UNIVERSITY OF MALAYA
Faculty of
Computer Science
And
Information Technology

ZARINA MOHAMED ALIAS
WEK 010400
PACKET SNIFFER AND ANALYZER

Supervisor: Encik Badrul Anuar Juma'at
Moderator: Mr. Ling Teck Chaw

ABSTRACT

With the proliferation of the Intranet, there is a vast amount of digital information stored in the file server and shared within an organization. It is extremely difficult to monitor all access and action on a network. With numerous incidents these days, the fact that a company cannot monitor all the actions on their network can contribute to a serious problem.

Packet sniffer and analyzer, also known as network protocol analyzer is a stand alone system used to monitor network. It has the capability of filtering network traffic based on Internet Protocol (IP) address, port number and protocols. It behaves as a useful tool for network administrators, network engineer and network.

Hackers use packet sniffer to obtain usernames, passwords, credit card numbers, personal information and other information that could be damaging to a person or a company.

When they obtain this information, hackers will use the information to do damage to others such as send a virus using the usernames and passwords captured.

However, in this project the system is built for the purpose of monitoring network traffic. Sniffalyzer is targeted to be implemented in a Local Area Network (LAN) environment. Its main function is to monitor, identify and control network traffic in real time. It helps to improve network manageability and overcome network problems.

Sniffalyzer can be effective as security measure for the network and shared resources. All packets passing throughout the network can be captured and analyzed and these help to secure the network.

ACKNOWLEDGEMENT

Uttermost greatful goes to Allah the Almighty for all the confidence and patience in the completion of this thesis. I wish to record my appreciation to everyone who has been so helpful and supportive in this project work and brought it to success.

I would like to express deep gratitude to my supervisor, Encik Nor Badrul Annuar Juma'at for the continuous help and useful advice he has given me during this project and also to my examiner Mr. Ling Teck Chaw.

I would also like to take this opportunity to express my thanks to all family members of FSKTM especially the family of Computer Systems and Technology Department for their constructive criticism and support to face the difficulties and challenging time.

Finally, I would like to extend my sincere thanks and love to my mother, Zubaidah Abdul Rahman who has been very, very patience and given me invaluable support throughout my university life. To my dear sister Azlyna, who is going through great difficulties and for being so brave, you are my inspiration and you have given me the courage to go through mine.

My gratefulness also goes to the rest of my family and friends who had helped me in critical time to complete this project.

Thank you with all my heart.

TABLE OF CONTENTS

CHAPTER 5 – SYSTEM DESIGN

CHAPTER 6 – SYSTEM DEVELOPMENT AND IMPLEMENTATION

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

unfamiliar or unwanted traffic wandering around the network and track down the problem. Example, when a network normally does not broadcast AppleTraffic, but the analyzer clearly show that AppleTalk packets traversing the network. The network administrator could investigate into the problem and find a solution to it.

The strength of Sniffalyzer is that it could capture traffic in real time environment on non-switched, switched network and dial-up connection. It has the capabilities to decode packets into a readable form and enable examination on them for protocol and application problems. Packet information such as source and destination address, packet length, types of protocol is displayed for analyzing purposes.

Sniffalyzer filters are used to accept and deny packets. Any unwanted packets can be discarded and this will help in reducing the time and amount of packets captured. Packet filtering in Sniffalyzer will be based on protocols and/or IP address. The system module will allow the user to key in the filter.

A graph is generated to display amount of packets being captured based on the protocols. This function eases the monitoring activities for the network administrator.

In addition, network administrator can analyze the network traffic based on the information displayed in text or graph. This helps them to increase the network performance, perform troubleshooting and these will lead to

reducing the number of malicious threat intended on the network and organization.

1.3 Project Objective

The objective of this project is listed below:

1. The system should be able to monitor network activities on LAN environment.
2. The system should be able to capture in real-time all traffic transport over switched network, non-switched network and dial-up connection network
3. The system should be able to filter traffic based on protocols and/or Internet Protocol (IP) addresses.
4. The system should be able to display filter traffic based on protocols and/or Internet Protocol (IP) addresses.
5. The system should be able to display and decode accurate packet information for analyzing purposes.
6. The system should be able to provide summary report for analyzing purposes.

1.4 Project Scope

The project scope outlines the limitations of the system. The scope of the project is listed below:

1. The system is designed to be used by network administrator and network engineer for the purpose of monitoring and analyzing network traffic.

2. The system is designed to increased network performances and ease network troubleshooting.

3. The system is designed to safeguard the network from any malicious threat and activities.

4. The packets sniffed and captured cannot be modified or altered in any way.

5. The system is not designed to capture and read confidential network information.

1.4.1 Target User

- **Network application developers** - To debug network application and examine network protocols.
- **Network administrators/engineer/analysis** - To diagnose network problems and help to ease troubleshooting.
- **IT professionals** - To supervise contents inside the internal network.
- **Consultants** - To help solve problems for customers.
- **Parents** - To find out what their children are doing on the internet.

1.5 Project Schedule

In developing and completing the project, it needs proper planning 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 figure 1.1.

ID	Task Name	Start	Finish	Duration
1	Requirement and Analysis Phase	6/23/2004	10/20/2004	17.2w
2	System Design Phase	9/1/2004	11/19/2004	11.6w
3	System Coding	12/13/2004	3/4/2005	12w
4	Testing Phase	12/13/2004	3/4/2005	12w
5	Maintenance	2/21/2005	2/21/2006	52.4w

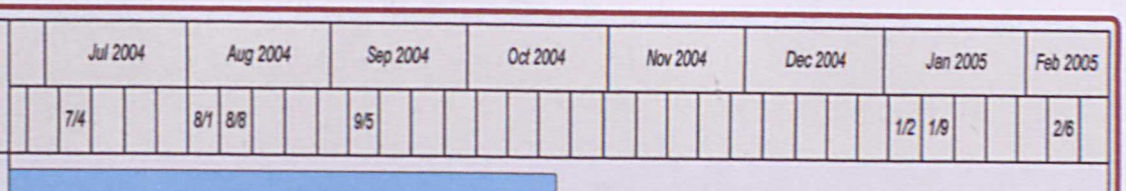


Figure 1.1 – Project Gantt Chart

CHAPTER 2
LITERATURE REVIEW

CHAPTER 2 - LITERATURE REVIEW

Literature review represents a demonstration and mastery of the concepts relevant to the research problem. It provides description of background information that is focused on the problem and represents a search of the sources and an exemplary reference format.

The objectives of this literature review:

1. To document facts and specification related to the project for current and future references.
2. To analyze the findings in order to provide better understanding on the system to be developed.
3. The assist in determining the requirements for this project.

2.1 Network

A computer network is a group of computers which are connected together and communicate with one another. It is a tool for communication that allows users to store and retrieve information, share printers and exchange information.[1] This is an example of computer networking, linking millions of computers around the world. There are three main types of computer network as described below:

2.1.1 Local Area Network (LAN)

Most LANs are confined to a single building or group of buildings. However, one LAN can be connected to other LAN over any distance. A LAN connected in this way is called a wide-area network (WAN).

LAN connects workstations and personal computers. Each node in a LAN is able to access data and devices anywhere on the LAN. This means that many users can share expensive devices, such as printers

as well as data. Users can also use the LAN to communicate with each other by sending e-mail or engaging in chat sessions.

There are many different types of LANs. Ethernet being the most common for personal computers.

2.1.2 Wide Area Network

WAN is computer network that spans a relatively large geographical area. Typically, a WAN consists of two or more local-area networks (LAN). The largest WAN in existence is the Internet.

2.1.3 Metropolitan Area Network

Metropolitan Area Network, a data network designed for a town or city. MAN is larger than LAN, but smaller than WAN. MAN is usually characterized by very high-speed connections using fiber optical cable or other digital media.

2.2 Ethernet

Ethernet is the most widely implemented method of networking computers in a LAN. An architecture developed by Xerox Corporation in cooperation with DEC and Intel in 1976. [2]

Ethernet uses a bus or star topology and supports data transfer rates at 10 Mbps. Ethernet uses the CSMA/CD access method to handle simultaneous demands.

2.3 OSI Model

Open System Interconnection (OSI) model is an ISO standard for worldwide communications that defines a networking framework for implementing protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one host, proceed to the bottom layer, over

the network to the next host and back up the hierarchy. OSI model layer are shown in figure 2.1.

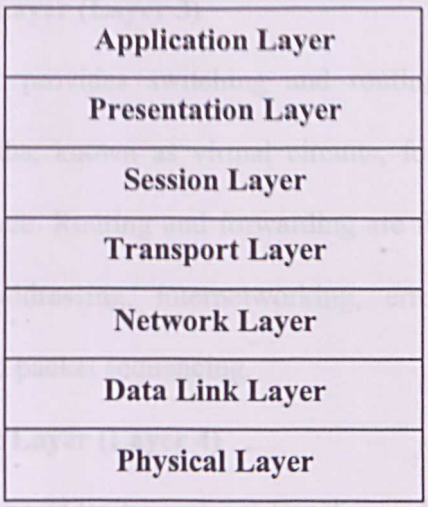


Figure 2.1 – OSI Model

2.3.1 Physical Layer (Layer 1)

This layer conveys the bit stream such as electrical impulse, light or radio signal through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier including defining cables, cards and physical aspects. Fast Ethernet, RS232, and ATM are protocols with physical layer components.

2.3.2 Data Link Layer (Layer 2)

At this layer, data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and frame synchronization. The data link layer is divided into two sub layers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sub layer controls how a computer on the network gains access to the data and permission to transmit it. The

LLC layer controls frame synchronization, flow control and error checking.

2.3.3 Network Layer (Layer 3)

This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing.

2.3.4 Transport Layer (Layer 4)

This layer provides transparent transfer of data between end systems or hosts and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer.

2.3.5 Session Layer (Layer 5)

This layer establishes, manages and terminates connections between applications. The session layer sets up, coordinates, and terminates conversations, exchanges and dialogues between the applications at each end. It deals with session and connection coordination.

2.3.6 Presentation Layer (Layer 6)

This layer provides independence from differences in data representation (e.g., encryption) by translating from application to network format and vice versa. The presentation layer works to transform data into the form that the application layer can accept. This layer formats and encrypts data to be sent across a network.

2.3.7 Application Layer (Layer 7)

This layer supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered and any constraints on data syntax are identified. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail and other network software services. Telnet and FTP are applications that exist entirely in the application level.

2.4 Packet

Packet is a piece of a message transmitted over a network. One of the key features of a packet is that it contains the source and destination address in addition to the data.

An example, in IP networks, packets are often called datagram. A datagram is a variable packet (size up to 65,536 bytes) consisting of two parts: header and data. The header can be from 20 to 60 bytes and contains information essential to routing delivery. See figure 2.2. [3]

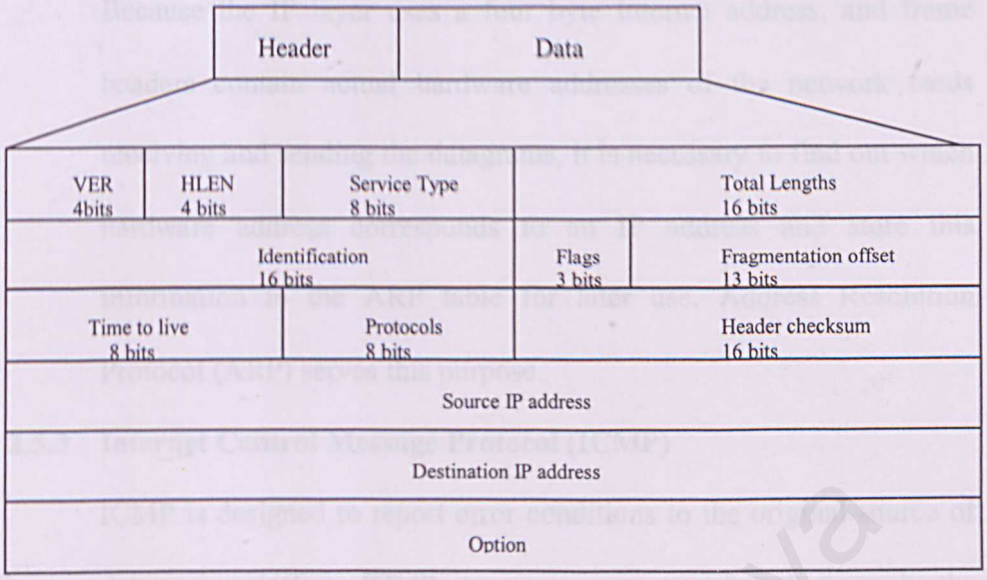


Figure 2.2 – IP datagram

2.5 Protocol

In networking, the term protocol refers to a set of rules that govern communications. For two devices on a network to successfully communicate, they must both understand the same protocols. In this project, the user will determine the type of protocol in the packet they want to capture. Lists of protocols are described below:

2.5.1 Internet Protocol (IP)

IP is responsible for moving packet of data from host to host. IP forwards each packet based on a four byte destination IP address. The internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.

Header Field: Refer Figure 2.2

2.5.2 Address Resolution Protocol (ARP)

Because the IP layer uses a four byte Internet address, and frame headers contain actual hardware addresses of the network cards receiving and sending the datagrams, it is necessary to find out which hardware address corresponds to an IP address and store this information in the ARP table for later use. Address Resolution Protocol (ARP) serves this purpose.

2.5.3 Internet Control Message Protocol (ICMP)

ICMP is designed to report error conditions to the original source of data transmission. ICMP messages are transferred through the network as the data portion of an IP datagram. ICMP messages themselves can be lost but no new error messages about ICMP errors are generated. ICMP is used by hosts and routers to send notification of datagram problem back to sender.

2.5.4 Transmission Control Protocol (TCP)

TCP is responsible for verifying the correct delivery of data from client to server and that the packets will be delivered in the same order in which they were sent. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.

Header Field: source and destination port address, sequence number, control, check, HLEN, etc.

2.5.5 User Datagram Protocol (UDP)

UDP, like TCP, is a higher encapsulation level protocol of the IP protocol. UDP uses IP to transport messages from one machine to

another. Unlike TCP, UDP does not provide a reliable mechanism for data transfer. It does not require acknowledgment of received data and does not provide a means to control data position in transmitted datagrams. Applications using UDP as a transport must accept full responsibility for making UDP a reliable communication process.

Header Field: source and destination port address, check sum and UDP length

2.5.6 Bootstrap protocol (BOOTP) and Dynamic Host Configuration Protocol (DHCP) – a client's server protocol used to provide IP address, subnet mask, router's IP address and server's IP address information.

2.5.7 Domain Name System (DNS) – a system protocol that map a name to an address and conversely an address to a name.

2.5.8 Other protocols- PPP, POP, NetBIOS, X.25, ATM, AARP, IEEE 802.11, IGRP and etc.

2.6 Internet Protocol Address (IP Address)

Networks using the TCP/IP protocol route messages based on the IP address of the destination. IP address to as Internet protocol address, it identifies a networked system so that it may communicate via Internet protocols.[4] IP addresses can be divided into two parts: the network and the host on the network. There are five IP classes:

Class A - supports 16 million hosts on each of 126 networks

Class B - supports 65,000 hosts on each of 16,000 networks

Class C - supports 254 hosts on each of 2 million networks

Class D - Used for multicasts

Class E - Used for experimental purposes only.

2.7 Port

A port is a logical connection place and the way a client program specifies a particular server program on a computer in a network.

Higher-level applications that use TCP/IP such as the Web protocol, Hypertext Transfer Protocol, have ports with preassigned numbers. These are known as well-known ports that have been assigned by the Internet Assigned Numbers Authority (IANA).

Other application processes are given port numbers dynamically for each connection. When a service (server program) initially is started, it is said to bind to its designated port number. As any client program wants to use that server, it also must request to bind to the designated port number.

Port numbers are from 0 to 65536. Ports 0 to 1024 are reserved for use by certain privileged services. For the HTTP service, port 80 is defined as a default and it does not have to be specified in the Uniform Resource Locator (URL). Some sample port numbers and the protocols with which they are most commonly associated, are listed in table 2.1 [5]:

Table 2.1 – List of Well Known Ports

20 (TCP): FTP data 21 (TCP): FTP control 23 (TCP): Telnet 25 (TCP): SMTP 37 (UDP): Time 43 (TCP): NICNAME/whois 53 (TCP/UDP): Domain Name System (DNS) 69 (UDP): Trivial FTP (TFTP) 79 (TCP): Finger 80 (TCP): HTTP 110 (TCP): POP3 119 (TCP): NNTP	137 (TCP/UDP): NETBIOS Name Service 123 (UDP): Network Time Protocol (NTP) 138 (UDP): NETBIOS Datagram Service 139 (TCP): NETBIOS Session Service 143 (TCP): IMAP 161 (UDP): SNMP 162 (UDP): SNMP trap 179 (TCP): Border Gateway Protocol (BGP) 443 (TCP): Secure HTTP (HTTPS) 520 (UDP): RIP 1080 (TCP): SOCKS 33434 (UDP): "Invalid" port used by trace route

2.8 Non-switched Network

A segment is a network architecture that resides behind a router, bridge, hub or switch in which every node is directly addressable from every other node. In a non-switched environment, frames are handled in a broadcast manner. That is, when a frame is transmitting from one host, it is 'seen' by every other active host on the segment. Each host, in turn, will briefly exam the frame to see if it is addressed to them. If it is not intended for them, the frame will be discarded.

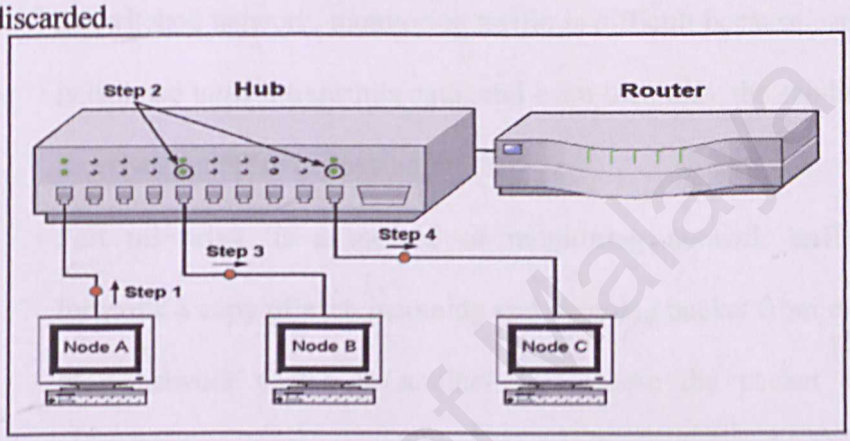


Figure 2.3 Non-switched Environment

2.9 Switched Network

In a switched environment, data frames are handled in a direct manner. When a frame is transmitting from one host, it is only seen by the addressed host. The hosts still perform the examination of the destination address even though the host is the intended host.

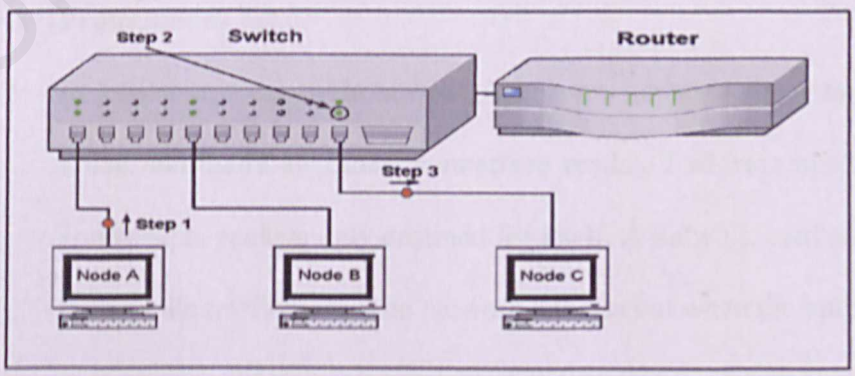


Figure 2.4 – Switched Environment

It is not difficult to sniff in a non-switched environment not difficult because all frames are broadcasted to every active port. While in the switched environment, since it does not broadcast most frames, in order for a host to be used as a sniffing agent, the network interface must be set to 'promiscuous' mode. After this mode is set, the network interface will no longer drop network frames which are addressed to other hosts.

2.9.1 Port Mirroring

In switched network, monitoring traffic is difficult because each port is isolated until it transmits data, and even then only the sending and receiving ports are connected.

Port mirroring, is a method of monitoring network traffic that forwards a copy of each incoming and outgoing packet from one port of a network switch to another port where the packet can be studied.[6] It enables the administrator to keep close track of switch performance and alter it if necessary.

Packet sniffer and analyzer can be placed on the port receiving the mirrored data to monitor the network traffic. The sniffer captures and evaluates the data without affecting the client on the original port. One example for port mirroring is SPAN (Switch Port Analyzer).

2.9.2 Promiscuous Mode

In packet sniffing environment, packets are captured in promiscuous mode. Normally an Ethernet interface reads all address information and accepts packets only destined for itself. A network card will only capture the traffic to its own network address but when the interface is

in promiscuous mode, it reads all information, regardless of its destination.

2.10 Network Interface / Adapter

Adapter is one of the essential parts in the system. It allows the system to choose which interface (network card) will be used to capture frames from.

2.11 Packet Filtering

Packet filtering is a method of enhancing network security by examining network packets as they pass through switch or router. Packet filter allow the user to determine which packet to pass or to capture. Packets may be filtered based on their protocol, sending or receiving port, sending or receiving IP address. Filter rules or filter syntax specifies the criteria that a packet must match.

2.11.1 Packet Filtering Definition

A host or network device, without packet filtering, looks at a packet's destination address and decides whether or not this packet has to be routed through the router or should remain on that interface. This is a basic principle that routing works under. When packet filtering is added, it adds another level of analysis for each packet. The first step is still examination of the destination address. Then, if the router or switch has determined it has to process the packet, it applies its filter rules.

Filter rules are a security policies implemented as approved and disapproved services. For instance, packets destined for particular host can restricted, based on the specific types of packets or filtering rules. The restriction can even be done on packets leaving LAN destined for the outside world. Packet

filtering can be very sweeping or specific down to individual machines and ports.

For instance, an organization running a web server on host X. Users on the internet are allowed to access the organization's web pages, but cannot telnet into that host on the LAN. Packet filtering can be used for this type of selective access to restrict their services.

2.11.2 Why Use Packet Filtering?

Packet filtering is most commonly used as a first line of defense against attacks from machines outside the local LAN. Since most routers have built-in filtering capabilities, packet filtering has become a common and inexpensive method of security. Packet filtering can be very flexible and powerful to guarantee the security of network and all the hosts and devices connected to it.

2.11.3 How Powerful is Packet Filtering?

Packet filtering allows to explicitly denies or allow packets by machine and/or port. One machine filtering is based on IP address. For instance, all packets destined for port 80 on all host on the local LAN are restricted except host X and Y.

2.11.4 Packet Filtering Downfall

The downfall of packet filtering is the lack of flexibility. Standard packet filtering allows or restricts packets to a location or from a location. There is no "sometimes" or "only from this person". If telnets from the outside world is disallowed into a particular host, that's that. No host on the across the network can telnet into the host specified in the filter. This sort of filtering is known as Static Filtering.

Dynamic Filtering is more flexible by allowing packets restriction only from certain users. For instance, stop all incoming telnet packets except those from user X, Y and Z. This is accomplished via an advanced security system which challenges the incoming user to provide a passkey before the network device will pass packets into the local LAN.

2.11.5 How to Configure Packet Filters

There are three basic steps to packet filtering [7]:

- ***Knowing what to permit and what to restrict.***

  As a first step, decision must be made on what packets are approved and which are restricted. For example, do all the hosts on the local LAN to accept telnet from the outside or only one central host. The best security policy is to restrict all packets except those expressly permitted.

- ***Formally defining packets that should be permitted and restricted.***

  After knowing what to accept or deny, define a way which allows easy translation into software syntax. A good template to work under is shown in table 2.2 [8]:

Table 2.2 – Packets Definition Template

Action	Source	Port	Destination	Port	Type
deny	xxx.xxx.xxx.xxx	#####	xxx.xxx.xxx.xxx	#####	(type)
allow	xxx.xxx.xxx.xxx	#####	xxx.xxx.xxx.xxx	#####	(type)

- ***Translating formal definitions to software syntax.***

  Meaning that translation is done in such a way that it can be understood by the network device or host running the packet sniffer software.

2.11.6 Types of Packet Filter

There are two types of packet filtering. One is static and the other is dynamic. [9]

2.11.6.1 Static Packet Filtering

This type does not track the state of network packets and does not know whether a packet is the first, a middle packet or the last packet. It does not know if the traffic is associated with a response to a request or is the start of a request.

2.11.6.2 Dynamic Packet Filtering

This type of filter tracks the state of connections to tell if someone is trying to fool the firewall or router. It can tell if traffic is associated with a response or request. This type of filtering is much more secure and flexible than static packet filtering.

Dynamic filtering also allows packets restriction from certain users. For instance, the filter could stop all incoming telnet packets except those from user X, Y and Z.

2.11.7 Basic Packet Filtering Modules

2.11.7.1 MAC addresses

Packet filtering based on MAC addresses enables only certain computers to transmit data through the filter.

Syntax : ether host 08:00:08:15:ca:fe

2.11.7.3 Internet Protocol (IP) addresses

Packet filtering based on IP address. This filter is used to permit only traffic destined to, or originating from, specific addresses to pass through the filter.

Syntax : host 192.168.0.10

2.11.7.3 Protocol

Protocol filtering based on the protocol field in the IP header, enabling only certain protocols to enter the system.

2.11.7.4 Port numbers

Packet filtering based on the source or destination port number enables it to be more specific about the types of traffic to allow into a computer or onto a network.

Syntax : tcp port 80

2.11.8 Advanced Packet Filtering Characteristics

2.11.8.1 Bit and Byte Value Patterns

It is done by specifying values at exact bit and/or byte locations in a packet. For example:

TCP headers with the SYN bit set to 1 (this statement indicates someone attempting to make a TCP connection on the network).

2.11.8.2 Boolean Operations

By mixing two different filter command into a single filter. For example (AND operation) [10]:

Packets with the ICMP type field value of 3 (Destination Unreachable)

AND

Packets with the ICMP Code value 3 (Port Unreachable)

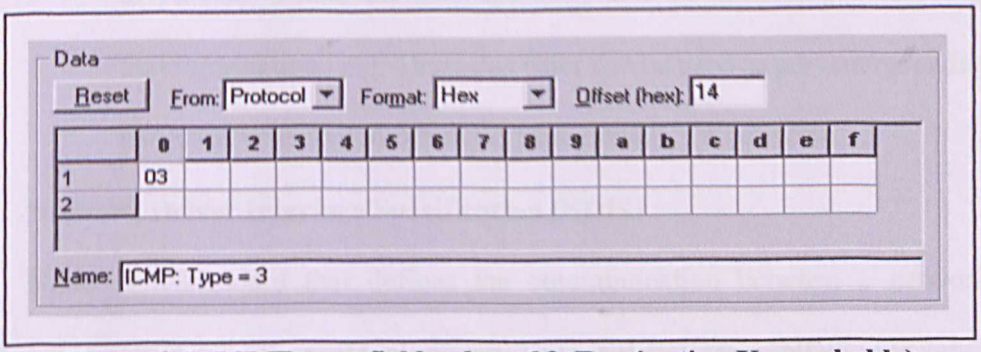


Figure 2.5- ICMP type field value of 3 (Destination Unreachable)

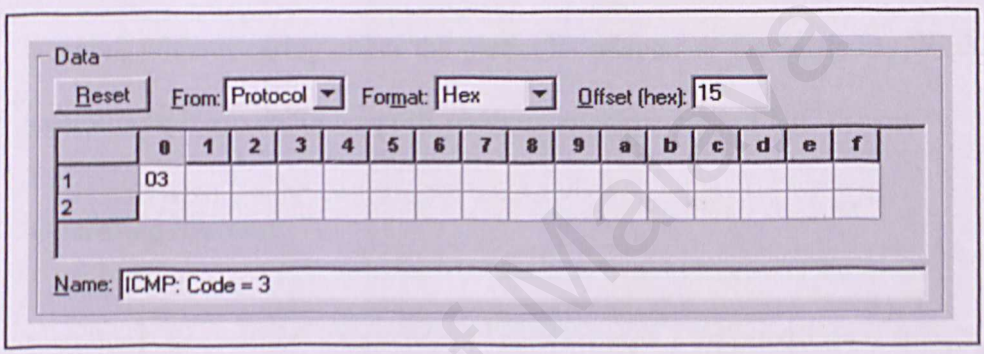


Figure 2.6 - ICMP Code value 3 (Port Unreachable)

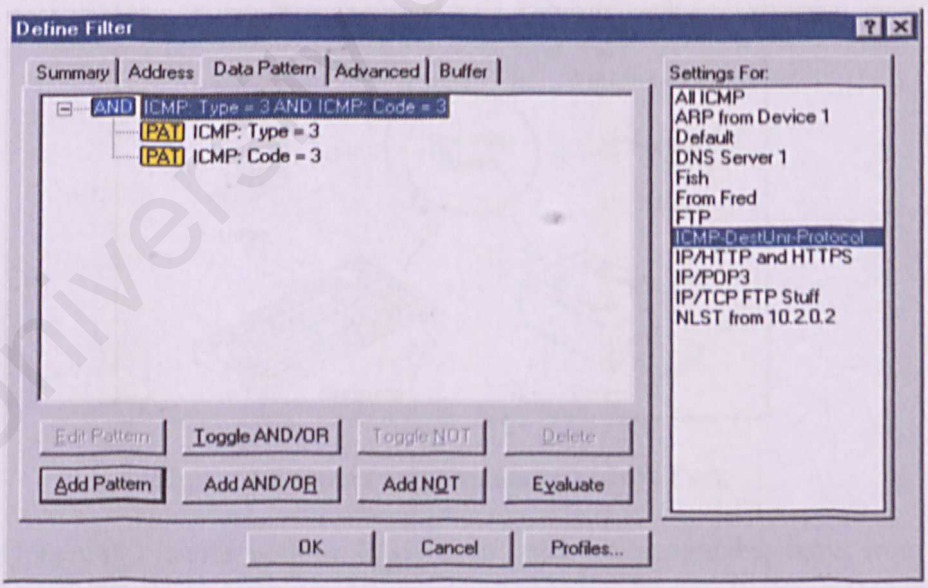


Figure 2.7 - The final data pattern filter

2.11.8.3 Application layer Filtering (ALF)

ALF can be used to filter a packet with abnormal information in the headers of a message and even within the data itself. This type of

filter modules can be set to look for specific character strings (words or phrases) within the message body and block messages based on that information [11]. Thus, this filter can be used to prevent receiving particular sensitive information from outside the network.

2.12 Network Driver Interface Specification (NDIS)

NDIS is a standard that defines the communication between a network adapter/driver and the protocol drivers. The main purpose of NDIS is to act as a wrapper that allows protocol drivers to send and receive packets onto a network without caring either the particular adapter or the particular Win32 operating system. NDIS supports network interface card or NIC drivers.

2.13 Operating Systems

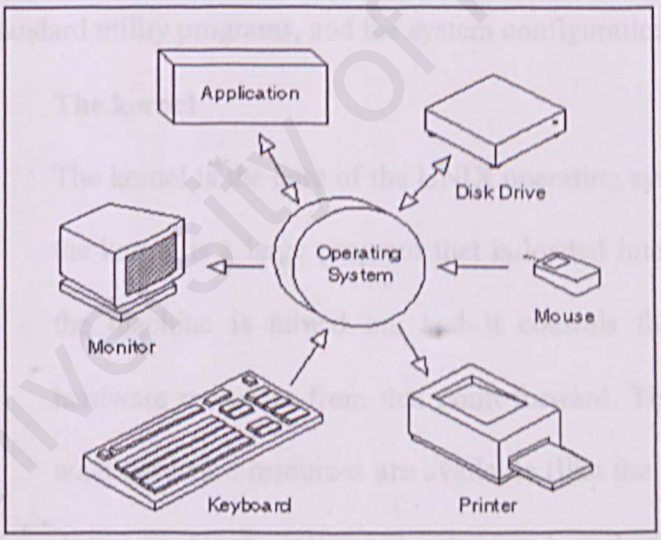


Figure 2.8 Operating Systems Control Task

Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.

For large systems, the operating system has even greater responsibilities and powers. It makes sure that different programs and users running at the same time do not interfere with each other. The operating system is also responsible for security, ensuring that unauthorized users do not access the system.

2.13.1 UNIX

UNIX is an operating system created in the late 1960s, in an effort to provide a multi-user, multitasking system for use by programmers. The philosophy behind the design of UNIX was to provide simple, yet powerful utilities that could be pieced together in a flexible manner to perform a wide variety of tasks.

The UNIX operating system comprises three parts: The kernel, the standard utility programs, and the system configuration files [12].

- **The kernel**

  The kernel is the core of the UNIX operating system. Basically, the kernel is a large program that is loaded into memory when the machine is turned on, and it controls the allocation of hardware resources from that point forward. The kernel knows what hardware resources are available (like the processor, disk drives, network interfaces, etc.) and it has the necessary programs to talk to all the devices connected to it.

- **The standard utility programs**

  These programs include simple utilities like cp, which copies files, and complex utilities, like the shell that allows you to issue commands to the operating system.

- **The system configuration files**

  The system configuration files are read by the kernel, and some of the standard utilities. The UNIX kernel and the utilities are flexible programs and certain aspects of their behavior can be controlled by changing the standard configuration files. One example of a system configuration file is the file system table "fstab", which tells the kernel where to find all the files on the disk drives.

2.13.2 Microsoft Windows XP

Windows XP Professional combines the security, manageability, and reliability of Windows 2000 with the user friendly environment of Windows 98 and Windows Millennium Edition. This combination creates the best computer operating system around for professional use.

Windows XP features include [13]:

- Windows XP Professional is built on the proven code base of Windows NT and Windows 2000, which features a 32-bit computing architecture and a fully protected memory model.
- Building on the device driver verifier found in Windows 2000, the Windows XP Professional will provide even greater stress tests for device drivers.
- Protects core system files from being overwritten by application installations. If a file is overwritten, Windows File Protection will restore the correct version.

- Critical kernel data structures are read only, so that drivers and applications cannot corrupt them. All device driver code is read-only and page protected.
- A system service that helps users installs, configure, track, upgrade, and remove software programs correctly.
- Designed to allow multiple applications to run simultaneously, while ensuring great system response and stability.
- IP security (IPSec) helps protect data transmitted across a network. It is an important part of providing security for virtual private networks (VPN), which allow organizations to transmit data securely over the Internet.
- Easily manage security resources with single, unified view of key settings, tools and access to resources.
- Windows XP Professional has a fresh visual design. Common tasks have been consolidated and simplified. New visual have been added to help users navigate their computers more easily.

2.13.2.1 Advantages

- **Built on the new Windows engine**

  Windows XP Professional will provide a dependable computing experience for all type of users.

- **Enhanced device driver verifier**

  Device drivers that pass these tests will be the most robust drivers available, which will ensure maximum system stability.

- **Windows File Protection**

  By safeguarding system files, Windows XP Professional mitigates many of the most common system failures encountered in earlier versions of Windows.

- **Improved code protection**

  Rogue applications cannot adversely affect core operating system areas.

- **Windows Installer**

  Will help minimize user downtime and increase system stability.

- **Preemptive multitasking architecture**

  Run your most demanding applications while still experiencing impressive system response time.

- **Fresh visual design**

  Allows the most common tasks to be exposed easily, helping users get the most of out of Windows XP Professional.

- **Adaptive user environment**

  A cleaner work environment allows the user to be more efficient. Users can find the crucial data and applications they need quickly and easily. All of these settings can be controlled using Group Policy, so IT administrators can decide what features are most appropriate for their environments.

- **Troubleshooters**

  Enables users to be more self-sufficient, resulting in greater productivity, fewer help desk calls, and better customer service.

2.14 Programming Language

A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks. Every language has its strengths and weaknesses. The choice of which language to use depends on the type of computer the program is to run on, what sort of program it is, and the expertise of the programmer.

The following are two types of programming language most commonly used:

2.14.1 Java

Java, formerly known as oak, is an object-oriented programming language developed by Sun. It shares many similarities with C, C++, and Objective C. [14]

The language was originally created because C++ proved inadequate for certain tasks. Since the designers were not burdened with compatibility with existing languages, they were able to learn from the experience and mistakes of previous object-oriented languages.

Even more importantly Java was designed from the ground up to allow for secure execution of code across a network. Most notably there are no pointers in Java. Java programs cannot access arbitrary addresses in memory. Java was designed not only to be cross-platform in source form like C, but also in compiled binary form and it is easier to write a bug free code

2.14.2 C and C++

C++ is an object oriented programming language created by Bjarne Stroustrup and released in 1985. It implements data abstraction using a concept called classes. C++ features allow object oriented

programming. Parts of C++ program are modifiable, reusable and extensible, existing code is easily modifiable without having to change the code. [15]

C++ concept called operator overloading not seen makes the creation of libraries much cleaner. C++ maintains aspects of the C programming language, yet has features which simplify memory management. Additionally, some of the features of C++ allow low level access to memory but also contain high level features. C++ could be considered a superset of C.

C programs can run on C++ compilers. C uses structured programming concepts and techniques while C++ object oriented programming and classes which focus on the data.

C is such a powerful tool because it can be used to write machine-independent programs.

2.15 Compiler

Compiler transforms a program written in a high-level programming language from source code into object code. Programmers write programs in a form called source code. Source code then will go through several steps before becomes an executable program (shown in figure 2.9). [16]

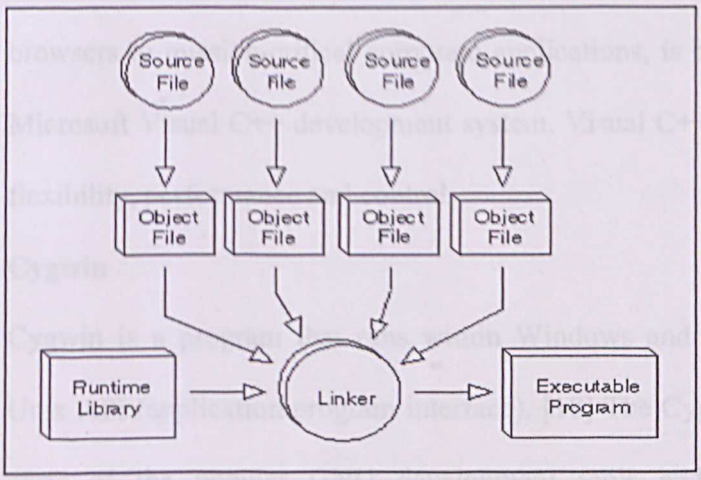


Figure 2.9 Compiler Architecture

First step - pass the source code through a compiler, which translates the high-level language instructions into object code.

Second step - object code is passed through a linker and the linker combines modules and gives real values to all symbolic addresses, thereby producing machine code.

2.15.1 Microsoft Visual C++

Microsoft Visual C++ is the most productive C++ tool for creating the highest-performance applications for Windows and the Web. Visual C++ is an application development tool introduced in 1993 by Microsoft for C++ programmers. [17]

Visual C++ supports :

- object-oriented programming of 32-bit Windows applications with an integrated development environment (IDE). Visual C++ was introduced in 1993,
- C/C++ compiler
- class library called the Microsoft Foundation Classes (MFC)

Nearly all world-class software, ranging from the leading web browsers to mission-critical corporate applications, is built using the Microsoft Visual C++ development system. Visual C++ 6.0 provides flexibility, performance and control.

2.15.2 Cygwin

Cygwin is a program that runs within Windows and emulates that Unix API (application program interface). [18] The Cygwin tools are ports of the popular GNU development tools and utilities for Windows NT and 9x. They function through the use of the Cygwin library. Meaning, it runs Unix applications within Windows. This allows Unix commands and applications running from within Windows. Cygwin runs under most Windows versions including 95/98/ME/NT/2000/XP.

Unlike other methods of running Unix applications on a Windows computer such as dual-booting, cygwin allows both Windows and Unix applications to be used simultaneously and it can be easily installed and uninstalled without the threat of losing Windows data.

Cygwin provides a large number of Unix packages including most basic Unix commands, several shells, programming tools, graphics programs including ghostscript and internet tools.

Besides UNIX packages provided by cygwin, many other applications can be compiled under cygwin. Cygwin can access Windows files. For example, to find command to search all documents on the C drive. The Windows system can also access cygwin files. For example, to

open a graphic created by a Unix application with a Windows program.

2.15.2.1 Disadvantage

It is difficult to print from Unix applications because cygwin has no printing system like lp or lpr.

- Unlike standard Unix, cygwin is case insensitive. This causes some Unix applications not to run properly.
- While there are a large number of supported applications, some Unix applications won't run under cygwin.
- Cygwin is not adequate for those who need a fully functional Unix installation.

However, Cygwin provides an excellent solution for people who want to run Unix commands and applications from within Windows without the complications of dual booting.

CHAPTER 3
SYSTEM METHODOLOGY

CHAPTER 3 – SYSTEMS METHODOLOGY

System methodology is method to create a system with a series of steps or operations and can be defined as system life cycle model. Every system development process includes system requirements (users, needs and resources) as inputs and finished product (system) as output. [19]

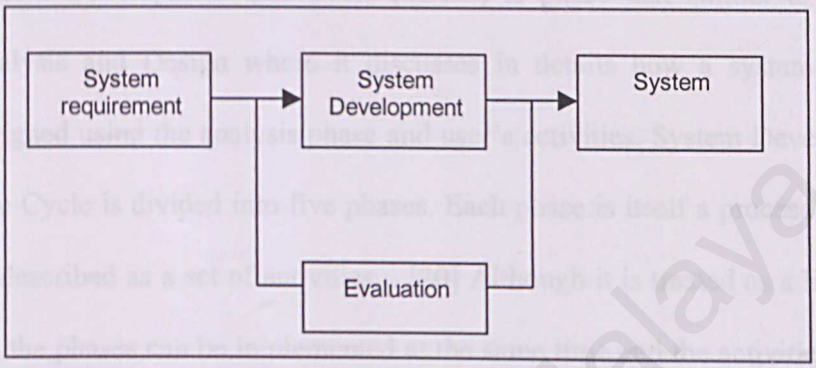


Figure 3.1 – System Operation

After comparing some of the system development and methodology approach, the waterfall model is used to build the system. The waterfall model as shown in figure 3.2 is used to describe the system development activities.

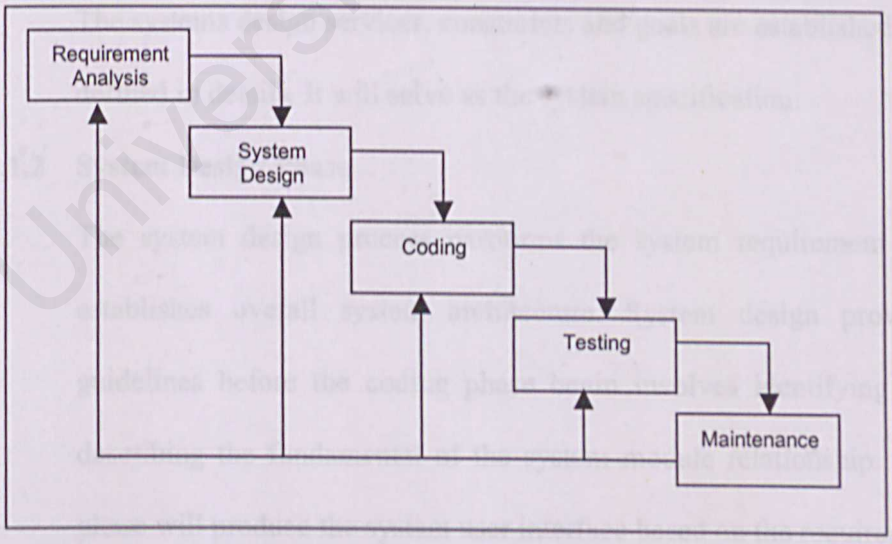


Figure 3.2 Waterfall Model

As the figure implies, the advantage of this model is that we can correct the error during the development process without waiting for other phase to complete. To understand further the waterfall model, we have to understand the system development life cycle of this model.

3.1 System Development Life-Cycle

System Development Life-Cycle (SDLC) is phase that similar to System Analysis and Design where it discusses in details how a system can be designed using the analysis phase and user's activities. System Development Life Cycle is divided into five phases. Each phase is itself a process and can be described as a set of activities. . [20] Although it is treated as a life cycle but the phases can be implemented at the same time and the activities can be separated as necessary. The five phases are:

3.1.1 Requirement Analysis Phase

Requirement is a feature of the system of what the system is capable of doing in order to fulfill the system purpose.

The systems design services, constraints and goals are established and defined in details. It will serve as the system specification.

3.1.2 System Design Phase

The system design process partitions the system requirement and establishes overall system architecture. System design provides guidelines before the coding phase begin involves identifying and describing the fundamental of the system module relationship. This phase will produce the system user interface based on the requirement specified.

3.1.3 Coding Phase

Code programming is executed at this phase by using the appropriate programming language to develop the system. The coding phase transforms the system design into a form readable by the computer. If the system design is complete and accurate, the coding can be established efficiently.

3.1.4 System Testing Phase

The system testing phase focus in finding fault that could cause failure in the system. Fault finding can increase the quality of the system. The failure in the system may result from missing requirement, requirement impossible to implement, fault in program design or incomplete algorithm in the program code.

3.1.5 Maintenance Phase

System development is complete when the system is operational, that is when the system is used by the user in the actual environment. Any work done to change the system after it is in operation is called maintenance. Maintenance phase is also implemented when changes or new requirements are discovered.

CHAPTER 4
SYSTEM ANALYSIS

4.1.2.2User friendly – Enhance and easy to use menu button and toolbars. Display simple dialog box and message for the user.

4.1.2.2 Managability – Easy to manage, simple browsing and not time consuming.

4.1.2.4 Performance – System speed is critical to output production. The system transaction must process in an acceptable range and not time consuming.

4.1.2.5 Flexible – The system developed should be able to be executed in a multi-user environment.

4.2 Hardware Requirement

The minimum hardware requirements to build the system are stated in table 4.1.

Table 4.1 – Hardware Requirement

No.	Device	Information
1.	CPU	Intel Pentium II processor 231 Mhz, 352 RAM of RAM
2.	Operating System	Microsoft Windows XP Professional Version 2002 Service Pack 2, v.2135
3.	Monitor	Dell D828L
4.	Network Adapter	3 Com 3C905TX Based Ethernet Adapater (Generic)

4.3 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

Sniffalyzer. The software requirements are chosen based on their functionality, affordability, easy to use and user friendliness feature.

4.3.1 Miscorsoft Visual C++ 6.0 Compiler

Microsoft Visual C++ is the most productive C++ tool for creating the highest-performance applications for Windows and the Web. Visual C++ 6.0 brings a new level of productivity to C++, without compromising flexibility, performance or control. Furthermore it work support wxWidgets class library.

4.3.1.1 Advantages

- Enjoy a new level of productivity with new features that significantly reduce less development time lead to less time in building applications, coding, compiling, and debugging.
- Better speed leads to faster code generation Visual C++ 6.0 is tuned in a number of places so that developers can build the fastest, smallest components and applications possible
- User interface enhancements makes navigation easier for user
- Allow creation of multimedia-based highly interactive, Dynamic HTML pages

4.3.2 wxWidgets version 2.4.2

wxWidgets also known as wxWindows is an easy to use API and sophisticated cross-platform C++ framework for writing advanced GUI applications on multiple platform and compilers. [22] It contains rich class of libraries ready to be use by programmer. It is also an application framework that provides architecture for using classes in developing complete applications.

The combination of the chosen compiler and programming tools will enable the development of a system's interface.

4.3.3 WinPCap version 3.0 Driver

The purpose of WinPcap is to access to Win32 applications. It provides facilities to capture raw packets and filter the packets according to user specified rules.

4.3.3.1 WinPcap Structure

WinPcap is an architecture for packet capture and network analysis for the Win32 platforms. It includes a kernel-level packet filter, a low-level dynamic link library (packet.dll) and a high-level and system-independent library (wpcap.dll). [23]

Packet capture is a low level mechanism that requires a strict interaction with the network adapter and with the operating system.

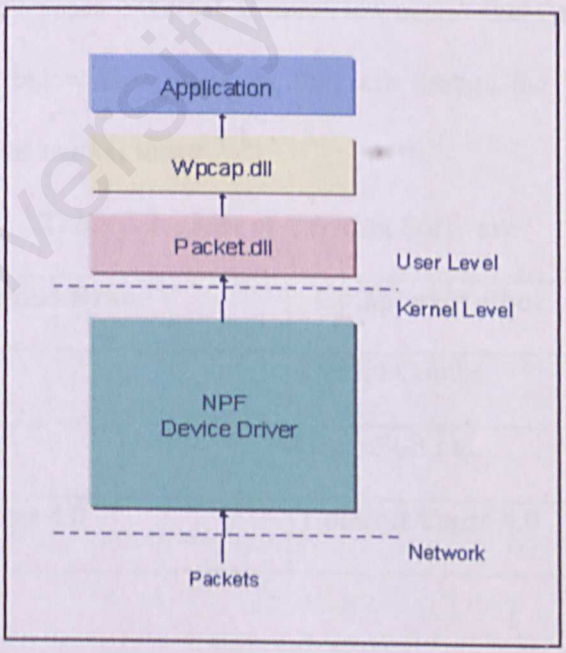


Figure 4.1 - WinPcap Architecture

The above figure (Figure 4.1) shows the various components of WinPcap:

Packet.dll is an application program interface (API) that can be used to directly access the functions of the packet driver, independent from the Operating System.

It provides functions to handle network adapters, read and write packets from the network, set buffers and filters in the driver and so on.

Wpcap.dll provides an application program interface (API) that work on any operating system. These features allow applications to capture packets on a network regardless of the type of operating system.

4.4 Existing Packet Sniffer and Analyzer Software

Three packet sniffer software are reviewed for analysis and feature comparison purposes. Ethereal, CommView and Packet Analyzer 4.0 as listed in table 4.2 below are chosen as they are among the best packet sniffer available in the market today.

Table 4.2 – List of Existing Software

No.	Product Brand	Company/Author	Availability
1.	Ethereal	Gerard Combs	Non-commercial
2.	CommView	TamoSoft Inc.	Commercial
3.	Packet Analyzer 4.0	Colasoft Capsa 4.0	Commercial

4.4.1 Ethereal

Ethereal is still technically beta software, but it has a comprehensive feature set and is suitable for production use.

Features:

- Data can be captured from a live network connection, or read from a captured file.
- Ethereal read capture files from tcpdump (libpcap)
- Live data can be read from Ethernet, FDDI, PPP, Token-Ring, IEEE 802.11
- Captured network data can be browsed via a GUI.
- Support 530 protocols
- Output can be saved or printed as plain text.
- Data display can be refined using a display filter.
- Display TCP streams

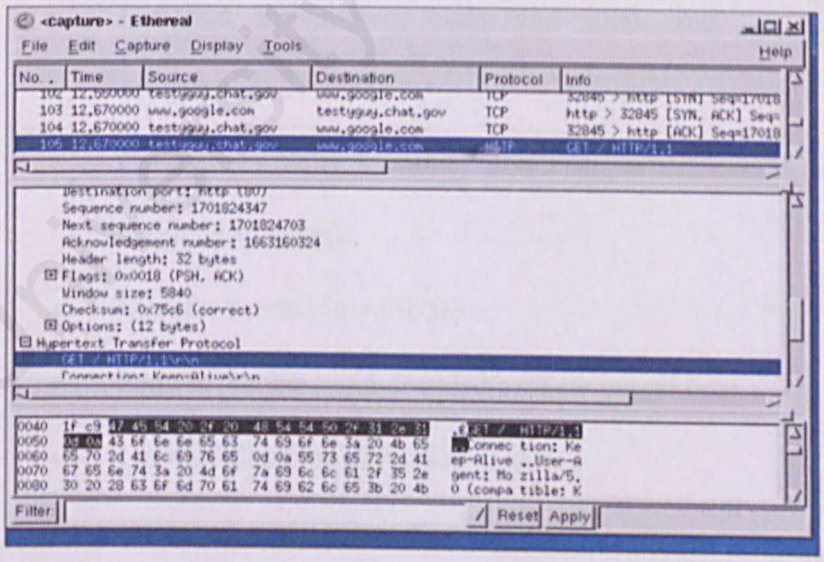


Figure 4.2 – Ethereal Screenshot

Disadvantage

- documentation user guideline are too simple

4.4.2 CommView

CommView is a program for monitoring Internet and Local Area Network (LAN) activity capable of capturing and analyzing network packets. It gathers information about data passing through your dial-up connection or Ethernet card and decodes the analyzed data.

Features

- Monitor internet and Local Area Network (LAN) activity
- Gathers information about data passing through dial-up connection or Ethernet card
- Decode packet with full analysis of protocols.
- Captured packets can be saved to log files for future analysis.
- Provide IP protocol and Ethernet protocol filters
- Designed for internet users and small and medium-sized networks
- Configure alarms to notify about important events, such as suspicious packets.
- Monitor bandwidth utilization.
- Browse captured and decoded packets in real time.
- Log individual or all packets to files.
- Load and view capture files offline.
- Filter packets based on Ethernet protocols and IP protocols

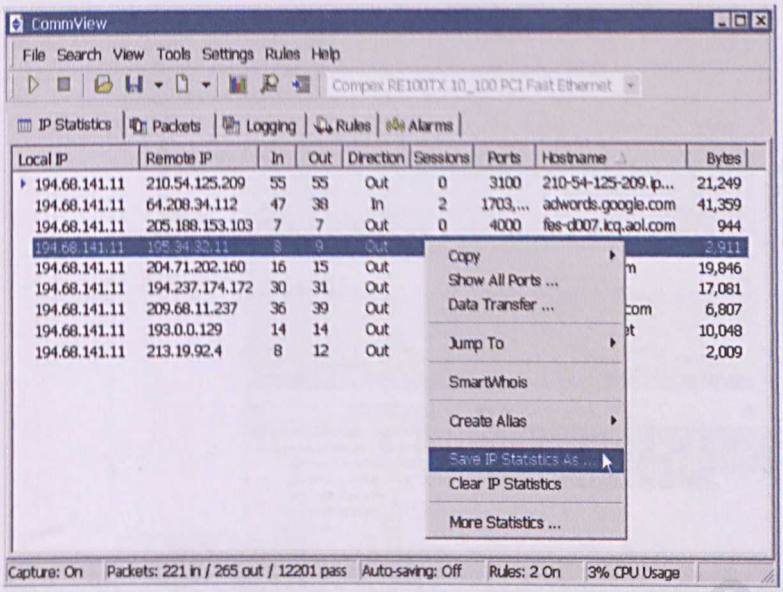


Figure 4.3 CommView Screenshot

Disadvantage

- does not provide installation guide and systems requirement.

4.4.3 Packet Analyzer 4.0

Packet Analyzer is a powerful but easy to use network monitor and analyzer designed for packet decoding and network diagnosing. It has the ability of real time monitoring and data analyzing.

Features

- Monitors and captures packets in real-time
- Display the analyzed data immediately after captured
- Captures email messages, web accesses and network transactions
- Resolves host name and address
- Decoding packets and packet header
- Displays IP address and MAC address of captured hosts
- Provide four filter types: packet filter, email filter, web` filter and transaction filter

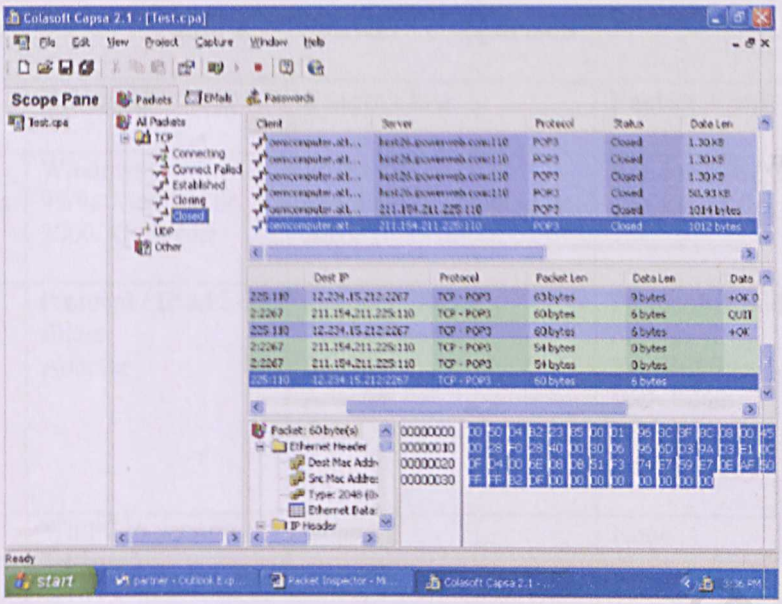


Figure 4.4 Packet Analyzer 4.0 Screenshot

Disadvantage

- does not support dial-up connection
- too many dialog box for filtering
- too many window frames for display
- does not provide installation guide and system requirement

4.5 Software Comparison

Table 4.3 shows the comparison between Ethereal, CommView and Packet Analyzer 4.0. The comparison was based on systems requirements, available modules, packet capture driver, number of protocols it supports and filters modules.

Table 4.3 – Software comparison

	Ethereal	**CommView**	**Packet Analyzer 4.0**
System Requirements	Windows 95/98/Me/NT4.0/ 2000/XP/Server	Windows 95/98/Me/NT/2000/XP/ 2003/	Windows 9x/XP/NT 3.x/NT/2000.
Modules	Protocol / IP address filters Adapter	Protocol Filters Packet Generator Alarm Adapter	Protocol/IP address Filters E-mail Filter Web Filter Transaction Filter Adapter
Driver	WinPCap version 3.1 beta3	None	None
Protocols	530	74	8
Filters	Protocol IP address	IP protocol Ethernet protocol	MAC address IP address Protocol Port Others

CHAPTER 5
SYSTEM DESIGN

CHAPTER 5 – SYSTEMS DESIGN

System design is the first of the three technical activities (design, code generation and testing) that are required to build and verify a system.[24]

The design process produces the function design and user interface design.

5.1 Systems Functions

Deals with the collaboration of each module to achieve the overall system functionality specification and interface design.

5.1.1 Structure Chart

Basically the system can be divided into four main modules as shown in figure 5.1.

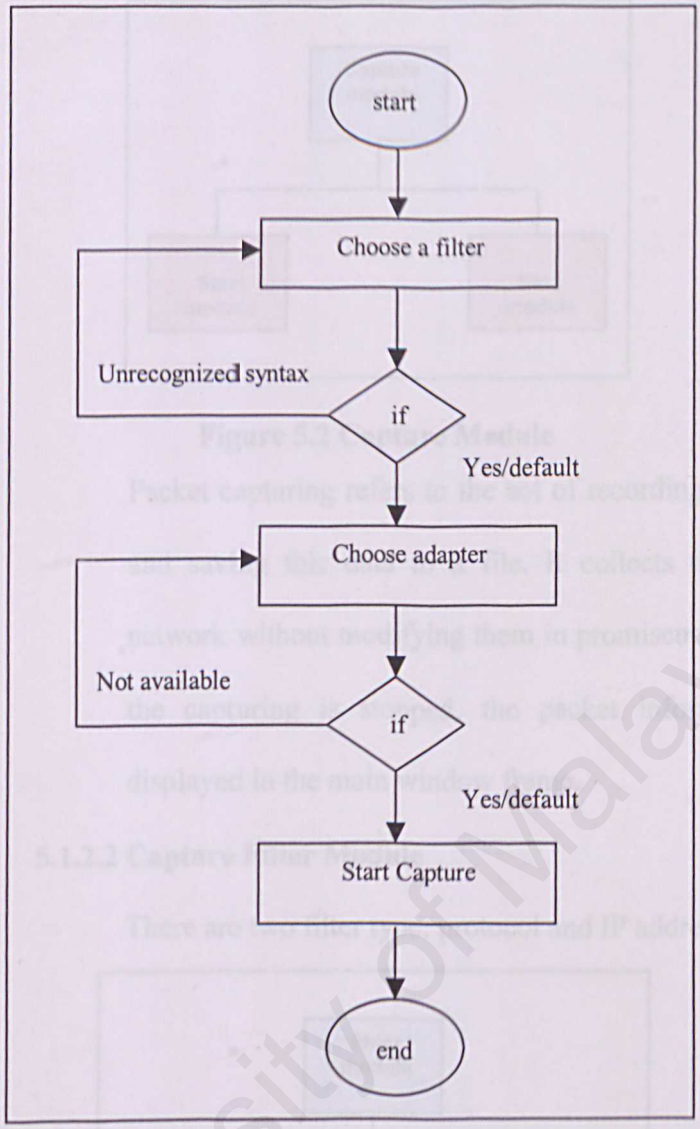


Figure 5.1 – Structure Chart for Sniffalyzer

5.1.2 Module Explanation

The following section will present detailed explanation for each module in the system.

5.1.2.1 Capture Module

There are two sub-modules in capture modules: start and stop capture.

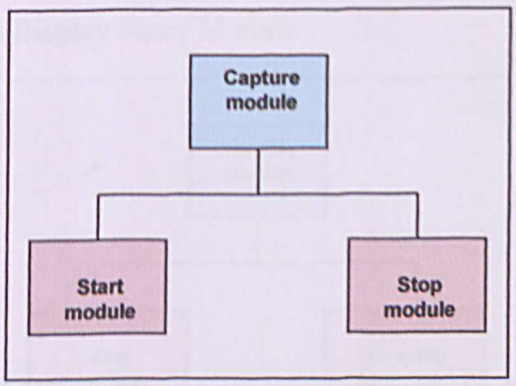


Figure 5.2 Capture Module

Packet capturing refers to the act of recording network traffic and saving this data to a file. It collects the packet from network without modifying them in promiscuous mode. When the capturing is stopped, the packet information will be displayed in the main window frame.

5.1.2.2 Capture Filter Module

There are two filter type: protocol and IP address.

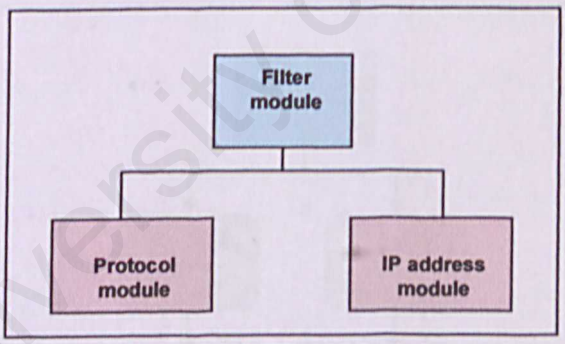


Figure 5.3 Filter Module

5.1.2.3 Adapter Modules

A number of adapters will be displayed in the adapter module.

5.1.2.4 Display Filter Module

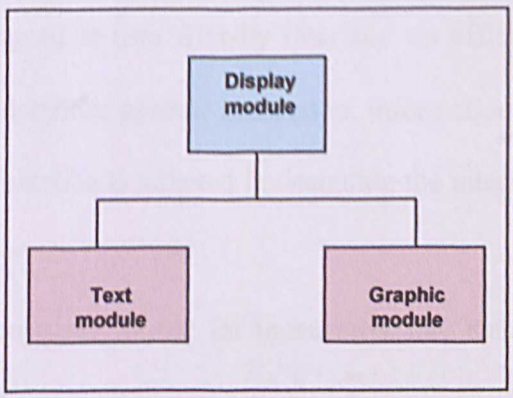


Figure 5.4 – Display Filter Module

This module consists of two sub-modules. Text module will display all or selected packet information in text and graphic module display packets based on protocol or IP address filter.

5.2 Data Flow Diagram (DFD)

Data flow diagram depicts the flow of data through a system or processing performed by that system.

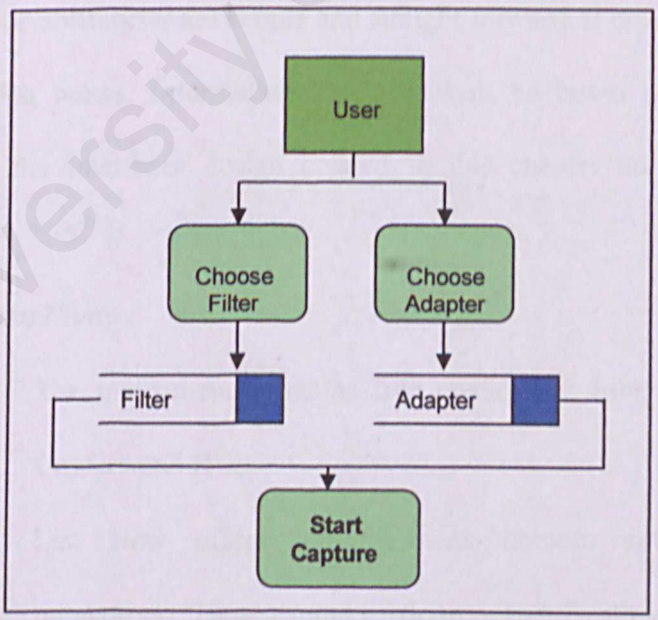


Figure 5.5 – Data Flow Diagram

5.3 User Interface Design

The system designed in user friendly interface yet efficient. Areas that are taken into consideration: general interaction, information display data input. The following guideline is adopted in designing the interface for the system. [26]

- Use a consistent format for menu selection, data display and other functions.
- Ask for verification for any destructive action such deletion of file.
- Reduce amount information memorize between actions.
- Seek Simple and efficiency in dialog box.

5.4 Systems Interface

Sniffalyzer interface created are based on the four main modules. User interface for Sniffalyzer are simple and straight forward. It does not offer too many dialog boxes. Information displayed will be based on user input. However, the interfaces design created in this chapter are subjected to changes.

5.4.1 Main Menu

- The main menu contains five menu bars: File, Edit, Option, Capture and Help.
- List view offers the following column options: Packet numbering, Time captured, Source and Destination Address, Protocol and packet Information.
- The window pane will display packets information and packet decode based on selected packet.

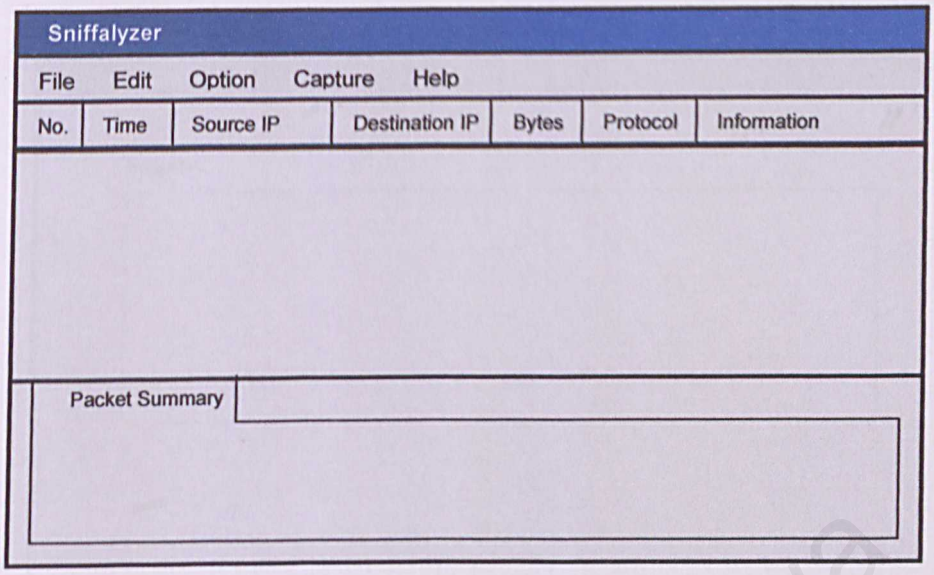


Figure 5.6 – Main Menu

5.4.2 Capture Module

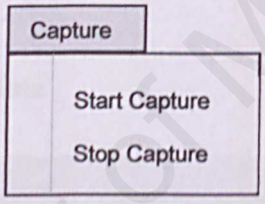


Figure 5.7 Capture Menu Bar

5.4.3 Capture Filter Module

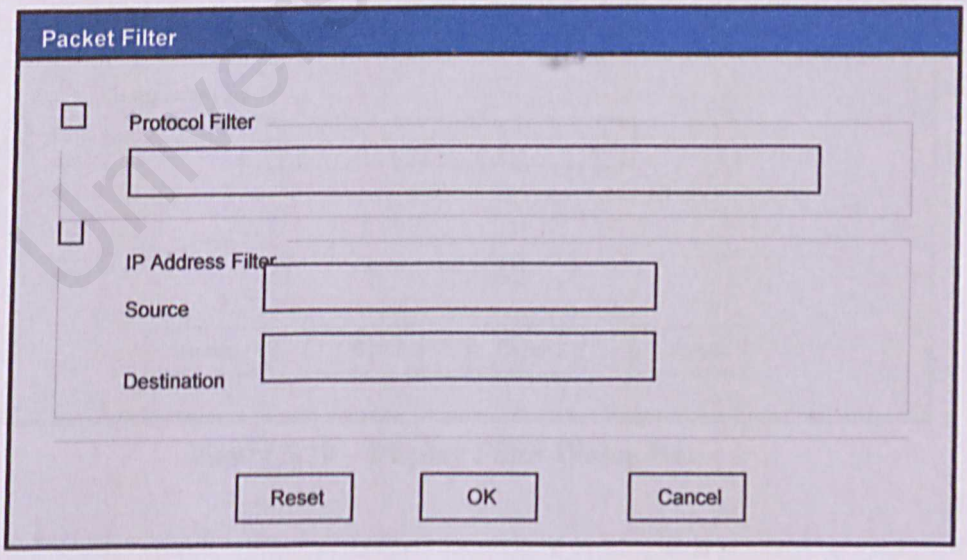


Figure 5.8 – Capture Filter Dialog Box

5.4.4 Adapter Module

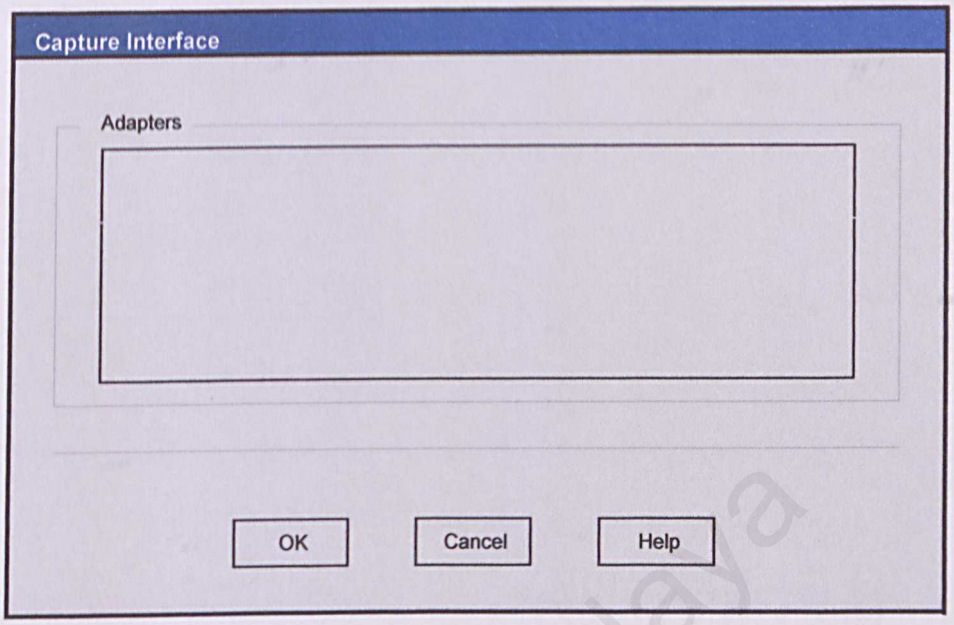


Figure 5.9 – Adapter Dialog Box

5.4.5 Display Filter Module

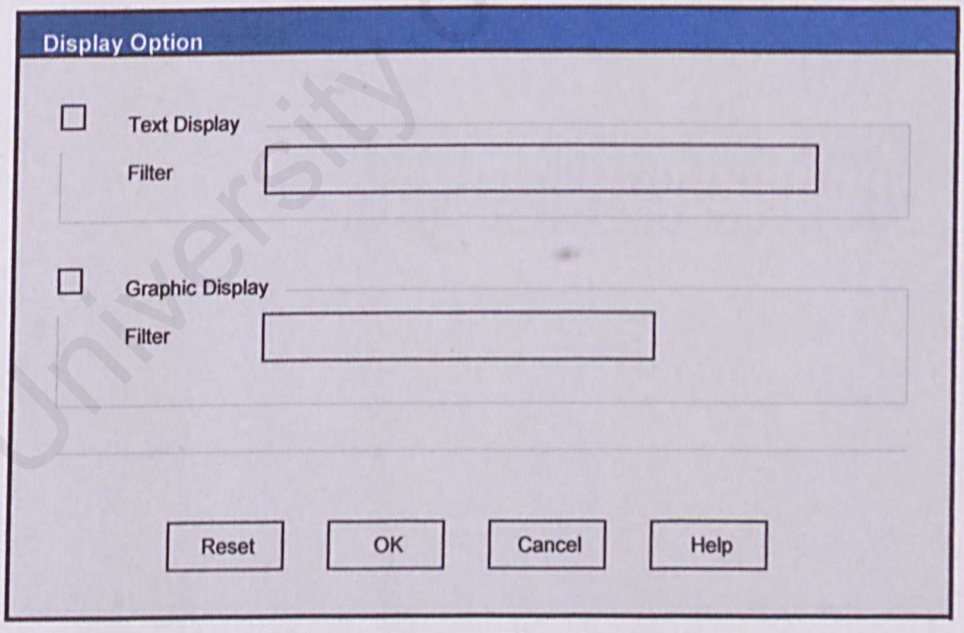


Figure 5.10 – Display Filter Dialog Box

CHAPTER 6
SYSTEM DEVELOPMENT AND IMPLEMENTATION

CHAPTER 6 – SYSTEM DEVELOPMENT AND IMPLEMENTATION

6.1 Introduction

Implementation is the process of translating the detailed design in to a program code. In this phase, the system requirement and design are being implemented and converted into program code. However due to certain limitations, modification are needed in order to develop the system in accordance to the limitation of development tools chosen to develop the system.

Each module in the system is initially being developed and tested phase by phase until it is fully becoming a functional system, after each module is able to run smoothly as an integrated system. The systems development involved code generation using Java language.

6.2 Development Environment

Table 6.1 – Hardware Requirement

No.	Device	Information
1.	CPU	Intel Pentium II processor 231 Mhz, 352 RAM of RAM
2.	Operating System	Microsoft Windows XP Professional Version 2002 Service Pack 2, v.2135
3.	Monitor	Dell D828L
4.	Network Adapter	3 Com 3C905TX Based Ethernet Adapter (Generic)

6.3 Development Tools

Table 6.2 – Development Software and Tools

No.	Software/Tools	Information
1.	Java 2 Platform, Standard Edition, v 1.4.2 (J2SE)	development environment for building applications, applets, and components using the Java programming language.
2.	JCreator LE 3.5	Integrated Development Environment (IDE) for Java
3.	Jpcap	set of Java classes which provide an interface and system for network packet capture
4.	Winpcap 3.1 beta 4	open source library for packet capture and network analysis for the Win32 platforms.
5.	Miscorsoft Visual C++ 6.0 Compiler	C++ Integrated Development Environment (IDE) for jpcap native codes

6.4 Program Development

Program development is the process of creating the programs needed to satisfy the information system's processing requirement. Program development consists of the following four steps.

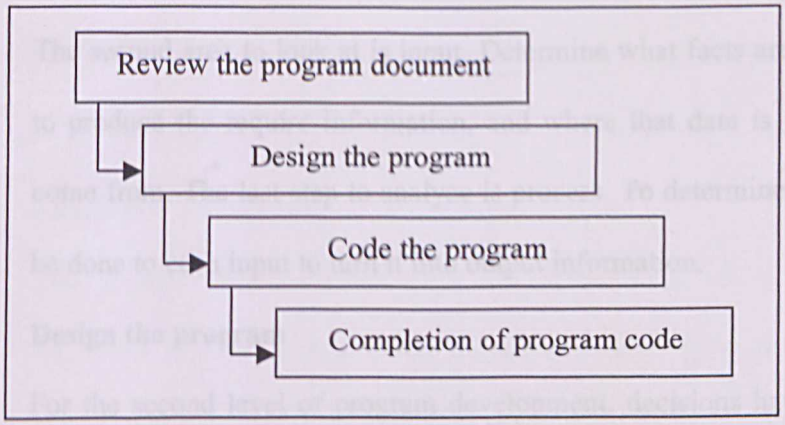


Figure 6.1 – Program Development Process

6.4.1 Review the program code

The first step in the program development is to review the program document that was prepared during the previous phase. The program document of consists of data flow of the system and the connection of module. The program document is then analyzed through these following steps:

- In written form, a complete definition of the requirements of the program.
- Understanding the written definition well enough to produce the desired result manually
- Defining the input required to produce the desired output.
- Identifying the source of the input.

Generally, the first area to analyze should be the output area of the program. This comes from the written definition of the requirements. The system's output will be show in the layout of on the screen, showing the information that should result from program running correctly.

The second area to look at is input. Determine what facts are needed to produce the require information, and where that data is going to come from. The last step to analyze is process. To determine what to be done to each input to turn it into output information.

6.4.2 Design the program

For the second level of program development, decisions have to be made on how the program can accomplish its tasks by developing a logical capturing solution to those program documents. The easiest way is to break the project into small pieces so and design the logic for each part of the problem.

6.4.3 Code the program

Coding programs is the process of translating the program design into the appropriate Java language to solve the problem. The activities in this process produce program modules that compile, build and run smoothly. Implementation of testing and analysis on the modules is to test its effectiveness and free of any error that could lead to system failure and malfunction.

6.5 System Coding

In system coding, every component of the program will look into this three aspects:

6.5.1 Control Structure

The control structure for the component proposed in the system design phase is translated into code. The program design structure

must reflex with the control structure design. In this project the coding is done using the bottom-up approach.

6.5.2 Algorithm

The system program code were designed based on a specific algorithm. Algorithm is a detail sequence of actions to perform to accomplish some task. An algorithm must reach a result after a finite number of steps The system were broken into several steps, get the device list for capturing, open the device list for capturing, loop the incoming and outgoing packet for handling, handling the packet for analysis, separate the packet based on the protocol and display the packet. This sequence of steps needed to solve logical problems.

6.5.3 Object Oriented Programming

Object Oriented Programming supports object technology. It is an evolutionary form of modular programming with more formal rules that allow pieces of software to be reused and interchanged between programs. OOP is thought to increase productivity by allowing programmers to focus on higher-level software objects. One of primary features of object-oriented is inheritance. In this project, the system's program code was written in two java source files: SniffBase.java and Sniffa.java. The device handling and packet handling process is placed in Sniffa.java, the derived class. While the SniffBase.java is the base class that extends the method of set() and get(). Sniffa.java inherits this method to display the GUI and packet capturing data and statistics.

6.6 Program Coding Approach

Factors to be taken into account when doing system coding:

6.6.1 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

6.6.2 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.

6.6.3 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.

6.6.4 Module

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

6.7 System Module

The system's module is divided into five: obtaining device/adapter list, opening adapter, capturing packet, display packet and statistics.

6.7.1 Obtaining device list

The system obtain the device or adapter list by using method getDeviceDescription() in jpcap.

6.7.2 Opening Adapter

The system will open the device for listening using method openDevice() in jpcap. The user will choose which adapter they want to use to listen to the network. The openDevice() method requires four arguments: the device name to be opened, the maximum number of bytes to read from the device at one time, a Boolean value specifying whether to put the device into promiscuous mode, and a timeout value

6.7.3 Capture Packet

After the device is open, the system will start listening through loopPacket() and start capturing packet. loopPacket() will capture packets until the maximum number of packets is reached specified in openDevice().

6.7.4 Display packet and packet summary

handlePacket() method is used to analyze a packet. This method is called every time a packet is captured. Packet strings and statistics will be processed here and send to the output area for display.

6.8 Program Coding

6.8.1 Coding Style

There are two standard methods of program design: the top-down approach and the bottom-up approach.

- Top-down programming involves writing code that calls functions that haven't defined and working through the general algorithm before writing the functions that do the processing. Top-down programming is, to a good degree, a very abstract way of writing code because it starts out by using functions that haven't been designed.
- The bottom-up approach to programming is the opposite: writes the basic functions then work up to the more complex parts of the program.

It's interesting that both of these approaches focus on the actions of the program rather than the objects the program manipulates - variables. Many times, the best way to write a program is to figure out the variables that need to work. By defining variables first and then working with functions that work on them, this always maintain a basic foundation of what the program should be doing. Finally, the code for each individual function is written.

6.8.2 Debug mechanism

Errors caused by faulty logic and coding mistakes are referred to as bugs. Finding and correcting these mistakes and errors that prevent the program from running and producing correct output is called debugging. Some common mistakes which cause program bugs are: mistakes in coding punctuation, incorrect operation codes, transposed characters, keying errors and failure to provide a sequence of

instructions needed to process certain conditions. The way of debugging the program code:

6.8.2.1 Runtime error

The program does something, but not as expected – a great way to make sure the code is getting executed.

6.8.2.2 Debugger

Debugging is the process of correcting or modifying the code in the program so that the program can build, run smoothly, act as expected and be easy to maintain later.

6.8.2.3Exception Handling

An exception is an indication of a problem that occurs during a program's execution. Exception handling enables the creation of application that can resolve or handle exception. Handling exception allows a program to continue executing as if no problem has been encountered.

CHAPTER 7
SYSTEM TESTING

CHAPTER 7 – SYSTEM TESTING

The main function of testing is to establish the pressure of defect. Testing is performed to ensure that it is working correctly and efficiently. and generally focused on two areas: internal efficiency and external effectiveness. The goal of external effectiveness testing is to verify that the system is functioning according to system design, and that it is performing all necessary functions or sub-functions. The goal of internal testing is to make sure that the computer code is efficient, standardized, and well documented. Testing can be a labor-intensive process, due to its iterative nature.

After system has been verified, it needs to be thoroughly tested to ensure that every component of the system is operating as it should and the system is performing exactly in accordance with the requirements.

7.1 Testing Methodology

There are two main methodologies of testing: white-box and black-box testing.

7.1.1 White-box testing examines the internal structure of a program and attempts to test each logical case. White-box testing can be thought of as transparent box testing: the tester can see and test a specific section of code.

7.1.2 Black-box testing also known input/output-driven testing in which the tester views the program as a black box, and as such, the inner workings of the program are unknown. The main tool used in black-box testing is the specification of the program: attempts to determine

what input causes the output of the program to be different from what the specifications would require.

7.2 Type of Testing

7.2.1 Module Testing

It is also referred to unit testing and it focuses on verification of the smallest unit of system design - the module. Using the detailed design specification as a guide, important control paths are tested to uncover errors within the boundary of the module.

Module testing were done on

- Obtaining device module

  to ensure that device is open and argument is passed correctly to the receiving function.

- Packet handling module

  to ensure that packets were actually captured and send to the appropriate method for handling and display purposes.

- Packet display module

  To ensure that the intended output were display correctly and at the correct output area.

Types of error occurred during module testing:

- **Algorithm error** - error in the assembly of program code results in the output display area
- **Syntax error** - innocent mistakes during keying in the program code.
- **Parameter passing error** - Data type of argument passed were different from the argument in method().

- **Event handling error** – Calling method from the inner class without declaring a new object

7.2.2 Integration Testing

Testing two or more modules or functions together with the intent of finding interface defects between the modules or functions. Testing completed at as a part of unit or functional testing, and sometimes, becomes its own standalone test phase. Integration testing can involve a putting together of groups of modules and functions with the goal of completing and verifying that the system meets the system requirements.

7.2.3 System Testing

It ensures that the system as a whole satisfies input and output specifications and that interfaces between modules, programs or subsystems are correct. Emphasis is placed on system access, security, performance, and recovery capabilities.

The modules tested in the integration tested were tested again as a complete system. The system testing will verify the accuracy of the systems process, input and output to ensure it follows the design specification and the system's requirement.

CHAPTER 8
SYSTEM EVALUATION AND DISCUSSION

CHAPTER 8 – SYSTEM EVALUATION AND DISCUSSION

This chapter will cover and discuss the system strength and limitation, the system's problem and solution, and a few suggestions to enhance the system in the future.

8.1 System Strength

The system developed, packet sniffer has achieved its main objective. The packet sniffer is used to listen to a live network and able to capture for analyzing purposes. Packet can be capture under promiscuous mode either from a dial-up connection and switched network. Throughout the development, the system was programmed and tested using Dell computer. Device adapter that were for capturing is Generic NdisWan Adapter and 3Com 3C90x Ethernet Adapter.

Generic NdisWan Adapter is used to listen to the network through a modem while 3Com 3C90x Ethernet Adapter is used to listen to network though a switched network.

Packet summary produced by the system displays the total of packets captured, TCP and UDP packet and total length of byte captured.

Most importantly the system was successfully built to capture packet from a live network and the function as packet sniffer.

This system uses jpcap facilities manipulated by the author based on winpcap packet capture library. The open source code can be retrieved freely and any programmer or user is allowed to change the coding and adding new modules to the code.

8.2 Systems Limitation and Constraint

The system does not produce relevant results to help the user analyze the packets captured from the network but it s capable of capturing the packet's raw data and information in the packet field.

The system was not able to decode packet header and data into a human readable form for the purpose of analyzing. This is a major drawback for the system.

The systems filter were setup inside the programming code. The code was program to capture TCP, UDP and ARP packets. The user does not have the ability to filter packet based on protocol or IP address.

The system can only display packet information in a form of raw data or string.

Packet summary presented by the system have restricted information on the packet detail. The statistics obtained were not enough to perform packet analyze function.

8.3 Problem and solution

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

8.3.1 Lack of programming experience

Problem:

It is a major drawback during the implementation of the project. The initial choice of programming language, C++ has been changed to Java. This decision was made because of the vast amount of classes and the complexity of event handling structure in C++.

Solution:

As time was the main factor, Java was chosen due to its less complicated programming style. Nevertheless, a lack of experience and skills in programming has been the major obstacle from achieving this project's objective.

8.3.2 Development Time Factor

Problem:

Small prior knowledge in Java programming style and environment, a lot of studies need to be done and learn within a short span of time. Due to this factor also, certain features is not implemented in this project.

Solution:

However some of the obstacles were resolved by doing personal studies and research through the Internet.

8.3.3 Installation and Setting

Problem:

Lack of experience, knowledge and skills in developing a system has turn out to be a major obstacle during the system development phase.

Installation and compilation of winpcap and jpcap were not done correctly during the early stage of development.

Solution:

Problem was overcome through a lot of research in the Internet, engagement in forum and documents available in the setup folder.

8.4 Future 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:

- Create a graphical user interface that could interact more with the user.
- Setting up packet filters so that the user can capture packets based on protocol, ip address or port number.
- To create more item in the packet summary such as total of received and dropped packets, statistics and so on.
- Statistic in a form of pie chart or graph. The graph and chart will represent then number of packets captured according to the protocol.
- Decode the packet data field into hexadecimal or ASCII for the purpose of analyzing the nature of the data packet.
- Implement security measure such as authentication through the use of password.

Jpcap provide a wide facility than can be used to build a better system. The packages inside jpcap facility can be manipulated to produce better functions and modules in the system.

Finally, the system can only display packets after the time passed to the getOpenDevice() method is out. It does display packet in real time.

CONCLUSION

After conducting analysis and testing, it is concluded that the project has achieved its main objective, capturing a packet from live network 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 simple packet sniffer, how the packets were captured from the live network and a lot more.

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.

APPENDIX A

APPENDIX A - USER MANUAL

A1. DOWNLOAD

No.	Platform and tools
1.	Java 2 Platform, Std Edition, v 1.4.2 (J2SE) http://java.sun.com
2.	JCreator LE 3.5 www.jcreator.com/download.htm
3.	Jpcap http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html
4.	Winpcap 3.1 beta 4 http://winpcap.polito.it/
5.	Miscorsoft Visual C++ 6.0 Compiler

A2. SETUP

1.	**Jpcap ver.0.4**
	1. Copy "lib\Jpcap.dll" into "[J2SDK directory]\bin" or "[JRE directory]\lib\ext\x86" 2. Copy "lib\jpcap.jar" into "[J2SDK directory]\lib\ext"
2.	**Winpcap 3.8 beta 4**
	1. Compile packet.dll Load the project contained in the directory PacketNTx\dll\project (or Packet9x\dll\project) in the Visual C++ IDE. There are four project configurations: • *PacketNT - Win32 Release*: standard release configuration • *PacketNT - Win32 Debug*: standard debug configuration • *PacketNT - Win32 NT4 Release*: release configuration able to run on NT4 does not include Wan and IP helper API support. • *PacketNT - Win32 NT4 Debug*: debug configuration able to run on NT4 does not include Wan and IP helper API support. • *WanPacket - Win32 Release*: release version of the WanPacket library, used to interface with NetMon API for Wan capture • *WanPacket - Win32 Debug*: debug version of the WanPacket library, used to interface with NetMon API for Wan capture

	2.	Compile wpcap.dll - *winpcap\wpcap\prj* of the WinPcap source code distribution. Load wpcap.dsw from the Visual C++ IDE and build the program. There are six build project configurations: • *Wpcap debug*: no support for DAG cards and Remote capture, debug version • *Wpcap release*: no support for DAG cards and Remote capture, release version • *Wpcap debug REMOTE*: support for Remote capture, no support for DAG cards, debug version • *Wpcap release REMOTE*: support for Remote capture, no support for DAG cards, release version • *Wpcap debug REMOTE DAG*: support for both DAG cards and Remote capture, debug version • *Wpcap release REMOTE DAG*: support for both DAG cards and Remote capture, release version
3.	**Java 2 Platform, Standard Edition, v 1.4.2 (J2SE)**	
	After installation, setup the environment variables : For Windows XP : Control Panel/Systems/Advanced Go to Environment Variable/User variables : include path and type C:\j2sdk (according to the installation directory)	

A3. PROJECT EXECUTION

Open, compile and build the java source file SniffBase.java and Sniffa.java.

Execute the Sniffa.java and the packet will automatically start capturing packets.

APPENDIX B

APPENDIX B – SYSTEM CODE

B.1 SniffaBase.java

```
import java.lang.Number;

import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import jpcap.*;

public class SniffBase// implements JpcapHandler
{
        public String output1="";
        public String output2="";
        int h=0;
        int i=0;
        int j=0;
        int k=0;
        int l=0;
        long m=0;
        int n=0;
        int o=0;


    public SniffBase()
    {}//end const. SniffBase

    public void setDevice(String out)
    {
        output1+=out;
    }

    public String getDevice()
    {
        return output1;
    }

    public void setDeviceName(String g)
    {
        output2+=g;
    }
    public String getDeviceName()
    {
        return "  Start sniffing on device " + output2;
    }
```

```
        public void setPacket(Packet p)
        {       h++;    }

        public int getPacket()
        {       return h;       }

        public void setTcp(Packet p)
        {       i++;    }

        public int getTcp()
        {       return i;       }

        public void setUdp(Packet p)
        {       j++;    }

        public int getUdp()
        {       return j;       }

        public void setIp(Packet p)
        {       k++;    }

        public int getIp()
        {       return k;       }

        public void setArp(Packet p)
        {       l++;    }

        public int getArp()
        {       return l;       }

        public void setLen(int p,int o)
        {       m+=p; n+=o;
                }

        public long getLen()
        {       return m;       }
        public long getcapLen()
        {       return n;       }

}//end class SniffBase
```

B.2 Sniffa.java

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import jpcap.*;
import java.io.* ;

public class Sniffa extends JFrame implements JpcapHandler
{
        Box box = Box.createVerticalBox();
        Container container;

        JTextField textField1,textField2;
        JTextArea textArea1, textArea2, textArea3,textArea4,textArea5;
        JTextArea editor;
        JMenu fileMenu, helpMenu;
        JMenuBar setMenuBar;
        JMenuItem
exitMenuItem,aboutMenuItem,saveMenuItem,copyMenuItem,pasteMenuItem;

        public static String area1="";
        public static String area2="";
        public static String area3="";
        public static String area4="";
        public static String output="";
        public static String output1="";
        public static String outpacket="";
        public static String[] device;

        public Sniffa()
        {
                super("Sniffalyzer");

                /*****************FILE MENU*****************/

                JMenu fileMenu = new JMenu("File");
                fileMenu.setMnemonic('F');

                JMenuItem saveMenuItem = new JMenuItem("Save");
                saveMenuItem.setMnemonic('S');
                saveMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
                                ActionEvent.CTRL_MASK));

                fileMenu.add(saveMenuItem);
                saveMenuItem.addActionListener(
```

```
            new java.awt.event.ActionListener()
            {
            public void actionPerformed(ActionEvent e)
            {       saveFile() ;    }
    }
) ;

        editor = new JTextArea () ;
        editor.setColumns(40) ;

        JMenuItem copyMenuItem = new JMenuItem("Copy") ;
        copyMenuItem.setMnemonic('C');
        copyMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C,
                        ActionEvent.CTRL_MASK));
        fileMenu.add(copyMenuItem);
        copyMenuItem.addActionListener(
        new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e)
        {       copy() ;        }
        }
        ) ;


        JMenuItem pasteMenuItem = new JMenuItem("Paste") ;
        pasteMenuItem.setMnemonic('P');
        pasteMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_P,
                        ActionEvent.CTRL_MASK));
        fileMenu.add(pasteMenuItem);
        pasteMenuItem.addActionListener(
        new java.awt.event.ActionListener(){
        public void actionPerformed(ActionEvent e)
        {       paste() ;       }
        }
        ) ;
        fileMenu.addSeparator();

        JMenuItem exitMenuItem = new JMenuItem("Exit");
        exitMenuItem.setMnemonic('E');
        exitMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_E,
                        ActionEvent.CTRL_MASK));
        fileMenu.add(exitMenuItem);
        exitMenuItem.addActionListener(
                new ActionListener(){//inner
                        public void actionPerformed(ActionEvent event)
                        {       System.exit(0);                 }
```

```
        }//end inner
);//listener

/**********************HELP MENU**********************/
JMenu helpMenu = new JMenu("Help");
helpMenu.setMnemonic('H');

JMenuItem aboutMenuItem = new JMenuItem("About");
aboutMenuItem.setMnemonic('A');
helpMenu.add(aboutMenuItem);
aboutMenuItem.addActionListener(
        new ActionListener(){//inner
                public void actionPerformed(ActionEvent event)
                {
                        JOptionPane.showMessageDialog(Sniffa.this,
                        "\tThis is a simple java packet sniffer",
                        "About",JOptionPane.PLAIN_MESSAGE);
                }
        }//end inner
);//listener

JMenuBar bar = new JMenuBar();
setJMenuBar(bar);
bar.add(fileMenu);
bar.add(helpMenu);

Container container = getContentPane();
container.setLayout(new FlowLayout());

JTextArea textArea1 = new JTextArea("",4,10);
textArea1.setBackground(Color.BLUE);
textArea1.setForeground(Color.WHITE);
textArea1.setFont(new Font("Arial-Narrow", Font.BOLD, 12));
box.add(new JScrollPane(textArea1));

JTextArea textArea2 = new JTextArea("",2,30);
textArea2.setBackground(Color.BLACK);
textArea2.setForeground(Color.WHITE);
textArea2.setFont(new Font("Arial-Narrow", Font.BOLD, 12));
box.add(new JScrollPane(textArea2));

JTextArea textArea3 = new JTextArea("",1,60);
textArea3.setText(
        "    Timestamp\t\tSource\t      Destination\t  Information");
textArea3.setFont(new Font("Arial-Narrow", Font.BOLD, 10));
```

```
        textArea3.setEditable(false);
        box.add(new JScrollPane(textArea3));

        JTextArea textArea4 = new JTextArea("",15,60);
        textArea4.setBackground(Color.PINK);
        textArea4.setForeground(Color.BLACK);
        textArea4.setFont(new Font("Arial-Narrow", Font.BOLD, 12));
        box.add(new JScrollPane(textArea4));

        JTextArea textArea5 = new JTextArea("",1,60);
        textArea5.setText("  PACKET SUMMARY");
        textArea5.setFont(new Font("Arial-Narrow", Font.BOLD, 12));
        textArea5.setEditable(false);
        box.add(new JScrollPane(textArea5));

        JTextArea textArea6 = new JTextArea(":",7,60);
        textArea6.setBackground(Color.CYAN);
        textArea6.setForeground(Color.BLACK);
        textArea6.setFont(new Font("Arial-Narrow", Font.BOLD, 12));

        box.add(new JScrollPane(textArea6));


        textArea1.setText(area1);
        textArea2.setText(area2);
        textArea4.setText(area3);
        textArea6.setText(area4);

        container.add(box);
        setVisible(true);
        setSize(600,1000);

}//end const. Sniffa

/***************PACKET HANDLING***************/

        public void handlePacket(Packet packet)
        {
                i+=1;
                SniffBase f = new SniffBase();

        outpacket+=    " "+i+". "+packet+"\n";

        if(packet instanceof Packet)
        f.setPacket(packet);
```

```
if(packet instanceof IPPacket)
f.setIp(packet);

if(packet instanceof TCPPacket)
{f.setTcp(packet);}

if(packet instanceof UDPPacket)
f.setUdp(packet);

if(packet instanceof ARPPacket)
f.setArp(packet);

f.setLen(packet.len,packet.caplen);

}//end handlePacket()


public static void main(String args[]) throws java.io.IOException
{

SniffBase s = new SniffBase();

/**********DISPLAY DEVICE**********/

String[] device1= Jpcap.getDeviceDescription();
for(int i=0; i<device1.length; i++){
output1+="  "+(i+1)+". "+device1[i]+"\n";

}

s.setDevice(output1);
area1+=s.getDevice();

String deviceName1=device1[0];
s.setDeviceName(deviceName1);
area2+=s.getDeviceName();


/**********OPEN DEVICE**********/
String[] device = Jpcap.getDeviceList();
for(int i=0; i<device.length; i++){
output+=(i+1)+". "+device[i]+"\n";

}
```

```
        String deviceName = device[1];

      Jpcap jpcap = Jpcap.openDevice(deviceName, 1028, true, 1);
      jpcap.loopPacket(20, new Sniffa());

/**************DISPLAY PACKET****************/

area3+=outpacket;

area4+=  "  Traffic\t\tCapture\n"+
         "  ---------------------------------------------------\n"+
         "  Packets\t\t"  +        s.getPacket() + "\n" +
         "  IP:" +s.getIp()+"\t\t"  + "TCP: "+s.getTcp()+" UDP: "+s.getUdp()+ "\n"
         +
         "  ARP    \t\t"+s.getArp()        + "\n" +
         "  Bytes  \t\t"+s.getLen()        +       "\n";

         Sniffa sniff = new Sniffa();
         sniff.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   }//end main


   /**************FILE-SAVE-COPY-PASTE************/

   private void saveFile() {

   JFileChooser fc = new JFileChooser () ;
   int returnVal = fc.showSaveDialog(this) ;
   if (returnVal == JFileChooser.APPROVE_OPTION) {
         File file = fc.getSelectedFile() ;
               try     {
                     editor.write(new FileWriter(file)) ;
                     }
               catch (IOException exp) {}
   }
   }//end save

   private void copy() {

   editor.copy() ;
   editor.requestFocus() ;
   }//end copy

   private void paste() {
```

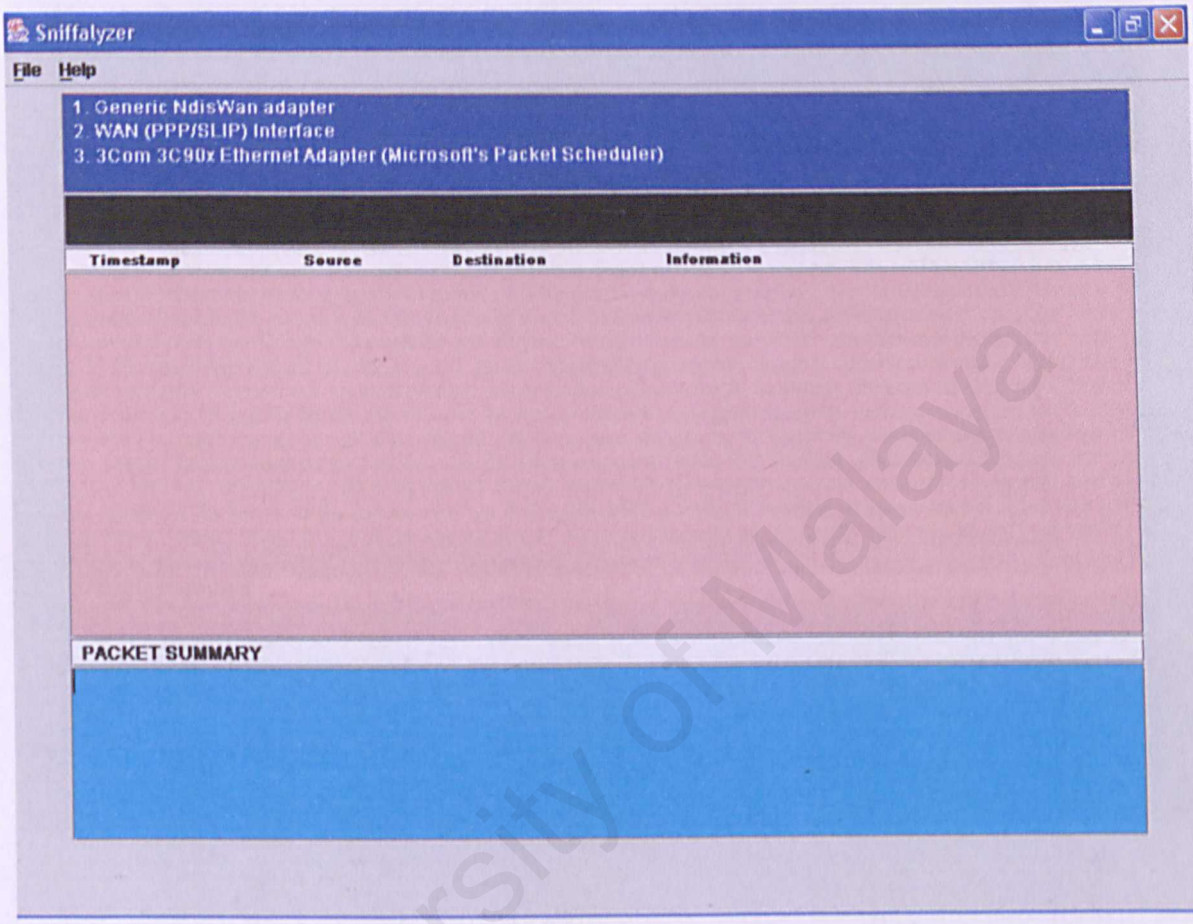
```
        editor.paste() ;
        editor.requestFocus() ;
        }//end paste

}//end class Sniffa
```

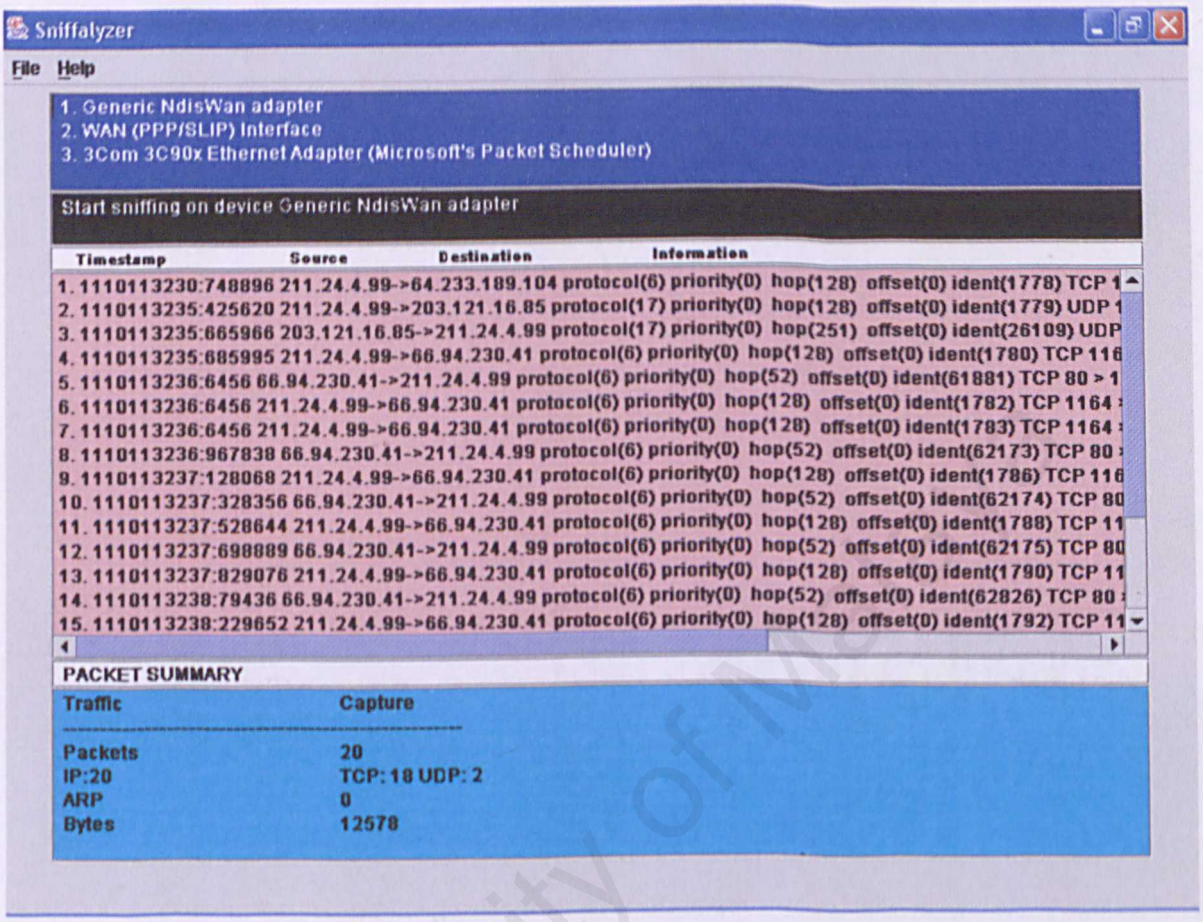
APPENDIX C

APPENDIX C – SYSTEM INTERFACE

C.1 THE SYSTEM'S INTERFACE BEFORE CAPTURING



C.2 THE SYSTEM'S INTERFACE AFTER CAPTURING



REFERENCE

REFERENCE

[1] Forouzan, A.B. (2001). *Data Communications and Networking.* 2nd ed. McGraw Hill.

[2] *Introducing Network Analysis* (2004). (URL-http://www.syngress.com/book_catalog/284_eps/sample.pdf),24/8/2004

[3] Myrom, E.S. (2000). *Data Communications* 4th ed.

[4] Cisco Systems, Inc. (2001). *Cisco Networking Academy Program: Second Year Companion Guide.* 2nd ed. Cisco Systems, Inc.

[5] Kesler, G.C. (2001). *The Power of Packet Filtering.* Windows NT Magazine

[6] EffecTech. (2004). (URL - http://www.effetech.com/help/cisco-span.htm), 15/7/2004)

[7] [8] (URL - http://www.support.psi.com/), 22/8/2004

[9][10] Chappell, L. (2002). *Packet Filtering: Capturing The Cool Packets.* 1st ed. Podbooks.com LIc

[11] Schinder, D.(2004). *Firewalls and VPNS.* http://www.windowsecurity.com/articles/Application_Layer_Filtering.html

[12] Blue Blazer (2004). *What is Unix.* http://bluelazer.netfirms.com/unixlinux.htm

[13] Microsoft Corporation. (2004). *Windows XP Professional Features.* http://www.microsoft.com/windowsxp/pro/evaluation/features.mspx

[14] Deitel, H.M & Deitel, P.J. (2003). *Java How To Program.* 5th ed. Prentice Hall

[15] Deitel, H.M & Deitel, P.J. (2000). *C How To Program.* 3rd ed. Prentice Hall

[16] (URL - www.webopedia.com), 1/9/2004

[17] ISONET Limited Company. (2004). *Microsoft Visual C++ 6.0* http://www.isonet.co.th/2/product/microsoft/c_60pro.htm.

[18] Focus On Unix (2004). *Cygwin – Unix Within Windows.* http://unix.about.com/library/

[19][20] Whitten, J.L., Bentley, L.D. & Dittman K.C. (2002). *Systems Analysis and Design Methods.* 5th ed. McGraw Hill

[21] Han, L.C. (2003). *Network Traffic Monitoring Systems.* Thesis. University of Malaya

[22] Smart, J. (2004). *What is wxWidgets.* www.wxwidgets.org

[23] Lawrence Berkeley National Laboratory. (2004). *WinPcap: The Free packet Architecture For Windows.* http://winpcap.polito.it/

[24] Sommerville, I. (2001). *Software Engineering.* 6th ed. Edison Wesley

[25] (URL - http://www.cs.uwaterloo.ca/~tmjvasig/CS134Testing.html), 22/1/2004

[26] SourceForge. (2005).*Network Capture Library For Java*). http://sourceforge.net/projects/jpcap

[27] Mustapha, Julia. (2003). *Network Game Monopoly.* Thesis. University of Malaya

[28] Kuen, C.M. (2002). *Network Game.* Thesis. University of Malaya