the program.



## 5.0 Conclusion

Viruses are very destructive programs that can be devastating to companies and individual. The best defense against malware is a combination of vigilant and sensible behavior on the Internet, proper computer usage, and anti-malware software. By erring on the side of caution when surfing the web, not opening strange links or emails from unknown senders, and regularly updating and running an anti-malware program, you'll be relatively safe from the manifold dangers of the Internet.



## 6.0 Summary

In this unit, we have been able to outline malware analysis, types of malwares and malware analysis



## 7.0 References/Further Reading

<https://www.techtarget.com/searchsecurity/definition/computer-forensics>

Årnes, A. (Ed.). (2017). *Digital forensics*. John Wiley & Sons.

Kävrestad, J. (2020). *Fundamentals of Digital Forensics*. Springer International Publishing.

Easttom, C. (2021). *Digital Forensics, Investigation, and Response*. Jones & Bartlett Learning.

Nelson, B., Phillips, A., & Steuart, C. (2019). Guide to Computer Forensics and Investigations, 2019. *structure*, 10, 26.

Dafoulas, G. A., & Neilson, D. (2019, October). An overview of digital forensics education. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)* (pp. 1-7). IEEE.

Pachghare, V. K. (2019). *Cryptography and information security*. PHI Learning Pvt. Ltd..

Lin, X., Lin, X., & Lagerstrom-Fife. (2018). *Introductory Computer Forensics*. Springer International Publishing.

Whitman, M. E., & Mattord, H. J. (2021). *Principles of information security*. Cengage learning.



## Unit 5                      Remote Method Invocation

### Contents

- 2.0    Introduction
- 1.1    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

We live in a very technologically advanced society. Technology and the use computers have become a part of our everyday life. Because of the increased knowledge and abundance of computer use, viruses have become a huge problem for users. Viruses are destructive programs that attack the computer and interfere with the operations of the computer. A virus can easily corrupt or delete data from your computer, which can become very costly to the owner of the computer. It is important that we learn about how viruses work so that we can avoid them at all cost.

Malware is any piece of software which is intended to cause harm to your system or network. Malware is different from normal programs in a way that they most of them have the ability to spread itself in the network, remain undetectable, cause changes/damage to the infected system or network, persistence. They have the ability to bring down the machine's performance to knees and can cause a destruction of the network. Consider the case when the computer becomes infected and is no longer usable, the data inside becomes unavailable – these are some of the malware damage scenarios. Malware attacks can be traced back to the time, even before the internet became widespread.



## 2.0 Intended Learning Outcomes (ILOs)

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

- Explain Malware Analysis
- Explain the Types of Malwares
- Explain the Types of Malware Analysis

### 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:**

1. A remote object is declared with a **remote interface**, an extension of the Java **interface**.
2. The remote interface is implemented by the object server.
3. 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<sup>L</sup>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.

#### 3.3 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.

### 3.6 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(RMIPortNum);
            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(RMIPortNum);
    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

SomeInterface obj = new SomeInterface("Server1");

// Register the object; rebind will overwrite 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 RMIRRegistry 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
  - locating the RMI Registry in the server host,  
and
  - 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

Viruses are very destructive programs that can be devastating to companies and individual. The best defense against malware is a combination of vigilant and sensible behavior on the Internet, proper computer usage, and anti-malware software. By erring on the side of caution when surfing the web, not opening strange links or emails from unknown senders, and regularly updating and running an anti-malware program, you'll be relatively safe from the manifold dangers of the Internet.



## 6.0 Summary

In this unit, we have been able to outline malware analysis, types of malwares and malware analysis



## 7.0 References/Further Reading

<https://www.techtarget.com/searchsecurity/definition/computer-forensics>

Årnes, A. (Ed.). (2017). *Digital forensics*. John Wiley & Sons.

Kävrestad, J. (2020). *Fundamentals of Digital Forensics*. Springer International Publishing.

Easttom, C. (2021). *Digital Forensics, Investigation, and Response*. Jones & Bartlett Learning.

Nelson, B., Phillips, A., & Steuart, C. (2019). Guide to Computer Forensics and Investigations, 2019. *structure*, 10, 26.

Dafoulas, G. A., & Neilson, D. (2019, October). An overview of digital forensics education. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)* (pp. 1-7). IEEE.

Pachghare, V. K. (2019). *Cryptography and information security*. PHI Learning Pvt. Ltd..

Lin, X., Lin, X., & Lagerstrom-Fife. (2018). *Introductory Computer Forensics*. Springer International Publishing.

Whitman, M. E., & Mattord, H. J. (2021). *Principles of information security*. Cengage learning.

**Unit 6****Using UML for Component-based Designs****Contents**

- 3.0 Introduction
- 3.13 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

We live in a very technologically advanced society. Technology and the use computers have become a part of our everyday life. Because of the increased knowledge and abundance of computer use, viruses have become a huge problem for users. Viruses are destructive programs that attack the computer and interfere with the operations of the computer. A virus can easily corrupt or delete data from your computer, which can become very costly to the owner of the computer. It is important that we learn about how viruses work so that we can avoid them at all cost.

Malware is any piece of software which is intended to cause harm to your system or network. Malware is different from normal programs in a way that they most of them have the ability to spread itself in the network, remain undetectable, cause changes/damage to the infected system or network, persistence. They have the ability to bring down the machine's performance to knees and can cause a destruction of the network. Consider the case when the computer becomes infected and is no longer usable, the data inside becomes unavailable – these are some of the malware damage scenarios. Malware attacks can be traced back to the time, even before the internet became widespread.



## 2.0 Intended Learning Outcomes (ILOs)

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

- Explain Malware Analysis
- Explain the Types of Malwares
- Explain the Types of Malware Analysis

### 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.

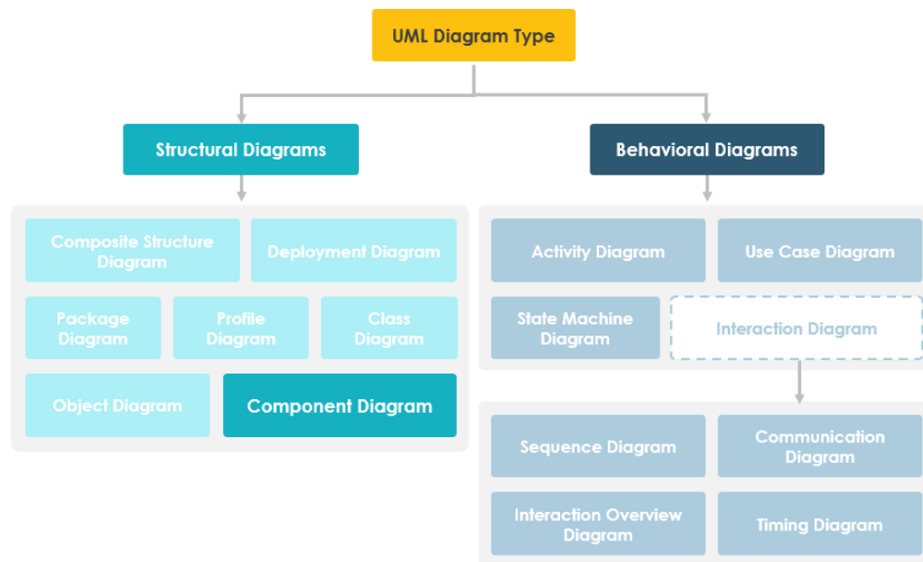


Figure 12: UML Diagram for Systems Designs

### 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.

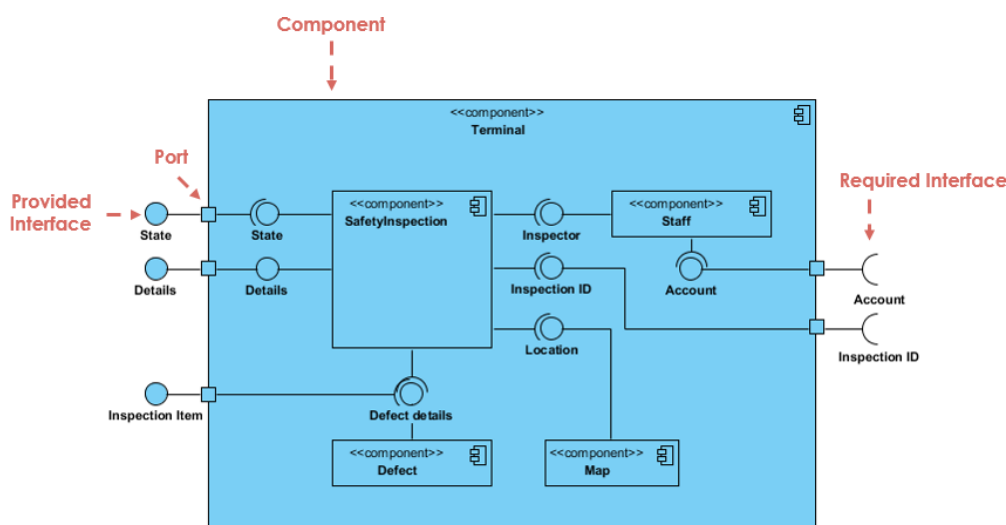


Figure 13: Component Diagram

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:

1. A rectangle with the component's name
2. A rectangle with the component icon
3. A rectangle with the stereotype text and/or icon

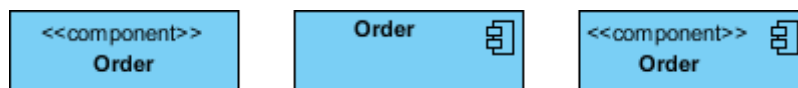


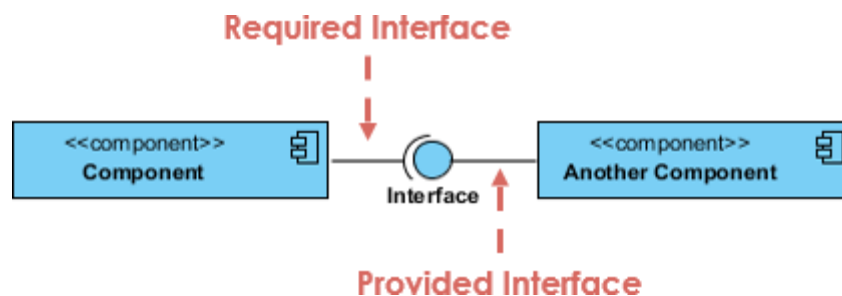
Figure 14: A high-level, abstracted view of a component

### 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).





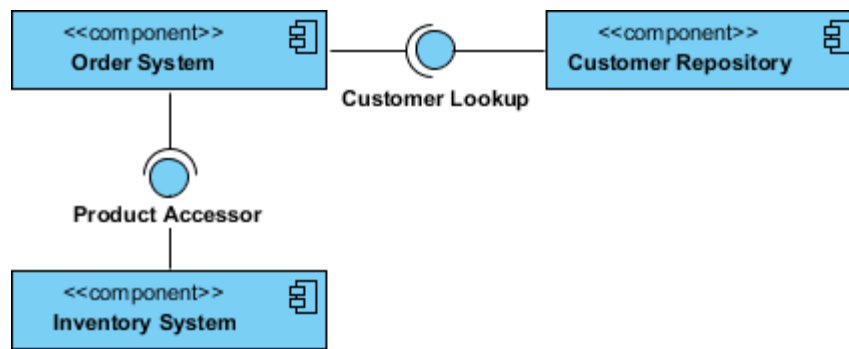


Figure 15: Component Diagram Example - Using Interface (Order System)

### 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.

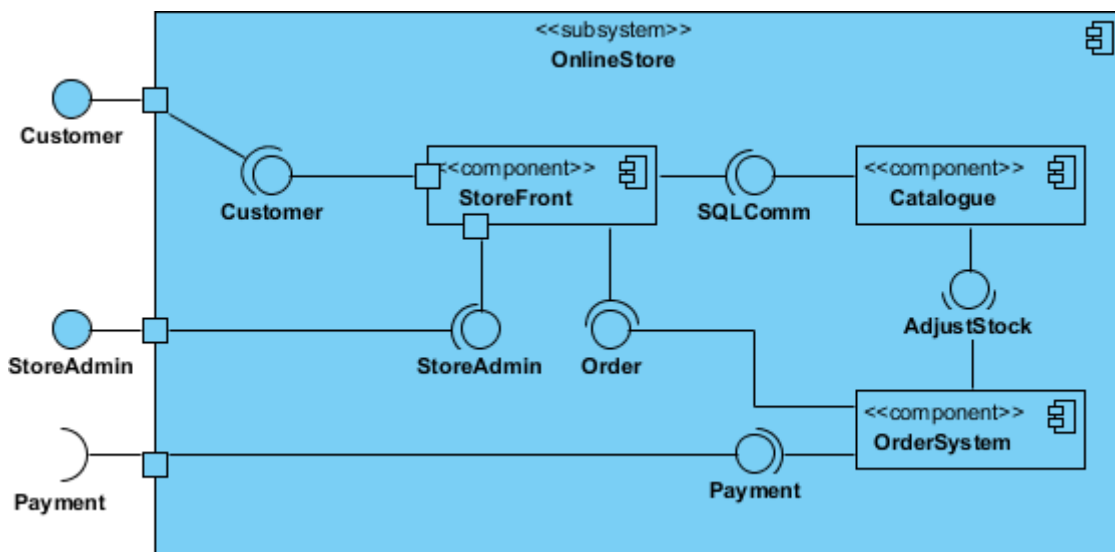


Figure 16: A Sub-system

### 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.

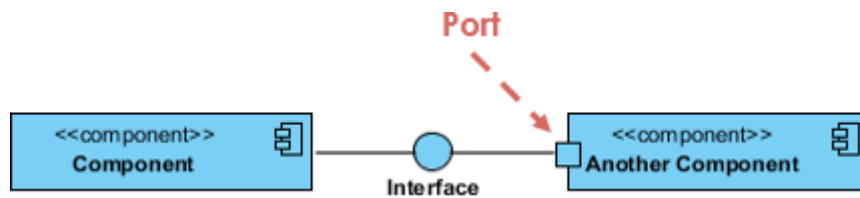
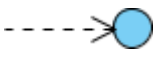
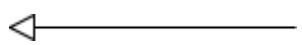


Figure 17: A Port

### 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
<b>Association:</b> <ul style="list-style-type: none"> <li>An association specifies a semantic relationship that can occur between typed instances.</li> <li>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.</li> </ul>	
<b>Composition:</b> <ul style="list-style-type: none"> <li>Composite aggregation is a strong form of aggregation that requires a part instance be included in at most one composite at a time.</li> <li>If a composite is deleted, all of its parts are normally deleted with it.</li> </ul>	
<b>Aggregation</b> <ul style="list-style-type: none"> <li>A kind of association that has one of its end marked shared as kind of aggregation, meaning that it has a shared aggregation.</li> </ul>	
<b>Constraint</b> <ul style="list-style-type: none"> <li>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.</li> </ul>	

<b>Dependency</b> <ul style="list-style-type: none"> <li>• 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.</li> <li>• This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s).</li> </ul>	
<b>Links:</b> <ul style="list-style-type: none"> <li>• A generalization is a taxonomic relationship between a more general classifier and a more specific classifier.</li> <li>• Each instance of the specific classifier is also an indirect instance of the general classifier.</li> <li>• Thus, the specific classifier inherits the features of the more general classifier.</li> </ul>	

### 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.

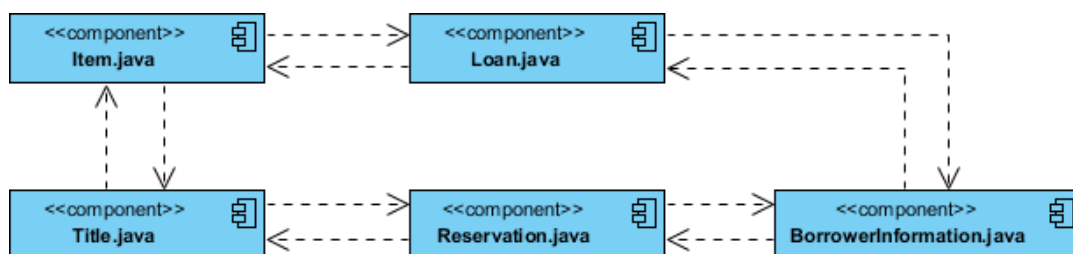
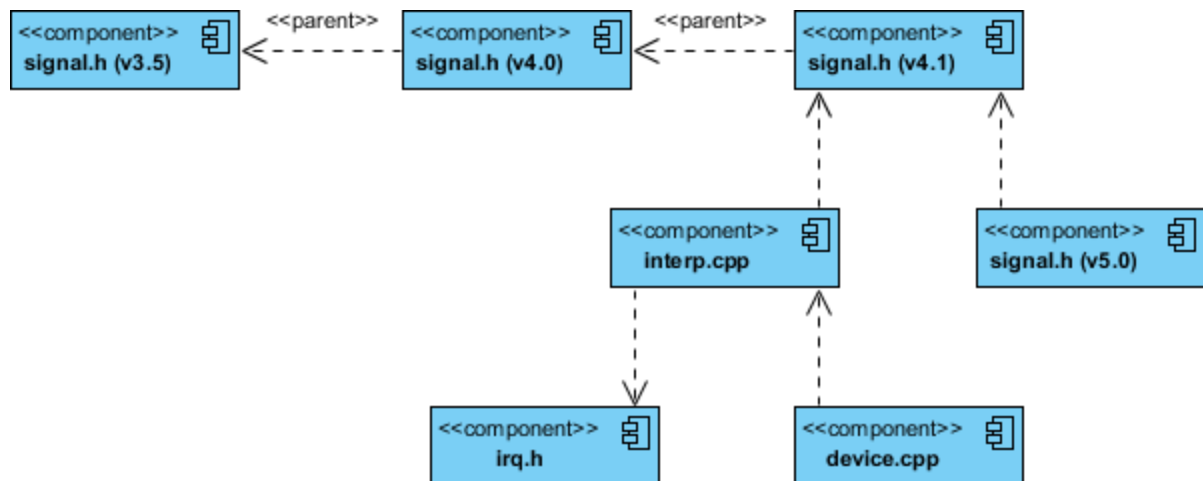
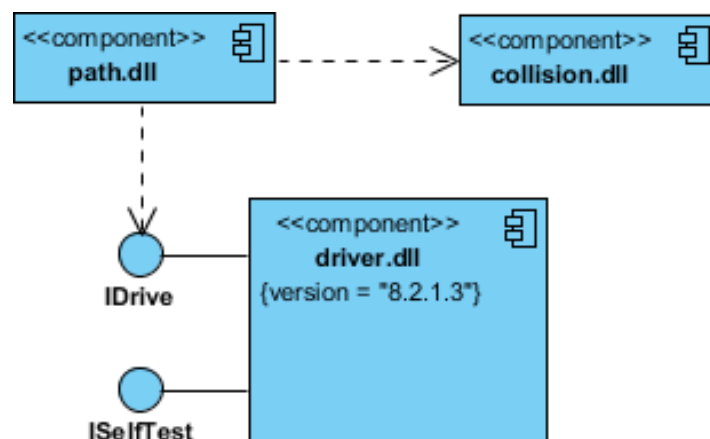


Figure 18: Component Diagram Example - C++ Code with versioning



### 3.9 Modelling an Executable Release

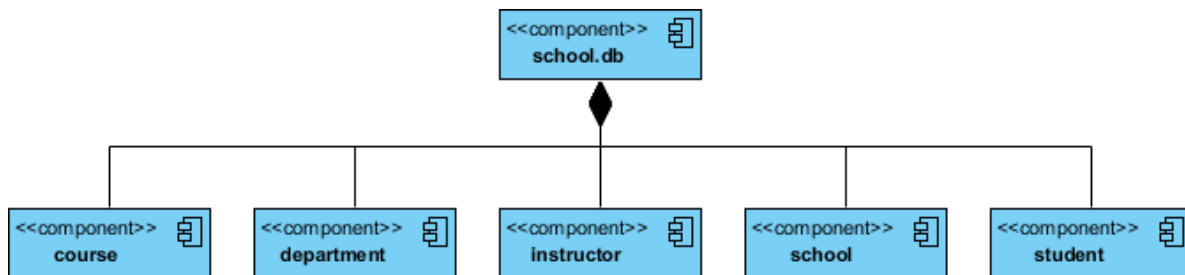
- Identify the set of components you'd like to model. Typically, this will involve some or all the components that live on one node, or the distribution of these sets of components across all the nodes in the system.
- Consider the stereotype of each component in this set. For most systems, you'll find 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

Viruses are very destructive programs that can be devastating to companies and individual. The best defense against malware is a combination of vigilant and sensible behavior on the Internet, proper computer usage, and anti-malware software. By erring on the side of caution when surfing the web, not opening strange links or emails from unknown senders, and regularly updating and running an anti-malware program, you'll be relatively safe from the manifold dangers of the Internet.



## 6.0 Summary

In this unit, we have been able to outline malware analysis, types of malwares and malware analysis



## **7.0 References/Further Reading**

[What is Component Diagram? \(visual-paradigm.com\)](http://visual-paradigm.com)

---

## Module 4: Distributed Transactions

---

### Introduction to Module

As soon as cyberspace and e-commerce were created in the mid-1990s, cybercrime flourished on a parallel track. Today, cybercrime has been doubling every single year in the number of incidents, as well as monetary losses. It is impossible to truly quantify cybercrime because most victims only see further losses in publicizing their inability to defend themselves from this modern day menace. The interesting note is that, of the cybercriminals who have been caught, the vast majority have pleaded guilty. The word ethics comes from the ancient Greek word *eché*, which means character. Every human society practices ethics in some way because every society attaches a value on a continuum of good to bad, right to wrong, to an individual's actions according to where that individual's actions fall within the domain of that society's rules and canons. In this module, Cyber crime Acts will be address which provide legal backings to human data and privacy.

Unit 1: Concept of Cyber Law

Unit 2: The INDIA cyber-Acts

Unit 3: The International Laws

Unit 4: Cyber Ethics

**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



## 1.0 Introduction

Technology has engendered new types of lawsuits or modified old ones. As, for example, the next generation of offences arose within the field of computer crimes (e.g., identity thefts), technology impacted on traditional rights such as copyright (1709) and privacy (1890), turning them into a matter of access, control, and protection over information in digital environments. This unit we explain the concepts of cyber law, the need of cyber law in the IT world and why is important to actually address cyber crime issues.



## 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, student will be able to

- Justify cyber crimes as sanctioned in cyber laws
- Demonstrate the concept of cyber law



## 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.

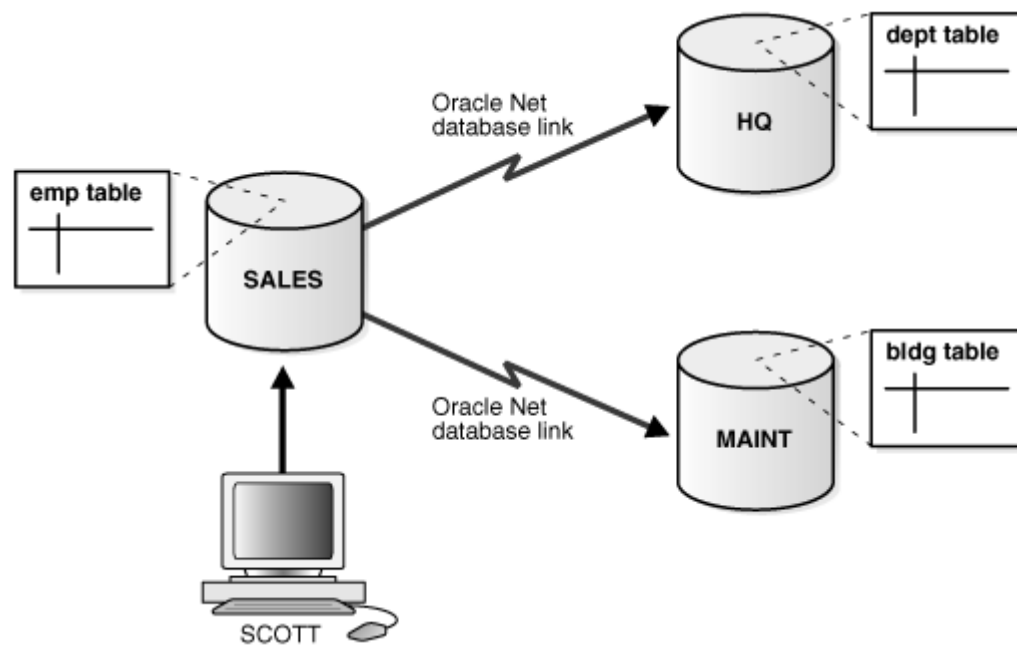


Figure 19: Sample Database on a Distributed Systems

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.

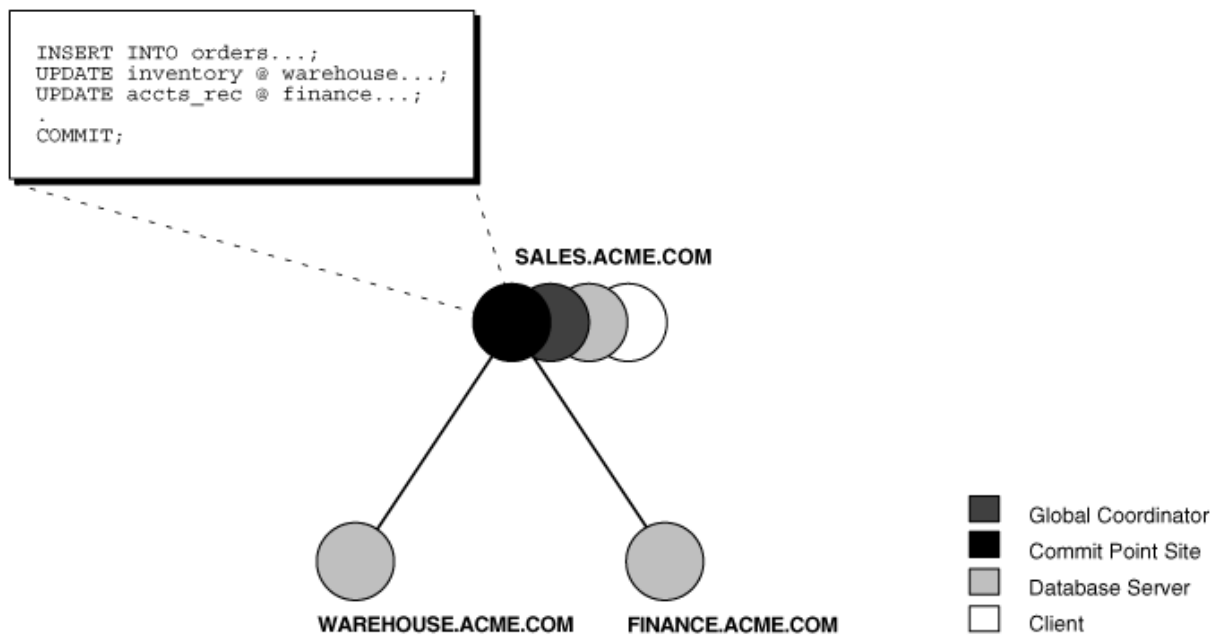


Figure 20: Example of a Session Tree

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.

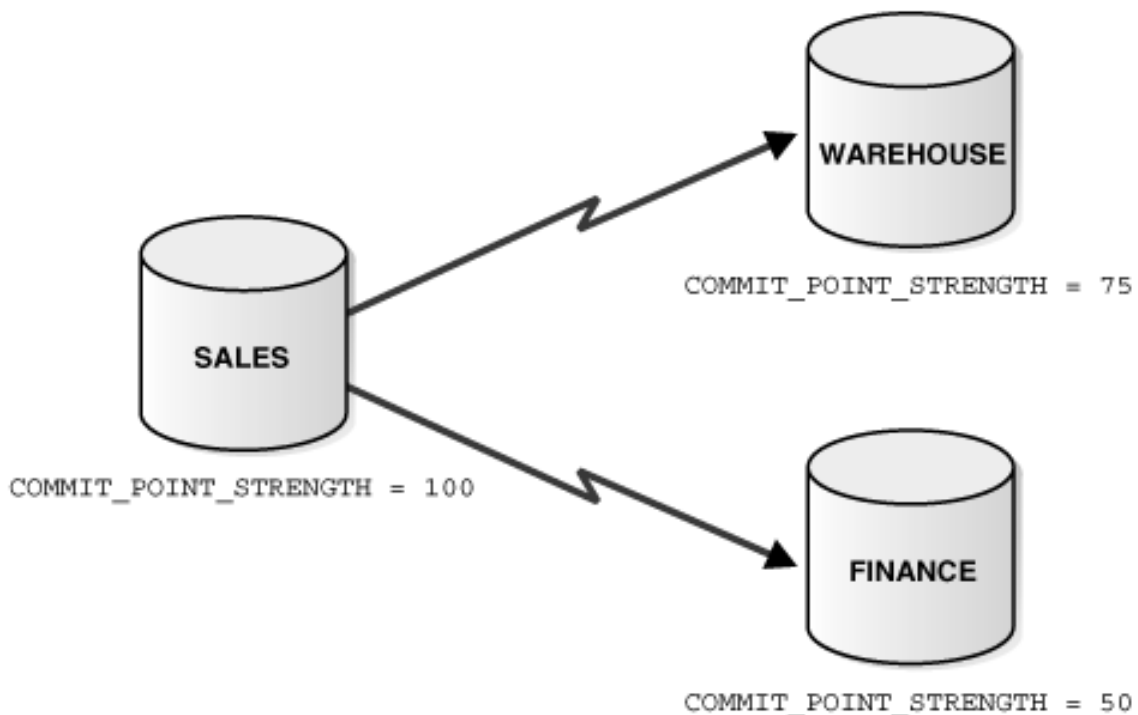


Figure 21: Commit Point Site

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:

1. 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.
2. 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.
3. 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

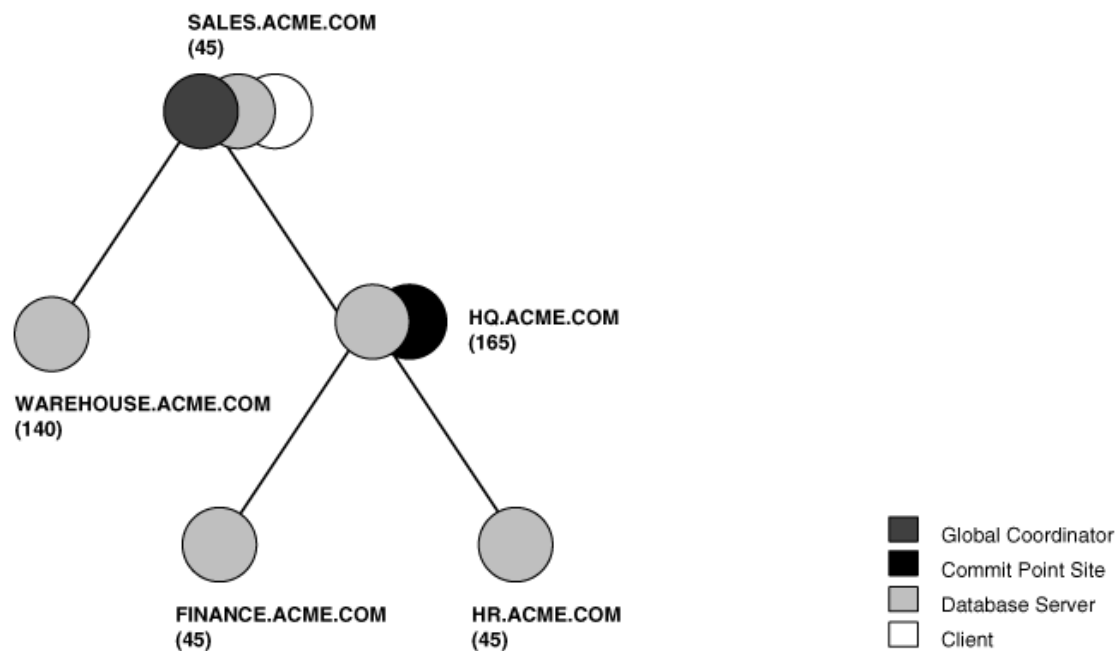


Figure 22: Commit Point Strengths and Determination of the Commit Point Site

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: <ul style="list-style-type: none"> <li>Only queries are issued at one or more nodes.</li> <li>No data is changed.</li> </ul> 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: <ul style="list-style-type: none"> <li>No data changes.</li> </ul> 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: <ul style="list-style-type: none"> <li>No data changes.</li> </ul> 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:

1. Releases resources currently held by the transaction and rolls back the local portion of the transaction.
2. 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:

1. The node requests that its **descendants**, that is, the nodes subsequently referenced, prepare to commit.
2. 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
3. The node allocates the resources it needs to commit the transaction if data is changed.
4. The node saves redo records corresponding to changes made by the transaction to its redo log.
5. The node guarantees that locks held for the transaction are able to survive a failure.
6. 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:

1. The global coordinator instructs the commit point site to commit.
2. The commit point site commits.
3. The commit point site informs the global coordinator that it has committed.
4. The global and local coordinators send a message to all nodes instructing them to commit the transaction.
5. At each node, the database commits the local portion of the distributed transaction and releases locks.
6. At each node, the database records an additional redo entry in the local redo log, indicating that the transaction has committed.

7. 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:

1. After receiving notice from the global coordinator that all nodes have committed, the commit point site erases status information about this transaction.
2. The commit point site informs the global coordinator that it has erased the status information.
3. 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:

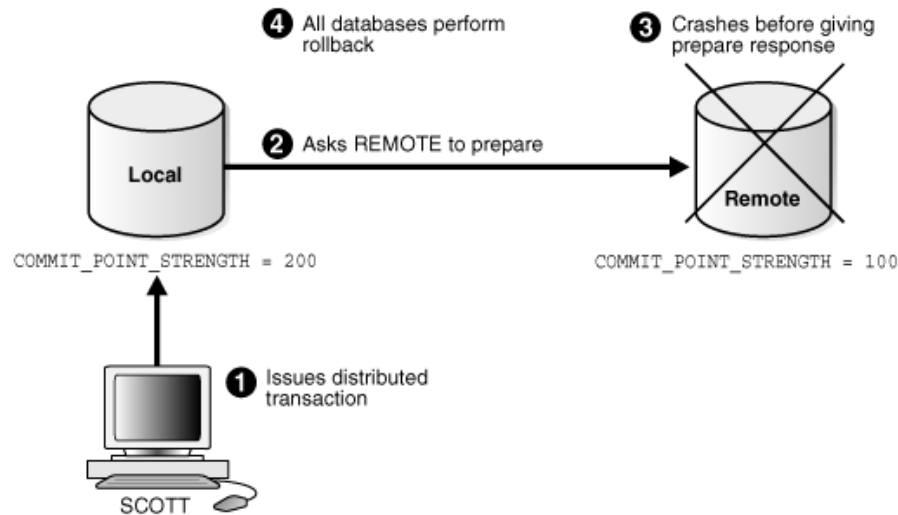


Figure 23: Failure During Prepare Phase

The following steps occur:

1. User **SCOTT** connects to **Local** and executes a distributed transaction.
2. 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.
3. The **remote** database crashes before issuing the prepare response back to **local**.
4. 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:

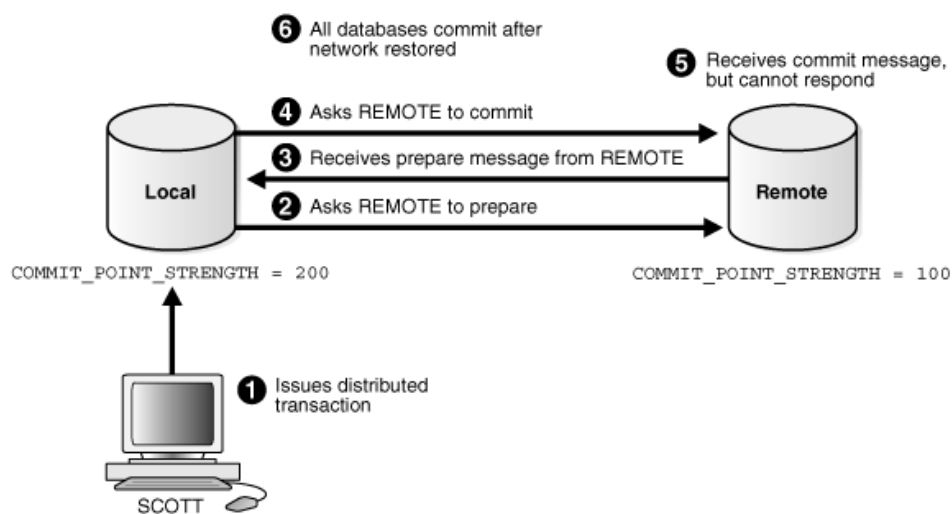


Figure 24: Failure During the Commit Phase

The following steps occur:

1. User Scott connects to local and executes a distributed transaction.
2. 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.
4. The commit point site commits the transaction locally, then sends a commit message to remote asking it to commit.
5. The remote database receives the commit message, but cannot respond because of a network failure.
6. 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:

1. An application establishes a connection using a database link.



2. The distributed transaction commits with the highest global SCN among all the databases involved.
3. 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

What is biggest crime ever committed in the cyber space?

## 4.0 Self-Assessment/Exercises

**Explain the different sources of law.**

**Answer**

- a) **Legislation:** - It is the formal enactment of law by the legislature created or authorized by the constitution. It stands in contrasted with judge made law. Legislation consists of written laws, as contrasted with judge made law or common law. It also stands in contrasted to customary law.
- b) **Common Law:** - It comprises the body of principle, which derive their authority solely from the decisions of courts. It is a body of law that develops and derives through judicial decisions different from legislative enactments. Its principals do not derive their validity from formal law making by anybody, but from their enunciation through decisions of courts.
- c) **Custom:** - Custom“ denotes a usage or practice of the people (including a particular social group or a group residing in a particular locality) which by common adoption and acquiescence and by long and unvarying habit, has become compulsory and has acquired the force of law with respect to the place or subject matter to which it relates.



## 5.0 Conclusion

Cyberlaw does concern you. As the nature of Internet is changing and this new medium is being seen as the ultimate medium ever evolved in human history, every activity of yours in Cyberspace can and will have a Cyber legal perspective. From the time you register your Domain Name, to the time you set up your web site, to the time you promote your website, to the time when you send and receive emails, to the time you conduct electronic commerce transactions on the said site, at every point of time, there are various Cyberlaw issues involved.



## 6.0 Summary

Cyber law describes the legal issues related to use of communications technology, particularly "cyberspace", i.e. the Internet. It is less a distinct field of law in the way that property or contract are as it is an intersection of many legal fields. Cyber law is an attempt to integrate the challenges presented by human activity on the Internet with legacy system of laws applicable to the physical world.



## 7.0 References/Further Reading

- Dudley, A., Braman, J., & Vincenti, G. (2011). *Investigating Cyber Law and Cyber Ethics: Issues, Impacts and Practices: Issues, Impacts and Practices* (Issue January).  
[https://books.google.com/books?hl=en&lr=&id=\\_-aeBQAAQBAJ&pgis=1](https://books.google.com/books?hl=en&lr=&id=_-aeBQAAQBAJ&pgis=1)
- Isha Upadhyay (September, 2020). *Cyber Law: A Comprehensive Guide For 2021*.  
<https://www.jigsawacademy.com/blogs/cyber-security/what-is-cyber-law/>. Last accessed: December, 2021.
- 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>

Trachtman, J. P. (2013). Cyberspace and Cybersecurity. In *The Future of International Law*.  
<https://doi.org/10.1017/cbo9781139565585.006>

Vikaspedia. *Cyber Laws*. <https://vikaspedia.in/education/digital-literacy/information-security/cyber-laws>. Last accessed: 30 December, 2021

**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

The cyberspace is anticipated to become even more complex in the upcoming years, with the increase in networks and devices connected to it. India as a nation has encountered several cyber-attacks which forced the government to impose cyber law that regulates the code and conducts of the people of India and international on the cyberspace. In this unit, we will discuss some of the regulations such as ITA 2000, IPC, National Cyber security policy and review some of the scenarios of cybercrime in India



## 2.0 Intended Learning Outcomes (ILOs)

At the end of this unit, the student will able to

- Understands laws binds to cyberspace
- Know their rights in data and privacy protection
- 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:

1. Flat transactions
2. 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.

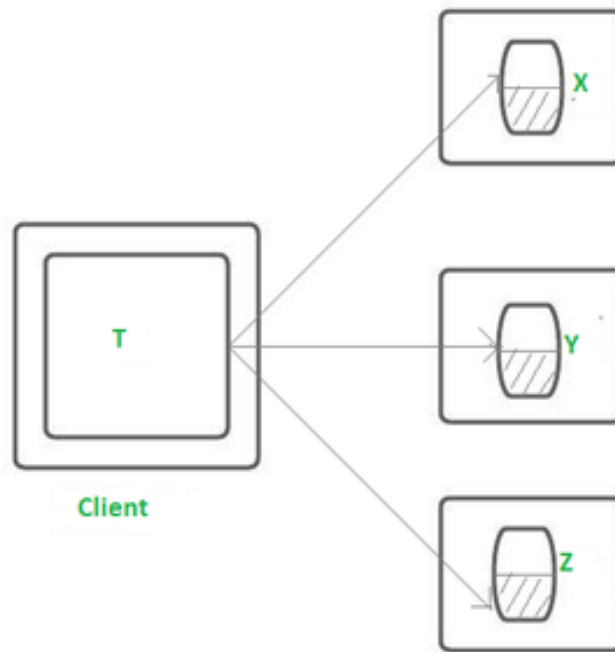


Figure 25: Flat Transactions

#### 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 a 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 diagram

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.

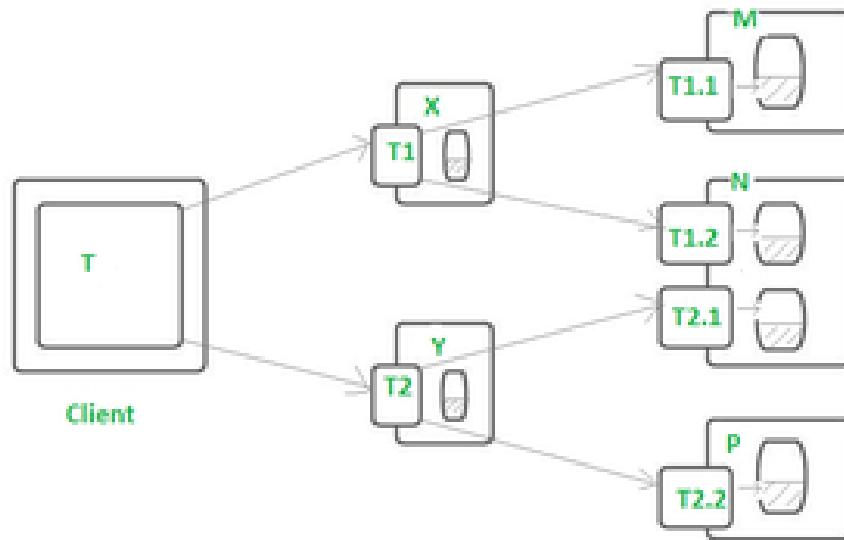


Figure 26: Nested Transactions

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:**

1. Account A is on server X



2. Account B is on server Y, and
3. 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 below, transaction T is designed as a set of four nested transactions: T1, T2, T3 and T4.

### 3.4.2.1 Advantage:

The performance is higher than a single transaction in which four operations are invoked one after the other in sequence.

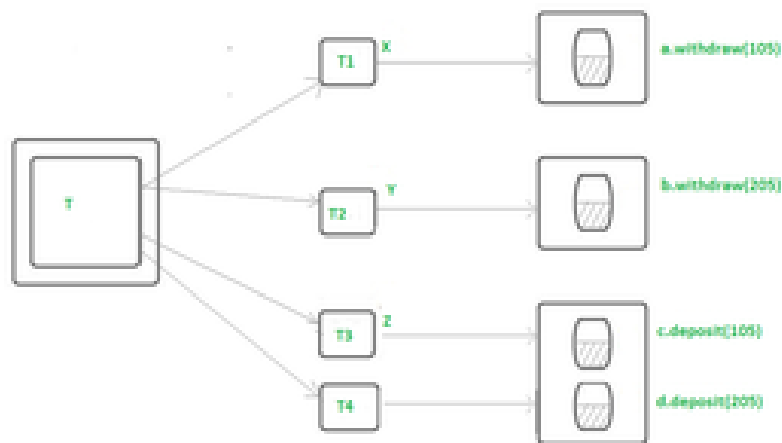


Figure 27: Nested Transactions

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 any two cybercrimes in your country.

## 4.0 Self-Assessment/Exercises

**Discuss the classification of crimes under the IT Act 2000.****Answer**

The following acts are cyber crime in the I.T. Act 2000:- Without permission of the authorized user

- i) Accessing or securing access to computer system or network.
- ii) Downloading, copying or extracting any data or information.
- iii) Introducing any computer, virus or contaminant in the computer.
- iv) Disrupting the working of the computer.
- v) Disrupting the access of the computer of an authorized user.
- vi) Providing assistance to ensure unauthorized access to the computer.
- vii) Tampering with computer source documents.
- viii) Hacking of computer system.
- ix) Carrying on activities that are not in compliance with the provisions of the Act.

**What are the amendments to the Indian Penal Code?****Answer**

The Indian Penal Code (IPC) details actions that constitute a crime and the punishments prescribed for such actions. It elaborately classifies crimes based on interests that are intended to be protected. The classification includes :-

- i) Offences against body
  - ii) Offences against property
  - iii) Offences against marriage
  - iv) Offences against public tranquility
  - v) Offences against state
- Some important aspects have to be weighed while determining whether a crime has been committed or not.

**5.0 Conclusion**

Cybercrimes are a new class of crimes which are increasing day by day due to extensive use of internet these days.



## **6.0 Summary**

Technology Act, 2000 was enacted with prime objective to create an enabling environment for commercial use of I.T. The IT Act specifies the acts which have been made punishable. The Indian Penal Code, 1860 has also been amended to take into its purview cybercrimes.



## **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'  $\lambda$  The complexity in types and forms of cybercrime increases the difficulty to fight back, fighting cybercrime calls for international cooperation  $\lambda$ . 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 international laws and treaties
- Explain international cyber-attacks previously occurred



## 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.

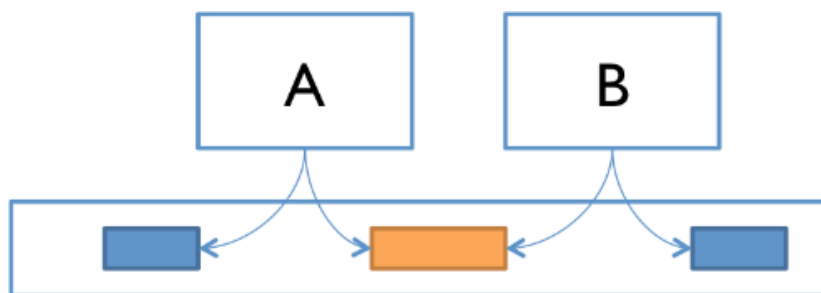
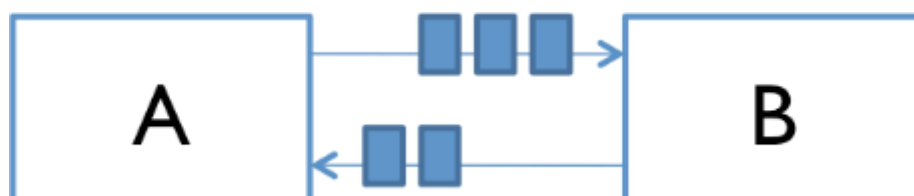


Figure 28: Shared Memory

#### 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.





### 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.

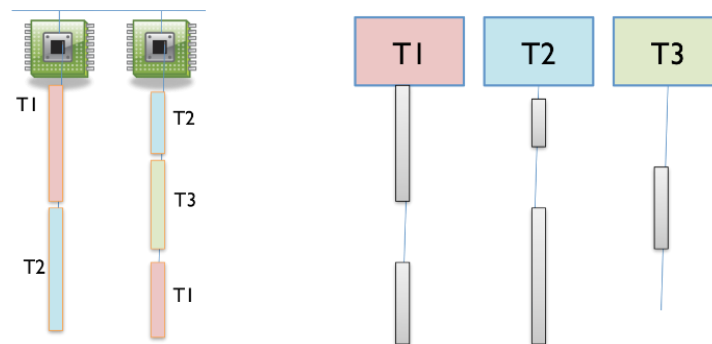


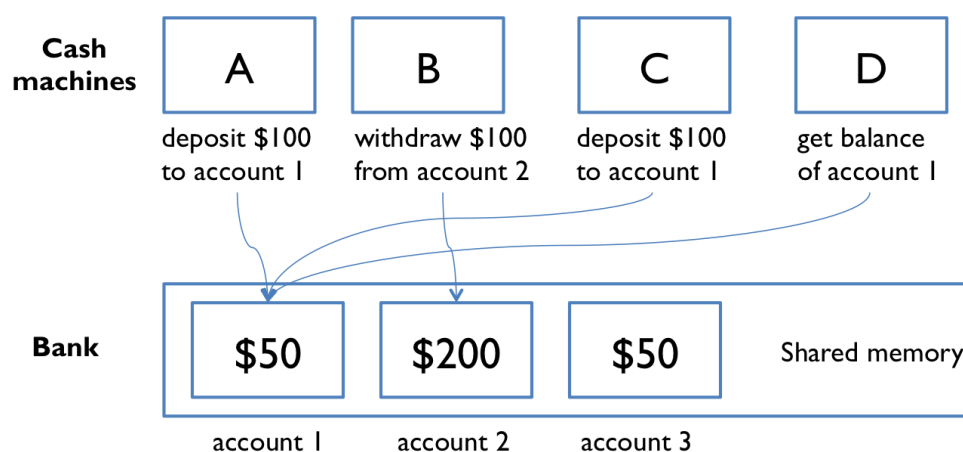
Figure 29: Many Concurrent Threads, One or Two Processors

### 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.



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();  
    }  
}
```

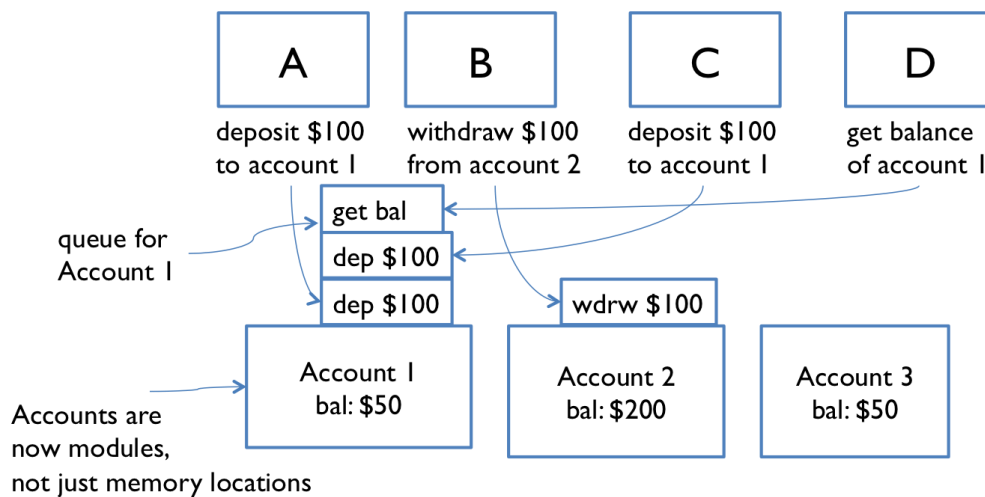
```
    if (answer == 0) throw new RuntimeException("answer wasn't ready!");  
}
```

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



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 section of the Information Technology Act (ITA) that sanction internet fraudsters? Explain the consequence according to the Act.



## 5.0 Conclusion

A country's participation in a particular international agreement becomes effective only if domestic laws are drafted and approved that legislate the intent of the signed international agreement.



## 6.0 Summary

Lawmakers and law enforcement agencies, around the world, advocate the need for cyber laws that are written in the cyber language. That is, laws that explicitly define cyber offenses and fully support the acceptance of cyber evidence. International bodies, responding to this call, have convened and produced treaties and conventions that, unfortunately, have fallen short of receiving total acceptance by the member countries.



## **7.0 References/Further Reading**

[Concepts: Concurrency \(uhcl.edu\)](http://uhcl.edu)

[Processes and Threads \(The Java™ Tutorials > Essential Java Classes > Concurrency\)  
\(oracle.com\)](http://www.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

Ethics is, therefore, the study of right and wrong in human conduct. Ethics can also be defined as a theoretical examination of morality or “theory of morals.” Other philosophers have defined ethics in a variety of ways. Robert C. Solomon, in *Morality and the Good Life*, defines ethics as a set of “theories of value, virtue, or of right (valuable) action.” O.J. Johnson, on the other hand, defines ethics as a set of theories “that provide general rules or principles to be used in making moral decisions and, unlike ordinary intuitions, provides a justification for those rules.” The word ethics comes from the ancient Greek word *eché*, which means character. Every human society practices ethics in some way because every society attaches a value on a continuum of good to bad, right to wrong, to an individual’s actions according to where that individual’s actions fall within the domain of that society’s rules and canons.



## 2.0 Intended Learning Outcomes (ILOs)

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

- Understand the use of ethical theories in ethical arguments.
- Articulate the ethical tradeoffs in a technical decision.
- Understand the role of professional codes of ethics.



## 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:

1. It logically represents a business activity with a specified outcome.
2. It is self-contained
3. It is a black box for its consumers
4. It may consist of other underlying services

To 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:

1. Business value is more important than technical strategy.
2. Strategic goals are more valuable than project-specific benefits.
3. Intrinsic interoperability is greater than custom integration.
4. Shared services over specific-purpose implementations.
5. Flexibility is given more importance than optimization.
6. 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

**Why is ethics relevant in the cyberspace?**

### 4.0 Self-Assessment/Exercises

#### 1. What are the ten commandments for computer ethics?

**Answer**

- i. Thou shalt not use a computer to harm other people.
- ii. Thou shalt not interfere with other people's computer work.
- iii. Thou shalt not snoop around in other people's files.
- iv. Thou shalt not use a computer to steal.
- v. Thou shalt not use a computer to bear false witness.
- vi. Thou shalt not use of copy software for which you have not paid.
- vii. Thou shalt not use other people's computer resources without authorization.
- viii. Thou shalt not appropriate other people's intellectual output.
- ix. Thou shalt think about the social consequences of the program u write.
- x. Thou shalt use a computer in ways to show consideration and respect.

#### 2. Explain the three levels of computer ethics.

**Answer**

- **First level:** - It is the basic level where computer ethics tries to sensitize people to the fact that computer technology has social and ethical consequences. Newspaper, TV news program, and magazines have highlighted the topic of computer ethics by reporting on events relating to computer viruses, software ownership law suits, computer aided bank robbery, computer malfunction etc.
- **Second level:-** It consists of someone who takes interest in computer ethics cases, collects examples, clarifies them, looks for similarities and differences reads related works, attends relevant events to make preliminary assessments and after comparing them.
- **Third level:** - It referred to as „theoretical“ computer ethics applies scholarly theories to computer ethics cases and concepts in order to deepen the understanding of issues.



All three level of analysis are important to the goal of advancing and defending human values.



## 5.0 Conclusion

The role of ethics is to help societies distinguish between right and wrong and to give each society a basis for justifying the judgment of human actions. Ethics is, therefore, a field of inquiry whose subject is human actions, collectively called human conduct, that are taken consciously, willfully, and for which one can be held responsible. According to Fr. Austin Fagothey, such acts must have knowledge, which signifies the presence of a motive, be voluntary, and have freedom to signify the presence of free choice to act or not to act.



## 6.0 Summary

The purpose of ethics is to interpret human conduct, acknowledging and distinguishing between right and wrong. The interpretation is based on a system which uses a mixture of induction and deduction. In most cases, these arguments are based on historical schools of thought called ethical theories. There are many different kinds of ethical theories, and within each theory there may be different versions of that theory. Let us discuss these next.



## 7.0 References/Further Reading

Alfreda D. et al. (2012). *Investigating Cyber Law and Cyber Ethics: Issues, Impacts, and Practices*. Information Science Reference, USA. ISBN 978-1-61350-133-7

Baldini, Gianmarco, Botterman, Maarten, Neisse, Ricardo, and Tallacchini, Mariachiara (2016) "Ethical Design in the Internet of Things," *Science and Engineering Ethics*, 1-21.

Bustard, John D. (2017), “Improving Student Engagement in the Study of Professional Ethics: Concepts and an Example in Cyber Security” *Science and Engineering Ethics*, 1-16.

Dipert, Randall R. (2010) “The Ethics of Cyberwarfare,” *Journal of Military Ethics* 9:4, 384-410

ICSI (2016). *Cybercrime Law and Practice*. THE INSTITUTE OF COMPANY SECRETARIES OF INDIA. ISBN : 978-93-82207795.

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>

Manjikian, Mary (2017) *Cybersecurity Ethics: An Introduction*, Routledge; 240 pp. Taddeo,

Mariarosaria and Glorioso, Ludovica (2017) *Ethics and Policies for Cyber Operations*, Springer. EC Council (2016) *Ethical Hacking and Countermeasures* (Book Series, 4 volumes), Cengage Learning

---

## Module 5: Mobile & Cloud Computing

---

### Introduction to Module

As soon as cyberspace and e-commerce were created in the mid-1990s, cybercrime flourished on a parallel track. Today, cybercrime has been doubling every single year in the number of incidents, as well as monetary losses. It is impossible to truly quantify cybercrime because most victims only see further losses in publicizing their inability to defend themselves from this modern day menace. The interesting note is that, of the cybercriminals who have been caught, the vast majority have pleaded guilty. The word ethics comes from the ancient Greek word *eché*, which means character. Every human society practices ethics in some way because every society attaches a value on a continuum of good to bad, right to wrong, to an individual's actions according to where that individual's actions fall within the domain of that society's rules and canons. In this module, Cyber crime Acts will be address which provide legal backings to human data and privacy.

Unit 1: Concept of Cyber Law

Unit 2: The INDIA cyber-Acts

Unit 3: The International Laws

Unit 4: Cyber Ethics

**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



## 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

- Justify cyber crimes as sanctioned in cyber laws
- Demonstrate the concept of cyber law



## 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.

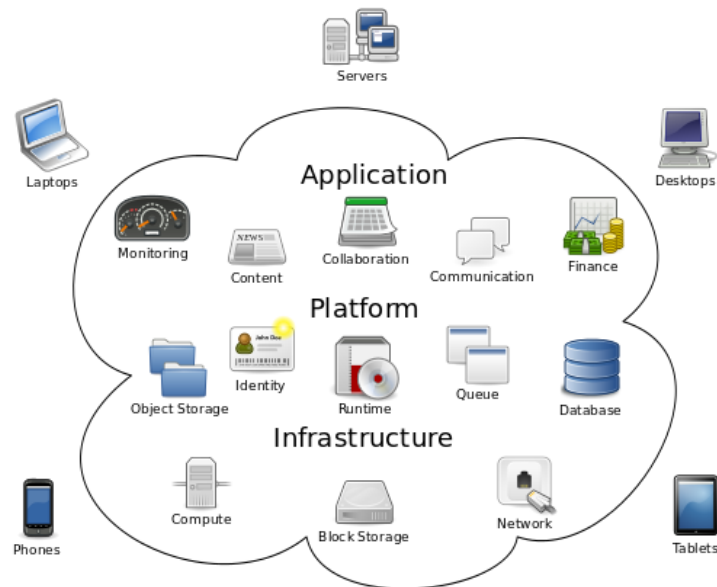


Figure 30: Cloud Computing

### 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 is biggest crime ever committed in the cyber space?

## 4.0 Self-Assessment/Exercises

**Explain the different sources of law.**

**Answer**

a) **Legislation:** - It is the formal enactment of law by the legislature created or authorized by the constitution. It stands in contrast with judge made law. Legislation consists of written laws, as contrasted with judge made law or common law. It also stands in contrast to customary law.

b) **Common Law:** - It comprises the body of principle, which derive their authority solely from the decisions of courts. It is a body of law that develops and derives through judicial decisions different from legislative enactments. Its principles do not derive their validity from formal law making by anybody, but from their enunciation through decisions of courts.

c) **Custom:** - Custom denotes a usage or practice of the people (including a particular social group or a group residing in a particular locality) which by common adoption and acquiescence and by long and unvarying habit, has become compulsory and has acquired the force of law with respect to the place or subject matter to which it relates.



## 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

Cyber law describes the legal issues related to use of communications technology, particularly "cyberspace", i.e. the Internet. It is less a distinct field of law in the way that property or contract are as it is an intersection of many legal fields. Cyber law is an attempt to integrate the challenges presented by human activity on the Internet with legacy system of laws applicable to the physical world.



## 7.0 References/Further Reading

1. 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>
2. Bahtovski, A., Gusev, M.: Cloud Computing in Mobile Technologies. The 9<sup>th</sup> Conference for Informatics and Information Technology (CIIT 2012).
3. 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/>
4. Wikipedia: Cloud Computing. [https://en.wikipedia.org/wiki/Cloud\\_computing](https://en.wikipedia.org/wiki/Cloud_computing)
5. Microsoft Azure: What is cloud computing? A beginner's guide.  
<https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/>
6. Rouse, M.: Definition: Software as a Service (SaaS). May, 2016.  
<http://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service>
7. Rouse, M.: Definition: Platform as a Service (PaaS). September, 2017.  
<http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS>
8. Mobile Cloud Computing – Pros and Cons. December, 2014.  
<https://www.getcloudservices.com/blog/mobile-cloud-computing-pros-and-cons/>
9. Kleiner, C., Disterer, G.: Ensuring mobile device security and compliance at the workplace. Conference on Enterprise Information Systems, HCist 2015, October 7-9, 2015.

10. 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 be able to

- Understands laws binds to cyberspace
- Know their rights in data and privacy protection
- Learn from existing scenarios of cybercrimes in India



## 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 cybercrimes in your country.

## 4.0 Self-Assessment/Exercises

**Discuss the classification of crimes under the IT Act 2000.**

**Answer**

The following acts are cyber crime in the I.T. Act 2000:- Without permission of the authorized user

- x) Accessing or securing access to computer system or network.
- xi) Downloading, coping or extracting any data or information.
- xii) Introducing any computer, virus or contaminant in the computer.
- xiii) Disrupting the working of the computer.
- xiv) Disrupting the access of the computer of an authorized user.
- xv) Providing assistance to ensure unauthorized access to the computer.

- xvi) Tampering with computer source documents.
- xvii) Hacking of computer system.
- xviii) Carring on activities that are not in compliance with the provisions of the Act.

### **What are the amendments to the Indian Penal Code?**

#### **Answer**

The Indian Panel Code (IPC) details actions that constitute a crime and the punishments prescribed for such actions. It elaborately classifies crimes based on interests that are intended to be protected. The classification includes :-

- vi) Offences against body
  - vii) Offences against property
  - viii) Offences against marriage
  - ix) Offences against public tranquility
  - x) Offences against state
- Some important aspects have to be weighed while determining whether a crime has been committed or not.



### **5.0 Conclusion**

Cybercrimes are a new class of crimes which are increasing day by day due to extensive use of internet these days.



### **6.0 Summary**

Technology Act, 2000 was enacted with prime objective to create an enabling environment for commercial use of I.T. The IT Act specifies the acts which have been made punishable. The Indian Penal Code, 1860 has also been amended to take into its purview cybercrimes.



## **7.0 References/Further Reading**

[Flat & Nested Distributed Transactions - GeeksforGeeks](#)

## 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 international laws and treaties
- Explain international cyber-attacks previously occurred



## **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:

$$S_t = S_0 K$$

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  $P_r/P_t = A/R^n$ , where  $A$  is some constant.

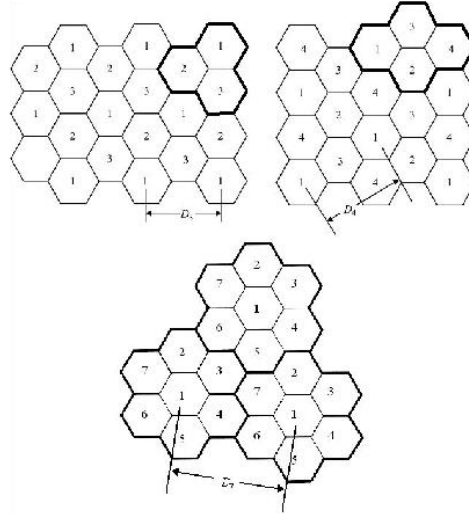


Figure 31: Frequency reuse patterns

**Figure 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.  $D_i$  shows the shortest distance between cells reusing the same frequency.

With this model, signal to interference ratios are estimated as

$$S/I = \frac{R^{-n}}{\sum_{i=1}^{i_0} D_i^{-n}}$$

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 1). In the case of hexagonal cell approximation the expression simplifies to equation 1:

$$S/I = \frac{(\sqrt{3K})^n}{i_0}$$

$n$  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 (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:

#### 3.1.3.1 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.

#### 3.1.3.2 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.

#### 3.1.3.3 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-CDMA).

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.

#### 3.1.4.1 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:

$$C = W \log_2(1 + S/N)$$

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.

### 3.1.4.2 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).

#### 3.1.4.2.1 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.

#### 3.1.4.2.2 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.

#### 3.1.4.2.3 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:

$$N = 1 + \frac{W}{R} \cdot \frac{1}{E_b/N_t} \cdot \frac{1}{\alpha} \cdot \frac{1}{1 + \beta} \dots\dots\dots 2.5$$

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 (2.5) 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).

#### 3.1.5.1 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 [1](#).

Modulation	Bits encoded by:	Example
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

What section of the Information Technology Act (ITA) that sanction internet fraudsters? Explain the consequence according to the Act.



## 5.0 Conclusion

A country's participation in a particular international agreement becomes effective only if domestic laws are drafted and approved that legislate the intent of the signed international agreement.



## **6.0 Summary**

Lawmakers and law enforcement agencies, around the world, advocate the need for cyber laws that are written in the cyber language. That is, laws that explicitly define cyber offenses and fully support the acceptance of cyber evidence. International bodies, responding to this call, have convened and produced treaties and conventions that, unfortunately, have fallen short of receiving total acceptance by the member countries.



## **7.0 References/Further Reading**

[Contents \(colorado.edu\)](http://colorado.edu)

## 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 Analog
- 4.0    Self-Assessment Exercises
- 5.0    Conclusion
- 6.0    Summary
- 7.0    References/Further Reading



## 1.0 Introduction

Ethics is, therefore, the study of right and wrong in human conduct. Ethics can also be defined as a theoretical examination of morality or “theory of morals.” Other philosophers have defined ethics in a variety of ways. Robert C. Solomon, in *Morality and the Good Life*, defines ethics as a set of “theories of value, virtue, or of right (valuable) action.” O.J. Johnson, on the other hand, defines ethics as a set of theories “that provide general rules or principles to be used in making moral decisions and, unlike ordinary intuitions, provides a justification for those rules.” The word ethics comes from the ancient Greek word *eché*, which means character. Every human society practices ethics in some way because every society attaches a value on a continuum of good to bad, right to wrong, to an individual’s actions according to where that individual’s actions fall within the domain of that society’s rules and canons.



## 2.0 Intended Learning Outcomes (ILOs)

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

- Understand the use of ethical theories in ethical arguments.
- Articulate the ethical tradeoffs in a technical decision.
- Understand the role of professional codes of ethics.



## 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:

5. It logically represents a business activity with a specified outcome.
6. It is self-contained
7. It is a black box for its consumers
8. It may consist of other underlying services

To 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:

7. Business value is more important than technical strategy.
8. Strategic goals are more valuable than project-specific benefits.
9. Intrinsic interoperability is greater than custom integration.
10. Shared services over specific-purpose implementations.
11. Flexibility is given more importance than optimization.
12. 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

**Why is ethics relevant in the cyberspace?**

### 4.0 Self-Assessment/Exercises

#### 1. What are the ten commandments for computer ethics?

**Answer**

- xi. Thou shalt not use a computer to harm other people.
- xii. Thou shalt not interfere with other people's computer work.
- xiii. Thou shalt not snoop around in other people's files.
- xiv. Thou shalt not use a computer to steal.
- xv. Thou shalt not use a computer to bear false witness.
- xvi. Thou shalt not use of copy software for which you have not paid.
- xvii. Thou shalt not use other people's computer resources without authorization.
- xviii. Thou shalt not appropriate other people's intellectual output.
- xix. Thou shalt think about the social consequences of the program u write.
- xx. Thou shalt use a computer in ways to show consideration and respect.

#### 2. Explain the three levels of computer ethics.

**Answer**

- **First level:** - It is the basic level where computer ethics tries to sensitize people to the fact that computer technology has social and ethical consequences. Newspaper, TV news program, and magazines have highlighted the topic of computer ethics by reporting on events relating to computer viruses, software ownership law suits, computer aided bank robbery, computer malfunction etc.
- **Second level:-** It consists of someone who takes interest in computer ethics cases, collects examples, clarifies them, looks for similarities and differences reads related works, attends relevant events to make preliminary assessments and after comparing them.
- **Third level:** - It referred to as „theoretical“ computer ethics applies scholarly theories to computer ethics cases and concepts in order to deepen the understanding of issues.

All three level of analysis are important to the goal of advancing and defending human values.



## 5.0 Conclusion

The role of ethics is to help societies distinguish between right and wrong and to give each society a basis for justifying the judgment of human actions. Ethics is, therefore, a field of inquiry whose subject is human actions, collectively called human conduct, that are taken consciously, willfully, and for which one can be held responsible. According to Fr. Austin Fagothey, such acts must have knowledge, which signifies the presence of a motive, be voluntary, and have freedom to signify the presence of free choice to act or not to act.



## 6.0 Summary

The purpose of ethics is to interpret human conduct, acknowledging and distinguishing between right and wrong. The interpretation is based on a system which uses a mixture of induction and deduction. In most cases, these arguments are based on historical schools of thought called ethical theories. There are many different kinds of ethical theories, and within each theory there may be different versions of that theory. Let us discuss these next.



## 7.0 References/Further Reading

Alfreda D. et al. (2012). *Investigating Cyber Law and Cyber Ethics: Issues, Impacts, and Practices*. Information Science Reference, USA. ISBN 978-1-61350-133-7

Baldini, Gianmarco, Botterman, Maarten, Neisse, Ricardo, and Tallacchini, Mariachiara (2016) "Ethical Design in the Internet of Things," *Science and Engineering Ethics*, 1-21.

Bustard, John D. (2017), “Improving Student Engagement in the Study of Professional Ethics: Concepts and an Example in Cyber Security” *Science and Engineering Ethics*, 1-16.

Dipert, Randall R. (2010) “The Ethics of Cyberwarfare,” *Journal of Military Ethics* 9:4, 384-410

ICSI (2016). *Cybercrime Law and Practice*. THE INSTITUTE OF COMPANY SECRETARIES OF INDIA. ISBN : 978-93-82207795.

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>

Manjikian, Mary (2017) *Cybersecurity Ethics: An Introduction*, Routledge; 240 pp. Taddeo,

Mariarosaria and Glorioso, Ludovica (2017) *Ethics and Policies for Cyber Operations*, Springer. EC Council (2016) *Ethical Hacking and Countermeasures* (Book Series, 4 volumes), Cengage Learning