Name : Noraini Nazaruddin

Matric No.: WEK 020172

Supervisor: En. Abdullah Gani

Moderator: Dr. Ling Teck Chaw

Title : Design and Develop a Generic Cut-Through Switch

Abstract

Today's advancement in network is related to the advancement of packet switching technology. Switching directs network traffic in an efficient manner - it sends information directly from the port of origin to only its destination port. Without switching, many networks are experiencing bandwidth shortages because of an increase in traffic due to the increased numbers of users, the amount of data transported between client/server applications, and the inefficient traffic patterns of some networks.

Transmission of data from one destination to another is un guaranty successful. Sometimes the data will lost or corrupted during transmission to destination port. To make data more reliable, we need to improved switch operation. Latency, throughput and delay can effect network performances. To resolve this problem, remodeling the switch will help we find out the weakness of the switch, so from that we can enhance the functional of switch.

Cut through switching method is used in this project to improve network performances. To make packet more reliable and prevent packet loss during transmission in network, the buffer will be manage properly and effectively. This project also emphasized on switch fabric, buffer, input and output port to make transmission of data more efficient. To prevent waste of buffer, divide packet into smaller partition (eg. 64 bytes) is required.

The scope of this project is to design and develop a generic cut-through switch using Java tools. The switch is modeled from Ethernet cut through switch which is operates at LAN. Transmission packet is based on Ethernet header MAC

address. Ethernet switch architectures are modeled from crossbar approaches. Dummy packet is use for packer generator. The expected outcome for this project is a simulator that show how cut-through switch work and a graph that show the ratio of frames dropped.

Acknowledgement

First of all, I would like to take this opportunity to express my gratitude to the following persons. Without them, this project would never have been completed. First and foremost, I would like to express my gratitude and appreciation to my respected supervisor, Mr. Abdullah Gani for devoting his precious time in guiding me throughout the semester with patience and dedication. His guidance, support, and suggestions have helped me a lot to understanding the project requirements.

Next, I would like to address my thankful to my moderator, Mr Ling Teck Chaw for evaluating the project, and at the same time, forwarding useful suggestions and comments in order for me to come up with a better project performances.

Besides that, I would like to express my gratitude to my partner in this project, Juliana Jamaluddin for her dedicate contribution in idea and time to make up this project and for fellow friends for their supports and help. Last but not least, special thanks to my family members for giving me continuous motivation and support throughout this project.

Table of Contents

Abstract ii

Acknowledgement iv

List of Figures xii

List of Tables xiii

CHAPTER 1 : INTRODUCTION

1.1 Background 1

1.2 Motivation 3

1.3 Statement Of Problems 4

1.4 Aim and Objectives 5

1.5 Solution 6

1.6 Scope 7

1.7 Limitation 8

1.8 Expected Outcomes 9

1.9 Project Schedule 10

1.10 Report Layout 11

CHAPTER 2 : LITERATURE REVIEW

2.0 Chapter Introduction 14

2.1 Switching 14

2.2 Switching Techniques 14

	2.2.1 Circuit Switching	15
	2.2.2 Packet Switching	15
	2.2.3 Message Switching	16
2.3	Switch	16
2.4	Switch Operation	17
2.5	Switching Method	17
	2.5.1 Store and Forward	18
	2.5.2 Cut Through	18
	2.5.2.1 Fragment Free	19
	2.5.2.2 Fast Forward	19
2.6	Ethernet	19
2.7	Carrier Sense Medium Access / Collision Detection CSMA/CD	20
2.8	Ethernet Frame	20
2.9	Ethernet Switch Architecture	21
2.10	Ethernet Switch Design	22
	2.10.1 Shared Memory Approach	22
	2.10.2 Crossbar Switch	23
	2.10.3 Shared Bus	24
2.11	Buffering	24
	2.11.1 Input buffering	24
	2.11.2 Output buffering	26
	2.11.3 Path buffering	27
2.12	Discrete Event Simulation	27

2.13 Existing System 28

2.14 Chapter Summary 29

CHAPTER 3 : SYSTEMS METHODOLOGY

3.0 Introduction 30

3.1 Object Oriented Approach 30

3.2 Project development tools 32

3.2.1 Java is simple 32

3.2.2 Java is distributed 33

3.2.3 Portability: Program once, Run anywhere (Platform Independence) 34

3.2.4 Java is interpreted 34

3.2.5 Security 35

3.2.6 Reliability 35

3.2.7 Java is robust 36

3.2.8 Java is portable 36

3.2.9 Java is multithreaded 37

3.3 Development tools 37

3.3.1 J2SE v1.4.2 38

3.3.1.1 Java Foundation Class (JFC) 39

3.3.2 JCreator LE 40

3.4 Unified Modeling Language 41

3.4.1 Goals of UML 41

3.4.2 UML Diagrams 42

3.5 Rational Unified Process 43

3.5.1 The Inception Phase 44

3.5.2 The Elaboration Phase 44

3.5.3 The Construction Phase 44

3.5.4 The Transition Phase 45

3.6 Summary 45

CHAPTER 4: SYSTEM ANALYSIS

4.0 Introduction 46

4.1 Simulator Specification 46

4.1.1 Crossbar 46

4.1.2 Speed-up 46

4.2 Requirement Analysis 47

4.2.1 Functional requirements 47

4.2.2 Non-functional Requirement 48

4.3 Hardware Requirement 49

4.4 Software Requirements 50

4.5 Chapter Summary 50

CHAPTER 5: SYSTEM DESIGN

5.0 Chapter introduction 51

5.1 UML Diagram 51

5.1.1 Use Case Diagram 51

5.1.2 State Diagram 52

5.1.3 Sequence Diagram 53

5.1.4 Class Diagram 54

5.2 Switch Architecture 55

5.3 Crossbar Switch Simulation Model 56

5.4 Pseudocode 57

5.4.1 Pseudocode for TrafficGenerator 57

5.4.2 Pseudocode for BufferedCrossbarSwitch 57

5.5 Interface prototype 58

5.6 Chapter Summary 59

CHAPTER 6: SYSTEM IMPLEMENTATION

6.0 Chapter introduction 60

6.1 Development environment 60

6.1.1 Software development environment 61

6.2 Development of the system 61

6.2.1 System coding 63

6.3 Program Coding Approach 64

6.3.1 Simplicity and Clarity 64

6.3.2 Use meaningful variable names 65

6.3.3 Establish effective commenting conventions 65

6.3.4 Module 65

6.4 System Module 66

6.5 Coding style 67

6.5.1 Formatting and indenting codes 67

6.5.2 Commenting codes 67

6.6 Simulation result 68

CHAPTER 7: SYSTEM TESTING

7.0 Chapter introduction 70

7.1 Compiling and executing 70

7.2 Debugging 71

7.3 Accuracy of execution 71

7.4 Unit testing 72

7.5 Module testing 73

7.6 Integration testing 73

7.7 System testing 74

7.8 Chapter summary 75

CHAPTER 8: SYSTEM EVALUATION AND CONCLUSION

8.0 Chapter introduction 76

8.1 System evaluation 76

8.1.1 Expectations achieved 77

8.1.2 System limitations 77

8.1.3 Problems encountered 77

8.1.4 Knowledge gained 78

8.2 Project enhancement 79

CONCLUSION 81

REFERENCES 83

Appendix A : User Manual a

List of Figures

No.	Title of Figure	Page
Figure 1.1:	WXES3181 Development Gantt Chart	10
Figure 1.2:	WXES3182 Development Gantt Chart	10
Figure 2.1:	Ethernet Frame Format	20
Figure 2.2:	Ethernet Switch Architecture	21
Figure 2.3:	Shared Memory Switch	22
Figure 2.4:	Crossbar Switch	23
Figure 2. 5:	Shared Bus Switch	24
Figure 2.6 :	Input-buffered Switches	25
Figure 2.7:	Output-buffered Switches	26
Figure 2.8 :	Path buffering	27
Figure 5.1:	Use Case Diagram	51
Figure 5.2:	State Diagram	52
Figure 5.3:	Sequence Diagram	53
Figure 5.4 :	Class Diagram	54
Figure 5.5:	Crossbar switch architecture	55
Figure 5.6 :	Crossbar switch simulation model	56
Figure 5.7 :	Interface	58
Figure 5.8 :	User key in total of frames	59
Figure 6.1 :	Graph Frame Drop Ratio	69

List of Tables

No.	Title of Table	Page
Table 6.1 :	Frame Drop Ratio	68

CHAPTER 1: INTRODUCTION

1.0 Chapter Introduction

This chapter provides an overview of the project which includes the objective statement, the scope and schedule outline. In section 1.1 backgrounds is presented and section 1.2 states the motivation of the project. Statement of problem is in Section 1.3, the aim and objectives of the project is presented in Section 1.4. Section 1.5 discussed the solution while Section 1.6 stated the scope of the project. Limitation and expected outcome are presented in section 1.7 and section 1.8. Section 1.9 showed the Project Schedule while developing and completing the project. In addition, the report layout of the project is also given in section 1.10.

1.1 Background

Today's advancement in network is related to the advancement of packet switching technology. Switching directs network traffic in an efficient manner - it sends information directly from the port of origin to only its destination port. It manages network traffic by reducing media sharing. Without switching, many

networks are experiencing bandwidth shortages because of an increase in traffic due to the increased numbers of users, the amount of data transported between client/server applications, and the inefficient traffic patterns of some networks. The switching function operates at the data link layer (layer 2) of the OSI Model. The switch establishes a connection between two segments and keeps the connection just long enough to send the current packet. Incoming packets, which are part of an Ethernet frame has a special header that includes the MAC address information for the source and destination of the packet. The packets then will be placed to a temporary memory area (buffer). Then the switch reads the MAC address that is in the frame header and compares the address to a list of addresses in the switch lookup table. Switches forward packets to specific locations according to a set of rules. These rules form the basis of packet routing. The organization of components (e.g., buffering and switching elements) in a switch is commonly called its "architecture". Switch architectures are based on the location of buffering (at input or output of a switch), the type of switching elements, and so on.

There are two types method of switching: cut-through and store-and-forward. Cut-through switching only examines enough of a frame to determine the destination MAC address. It then establishes a connection to the interface through which that address can be reached and the frame is sent out. The frame is forwarded through the switch before the entire frame is received. No Cyclic Redundancy Check (CRC) verification is done in these switches.

Store and forward switching saves the entire packet to the buffer and checks the packet for Cyclic Redundancy Check (CRC) errors or other problems. If the packet has an error, the packet is discarded. Otherwise, the switch looks up the MAC address and sends the packet on to the destination node. This thesis is to design and develop Ethernet cut-through switch using Java.

1.2 Motivation

- Transmission of data from one destination to another is un- guaranty success. Sometimes the data will lost or corrupted during transmission to destination port. To make data more reliable, we need to improved switch operation.
- Latency, throughput and delay can effect network performances. To resolve this problem, remodeling the switch will help we find out the weakness of the switch, so from that we can enhance the functional of switch.
- To have deeper understanding of how switch works in a real environment.
- To learn and analyze how cut-through switch operate and its behaviors
- To understand the design of a switch model that can operates effectively.
- To learn the tool that can develop a cut-through switch model.

1.3 Statement of Problem

- Packet loss during transmission in network will increase network traffic because the packet will be transmitting again.
- Slow transmission of packet can add latency and delay that will be effect network performances
- High utilization of bandwidth can make congestion on network
- Wasting the use of buffer will needed buffer management

1.4 Aim and Objectives

The objectives of this project are listed below:

- The main aim and objective of this project is to design and develop a generic cut through switch model.
- To understand how cut-through method is applied while transmitting packet from origin port to its destination port.
- To analyze the performances of cut -through switch.
- Use Discrete Event Simulation to building up models to observe the time based (or dynamic) behaviors of a system.
- To develop a cut-through switch by using Java that based on object-oriented approach.

1.5 Solution

- Cut through switching method is used in this project to improve network performances
- To make packet more reliable and prevent packet loss during transmission in network, the buffer will be manage properly and effectively.
- This project also emphasized on switch fabric, buffer, input and output port to make transmission of data more efficient.
- To prevent waste of buffer, divide packet into smaller partition (eg. 64 bytes) is required.
- Designed crossbar (crosspoint) switches architecture .Crossbar switches have higher performance since the switches have the flexibility of connecting any input to any output. They are non blocking, guaranteeing that any input can find an uncongested path to an output, and eliminating the bandwidth limitations of one-at-a-time connections.

1.6 Scope

The scope of this project is to design and develop a generic cut-through switch using Java. The switch is modeled from Ethernet cut through switch which is operates at LAN. Ethernet networks operate at a data transmission rate of 10 Mbps. Transmission packet is based on Ethernet header MAC address. Ethernet switch architectures are modeled from crossbar approaches. Dummy packet is use for packet generator.

Switch architecture is designed with input and output port, buffers and switch fabric. Switching fabric includes data buffers, the integrated circuits that they contain, the programming that allows switching paths to be controlled and the use of shared memory and bus technology to move data between nodes.

Packet-switching technique offers high utilizations efficiency, links can be dynamically shared, support of multiple data rates for different types of terminals, no blocking when load on the network is high, high delivery time, priority deliveries can be used for important packets and robustness if one route becomes unavailable another route can be used.

1.7 Limitation

- Time constraints - the length of project and the time given to complete this project is not enough as the project need more time to study and analysis.
- There are few of resources and information about this project. The existing system also hard to find. It's needed more knowledge, reading, and analysis to know how to build the tools.
- Environment constraint – the project is focused on Ethernet switch

1.8 Expected Outcome

The expected outcomes for this project is

- Show how switch is modeled for transmission of packet from input port to output port in network
- Obtained the rate of packet loss during transmission of packets to output port.
- The information of each packets that are generated by traffic generator such as time arrival, source and destination MAC address, input and output port, buffers current and balance size and which frame is dropped.

1.9 Project Schedule

In developing and completing the project, proper planning is needed to meet the project objectives. The project schedule has to be met so that the project would not be delayed. Project schedule are listed in the Gant Chart below (Figure 1.1):

	Chapter	Start	Finish	Duration
1	Introduction	3/28/2005	4/18/2005	3w 1d
2	Literature Review	4/11/2005	4/26/2005	2w 2d
3	Methodology	4/22/2005	4/29/2005	1w 1d
4	System Analysis	4/29/2005	5/6/2005	1w 1d
5	System Design	5/5/2005	5/12/2005	1w 1d
6	Documentation/Presentation	5/12/2005	5/20/2005	1w 2d

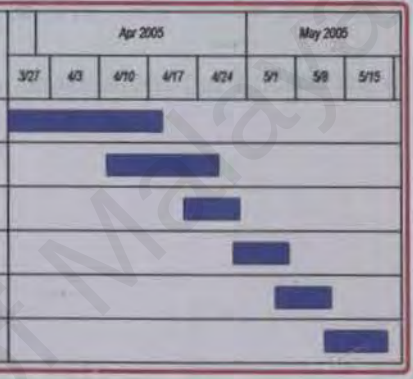

Figure 1.1: WXES3181 Development Gantt Chart

	Chapter	Start	Finish	Duration
1	System Implementation	5/8/2006	5/29/2006	16d
2	System Testing	5/19/2006	6/13/2006	18d
3	System Evaluation and Conclusion	6/5/2006	6/15/2006	9d
4	Documentation and Presentation	6/9/2006	6/22/2006	10d

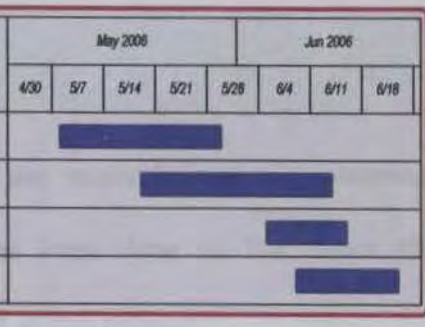

Figure 1.2: WXES3182 Development Gantt Chart

1.10 Report Layout

This report covers the project introduction, literature review, project development methodology scheme involved, system analysis and finally the system design:

a) Chapter 1 – Introduction

This chapter of the project presented an overview of the project. Some objectives of the project also been drawn up. In addition, the limitations of the project are also given.

b) Chapter 2 – Literature Review

This chapter explains about the research and study that has been done to develop the project. It covers the fundamental concepts of the proposed title. The existing system also will be analyzed to getting idea how to develop the system.

c) Chapter 3 - Methodology

This chapter describes and illustrates the methodology and planning involved in the project. A detailed study has been done to list down the requirements and the development tools involved in this project.

d) Chapter 4 – System Analysis

The System Analysis chapter elaborates on the functional and non-functional requirements of the project and the hardware and software needed to build the system.

e) Chapter 5- Design

This chapter presented the physical and logical design of the project. UML model such as Use Case, State, Class and Sequence Diagram will be presented and discussed.

f) Chapter 6 – System Implementation

This section discusses about how the system was implemented. It includes about the programming language used, some of the coding, and the coding style.

g) Chapter 7- System Testing

This chapter discusses about the testing that was done throughout the whole project.

h) Chapter 8 – System Evaluation & Conclusion

This chapter includes about the evaluation of the system which includes objectives achieved, system enhancement, problems faced, and discussion

1.10 Chapter Summary

Hopefully this project will be a stepping stone for those who want to enhance the project.

CHAPTER 2: LITERATURE REVIEW

2.0 Chapter Introduction

This chapter presents the overall study of Ethernet switch architecture. This chapter will begin with the importance of switching in networking .The functional and non functional of a switch will also be explained.

Going on, the chapter will discuss about cut through switch method. Finally the existing switch will be discussed and analyzed.

2.1 Switching

Switching directs network traffic in an efficient manner - it sends information directly from the port of origin to only its destination port. It manages network traffic by reducing media sharing.

2.2 Switching Technique

There are three types of switching technique:-

i) Circuit Switching

ii) Packet switching

iii) Message switching

2.2.1 Circuit Switching

This method involves the physical interconnection of two devices. A good example of circuit switching involves the Public phone network. Normal telephone service is based on a circuit-switching technology, in which a dedicated line is allocated for transmission between two parties. Circuit switching is ideal when data must be transmitted quickly and must arrive in the same order in which it's sent. This is the case with most real-time data, such as live audio and video.

2.2.2 Packet Switching

Refers to protocols in which messages are divided into packets before they are sent. Each packet is then transmitted individually and can even follow different routes to its destination. Once all the packets forming a message arrive at the destination, they are recompiled into the original message.

Most modern Wide Area Network (WAN) protocols, including TCP/IP, X.25, and Frame Relay, are based on packet-switching technologies. Packet switching is more efficient and robust for data that can withstand some delays in transmission, such as e-mail messages and Web pages.

2.2.3 Message Switching:

A method of handling message traffic through a switching center, either from local users or from other switching centers, whereby the message traffic is stored and forwarded through the system. Message Switching techniques were originally used in data communications. E-Mail delivery is example of message switching.

2.3 Switch

Switches are a fundamental part of most networks. Switches enable several users to send information over a network. Users can send the information at the same time and do not slow each other down. Just like routers allow different networks to communicate with each other, switches allow different nodes of a network to communicate directly with each other. A node is a network connection point, typically a computer.

Switch is an OSI Layer 2 device. It operates at the Data Link layer. Data Link layer contain two sub layers: Logical link Layer (LLC) and Media Access Control (MAC). LLC initiates the communication link between two nodes and ensures the link not broken. MAC examines the addressing information that

contains in a network frame and control how device share communication on the same network.

2.4 Switch Operation

LAN switches rely on packet switching. The switch establishes a connection between two segments and keeps the connection just long enough to send the current packet. Incoming packets, which are part of an Ethernet frame, save to a temporary memory area. The temporary memory area is a buffer. The switch reads the MAC (Media Access Control) address that is in the frame header and compares the address to a list of addresses in the switch lookup table. In a LAN with an Ethernet basis, an Ethernet frame contains a normal packet as the payload of the frame. The frame has a special header that includes the MAC address information for the source and destination of the packet.

2.5 Switching Method

Two types of architectures of switching: cut-through and store-and-forward.

2.5.1 Store and Forward

Store-and-forward switching, the destination and source addresses are read and filters are applied before the frame is forwarded. Buffers incoming packets in memory until they are fully received and a cyclic redundancy check (CRC) are run. Buffered memory adds latency to the processing time and increases in proportion to the frame size. This latency reduces bad packets and collisions that can adversely effect the overall performance of the segment. Utilizing this method, a switch reads an entire frame into an internal buffer. It then examines the MAC address. It compares the MAC address against an internal table of addresses, which tells the device which MAC addresses are on each interface. Once it has the interface identified, it sends the frame out that interface. The advantage to this method is that corrupted frames are identified and discarded without being forwarded. The disadvantage is that a buffer memory is required to store frames arriving on busy interfaces.

2.5.2 Cut Through

Cut-through switching only examines enough of a frame to determine the destination MAC address. It then establishes a connection to the interface through which that address can be reached and the frame is sent out. The frame is forwarded through the switch before the entire frame is received.

The advantage of this method is very fast operation, reduces transmission latency between ports. The disadvantage is it reduces error detection so the corrupted frames will be forwarded. It also can propagate broadcast storms to the destination port. No Cyclic Redundancy Check (CRC) verification is done in these switches. Cut through haves 2 categories: - Fragment Free and Fast Forward.

2.5.2.1 Fragment-Free

Checks that there are no collisions within the first 64 bytes of the packet, the minimum valid message size required by the IEEE 802.3 specifications. This guarantees that message fragments less than 64 bytes (runts) are not forwarded to other network segments. Runts are typically the result of collision fragments.

2.5.2.2 Fast Forward

Immediately forwards a packet after reading the first 14 bytes. There may be transmitted the packets with error.

2.6 Ethernet

Ethernet is a frame-based computer networking technology for LAN. It standardized as Institute of (IEEE)'s 802.3. Access mechanism used in Ethernet is called CSMA/CD.

2.7 Carrier Sense Medium Access / Collision Detection CSMA/CD

Before send the data, the sender listen for existing traffic on the line or channel, by trying to sense the signal or carrier. If no transmission is sensed, multiple accesses allow anyone onto the media to transmit data. If another node has access at the same time, collision occurs. If collision detected, the node will quits the transmission and wait a random of time for line to be clear, then sends data again later.

2.8 Ethernet Frame

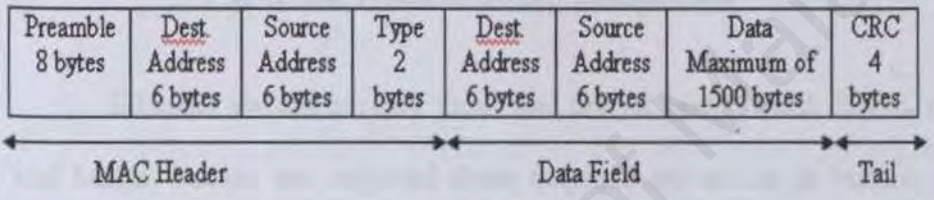

Figure 2.1: Ethernet Frame Format

- Preamble Field used for synchronization, 64-bits
- Destination Address is Ethernet address of the destination host, 48-bits
- Source Address is Ethernet address of the source host, 48-bits
- Type is Type of data encapsulated, e.g. IP, ARP, RARP, etc, 16-bits.
- Data Field is data area, 46-1500 bytes, which has Destination Address of destination host and Source Address of source host
- CRC means Cyclical Redundancy Check, used for error detection, the CRC check is still made and, if errors are found, the error count is updated.

2.9 Ethernet Switch Architecture

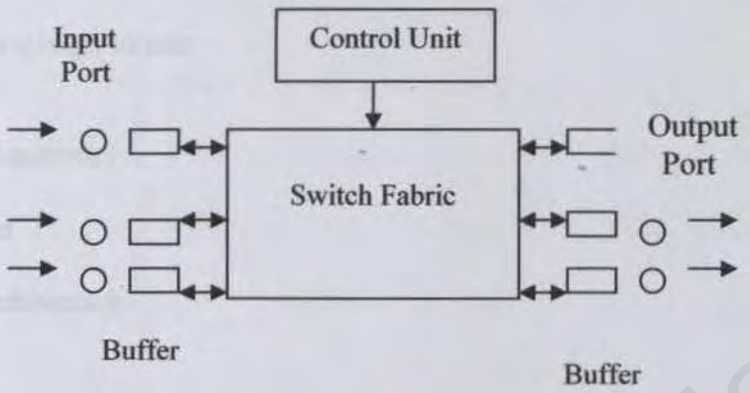

Figure 2.2: Ethernet Switch Architecture

Ethernet switch contain input and output port, switch fabric, control unit and buffer. Packet are received from network are stores in buffers (temporary repository for arriving packets while they wait to be processed) during processing, after they are queued for transmission. Control unit controls entire operation of switch. The control processor loads the forwarding table for routing or signaling. Switching fabric includes the switching units in a node, the integrated circuits that they contain, and the programming that allows switching paths to be controlled. There are three types of switch fabric: - Shared Memory, Shared Bus and Crossbar.

2.10 Ethernet Switch Design

LAN switches different in physical design. Currently, there are three popular design in use:

- Shared-memory
- Crosbar
- Bus-architecture

2.10.1 Shared Memory Approach

Figure 2.3 shows the basic structure of a shared memory switch. Incoming packets are converted from serial to parallel form, and written sequentially to a dual port Random Access Memory. A memory controller decides the order in which packet are read out of the memory, based on the packet headers with internal routing tags. Outgoing packets are demultiplexed to the outputs and converted from parallel to serial form.

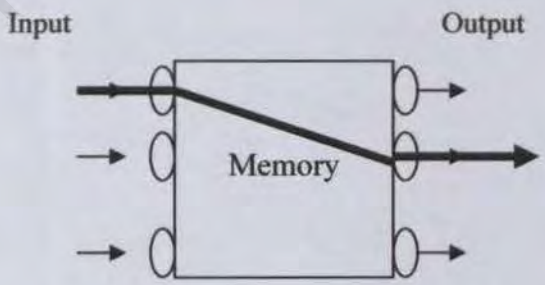

Figure 2.3: Shared Memory Switch

2.10.2 Crossbar Switch

This type of switch has an internal grid with which the input ports and the output ports cross each other. When the switch detects a packet on an input port, the switch compares the MAC address to the lookup table to find the appropriate output port. The switch then makes a connection on the grid where these two ports intersect.

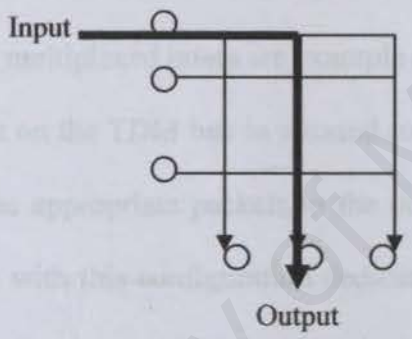

Figure 2.4: Crossbar Switch

2.10.3 Shared Bus

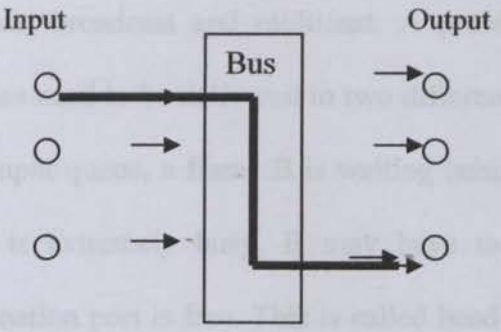

Figure 2. 5: Shared Bus Switch

Time-division multiplexed buses are example of this approach. Packets are sequentially broadcast on the TDM bus in a round-robin manner. At each output, address filters pass the appropriate packets to the output buffers, based on their routing tag. A switch with this configuration dedicates a memory buffer to each port. There is an application-specific integrated circuit (ASIC) to control the internal bus access.

2.11 Buffering

Three types of buffering techniques are used in switch architectures. They are:

2.11.1 Input buffering : Frames are buffered at each input port. So the incoming frame is stored in the buffer as it is received. This is helpful in situations when more that one station is trying to send to

the same output port. The input is buffered and forwarded only when the destination port is free. Input buffering may be used to support broadcast and multicast. A problem might occur when frames need to be delivered to two different destination ports. If in the input queue, a frame B is waiting behind A whose destination port is extremely busy, B may have to wait even though its destination port is free. This is called head-of-line blocking. Some switches can avoid such a problem if the control logic can look into the destination address of the frames in the queue and if destination address is free, forward them bypassing the first frame. Such switches are more efficient. Or use a well-known buffering scheme called Virtual Output Queueing (VOQ) in which each input maintains a separate queue for each output.

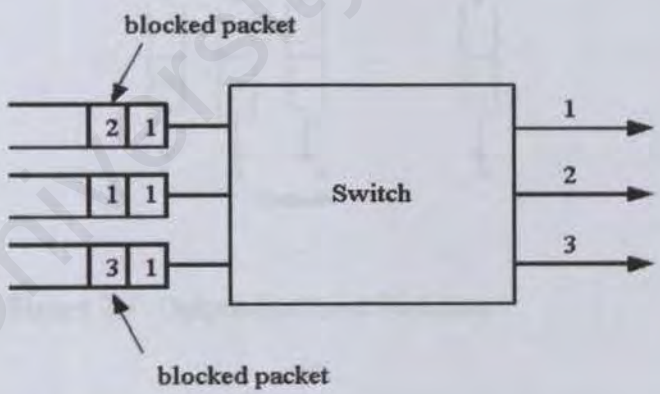

Figure 2.6 : Input-buffered Switches

2.11.2 Output buffering. This solves the problem of head-of-line blocking as frames received are forwarded to the destination, and buffered in output buffer if the output port is busy. Common buffer pools may be used where multiple output streams may be buffered in a common memory space. Fast memory with multiple access is required in this case. Output buffer is useful when the traffic to a destination port is heavy. Problem might occur when one station is sending heavy load to a destination port, then other stations sending to the same destination port may be denied access. So this is not a fair method.

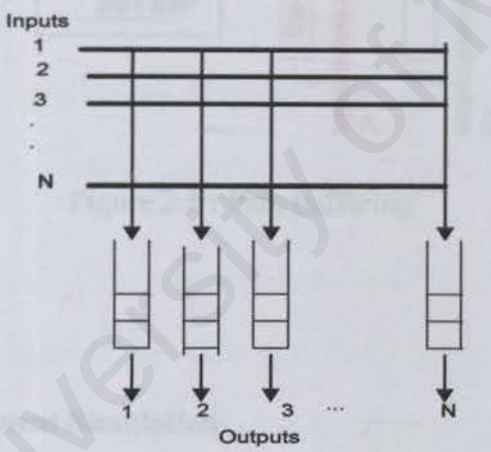

Figure 2.7: Output-buffered Switches

2.11.3 Path buffering. Both of the above problems can be avoided with this method as separate buffers exist for each input and output port pair.

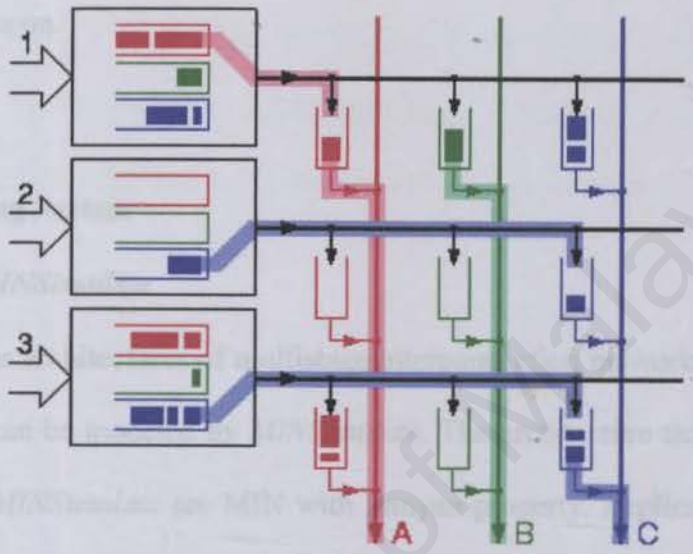

Figure 2.8 : Path buffering

2.12 Discrete Event Simulation

Simulation is the technique of a building a model of a real or proposed system so that the behaviour of the system under specific conditions may be studied. One of the key powers of simulation is the ability to model the behaviour of a system as time progresses.

Discrete event simulation is one way of building up models to observe the time based (or dynamic) behaviour of a system. There are formal methods for building simulation models and ensuring that they are credible. During the experimental phase the models are executed (run over time) in order to generate results. The results can then be used to provide insight into a system and a basis to make decisions on.

2.13 Existing System

- **Simulator *MINSimulate***

Various architectures of multistage interconnection networks exists. Some architectures can be modeled by *MINSimulate*. The architecture that are modeled by simulator *MINSimulate* are MIN with Banyan property, Replicated MIN and Multilayer MIN. Simulation is performed by c++ code.

The simulator is packet based and has Graphical User Interface (GUI). The tool's GUI support the wide variety of simulation parameters is easily accessible to the user. *MINSimulate* is able to simulate simple crossbars, multistage interconnection networks (MINs) with the delta property, and MINs that are arranged in multiple layers. Simulations can be performed using a wide variety of input parameters such as offered load, multicast traffic pattern, routing scheme or buffer configuration. The simulation based on performance measures such as throughput, mean delay, delay time distributions or mean buffer queue lengths in individual network stages.

During stimulation, all data is evaluated according to pre-set levels of confidence and accuracy using the statistical library Akaroa. Transient network behavior can also be evaluated with *MINStimulate*, that is useful for studies of the fine grain time-dependent performance, especially when observing traffic that changes with time. The presented tools allow for evaluating various network configurations under different traffic conditions. Therefore, one can easily establish knowledge whether a particular network design suits a task at hand.

2.14 Chapter Summary

The overview of fundamental concept of project has been presented in this chapter. More reading and analysis is needed to implement this information into my project.

CHAPTER 3 : SYSTEMS METHODOLOGY

3.0 Chapter Introduction

This chapter entails the entire methodology for building the switch simulator. The tools to build the simulator will also be discussed. Methodology here means a way of developing a software product.

3.1 Object Oriented Approach

This project is build using Java programming. Java implements the object-oriented approach. Object oriented programming is a type of programming in which it defines not only the data type of data structures but also the type of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an object that includes both data and functions. In addition, relationship between one object and another can be created. For example, objects can inherit characteristics from another object.

One of the advantages of object-oriented programming techniques programming is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply

create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify.

The concepts and rules used in object-oriented programming provide these important benefits:

- The concept of a data class makes it possible to define subclasses of data objects that share some or all of the main class characteristics. Called inheritance, this property of OOP forces a more thorough data analysis, reduces development time, and ensures more accurate coding.
- Since a class defines only the data it needs to be concerned with, when an instance of that class (an object) is run, the code will not be able to accidentally access other program data. This characteristic of data hiding provides greater system security and avoids unintended data corruption.
- The definition of a class is reusable not only by the program for which it is initially created but also by other object-oriented programs (and, for this reason, can be more easily distributed for use in networks).
- The concept of data classes allows a programmer to create any new data type that is not already defined in the language itself.

One of the first object-oriented computer languages was called Smalltalk. C++ and Java are the most popular object-oriented languages today. The Java

programming language is designed especially for use in distributed applications on corporate networks and the Internet.

3.2 Project Development Tools

In this project Java programming language would be used to build the switch simulator. This is because Java implements the object oriented approach. Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, multithreaded, and dynamic.

3.2.1 Java is Simple

No language is simple, but Java considered a much simpler and easy to use object-oriented programming language when compared to the popular programming language, C++. Partially modeled after C++, Java has replaced the complexity of multiple inheritance in C++ with a simple structure called interface, and also has eliminated the use of pointers.

Java is Object-oriented programming models the real world. Everything in the world can be modeled as an object. For example, a circle is an object, a person is an object, and a window's icon is an object. Even a mortgage can be perceived as an object. Java is object-oriented because programming in Java is centered on creating objects, manipulating objects, and making objects work together.

An object has properties and behaviors. Properties are described by using data, and behaviors are described by using methods. Objects are defined by

using classes in Java. A class is like a template for the objects. An object is a concrete realization of a class description. The process of creating an object class is called instantiation. Java consists of one or more classes that are arranged in a treelike hierarchy, so that a child class is able to inherit properties and behaviors from its parent class. An extensive set of pre-defined classes, grouped in packages that can be used in programs are found in Java.

Object-oriented programming provides greater flexibility, modularity and reusability. For years, object-oriented technology has been perceived as an elitist, requiring substantial investments in training and infrastructure. Java has helped object-oriented technology enter the mainstream of computing, with its simple and clean structure that allows the programmer to write easy to read and write programs.

3.2.2 Java is Distributed

Distributed computing involves several computers on a network working together. Java is designed to make distributed computing easy with the networking capability that is inherently integrated into it. Writing network programs in Java is like sending and receiving data to and from a file.

3.2.3 Portability: Program once, Run anywhere (Platform Independence)

One of the most compelling reasons to move to Java is its platform independence. Java runs on most major hardware and software platforms, including Windows 95 and NT, the Macintosh, and several varieties of UNIX. Java applets are supported by all Java-compatible browsers. By moving existing software to Java, you are able to make it instantly compatible with these software platforms. JAVA programs become more portable. Any hardware and operating system dependencies are removed.

3.2.4 Java is Interpreted

An interpreter is needed in order to run Java programs. The programs are compiled into Java Virtual Machine code called bytecode. The bytecode is machine independent and is able to run on any machine that has a Java interpreter. Usually, a compiler will translate a high-level language program to machine code and the code is able to only run on the native machine. If the program is run on other machines, the program has to be recompiled on the native machine. With Java, the program need only be compiled once, and the bytecode generated by the Java compiler can run on any platform.

3.2.5 Security

Java is one of the first programming languages to consider security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind. The compiler, interpreter,and Java-compatible browsers all contain several levels of security measures that are designed to reduce the risk of security compromise, loss of data and program integrity, and damage to system users. Considering the enormous security problems associated with executing potentially untrusted code in a secure manner and across multiple execution environments, Java's security measures are far ahead of even those developed to secure military systems. C and C++ do not have any intrinsic security capabilities.

3.2.6 Reliability

Security and reliability go hand in hand. Security measures cannot be implemented with any degree of assurance without a reliable framework for program execution. Java provides multiple levels of reliability measures, beginning with the Java language itself. Many of the features of C and C++ that are detrimental to program reliability, such as pointers and automatic type conversion, are avoided in Java. The Java compiler provides several levels of additional checks to identify type mismatches and other inconsistencies. The Java runtime system duplicates many of the checks performed by the compiler

and performs additional checks to verify that the executable byte codes form a valid Java program.

3.2.7 Java is Robust

Robust means reliable and no programming language can really assure reliability. Java puts a lot of emphasis on early checking for possible errors, as Java compilers are able to detect many problems that would first show up during execution time in other languages. Java eliminates certain types of programming constructs in other languages that are prone to errors. For instance, Java does not support pointers, which eliminates the possibility of overwriting memory and corrupting data. Java has a runtime exception-handling feature to provide programming support for robustness, and can catch and respond to an exceptional situation so that the program can continue its normal execution and terminate gracefully when a runtime error occurs.

3.2.8 Java is Portable

One advantage of Java is that its programs can run on any platform without having to be recompiled. This is one positive aspect of portability. It goes on even further to ensure that there are no platform-specific features on the Java language specification. For example, in some languages, such as Ada, the

largest integer varies on different platforms. In Java, the size of the integer is the same on every platform, as is the behavior of arithmetic. Having a fixed size for numbers makes Java programs portable. The Java environment itself is portable to new hardware and operating systems, and in fact, the Java compiler itself is written in Java.

3.2.9 Java is Multithreaded

Multithreaded is the capability for a program to perform several tasks simultaneously within a program. For instance, downloading a mp3 file while playing the file would be considered multithreading. In Java, multithreaded programming has been smoothly integrated into it, while in other languages, operating system-specific procedures have to be called in order to enable multithreading. Multithreading is especially useful in graphical user interface (GUI) and network programming. In GUI programming, many things can occur at the same time. For example, a user is able to listen to a mp3 file and surf the Web at the same time. In network programming, a server can serve multiple clients at the same time. Multithreading is a necessity in visual and network programming.

3.3 Development Tools

For this project the development tools used are as follows:

- Java 2 Platform, Standard Edition (J2SE 1.4.2)
- JCreator LE
- The system to be developed under Windows XP Professional

3.3.1 J2SE v1.4.2

The Java 2 Platform, Standard Edition is at the core of Java technology, and version 1.4.2 raises the java platform to a higher standard. It provides a complete environment for applications development on desktops and servers. It also primarily involve GUI, connectivity, virtual machine for the Java platform* (Java Virtual Machine (JVM)) and core libraries. From client to server, from desktop to supercomputer, improvement has been made to J2SE across the board. With version 1.4.2, enterprises can now use Java technology to develop more demanding business applications with less effort and in less time.

Version 1.4.2 builds upon the current J2SE platform and provides even more features for developers to build into their applications. More functionality in 1.4.2 means developers can now spend less time writing custom code to accomplish what is now part of the core J2SE platform. The result is faster applications programming with more consistency for enterprise development initiatives. New features in J2SE v1.4.2 also reduce the developer's reliance on other technologies such as C or C++, PERL, SSL and DOM implementation in browsers. This allows

developers to use a single technology to develop, test and deploy end-to- end enterprise application and software. Most anything you want to do, you can do it in J2SE user interface.

Version 1.4.2 provides more ways for developers leverage existing systems without changing their underlying platforms. It provides additional support for industry standards technologies such as XML, DOM, SSL, Kerberos, LDAP and CORBA to ensure operability across heterogeneous platforms, systems and environments. Additionally developers and software vendors may now use a new endorsed standard override mechanism in version 1.4.2 to provide newer versions of endorsed standard such as CORBA, as they become available.

3.3.1.1 Java Foundation Class (JFC)

JFC are a set of Java class libraries provided as part of the Java platform to support building graphics user interface and graphics functionality for Java technology-based client applications ("Java applications"). JFC includes an extensive set of technologies that enable developers to create a rich interactive user interface for client applications that can run not only on Microsoft Windows but also on other increasingly popular platform such Mac OSX and Linux.

Features of JFC :

- Abstract Window Toolkit (AWT): APIs that enable programs to integrate into native window system, including APIs for Drag Drop.
- Java 2D: APIs to enable advanced 2D graphics, imaging, text and printing.

- Swing GUI components
- Accessibility: APIs and assitive technologies for ensuring an applications is accessible to users with disabilities and meets government requirements for accessibility.
- Internationalization: API JFC technologies include support for creating application that can interact with users around the world using the user's own language. This includes the Input Method Framework API.

These five technologies are designed to be used together to enable developers to build fully functional GUI client applications that run and integrate on any client machine that supports J2SE, including Microsoft Windows, Solaris, Linux and Mac OSX.

3.3.2 JCreator LE

JCreator is a powerful IDE (Integrated Development Environment) for Java technologies. It provides the user with templates, class browsers, a debugger interface, syntax highlighting, wizards and a fully customizable user interface.User can directly compile or run their Java programming without having to activate the main document first. JCreator will automatically find the file with the main method or the html file holding the applet, and then start the appropriate tool.

Users can also create their own tools for calling Java Development Kit (JDK) applications such as the complier, interpreter or application viewer. JCreator also supports multiple complier tools that can be switched with the runtime configuration dialogue box.

3.4 Unified Modeling Language

The UML is a complete language that is used to design, visualize, construct and document systems. It is largely based on the object-oriented paradigm and is an essential tool for developing robust and maintainable software systems.

3.4.1 Goals of UML

The primary goals in the design of the UML were:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.

5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

3.4.2 UML Diagrams

Use Case Diagram:

An actor is represents a user or another system that will interact with the system. A use case is an external view of the system that represents some action the user might perform in order to complete a task. A use case diagram displays the relationship among actors and use cases.

Class Diagram:

Class diagrams are used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. It also displays relationships such as containment, inheritance, associations and others.

Sequence Diagram:

The Sequence Diagram describes messages exchanged between classes to accomplish tasks.

State Diagram:

State Diagrams model the dynamic behavior of a system by showing the various states that an object can get into and the transitions that occur between the states.

3.5 Rational Unified Process

The Rational Unified Process has been design with techniques similar to techniques that are used to design software. Especially, the process has an underlying object-oriented model, using Unified Modeling Language UML. The Rational Unified Process divides a development cycle into four phases. The phases are:

- The inception phase
- The elaboration phase
- The construction phase
- The transition phase

3.5.1 The Inception Phase

In this phase, a basic use case model, project plan, initial risk assessment and a document that generally describes the project (the core project requirements, constraints and key features) are realized.

3.5.2 The Elaboration Phase

The Elaboration phase is where big things happen, in this phase the problem domain analysis is made and also the architecture of the project gets its basic form. There is big step from this to next because this step means transition from low-risk operation to high-risk operation.

3.5.3 The Construction Phase

In this phase the main focus goes to the development of components and other features of the system that is being developed. This is the phase when coding takes place.

3.5.4 The Transition Phase

The transition is phase where the product moved from the development organization to the end user.

3.6 Chapter Summary

In this chapter, the design methodology and the development tools have been discussed. Java has significant advantages not only as a commercial language but also as a teaching language. It allows students to learn object-oriented programming without exposing them to the complexity of C++.

CHAPTER 4: SYSTEM ANALYSIS

4.0 Chapter Introduction

In this chapter all the requirements and consideration are analyzed and translated into a software model. All the specification needed to build the simulator will be discussed.

4.1 Simulator Specification

4.1.1 Crossbar

Although most switch architectures for modern systems are non-blocking, three types of blocking can limit performance when multiple input ports are contending for an output port: Head-of-Line (HOL) blocking, input blocking, and output blocking. HOL blocking can waste nearly half a crossbar switch's bandwidth if the cells waiting at each input are stored in a single First-In, First-Out (FIFO) queue.

4.1.2 Speed-up

Speed-up is a common way to reduce input and output blocking by running the crossbar switch faster than the external line rate. For example, if the crossbar switch runs twice as fast as the external line, the traffic manager can transfer two cells from each input port, and two cells to each output port during each cell time. The advantage of speed-up is obvious — it offers more predictable delay and jitter

across the switch ports by delivering more cells per cell time, and thus reducing the delay of each cell through the switch. In fact, sufficient speed-up can guarantee that every cell is immediately transferred to the output port, where its departure time can be precisely scheduled.

4.2 Requirement Analysis

Requirement analysis includes functional requirement and non-functional requirement.

4.2.1 Functional requirements

Specify a function that a system or a system's component must be able to perform. These are software requirements that define the behaviors of a system that is the fundamental process of transformation that software and hardware components of the system perform on inputs to provide expected output. Below is the information that switch simulator must keep.

There are five components recognized as the most important functional requirement for the project.

1. Generate packet

   Traffic generator will be used to generate the packet.

2. Transmit and receive packet

Switch fabric will determine the path of packet from input port to output port.

3. Process packet

The switch model will be designed with lookup table that contains information of input port, output port, source MAC address and destination MAC address to process packet while transmission .

4. Generate table

The table is created to list the result of the crossbar simulation. The results are time arrival of each frame, source and destination MAC address of frames, scheduling algorithms and total of packets loss.

4.2.2 Non-functional Requirement

Non-functional requirement is a description of the features, characteristics and attributes of the system. The non-functional requirements for this project are as follows:

(i) User interface

The simulator will be a stand alone system using a Java interface. The user interface will be simple for the ease of users.

(ii) Usability

The cut through switch is modeled to see the result of simulators. It is build to simulate the process and performances of the crossbar switch architecture. From that, we can enhance the research or apply it to real environment.

(iii) Reliability

To which extent a system can be expected to perform its extended function. System must be reliable to provide accurate result.

(iv) User friendly

User can easy to understand how to use menu button and toolbars.

(v) Maintainability

System maintenance is a must for this system, as it allows certain changes or modifications to be made over the system.

4.3 Hardware Requirement

The minimum hardware requirements to build the system are:

- Windows 95/ 98/ ME/XP/ 2000/ NT 4.0
- 64 MB RAM
- 4.0 GB Hard Disk space

4.4 Software Requirements

Software requirements are a combination of tools to develop all the modules specified. Listed below are the software and tools required to build cut through switch. The software requirements are chosen based on their functionality, affordability, easy to use and user friendliness feature.

As this projects going to be built using Java, the Java plug-in must be installed in order to view the interface. This software can be run in any platform as Java is platform independence.

- Java 2 Platform, Standard Edition (J2SE v1.4.2)
- JCreator LE
- Windows XP Professional

4.5 Chapter Summary

All the requirements for building switch simulator have been specified and analyzed. This includes functional and non-functional requirement, hardware and software specification.

CHAPTER 5: SYSTEM DESIGN

3.0 Chapter Introduction

System Design shows the Use Case, State, Sequence Diagram and Class Diagram for my project using UML approaches. System design also focuses on the development of switch simulator.

5.1 UML Diagram

5.1.1 Use Case Diagram

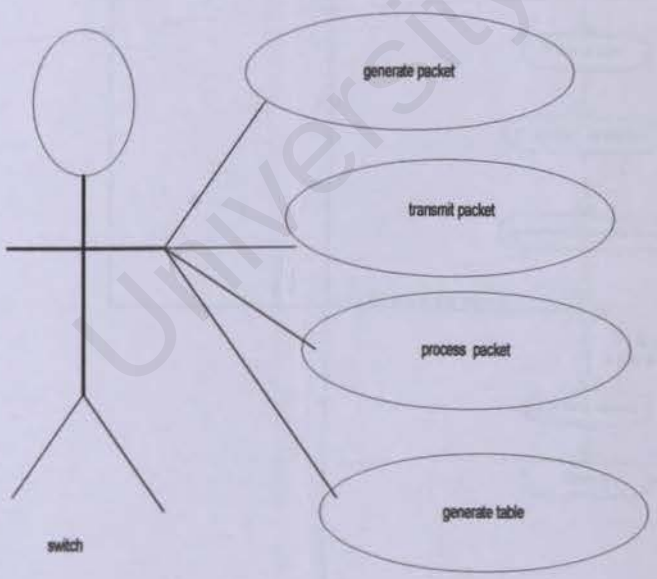

Figure 5.1: Use Case Diagram

An actor is represents a user or another system that will interact with the system. A use case is an external view of the system that represents some action the user might perform in order to complete a task. From Figure 5.1 stated the function of switch: Generates packet, Transmits packet, Process packet and Generate table.

5.1.2 State Diagram

Figure 5.2 presented the State Diagram of my project.

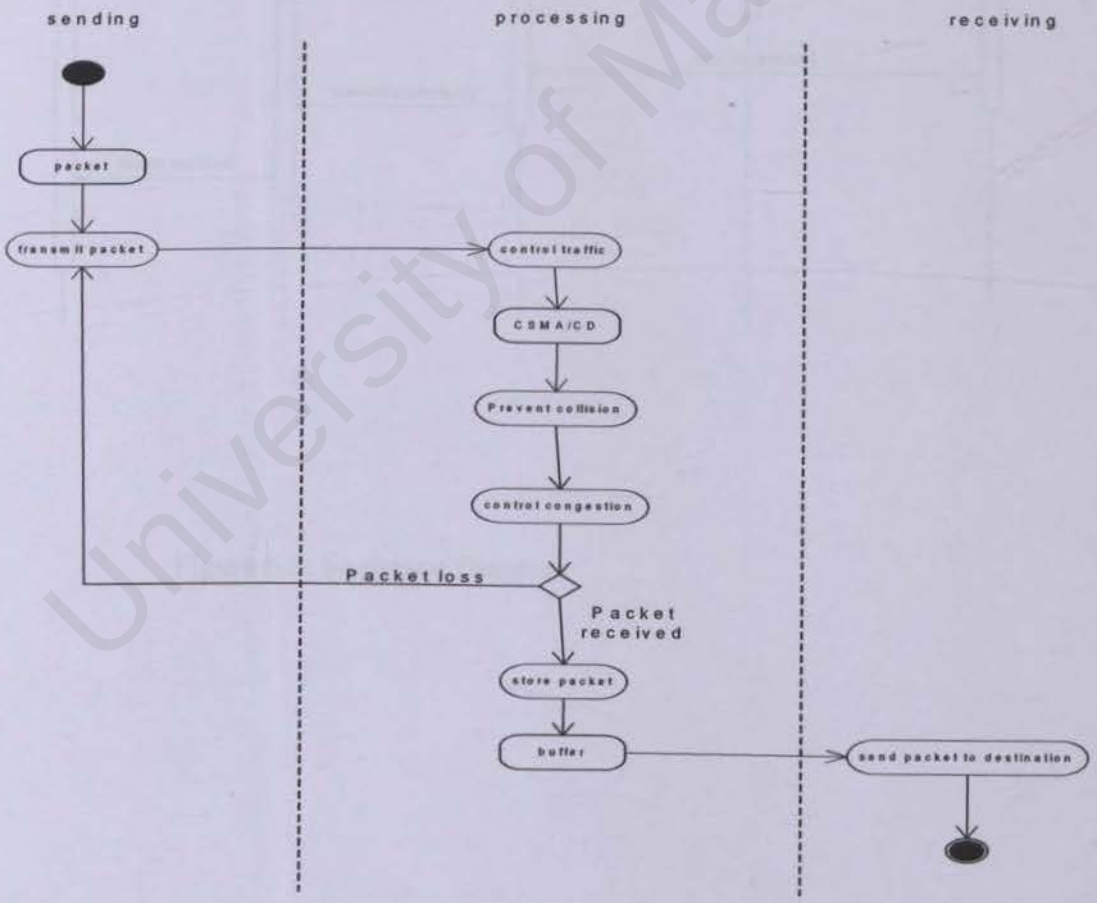

Figure 5.2: State Diagram

5.1.3 Sequence Diagram

Figure 5.3 presented Sequence Diagram for my project.

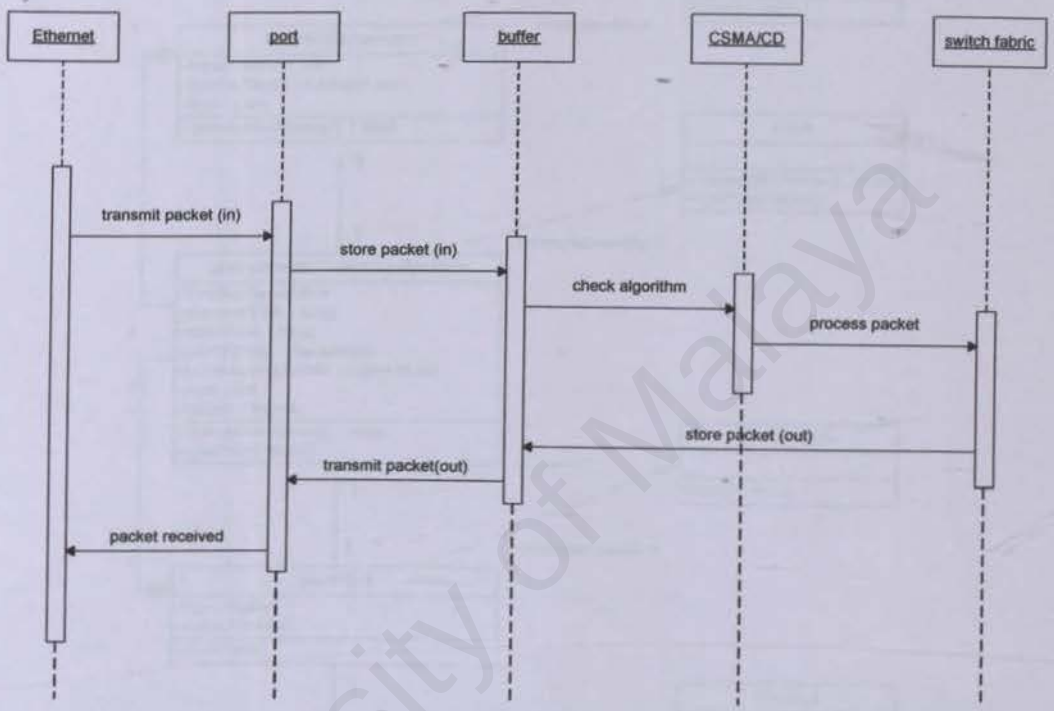

Figure 5.3: Sequence Diagram

5.1.4 Class Diagram

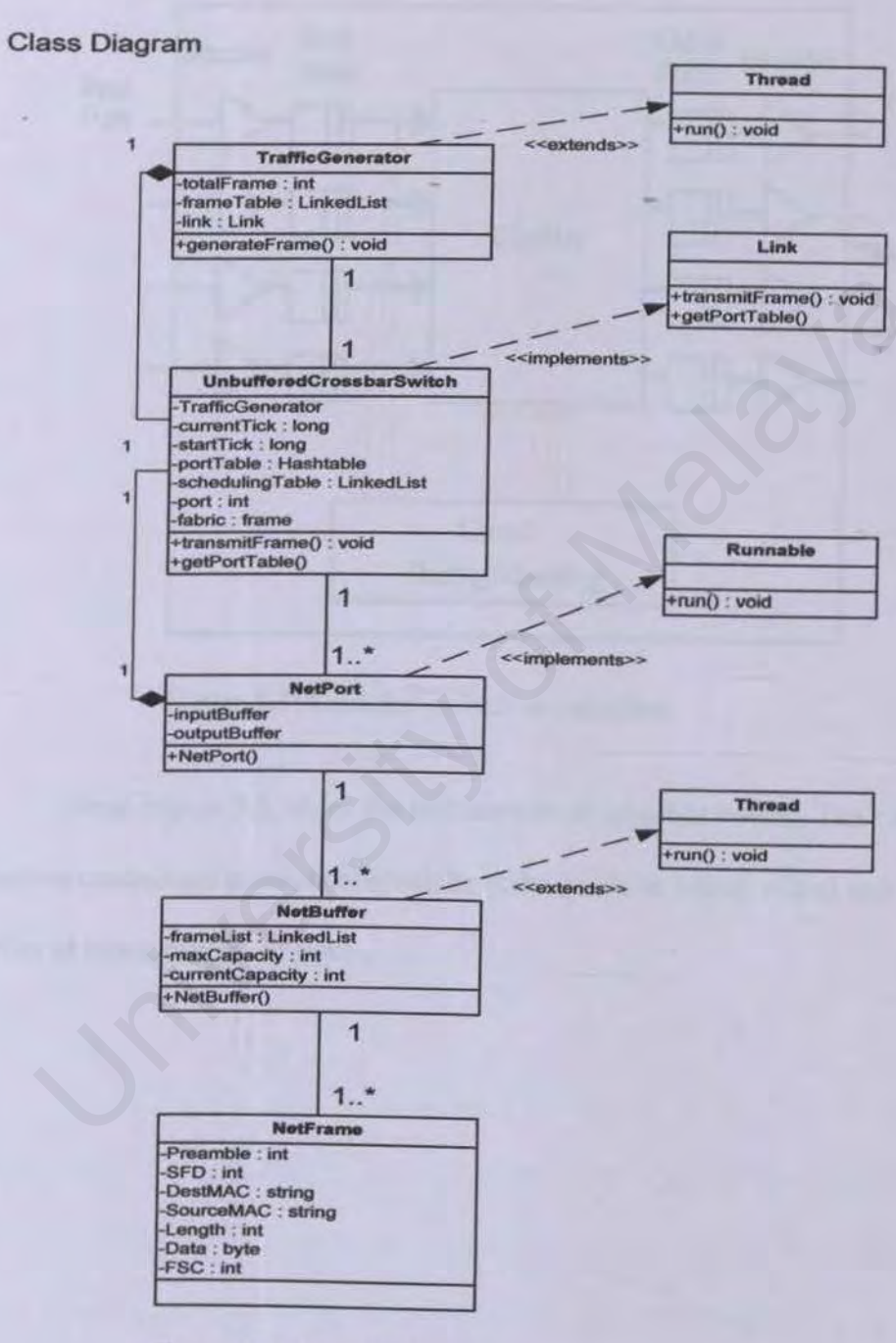

Figure 5.4 :Class Diagram

5.2 Switch Architecture

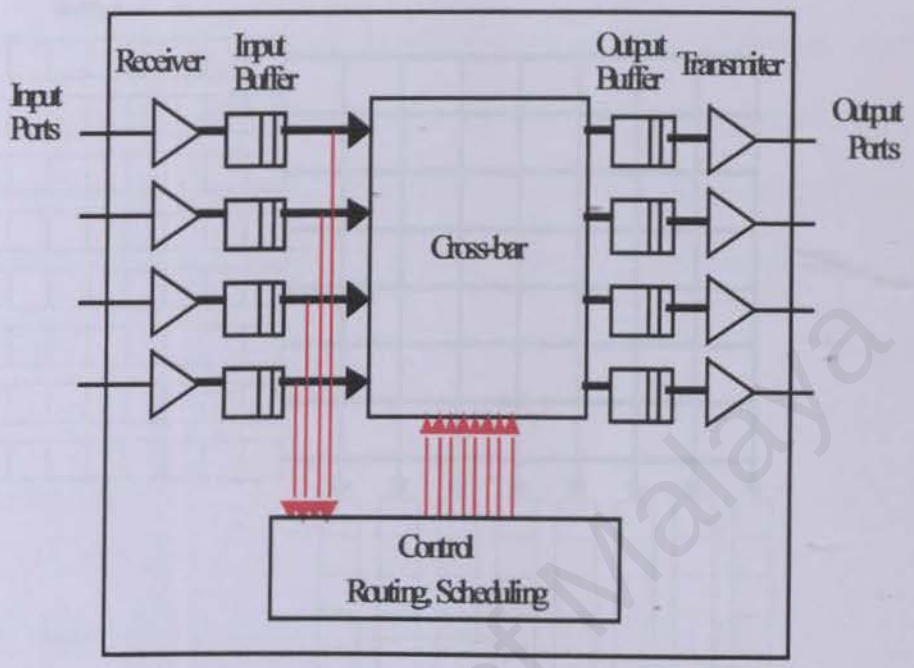

Figure 5.5: Crossbar switch architecture

From Figure 5.5, show the architecture of crossbar switch. The switches requires centralized controller/scheduler (who sends to whom when) and can buffer at input, output, or both .

5.3 Crossbar Switch Simulation Model

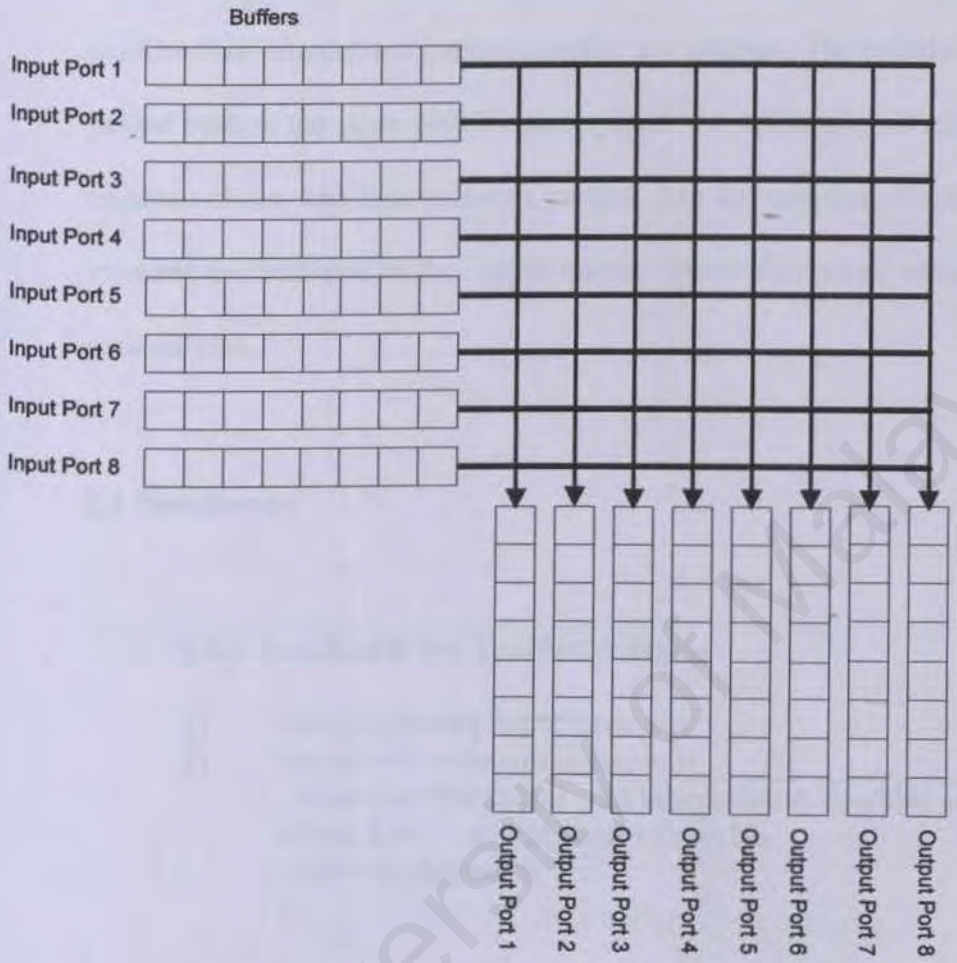

Figure 5.6 : Crossbar switch simulation model

As shown in Figure 5.6, switch simulation model consists of three main parts: 8 ports, 8 buffers in each ports, and a centralized scheduler for configuring the crossbar. When a frame arrives at a port, it is buffered in an input queue according to its destination. The frame awaits a decision by the scheduler allowing it to traverse the crossbar switch fabric. We assume one frame has 64 bytes length and maximum frames can store in each port is 8 frames

(1 buffers can store 1 frames). The frame will be queue in input port based on FIFO (First In First Out)..The scheduler examines the contents of all input queues, decides upon the configuration of the crossbar, and chooses a set of conflict-free connections between inputs and outputs. The scheduling decision is passed back to the ports which communicate the configuration information to the crossbar slices, and then transmit packets into the crossbar. Packets leaving the crossbar are buffered in the output queues where they await transmission to the external line.

5.4 Pseudocode

5.4.1 Pseudocode for TrafficGenerator

```
1)      Get totalPackets input from user
2)      for (int i=0; i<totalPackets;i++){
          //Generate the packet with sourceMac & destMac randomly
        assign &&      sourceMac != destMac
        //transmit the packet
        }
```

5.4.2 Pseudocode for BufferedCrossbarSwitch

```
1)      When FrameA arrives at the switch
        Port1= check source Mac of FrameA

        BuferC = check destination MAC of FrameA
        If (Buffer1C.current_size + FrameA.length <=
        Buffer1C.max_capacity ) {
Buffer1C.add (FrameA); //put the packet into buffer
Buffer1C.current_size = Buffer1C.current_size + FrameA.length;
        }
else{
FrameA = null; //drop the packet
}
```

```
2)      if (Port3.booFree = true) { // destination port is free
        loop allPorts {
if( Buffer1C.curent_size != 0) {
remove frameA from buffer;
Buffer1C.current_size = Buffer1C.current_size – FrameA.length;
transmit FrameA to Port3;
Port3.boofree = false;
}
else {

//FIFO queue
   }
  }
}
```

5.5 Interface prototype

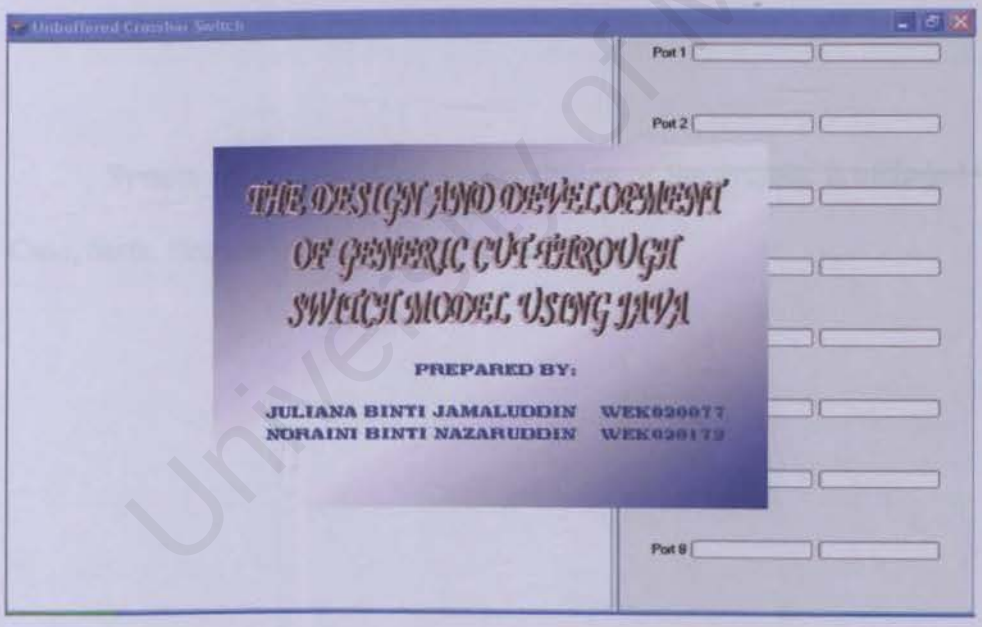

Figure 5.7 : Interface

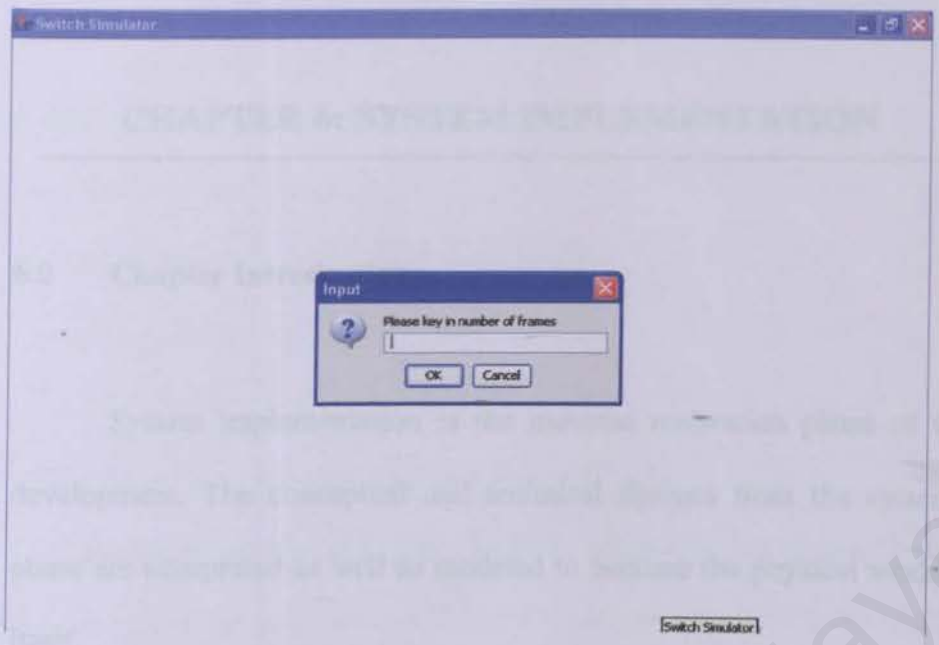

Figure 5.8 : User key in total of frames

5.6 Chapter Summary

System Design provided logical design of the project; it included the Use Case, State, Sequence Diagram and Class Diagram.

CHAPTER 6: SYSTEM IMPLEMENTATION

6.0 Chapter Introduction

System implementation is the material realization phase of the system development. The conceptual and technical designs from the system analysis phase are interpreted as well as modeled to become the physical working system itself.

The following subchapters will explain the development environment as well as the development of the system itself, some system coding and the coding style and approach and object oriented technique applied in the simulation of the switch.

6.1 Development environment

The hardware configured for the development environment is the underlying element of the whole system. The hardware used in the system implementation phase plays an important role in realizing the final system architecture.

The hardware configuration for the development environment is as below:

- AMD Athlon Processor
- 654 Mhz
- 64 MB RAM

6.1.1 Software development environment

Hardware and software form a tightly coupled cohesion that operates in unison to performed programmed tasks. Without software, the fastest, biggest or the most powerful computer will also be inoperative and idle in the corner.The software tools utilized in the development environment are listed as below:

- Java 2 SDK Software Development environment version 1.4.2_06
- JCreator LE version 2.50

6.2 Development of the system

To be able to use Java as the programming language to code or to develop a program or a system, one needs to gain an understanding of the concepts of the objects oriented. Understanding of what an object is, what a class is, how objects and classes are related, how objects communicate by using messages is much needed.

In definition an object is a software bundle related variables and methods. "Objects" is a key to object oriented technology. One can look around now and see many examples of real-world objects: a car, a bicycle, a desk a, television set

and a computer. These real world objects share two characteristics: they all have a state and behavior. For example cars have a state (brand, color, size) and behavior (accelerating, braking).

Software objects are modeled after real world objects in that they too have state and behavior. A software objects maintain its state in one or more variables. A variable is an item of data named by an identifier. A software objects implements its behavior with methods. A method is a function associated with an object.

Methods surround and hide the objects nucleus from other objects in the program. Packaging an object's variables within the protective custody of its method is called encapsulation. Encapsulating related variables and methods into a neat software bundle is a simple yet powerful idea that provides two primary benefits to software developers:

- Modularity: the source code for an object can be written and maintained independently of the source code for other objects. Also, an object can be easily passed around in the system. One can give a bicycle to someone else and it still work.
- Information hiding: an object has a public interface that other objects can use to communicate with it. The object can maintain private information and methods that can be changed at any time without affecting the other objects that depend on it. One does not need to understand the gear mechanism in his bike to use it.

A single object is generally not very useful. Instead, an object usually appears as a component of a larger program or an application that contain many other objects. Software objects interact and communicate with each other by sending messages to each other.

A class is a blueprint or prototype that defines the variables and the methods common to all objects of a certain kind. For example, a bicycle is just one of the many bicycles in the world. In object oriented software it's also possible to have many objects of the same kind that share characteristics: rectangles, employee records, video clips and so on. A software blueprint for objects is called a class.

6.2.1 System coding

After researches and studies have been done, a decision was made to codethe simulation system using the Java programming language, and to be able to run the simulation as a standalone windows application and as an applet which can be executed using the Xinox Software's JCreator Light Edition (LE) v2.50 integrated development environment.

The switch simulation system utilizes five java platform packages. The four packages are **java.awt , java.swing, java.net** and **java.util.**.

Package **java.awt** and **java.swing.** contains all of the classes for creating user interfaces and for painting graphics and images. A user interface object such as a button pr a scrollbar is called, in AWT terminology, a component. The Component class is the root of all AWT components. Some components fire

events when a user interacts with the components. The AWT Event class and its subclasses are used to represent the events that AWT components can fire. The **java.swing**. is latter version.

A container is a component that can contain components and other containers. A container can also have a layout manager that controls the visual placement of components in the container. The AWT package contains several layout manager classes and an interface for building your own layout manager.

Package **java.util** contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array). Package **java.net** provides the classes for implementing networking applications. Using the socket classes, you can communicate with any server on the Internet or implement your own Internet server. A number of classes are provided to make it convenient to use Universal Resource Locators (URLs) to retrieve data on the Internet.

6.3 Program Coding Approach

Factors to be taken into account when doing system coding:

6.3.1 Simplicity and Clarity

More than a few misguided programmers believe that the more complex and convoluted their code, the more sophisticated their skills. A good program is generally quite simple. The underlying meaning of the

procedure represented in programming language source code should be easy to understand and clear for the programmer

6.3.2 Use meaningful variable names

In general, variables and data structures should be named in a manner that enables the programmer to infer their meaning within the context of the procedure at hand and their correlation with some real-world object.

6.3.3 Establish effective commenting conventions

- Start with an effective prologue
- Describe blocks of code, rather than commenting every line
- Use blank lines and indenting so that comments can be readily distinguished from code.

6.3.4 Module

Separate function structure so it can function independantly and easy for modifications

6.4 System Module

The switch simulation program is controlled by a main class of `UnbufferedCrossbarSwitch.java`. In this program, Packet generator used to generated the total of frames that the user had key in.

Then, each of the frames will map to MAC address statically and randomly. This is created by using `Hashtable`. Each port has an input buffer and an output buffer.

```
NetBuffer inputBuffer;
NetBuffer outputBuffer;
```

The switch port is set to 8 and each buffer can limit only 8 frames, and each frame sent is 64 bytes. Here is `NetFrame.java` class:

```
Class NetFrame{
int Preamble;
int SFD;
String DestMAC;
String SourceMAC;
int Length;
byte [] Data;
int FSC;
}
```

It describes an Ethernet frame. In this program, linked list is used to queue the frames into buffer.

```
LinkedList frameList;
```

The clock or thread keep track of the current simulation time in seconds, and the model increments the clock once every second. The schedule is responsibilities for scheduling the arrival and departure of frame on each port.

6.5 Coding style

6.5.1 Formatting and indenting codes

Formatting and indenting codes are constantly associated to good coding practice.A code that is written without proper formatting or indenting will function or what as well as a formatted code. However, this can make exceptionally difficult to see where an error is coming from. Indentation principally makes the structure of the code stand out and easier to be read. This eventually will help in detecting and removing the common programming errors. Creator provides user a good formatting and indenting facility where user is associated to format and indent codes automatically in the environment while coding a Java source.

6.5.2 Commenting codes

Comments are part of this program code. It is a good and recommended practice.Commenting will help the reader of the code to understand what and why the coding was written .In addition, this also makes it easier for others especially collaborating programmers to understand the coding.

6.6 Simulation result

From the switch simulation that I have been developed, I key in a numbers of frames to analyze frames drop percentage .Below is the frame drop ratio that I get. Figure 6.1 is the graph based on the Table 6.1.

Total Frame	Frame Drop Ratio, %
0	0
100	3
200	14
300	21.67
400	27.75
500	31.4
600	33.5
700	35.57
800	36.22
900	39.11
1000	42.5

Table 6.1 : Frame Drop Ratio

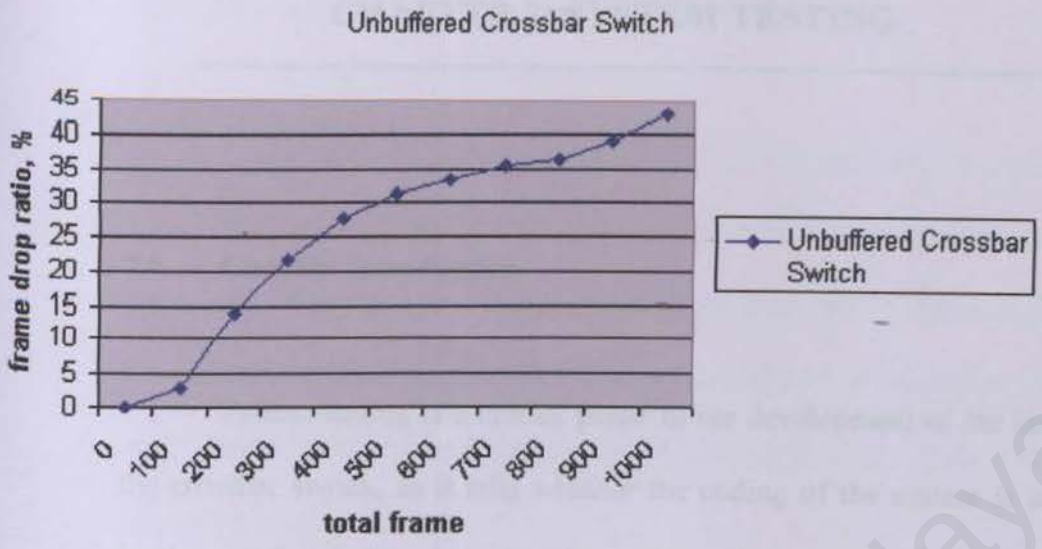

Figure 6.1 : Graph Frame Drop Ratio

CHAPTER 7: SYSTEM TESTING

7.0 Chapter Introduction

System testing is a crucial phase in the development of the simulation of the crossbar switch, as it tells whether the coding of the system is successfully implemented, whether the executing system visualizes the simulator accurately and whether the code needed to be modified, enhanced, deleted, added or debugged. Testing is done throughout the system development and not just at the end.

7.1 Compiling and executing

Once the coding of the system is completed, and all the classes designed are fully coded, the java source needs to be compiled to see whether there any bugs or errors in the coding.

If the java application can be successfully compiled without any error, then the testing phase can proceed to executing the application. Otherwise, if the java application is compiled with errors, the testing phase needs to jump to the debugging phase, before it is recompiled again.

7.2 Debugging

Once the application is compiled with errors, the error message need to be scrutinized to identify where the errors have occurred on the source code. The error might be caused by syntax mistakes, such as left out of semi-colon, curly braces and any other symbols. Some errors might also be caused by logical errors such as errors in dereferencing, errors in calling methods, or errors in passing arguments.

The process of debugging is to check on those mistakes and correct them. It is a process eliminating errors or bugs from the source code in order for the system to compile successfully.

7.3 Accuracy of execution

After the source code has all the errors eliminated and compiled successfully it will then be executed or in other words run. The target of the execution is for the users to use the system or interact with the system through the system interface.

In the context of switch simulation, the compiled source is executed checked and verified if the simulation acts according to the user's input or not. There may be logical errors where source code compiled successfully but simulation does not perform according to the users input. If the system does not

act properly according to the user's response, the testing phase will go back to debugging-compiling-executing process until the system is able to work accurately according to user's input.

7.4 Unit testing

Unit testing verifies that the component functions properly with the types of input expected from studying the component's design. The first step is to examine the program code by reading through it, trying to spot data and syntax faults. This is followed by comparing the code with specifications and with the design to make sure that all the relevant cases have been considered. Next view the result and eliminate remaining syntax faults if necessary. Finally test cases are developed to show that the input is properly converted to the desired output. Unit testing tries to look for all the possible errors that will occur in a program. A complete test process should test all of the following categories of test data.

- Normal data - to test a given correct data will produce the expected results.
- Erroneous data - for a given erroneous data, like invalid date format, does the system detect or not.
- Boundaries value analysis - data that are out of the range specified will be used to test the system because errors may occur at the extreme point.

7.5 Module testing

When individual components are working properly and meets the objectives,these components are combined into a module. A module is a collection of dependent components. Module testing enables each module to be tested independently. This testing will ensure that the module calling sequences in this project is systematic

7.6 Integration testing

When the individual components are working correctly and meet the objectives,these components are integrated into a working system. In other words, integration testing is the process of verifying that the system components work together as described in the system and program design specification.

Integration testing is used on switch simulation for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit-tested modules and build a program structure that has been dictated by design. This testing will ensure that the interfaces in the simulation are systematized and linked to the correct document of the system.

7.7 System testing

The last testing procedure done is the system testing. Testing the system is very different from unit testing and integration testing. The objective of unit testing and integration testing is to ensure that the code has implemented the design properly. In other words, the code is written to do what the design specifications intended. In the system testing, a very different objective is to be achieved, that is to ensure that the system fulfills user requirements or my supervisor's requirements.

Testing performed:

- Function testing

Function testing is based on the system's functional requirements. The testing is carried out for the system's every module. Each module is tested individually to determine whether the system performs as required.

- Performance testing

Performance testing addresses the non-functional requirements of the application. The types of performance tests carried out for this application are:

- Volume tests

The data field and address field are checked to see if they can accommodate all expected data.

- Timing tests

System performance is timed to ensure that it meets user's requirements.

7.8 Chapter Summary

During the testing phase, several testing strategies were being used to ensure the system is integrated and developed successfully. Approaches were employed to recover faults in the system. Unit, module, integration and system testing has been carried out for this system. The objective of a system will only be achieved after all the thorough testing done by the different user with different aspects.

CHAPTER 8: SYSTEM EVALUATION AND CONCLUSION

8.0 Chapter Introduction

This chapter explains all the evaluation procedures taken to identify simulation's strengths and limitations. Throughout the switch simulation development process, a lot of problems were encountered and solved eventually. This chapter also discusses future enhancement to the system.

8.1 System Evaluation

This section discusses the expectations achieved from the initial goals set at the beginning of this project. The systems strengths and limitations are also listed. The main problems encountered while developing this simulation is also explained along with all the knowledge gained from this project.

8.1.1 Expectations achieved

In general, the initial expectations set by the project objectives had been achieved. The following are the achieved goals:

i. Show the simulation of crossbar switch

ii. All the basic function performed by crossbar switch like transmitted, received and processing frame from input port to output port.

iii. Scheduling algorithm based on FIFO queue.

8.1.2 System limitations

Although the system performs the basic function, there are limitations in the system. The limitations are as follows:

i. The system only transmits the packet to a random port.

ii. The performances of simulation are based on FIFO input queue only.

8.1.3 Problems encountered

During the system requirement and analysis phase, a lot of study and research has been carried out. The problem faced during the analysis and requirement phase were not as crucial as during the implementation phase. A lot of modification and work cannot be carried out due to lack of knowledge in certain areas and time constraint. Below are some of the problems encountered:-

i) Problem with the programming language.

I had no prior knowledge in the java programming language before the implementation of the system started. I needed to get a deep detailed idea of object oriented programming in java before coming out with the architectural basis design of the crossbar switch simulation. Objects of the system were to be identified and verified of their applicability and flexibility to be coded. Once the coding of the system started, I encountered a lot of syntax and logical errors, which consequently, forced me to spend a huge amount of time in debugging and even seeking help from experts in forums.

There are a lot of built in classes in Java. It was a bit confusing to use which class and I took some time to learn about all the classes.

ii) Insufficient detailed references

There were little references and information for my project. I had to design and build my switch architecture and simulation based on my own understanding. Therefore the end result was not as good as expected.

8.1.4 Knowledge gained

During the whole project I learned a lot of things. Firstly of course learned the more about Java simulation language itself. Before the project I did not know a thing about simulation. This project made me learn a new

programming language which will be an advantage for me. I had a better understanding of how the cut-through switch works in real networks.

Developing a switch model has taught me in so many management skills.Time management was among the most important thing and while developing this project, time was really a big matter as other works also need to be accomplished.

Developing this project has actually helped me to work more independently. This is a good practice for me before I really face the world outside which have more challenge and obstacles.

Developing this project has also taught me to be patient. Patient is very important because it requires time to learn, time to develop and time for success. Writing this report enhanced my report writing skill which would be good and essential for future purposes.

8.2 Project enhancement

Due to time limitation, not all of the target objectives and ideas could be incorporated in this project. Future enhancement is essential to make the system more up-to-date, interesting and dynamic. These factors are crucial to create an interest on the user to use the system.

Ideas for future enhancement:

- Designed the crossbar using other algorithm such as Round Robin
- Compared the performance of switch using output buffer queue and virtual output queue to know how many frame is dropped.
- Do animation of transmission path of frames from input port to output port.

CONCLUSION

After conducting analysis and testing, it is concluded that the project has achieved its main objective, design and build switch simulation even though some of the requirement and targeted objective are not fulfilled.

There are more researches to be done in developing the system. With the first step taken, enhancement can still be made in the future to this version of system. The system could be made more up to date, dynamic and detail.

As the project has to be done in a short period of time and a lot of technical issue arises and need to resolve, a few problems has been encountered. Solution has been sought during testing. Encountering with problem has been proven to be a valuable learning experience.

I learnt that a good knowledge of software development life cycle could accommodate a developer to manage their project smoothly. All five phases, requirement analysis, system design phase, system coding, testing and maintenance need to be followed accordingly in order to build a good system. To build a good systems also require time, effort and patience. One the most essential knowledge gained from this project is the technique on problem solving. I was

also able to practice my skill in programming Java language and gain a sufficient knowledge on how to build a simulation.

This project has helped me a lot in recognizing my poor skill in time management, project management and communication. These experiences and knowledge gained would certainly help me to manage and organize any future project and will make me become a better programmer.

References

-G.B. Bleazard, Why Packet Switching? , The National Computing Centre .

-Stephan R. Schach (2005), Object Oriented and Classical Software Engineering , 6th Edition, McGraw Hill.

-Behrouz A. Forounzan (2001), Data Communication and Networking, 2nd Edition, McGraw Hill.

-(URL-http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs010.htm)

-(URL-http://www.itmweb.com/essay522.htm)

-(URL-http://w2.rad.com/networks/1994/ packet-switchtheory/sharmem.html)

-(URL-http://netlab1.bu.edu/~staro/546projects/switchedethernet/proj/overview.htm)

-(URL-http://www.inetdaemon.com/tutorials/lan/switching/architectures/crossbar.html)

-(URL-http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/switch.html)

-(URL-http://www.howstuffworks.com)

-(URL-http://archvlsi.ics.forth.gr/bufxbar/)

-(URL- http://en.wikipedia.org/wiki/Crossbar)

The references and links to website have been checked correct as of 20th June 2006.

Appendix A : User Manual

Table of contents ..... a

List of Figures ..... b

Section A : Introduction c

Section B : Hardware specification c

Section C : Software specification c

Section D : I) Installing J2SDK 1.4.2_06 ..... c-g

II) Installing JCreator LE version 2.5 ..... h-l

Section E : Running the program ..... m-q

List of Figures

No.	Number Of Figure	Page
1.	Figure A1, Figure A2	d
2.	Figure A3	e
3.	Figure A4, Figure A5	f
4.	Figure A6, Figure A7	g
5.	Figure A8, Figure A9	h
6.	Figure A10, Figure A11	i
7.	Figure A12	j
8.	Figure A13, Figure A14	k
9.	Figure A15, Figure A16	l
10.	Figure A17	m
11.	Figure A18	n
11.	Figure A19	o
13.	Figure A20	p
14.	Figure A21	q

A. Introduction

Switch cut-through system is a simulation of how the switch works in a network.

B. Hardware specifications

Below are the minimum hardware requirements to run the switch simulation:

- Windows XP Professional.
- 654 Mhz.
- 64 MB RAM.

C. Software specifications

Below are the softwares required to run the Flash memory simulation:

- Java 2 SDK Software Development environment version 1.4.2_06 (J2SDK 1.4.2_06).
- JCreator LE version 2.50.

D. I) Installing J2SDK 1.4.2_06

- Download the J2SDK from the http://www.java.sun.com website. It is a free software so it does not need to be licensed. Save it in your hardrive.
- Double click the on the **Setup** launcher. The **InstallShield Wizard** will appear as in Figure A1. Please wait for the **license** screen to appear as in Figure A2.

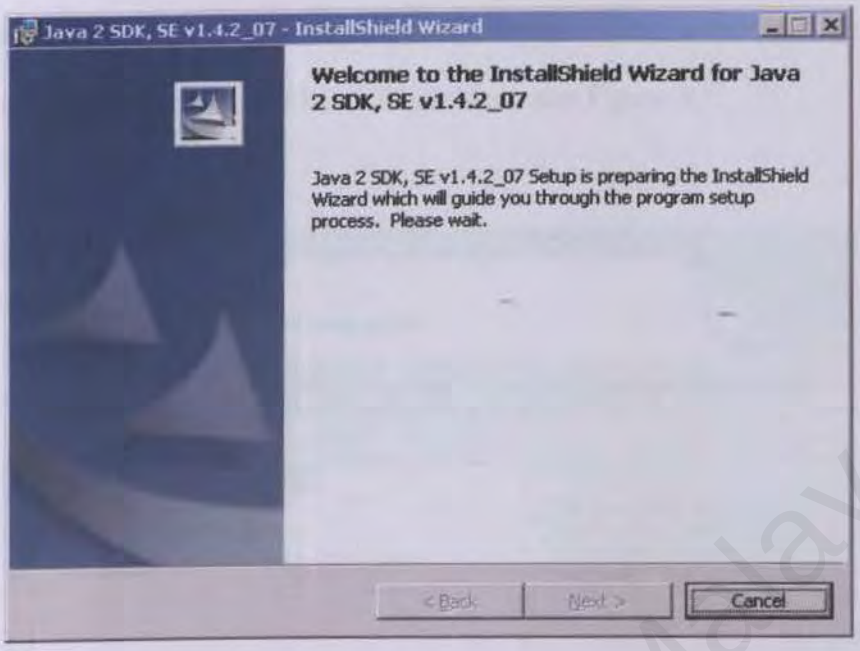

Figure A1

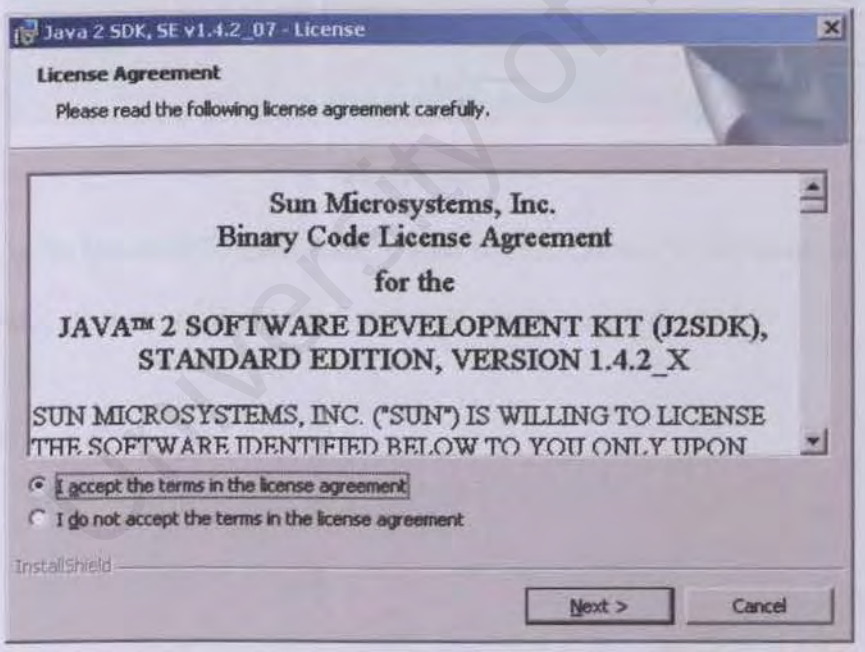

Figure A2

- Click the **I accept the terms in the license agreement** radio button and click **Next** button.

- In the **Custom Setup** wizard screen, click **Change** button to choose which directory to install your J2SDK to. Click **Next.** See Figure A3.

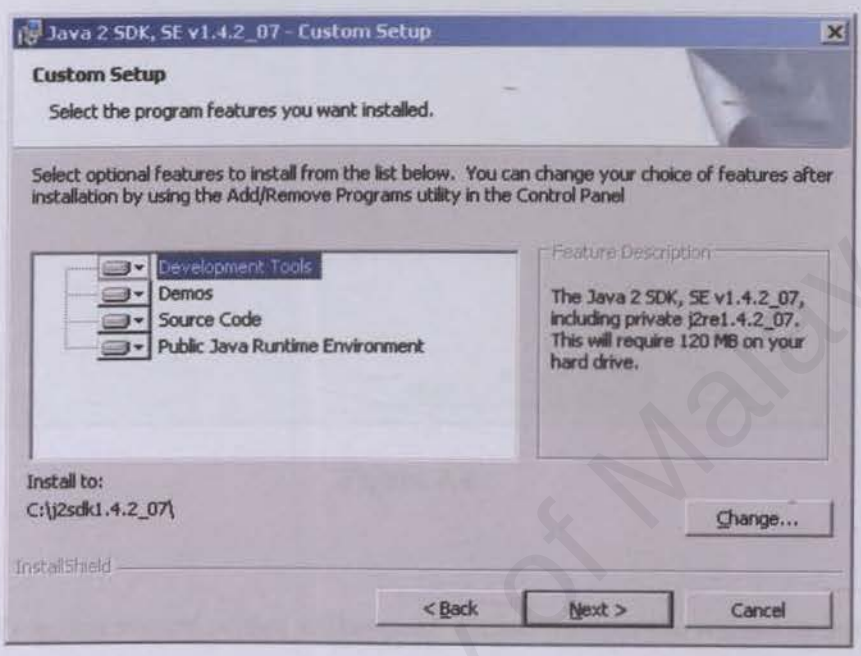

Figure A3

- Next is the **Browser Registration** wizard screen. Choose which browser you want and click the **Install** button to install. See Figure A4.

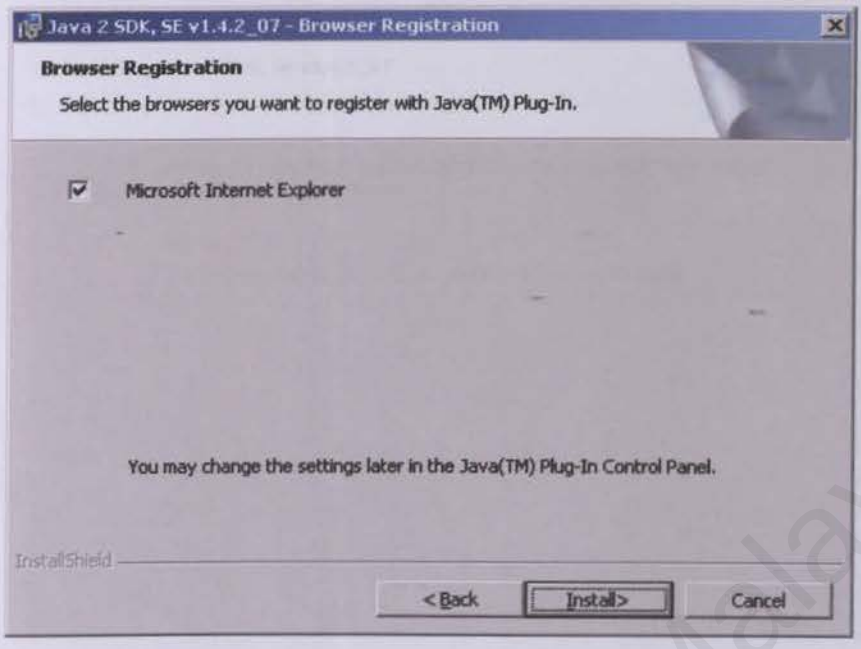

Figure A4

- The **Progress** wizard screen will appear. Please wait for the wizard to install Java in your computer. See Figure A5 and Figure A6.

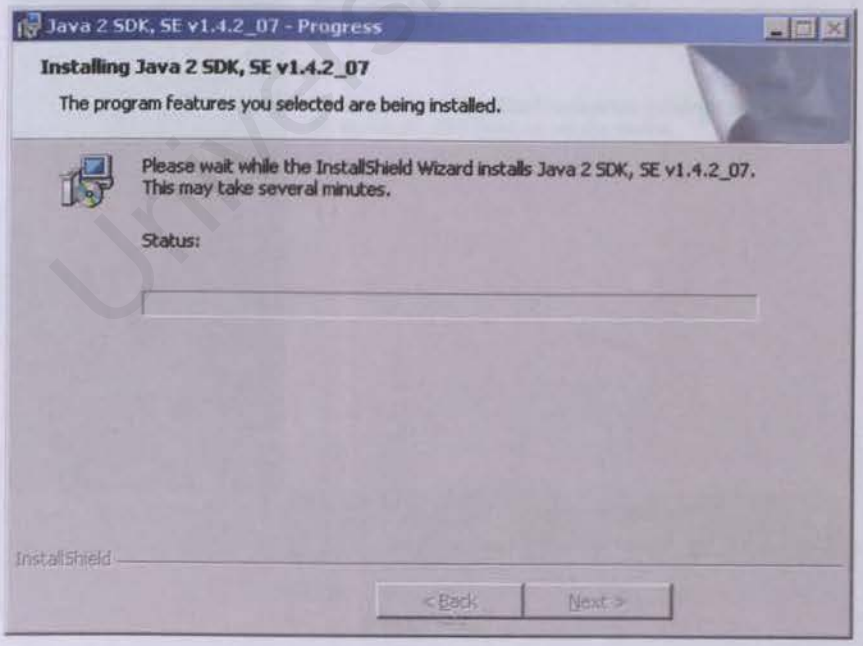

Figure A5

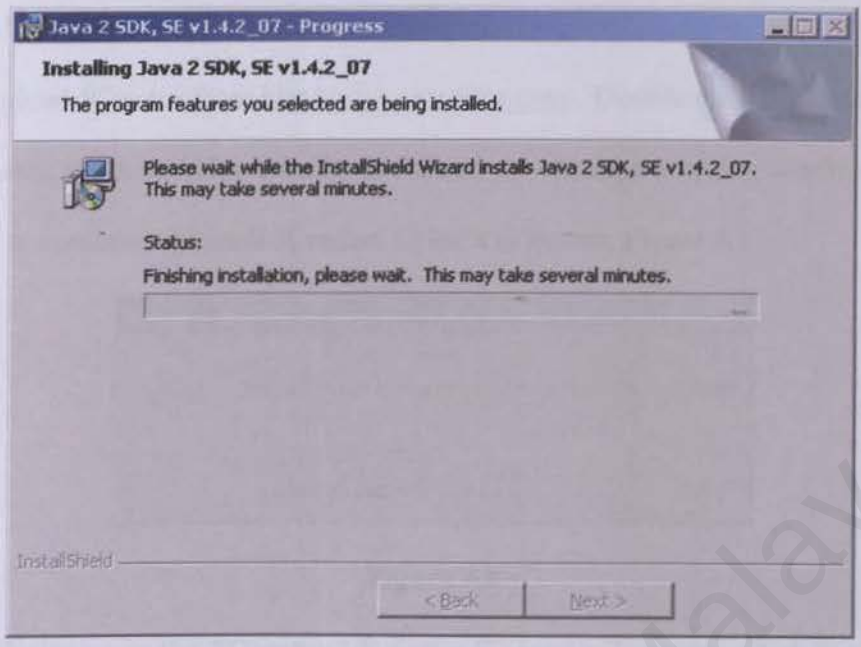

Figure A6

- The **Complete** wizard screen will appear. Click Finish button to finish the installation, **Figure A7.**

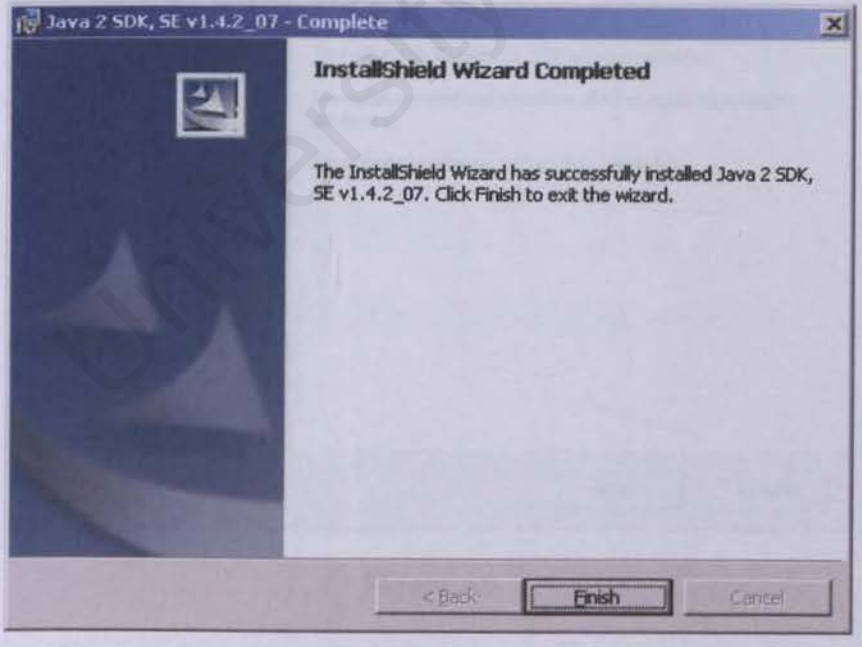

Figure A7

II) Installing JCreator LE version2.5

- Download JCreator from http://www.jcreator.com . Double click the **Setup** icon to launch the installation wizard. A dialog box will appear asking whether you want to continue to install JCreator. Click **Yes** button, Figure A8.

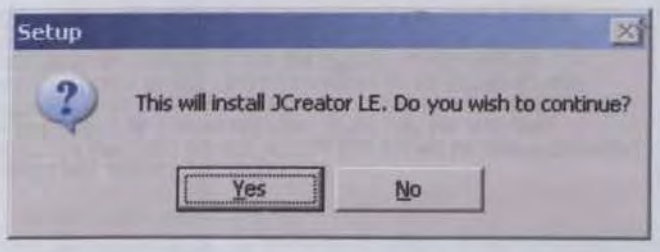

Figure A8

- The **Welcome to the JCreator LE Setup Wizard** will appear. Click **Next** button. See Figure A9.

Figure A9

- Click the **I accept the agreement** in the **License Agreement** screen and click the **Next** button. See Figure A10.

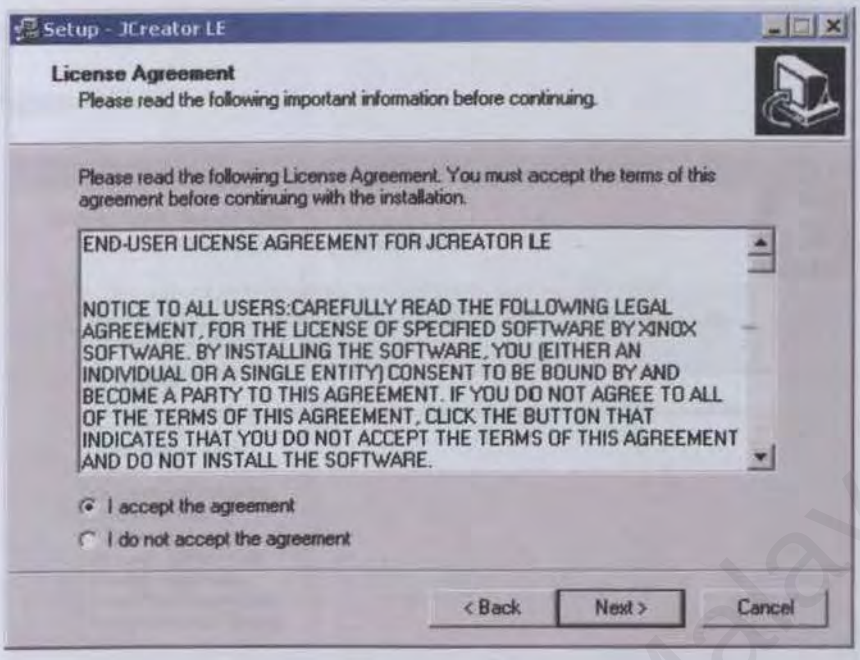

Figure A10

- In the next screen choose the destination directory where you want the JCreator to be installed. See Figure A11.

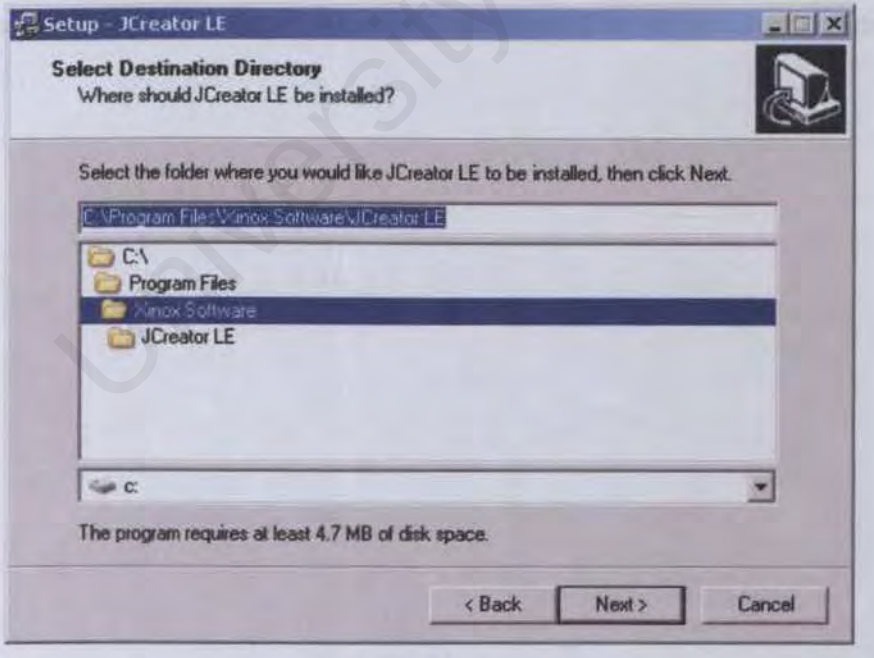

Figure A11

- In the next screen choose where should the Setup place the program's shortcut.

  See Figure A12.

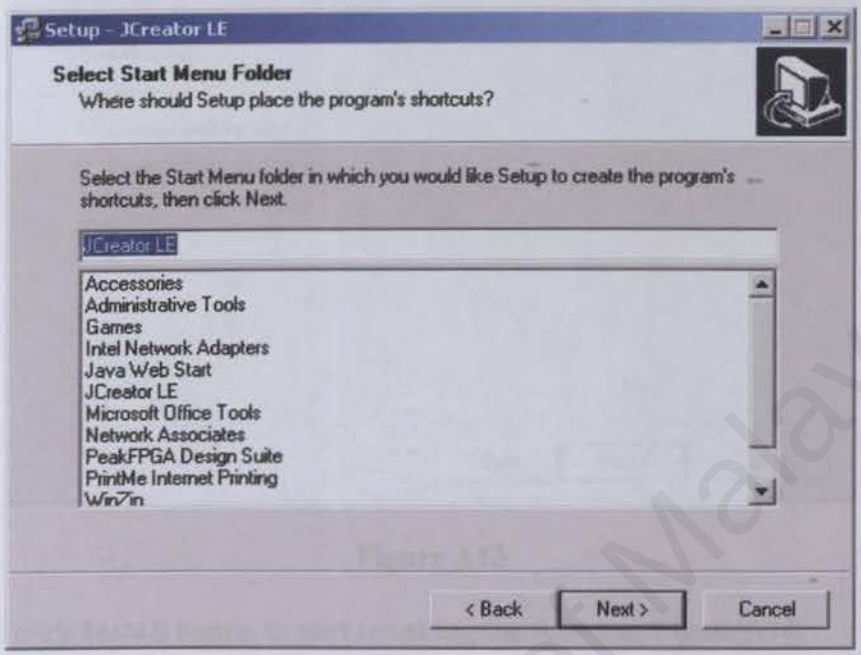

Figure A12

- Choose which additional tasks should be performed in the next screen. See Figure A13.

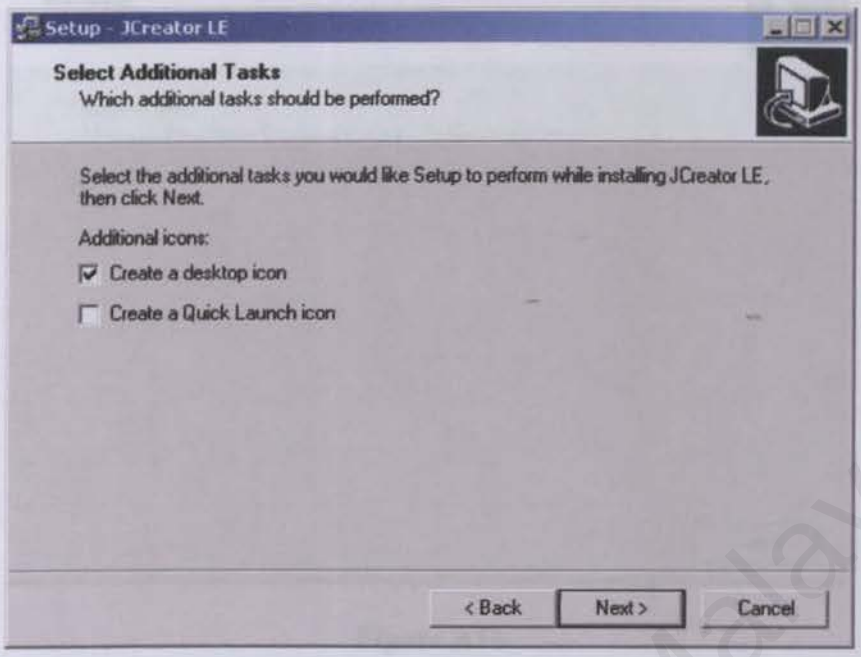

Figure A13

- Next, click **Install** button to start installing the JCreator, Figure A14.

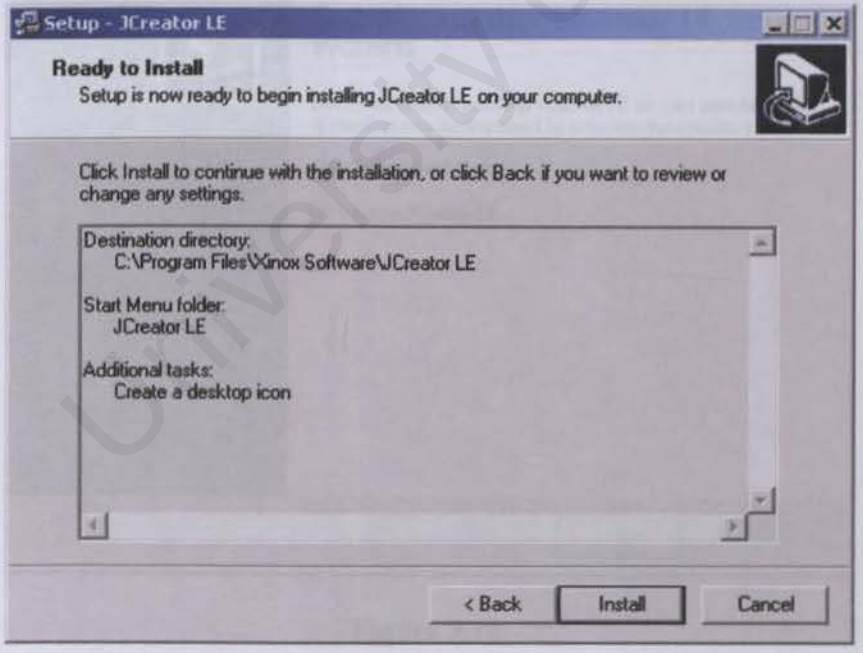

Figure A14

- Please wait while the Setup is installing the JCreator, Figure A15.

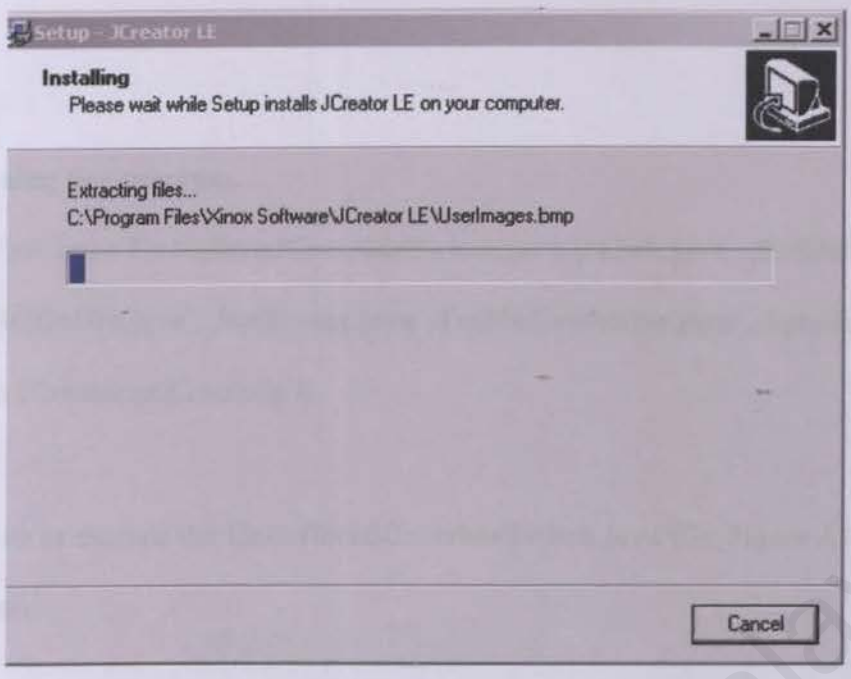

Figure A15

- Lastly click **Finish** button to finish the installation, Figure A16.

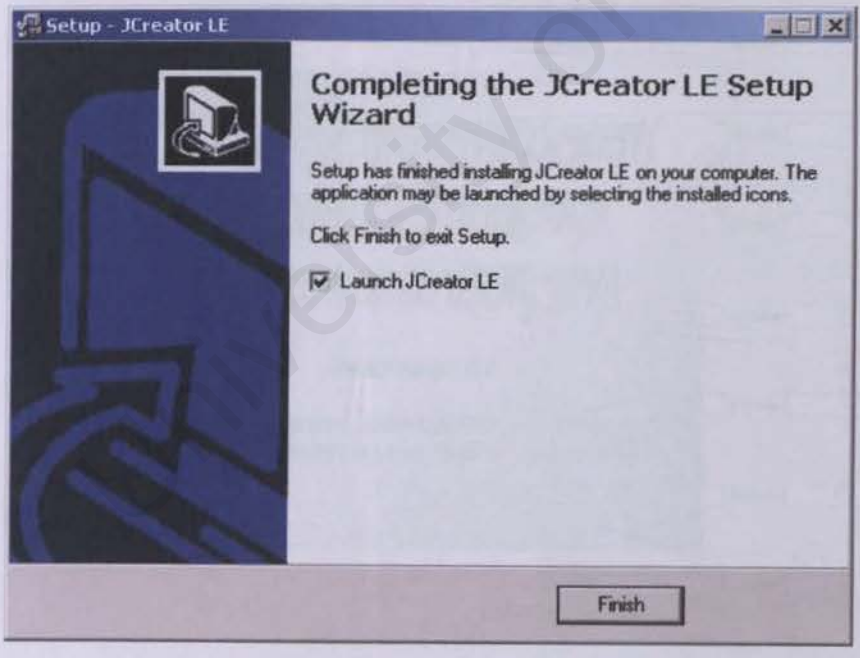

Figure A16

E. Running the program.

1) First open **UnbufferedCrossbarSwitch.java , Link.java ,NetBuffer.java , NetBuffer.java , NetFrame.java ,TrafficGenerator.java , Splash.java ,**file in JCreator and compile it.

2) Run or execute the **UnbufferedCrossbarSwitch.java** file. Figure A17 will appear.

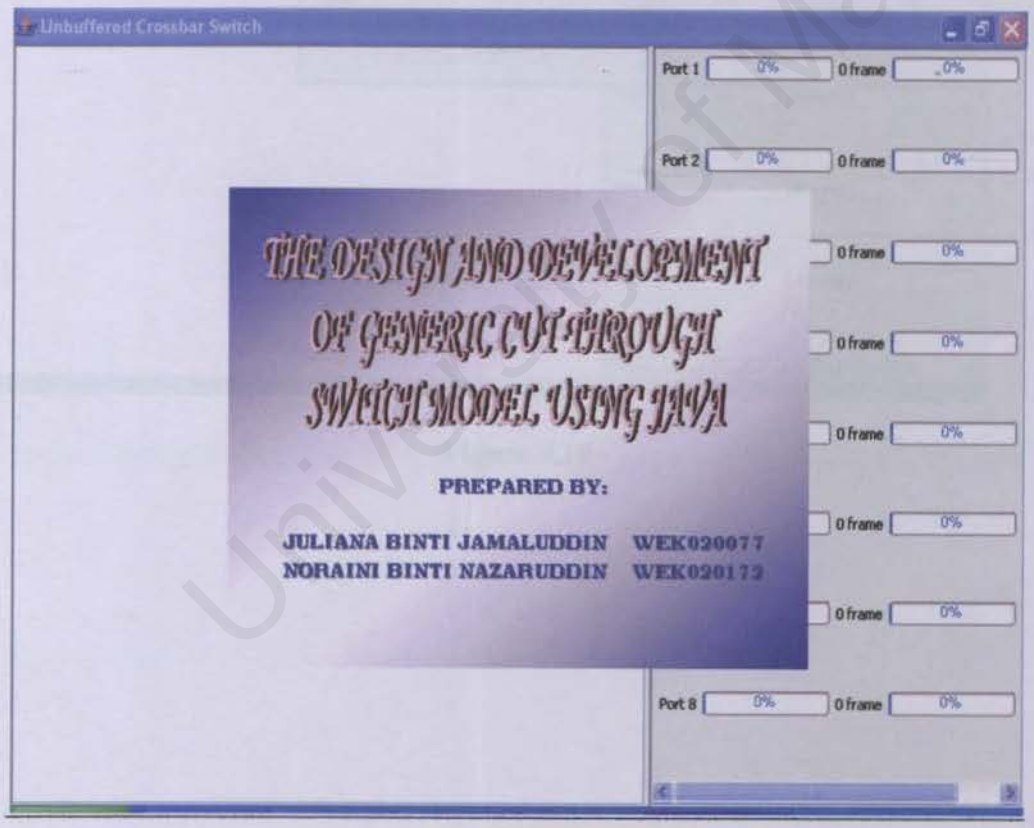

Figure A17

3) Next Figure A18 will appear. The dialog asks user to key in number of frames.

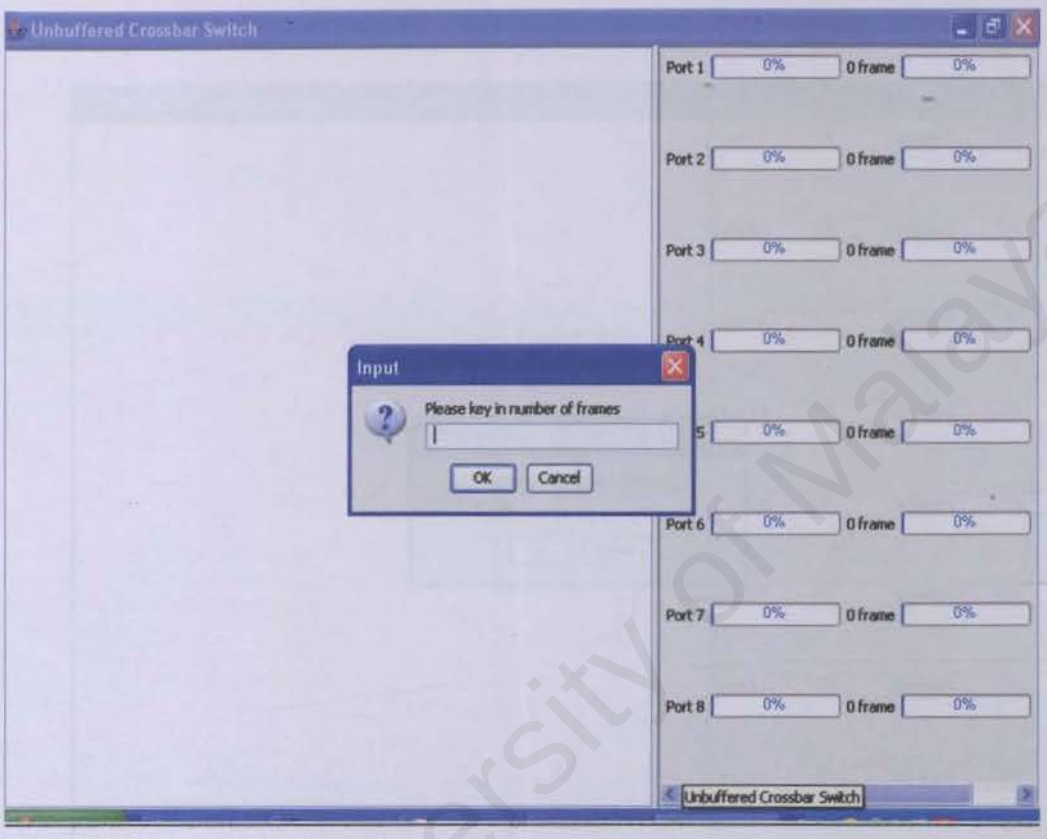

Figure A18

4) For example, the user key in 100 frames. Then click OK or ENTER button on the keyboard.

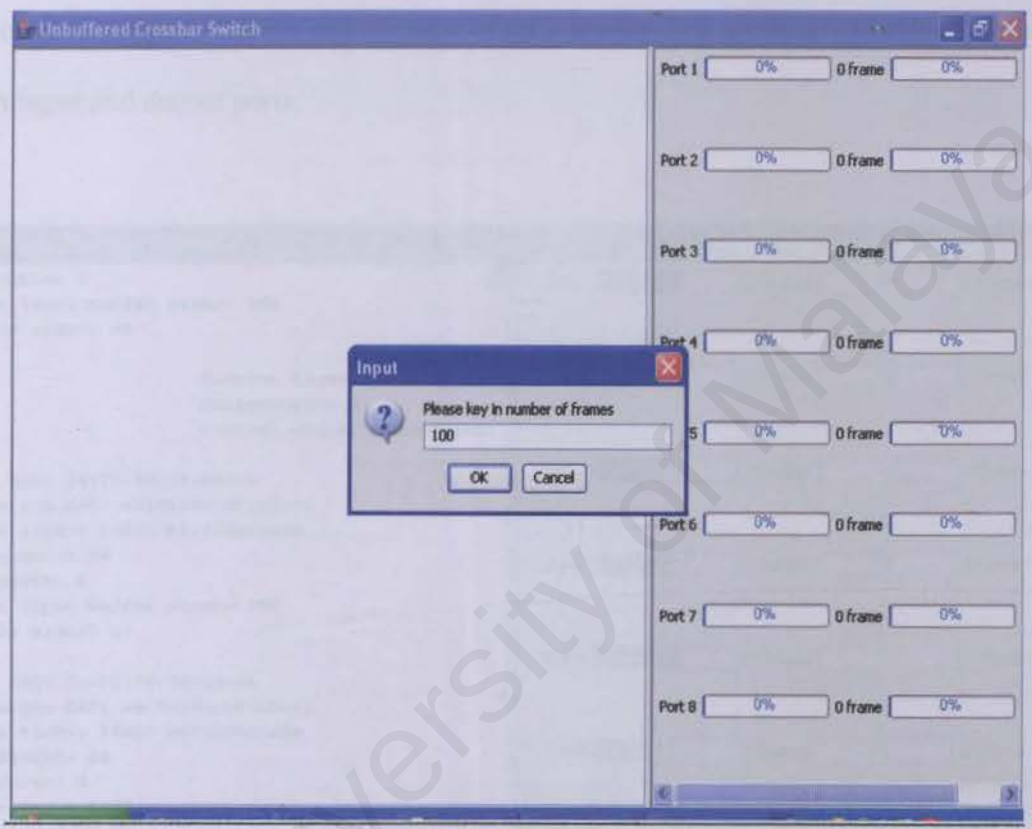

Figure A19

5) From the figure below (**Figure A20),** we can see the output of the simulation of the cut through switch. The information of frames such as source and destination MAC address, time arrival, time delivered in output port are listed. On the right, is the progress bar that show the progress of the packet while transmitting from input port to output port. Besides the each of the port, there is information of total frames that are received and delivered in each input and output ports.

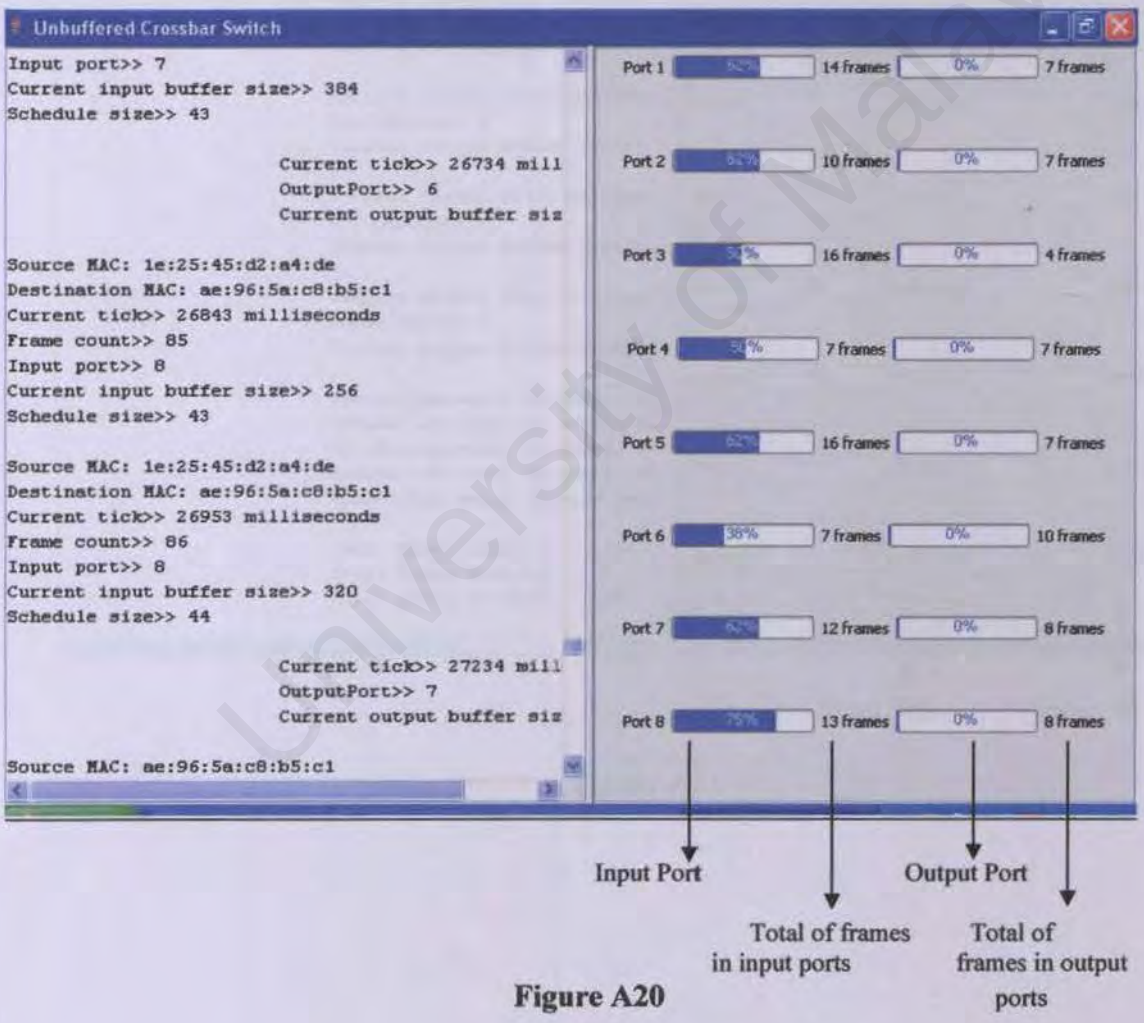

Figure A20

6) End of the simulation, the total of frames sent, dropped and received is listed. The total of frames in input ports and output is same, 95. The balance 5 is dropped before it enters input port because the buffer in input ports was full.

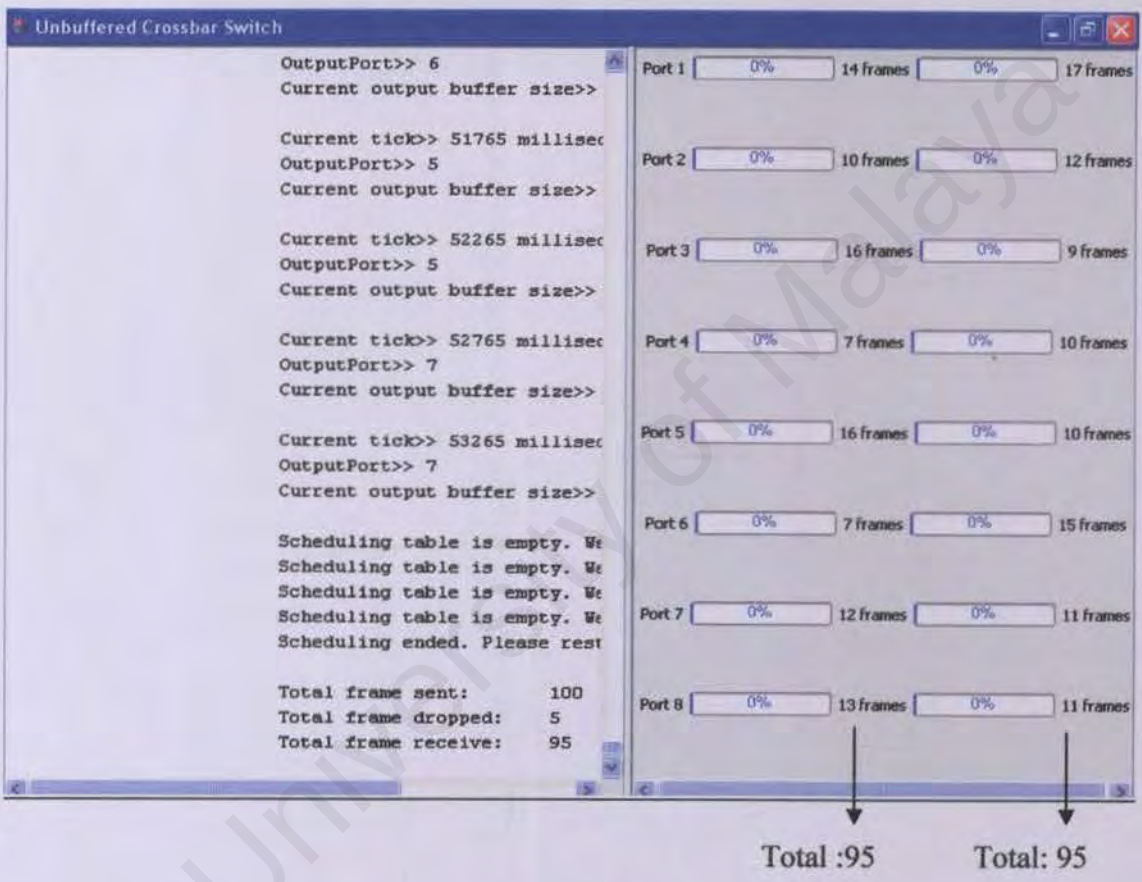

Figure A21