FACULTY OF COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY,
UNIVERSITY MALAYA.

SESSION 2004/2005

NETWORK SCANNER

ANISYAHAYATI ISMAIL (WEK020012)

Under the Supervision of

MR. NOR BADRUL ANUAR BIN JUMA'AT

Moderator

MR. LING TECK CHAW

This project is submitted to the Faculty of Computer Science & Information Technology, University of Malaya in partial fulfillment of requirement of the Bachelor of Computer Science.

ABSTRACT

This report is submitted for WXES3182, Latihan Ilmiah Tahap Akhir which is required for final year students of the Faculty of Computer Science and Information Technology who will be graduating.

The Network Scanner is a program that is developed to provide information on the network vulnerabilities especially open service port so that network security actions can be performed earlier. A lot of researches have been done in order for me to understand the basics of network scanning and what information it provides.

From the researches, I have gained a lot of information regarding the purpose of the network scanner, its functions, techniques that have been used in network scanning and a little insight of the design process of a network scanner. Finally I have completed developing the Network Scanner project.

Although this project is not much compared to the current Network Scanners in the market, I still hope that this Network Scanner will bring knowledge to other people like it did to me.

This course should be maintained for students as it actually encourage students to implement the knowledge they have acquired during the minimum of six semesters of studies in the faculty.

ACKNOWLEDGEMENTS

Heaps of gratitude towards Allah for without His blessings I would never be able to complete the Final Year Project report.

A special thank you is extended to Mr. Nor Badrul Anuar Bin Jumaat and Mr. Ling Teck Chaw for your encouragement, patience and all your time spent providing me guides and resource in achieving this report.

And to my loving family, my father, mother and sisters, thank you all very much for your support, love and care over the years. Without your support, life would have been miserable.

This special thank you is also extended to the staff of the Faculty for giving support to students who are doing the thesis project for this semester and all other semesters that have passed and will be coming.

And last but not least, a heart warming thank you to my supporting partner and friends for without their encouragement and their share of ideas and thoughts also I may not come to completing this report.

Anisyahayati Ismail,

Faculty of Computer Science and Information Technology.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

The Network Scanner project is the project that I have selected to be conducted for my final year project, WXES3181/3182. This introduction chapter will describe in details about the definition of the project nature, its purpose, goals and scope definitions. This chapter marks the initiation of the development of the project inline with the objective and scope so that it will not differ from the original purpose.

1.1 OVERVIEW

In the last few years, the face of computing changed dramatically. Computer networking and the internet gives a whole new meaning to computing. In just a few years, we are now in a wired and wireless world where information of any type is accessible from anywhere and by anyone all over the world. Back in those years, computers were just tools to assists in complex computing and in daily task. Now, computer is one of the most important sources of information over the internet. Computer networking has made communications so much easier. Thus, more and more businesses and organization nowadays began to rely heavily on networked system in order to conduct their businesses and increase their productivity through resource sharing. However, when more businesses and organization join the internet, the amount valuable information increases on the internet. This will actually build up the potential of information thefts and damage, which is why network security now plays a more vital role than ever before.

1.2 PROJECT DEFINITION

Since security is the main issue here, there need to be ways to overcome the security problems and maintains security over the network. One of the solutions includes the implementation of a security tool or application on the network. These applications are also called security tools. The Network Scanner itself is actually a security tool that is developed to scan the computing systems in a network in order to identify vulnerabilities of the system such as open service ports that could be exploited or would become a threat to the whole network if no proper action was taken. Network Scanner is a very useful tool that can be employed as an information provider to tighten up the security of a network. However, to a potential attacker, the Network Scanner can be a tool for information gathering to prepare for a likely successful attack.

1.3 OBJECTIVE

The objectives of the Network Scanner project are:

i) To learn and understand the design purpose of a network scanner.

Before going through the project, I must first introduce myself to the project which is the network scanner software. Most of the information was obtained from the internet. I also tried different network scanner products such as GFI LANguard Network Security Scanner, SoftPerfect Network Scanner and Advanced LAN Scanner to actually see their functions to give me a picture of what I should develop. I have also searched for other materials on network scanner at the library.

ii) To gain knowledge and understanding of the network architecture and the current existing network scanning techniques.

To achieve this objective, I have taken the steps to do a literature review on the networks scanner and also the overview on computer networking, network components and network security which is the main reason why network scanner existed. I have also done a review on current techniques used to scan networks and made comparisons between those techniques and the tools used so that I can select the most suitable and applicable technique to be implemented in the system.

iii) To define the requirements of a network scanner and finding the most suitable development methodology.

The requirements for the network scanner can be obtained from the library, the internet and from past theses. This includes the user requirements, what the system should do and what the system should produce from the tasks and functions assigned.

iv) To design modules for network scanning techniques that was selected for implementation.

Different modules are to be discovered in developing the system. I will be relying highly on the source available on the internet so that I can further understand the techniques involved in network scanning by slowly understanding the concepts of data flow and distribution in each of the known techniques.

v) To promote the use of open source programming tools in the translation of the network scanning system concept into a software representation that is understood by the computer.

This is where all the modules that have been discovered previously are compiled as one software system called the 'Network Scanner' using open source tools with C++ programming language I have been studying before.

1.4 PROJECT SCOPE

The scope of this project is divided into three different groups: the environment in which the Network Scanner is intended for use, target users and usage time.

1.4.1 Environment

The Network Scanner is developed to monitor machines in the network. Network here refers to Local Area Networks (LANs) which normally belong to companies or organisations.

1.4.2 Target Users

Generally the Network Scanner is intended for network engineers and network administrators to assist in their daily tasks. But the software is also intended for users who are interested in computer and network security. This will allow the achievement of the software aim to help in research and studies in security field.

1.4.3 Usage Time

In real time, the network usage has no limited time. A user can be online for hours or even days. The Network Scanner is designed to scan the network for vulnerabilities at one time. However, scheduled scans should be able to provide the network administrators the condition of the network from time to time and allow preventive actions to be performed in time.

1.5 PROJECT EXPECTATIONS

It is hoped that the project will fulfil the requirements of network systems and be of assistance in the development of the information technologies in the near future. This can be achieved through constant studies on network vulnerabilities identified through the usage of this software.

1.6 PROJECT TIMELINE

The Gantt chart shown below reflects the Network Scanner Project time line beginning from 12th July 2004 until 4th March 2005.

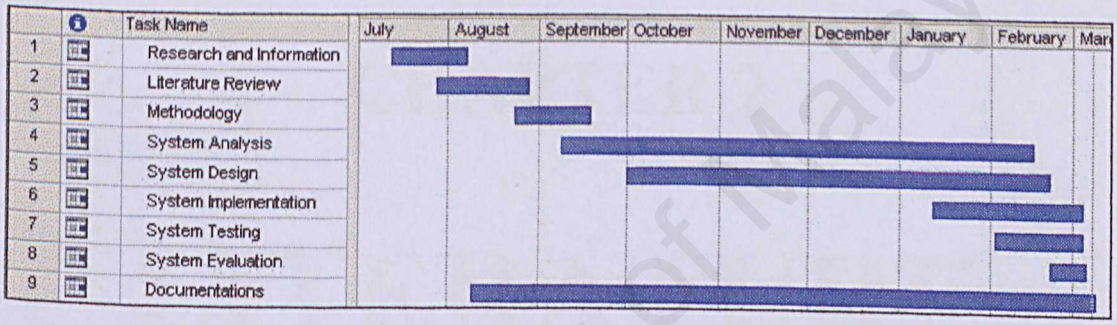

Figure 1.1: Gantt Chart: Network Scanner Project Timeline

CHAPTER 2

LITERATURE REVIEW

CHAPTER 2

LITERATURE REVIEW

A literature review for this project is a comprehensive survey of publications and other materials in the specific field of study or related to the Network Scanner project and also other matters and factors that are related to the project development. Literature reviews can be used to identify the requirements that can be implemented during the project development.

2.1 COMPUTER NETWORK

A network is a group of two or more computer systems linked together (Webopedia, 2004). A computer network allows communications and resource sharing between devices. There are several types of computer network, however the most widely used and popular are LANs and WANs.

2.1.1 Local Area Networks (LANs)

A local area network (LAN) is a computer network that covers a local area such as a home, office, or a small group of buildings like schools and college (Wikipedia, 2004).

Most LANs connect workstations and personal computers. Usually each *node* (individual computer) in a LAN is able to access data and devices anywhere on the LAN. This means that users can share expensive devices, such as laser printers, as well as data and information. Users on the LAN can communicate with each other, by sending e-mail or engaging in chat sessions.

The most popular LAN technologies are the Ethernet and Token Ring for PC and Apple's Apple Talk for Macintosh computers. One of the commonly used protocols on LANs is the Transfer Control Protocol/Internet Protocol (TCP/P).

LANs are capable of transmitting data at very fast rates, much faster than data can be transmitted over a telephone line; but the distances are limited, and there is also a limit on the number of computers that can be attached to a single LAN.

2.1.2 Wide Area Networks (WANs)

A Wide Area Network is a computer network that spans a wide geographical area. Typically, a WAN consists of two or more local-area networks (LANs). Many WANs are built for particular organizations. Other WANs are built by Internet Service Providers (ISPs) to provide internet access to organizations' LANs. Computers connected to a wide-area network are often connected through public networks, such as the telephone system. They can also be connected through leased lines or satellites. The best example and also the largest WAN in existence is the Internet (Wikipedia, 2004; Webopedia, 2003).

2.2 NETWORK SECURITY

In computer industry, security is the protection of data from theft, loss or unauthorized access, use or modification. With the constantly evolving nature of the internet it is vital that users continuously protect themselves and their information.

2.2.1 Vulnerability Factors

Increased Usage of Networks

More businesses nowadays rely heavily on networked systems and the internet to conduct business. When more people use the internet, the number of potential victims grows. The more businesses join the internet; more valuable information is at stake which increases the potential for theft and damage.

Always-on Connections

In the need of greater speed and to increase the information carrying capacity, even small or home businesses rely on high bandwidth always-on connection to the internet such as Digital Subscriber Lines (DSL) or cable modems. There are two important characteristic that increase vulnerability:

i) Because the connection is always on, the network or system is always available for potential hackers to access.

ii) Machines or systems with always on connection usually have static or unchanging IP address which makes it an easy target.

Insecure Technology

To cope with the astounding rate of technological change, software engineers nowadays focus more on producing user friendly software but neglects the security aspect of the software. This situation puts the end-user to risk because they were unaware of such insecurity.

Lack of Education

Most of businesses and individuals lacked of information about the threats that existed on the internet which is another reason they are often made as targets.

2.2.2 How Hackers Enter a System

Through Port Scanning

Port scanning is a hacking technique used to check TCP/IP ports to reveal what services are available in order to plan an exploit involving those services, and to determine the operating system of a particular computer. In TCP/IP and UDP networks, a port is an endpoint to a logical connection. The port number identifies what type of port it is. For example, port 80 is used for HTTP traffic. Since a port is a place where information goes into and out of a computer, port scanning identifies open doors to a computer. Port scanning has legitimate uses in managing networks, but port scanning also can be malicious in nature if someone is looking for a weakened access point to break into your computer.

Through Vulnerabilities, Exploits and Bugs

Potential hackers uses flaws in operating systems or software applications to break into a system and do damage. These flaws can be manipulated to be a backdoor to the system without anyone noticing it. A bug is an error or defect in software or hardware that causes a program to malfunction. Often a bug is caused by conflicts in software when applications try to run in tandem.

2.2.3 Types of Attacks

Denial of Service (DoS)

Denial of Service is a type of attack on a network that is designed to bring the network to its knees by flooding it with useless traffic. Many DoS attacks, such as the *Ping of Death* and *Teardrop* attacks, exploit limitations in the TCP/IP protocols. For all known DoS attacks, there are software fixes that system administrators can install to limit the damage caused by the attacks. But, like viruses, new DoS attacks are constantly being dreamed up by hackers.

Viruses and Malicious Codes

Viruses and malicious codes are programs or pieces of codes that is loaded onto your computer without your knowledge and runs against your wishes. Viruses can also replicate themselves. All computer viruses are manmade. A simple virus that can make a copy of itself over and over again is relatively easy to produce. Even such a simple virus is dangerous because it will quickly use all available memory and bring the system to a halt. An even more dangerous type of virus is one capable of transmitting itself across networks and bypassing security systems.

Since 1987, when a virus infected ARPANET, a large network that is used by the Defense Department and many universities, many antivirus programs have become available. These programs periodically check your computer system for the best-known types of viruses.

Trojan Horses

Trojan horses are destructive programs that masquerades as a harmless application. Unlike viruses, Trojan horses do not replicate themselves but they can be just as destructive. One of the most insidious types of Trojan horse is a program that claims to rid your computer of viruses but instead introduces viruses onto your computer.

The term comes from a Greek story of the Trojan War, in which the Greeks give a giant wooden horse to their foes, the Trojans, ostensibly as a peace offering. But after the Trojans drag the horse inside their city walls, Greek soldiers sneak out of the horse's hollow belly and open the city gates, allowing their compatriots to pour in and capture Troy.

Worms

Worms are programs or algorithms that replicates itself over a computer network and usually performs malicious actions, such as using up the computer's resources and possibly shutting the system down. A worm is a special type of virus that can replicate itself and use the system's memory, but cannot attach itself to other programs

2.2.4 Risks Involved

If security is not practiced in computer networks, there are a few risks to be faced. First is the loss of information should be the system was attacked and data thefts that might occur under our consciousness. Another risk involved is the risk of the system becoming a platform to launch attacks on other system once the system is occupied by hackers.

2.2.5 Preventive Measures

Among the preventive measures that can be taken is to evaluate our own computer network vulnerabilities so that appropriate actions could be taken earlier to prevent the vulnerabilities in our system to be manipulated by others. To achieve this objective is by constantly scanning the network for vulnerability using computer network security scanners.

2.3 NETWORK SCANNING

Network scanning is the art of detecting which systems are alive and reachable via the network, and what services they offer, using techniques such as ping sweeps, port scans and operating system identification (Arkin, 1999). Network scanning builds a clearer picture of accessible host and their network services through a process of acquiring the information by connecting to a network and quickly and sequentially transmitting request packets to the target host on the network. Scanning can be compared with a thief checking all the doors and windows of a house he wants to break into. Today the number of automated scanners is constantly increasing, and as a result, more and more attacks are successfully initiated. In order to be better prepared, we need to fully understand the scanning tools and the methods that these tools are using against us.

2.3.1 Network Scanning Techniques

There are several network scanning techniques that are popular and widely used nowadays. Below is a brief discussion of each technique.

i) **PING Sweeps**

PING Sweeps is usually used to determine whether a target IP address is alive or not. There are a few types of scan that gives PING sweep results.

- **ICMP sweeps (ICMP ECHO request)**

ICMP packets can be used to determine whether a target IP address is alive or not by simply sending an ICMP ECHO request packet to the target system. If an ICMP ECHO reply is received, the target is alive. Otherwise the target host is down.

- **Broadcast ICMP**

ICMP ECHO request that is sent to network or broadcast addresses will produce all information that is needed for mapping a target network in a much simpler way. The request is broadcasted to all hosts on the network, and alive hosts will send the ICMP ECHO reply to the sender's source IP after only one or two packets sent.

- **Non-ECHO ICMP**

Many firewalls are configured to block ICMP ECHO traffic, thus the alternative is the non-ECHO requests. An example request that is used for this kind of scanning is the ICMP timestamp request which allows a system to query another for the current time.

- **TCP Sweeps**

With this technique, instead of sending ICMP ECHO request packets, TCP ACK and TCP SYN packets are sent. These packets are used in the TCP connection establishment process called "the three way handshake". It has three segments.

1. A client sends a SYN segment specifying the port number of the server the client wants to connect to, and the client's *initial sequence number* (ISN).
2. If the server's service port is active, it will respond with its own segment containing the server's initial sequence number. The server will also acknowledge the client's SYN by sending ACK with the client's SYN+1. If the port is not active, the server will send a RESET segment which will reset the connection.
3. The client will acknowledge the server's SYN by sending ACK with the server's ISN+1.

However, RESET packet for an IP address can be spoofed by firewalls, so TCP sweeps may not be as reliable.

- **UDP Sweeps** (Also known as UDP Scans)

This method relies on a message initiated by a closed UDP port, the ICMP PORT UNREACHABLE message. If no such message is received after sending the UDP datagram to a UDP port that is to be examined on a targeted system, we may assume the port is opened. However this method is sometimes unreliable because

- Routers can drop UDP packets.
- Many UDP services do not respond when probed.
- Firewalls are usually configured to drop UDP packets (except for DNS).

Port Scanning

Port scanning is a method to determine what services are running or in a listening state on the targeted system by connecting to the TCP and UDP ports of that

system. Port scanning can also help identify the operating system and application in use on the targeted system. Types of port scanning are:

- **TCP connect() scan**

This type of scan uses the basic TCP connection establishment mechanism, the three-way handshake, where the connection is terminated after the full connection establishment process has been completed. The disadvantage is, this kind of scan is easily detected from the target system log.

- **TCP SYN Scan (half open scanning)**

In this type of scan, a full TCP connection is not opened. A SYN packet is sent to initiate the three-way handshake. If RST/ACK is received, it indicates a non-listening port. If SYN/ACK is received, that indicate the port is listening and the connection is immediately tear down by sending a RESET. Because the three-way handshake was not completed, some system may not log these scanning attempts.

- **Stealth Scan**

This type of scan is almost similar to half-open scanning. Stealth scanning is done to pass through filtering rules on the firewall, to avoid from being logged by the targeted system logging mechanisms and to hide from the usual site or network traffic.

- **Proxy Scanning/FTP Bounce Scanning**

This scan can be used to scan TCP ports from a "proxy" FTP server. A control communication connection to an FTP server is made. If there is a writable directory, the FTP server can be requested to send data to ports. If the transfer is successful, the target host is listening on the specified port scanned. Otherwise, a "452 Can't build data connection: Connection refused" message

will be received. This scan is quite slow. Some FTP server disable the "Proxy" feature, but there are still many who do not, making this kind of scanning still available.

- **TCP Reverse Ident Scanning**

By communicating with port 113, the ident (identification) protocol is used to determine the owner username of a particular TCP connection (Johns, 1993). A full TCP connection to the target machine port is needed. This scanning method gives information that helps determine which server is vulnerable.

Current techniques used for port scanning are

- **'Random' Port Scan**

In this method, the probing of ports is randomized to avoid from detection. This is because many commercial intrusion detection system and firewall scans for sequential connection attempts. A port scan is reported when the pattern is matched.

- **Slow Scan**

This technique takes a long period where the scan rate can be as low as 2 packets per day per target. It is because intrusion detecting system can determine if a specific IP tries to port scan the network they are defending. It is done by analyzing the network traffic for a certain amount of time.

- **Fragmentation Scanning**

All IP packet that carry data can be fragmented. Some filtering devices and intrusion detection systems may incorrectly reassemble or completely miss portions of the scan that may make them assume that this was just another segment of traffic that passed through their access list.

- **Decoy**

This type uses spoofed addresses for attacks. It makes it appears to the attacked network or host that the hosts specified as decoys are scanning them as well. Finding the real attacker will be nearly impossible when the intrusion detection system 'think' that the target network is being scanned by all the hosts.

- **Coordinated Scans**

This technique uses multiple IPs to probe a target network, each one probes for certain services on a certain machine in a different time period. Therefore detecting these scans is almost impossible.

Operating System Detection

Usually this is done because many security holes are operating system dependent, so identifying which operating system runs on the target host is important to a hacker. Techniques used to identify operating system include:

- **Banner Grabbing**

Some services can be used to identify an operating system. One of the most notable examples is the TELNET service. Just by telnet-ting to a system that provides the service, looking at the welcome banner will enable the identification of the operating system. Other services that have banners are the mail server.

- **DNS HINFO Record**

The Host information record is a pair of strings identifying the host's hardware type and the operating system. This technique is an old technique that is rarely effective today because administrators avoid using those records.

- **TCP/IP Stack Fingerprinting.**

This technique uses distinct variations in TCP stack implementation to determine the type of the remote operating system. A number of 'specific' TCP packets is sent to the target IP and the response is observed.

Firewalking

Firewalking is a program that uses a traceroute style of packets to scan a firewall and attempt to deduce the rules in place on that firewall. By sending out packets with various time-to-lives and seeing where they die or are refused. Firewalls are tricked into revealing rules.

2.3.2 Reviews on Network Scanning Utilities

There are a lot of network scanning tool available for downloads on the internet. Here are the few tools that I have looked into for literature review.

Nmap

Nmap, the "Network Mapper" is an open source utility for network exploration or security auditing. It is a low-level network port scanner. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services ports they are offering, what operating system and its version they are running, what type of packet filters or firewalls are in use, and dozens of other characteristics. Nmap also supports a number of performance and reliability features such as dynamic delay time calculations, packet timeout and

retransmission, parallel port scanning, detection of down hosts via parallel pings. Nmap also offers flexible target and port specification, decoy scanning, determination of TCP sequence predictability characteristics and output to machine parseable or human readable log files. Nmap runs on most types of computers, and both console and graphical versions are available.

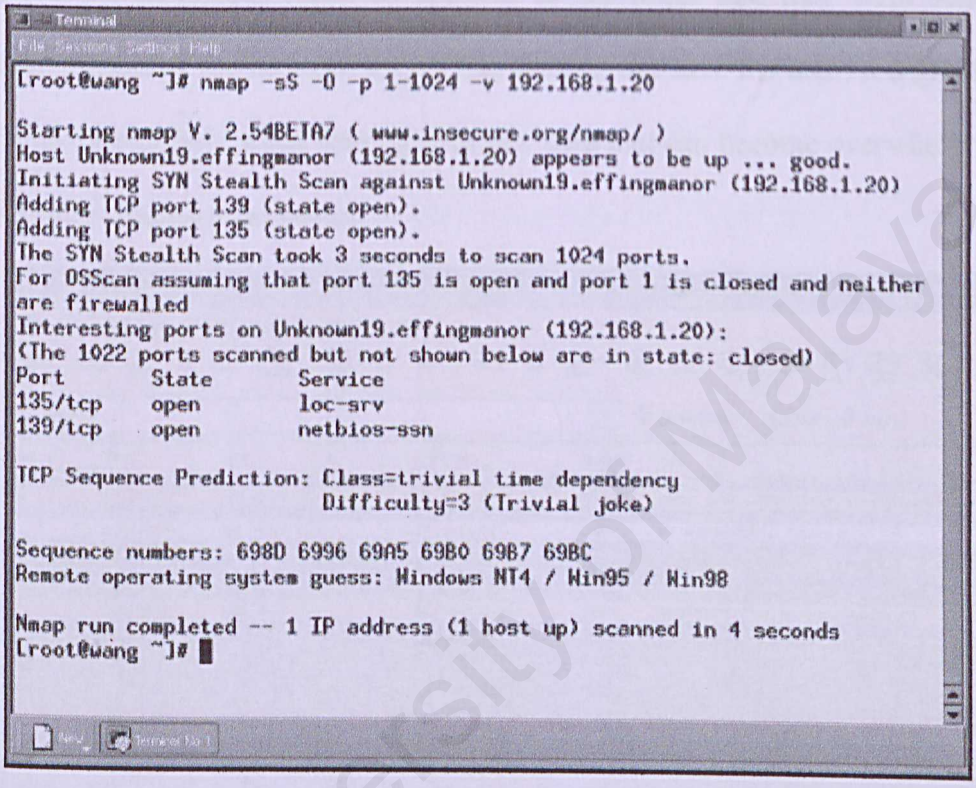

Figure 2.1: Example of text output from Nmap.

Ethereal

Ethereal is a feature-rich protocol analyzer that currently runs on most UNIX platforms and various Windows platforms. It requires GTK+, GLib, libpcap and some other libraries in order to run. It is open source software. Ethereal can read a huge number of different datafile formats including Sniffer, TCPdump, Snoop, Cisco's IDS, and several others. Ethereal supports a large number of physical network media via the host operating system's interface drivers. Ethereal can run

either as a graphic user interface (GUI) or as a text mode interface. Having the text mode interface as a fallback can be tremendously useful, especially in running multiple Ethereal probes in different locations, as it can be very easy to connect to each probe remotely. This is one of the features that sets Ethereal apart from many protocol analyzers. Ethereal can reconstruct a TCP data stream such that all of the packets are in logical order, as opposed to the order that they were sent and received through the network itself. This feature can save the analyst a great deal of time, particularly when dealing with raw data that can become overwhelming if not easily ordered in sequence.

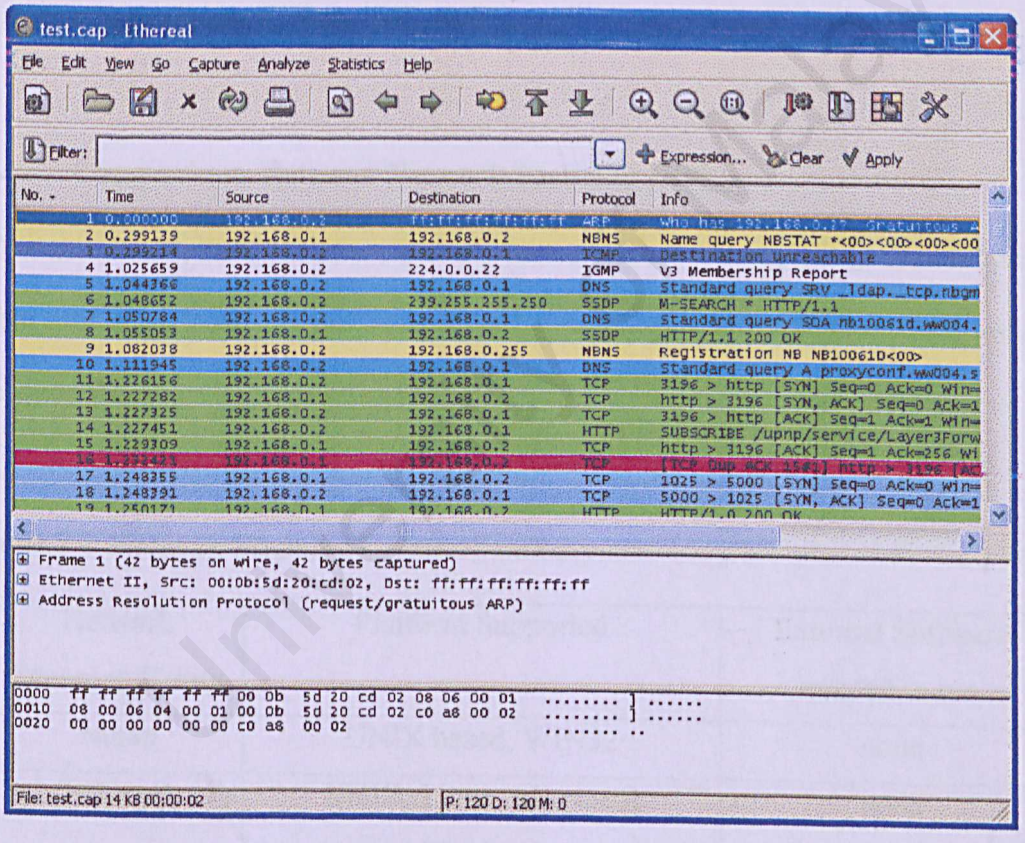

Figure 2.2: Ethereal Screen Shot

Hping

Hping is software to send raw TCP/IP packets of many different kinds and see the reply coming back from the host. Version two was a UNIX command with an

interface very similar to the ping program, but just with many more features: more protocols (not only ICMP), traceroute mode, many options to control different fields of outgoing packets, support for IP spoofing, and so on. Since version 3, that's now in alpha stage, Hping is trying to not be just a little tool but to become a framework for scripting related to TCP/IP testing and security. The latest version of Hping, Hping3, integrates two main new things: the first is an engine called APD that is able to translate simple packet descriptions in form of strings into a packet ready to be sent, and generate the representation from a real packet. The second is the TCL scripting language. So Hping3 can be imagined as a scriptable TCP/IP stack.

2.3.3 Comparisons Between Network Scanning Utilities

The table below summarizes a few of the network scanning utility features that were not mentioned in the review and several comparisons that were made between these network scanning utilities.

Table 2.1: Network Scanning Utilities Summary

Network Scanning Utility	Platform Supported	External Software needed to run
Nmap	UNIX based, WIN32	none
Ethereal	UNIX, Linux, Microsoft Windows	none
Hping	Microsoft Windows, Mac OS	WinPcap for Windows

Table 2.2: Network Scanning Utility Features Comparison

Network Scanning Utility	Captures network packets	Generates and sends packets	Examine network protocols
Nmap	No	Yes	Yes
Ethereal	Yes	No	Yes
Hping	No	Yes	No

2.4 OPEN SOURCE PROGRAMMING

More software are developed using open source programming. Users can use the software, modify and study and redistribute the software freely without worrying to pay royalty to previous developers. Open source eliminate the boundary of programming ideas generation since new open source tool is developed every day.

2.4.1 Programming Language

Any programming language can be used for creating open source software provided that there are tools that allow the types of compilation needed for some software. In this project literature review, I will focus on C++ programming language.

Why C++?

C++ has certain characteristics over other programming languages. Most remarkable are:

- **Object-oriented programming**

The possibility to orientate programming to objects allows the programmer to design applications from a point of view more like a communication between

objects that on a structured sequence of code. In addition it allows the reusability of code in a more logical and productive way.

- **Portability**

The same C++ code can be practically compiled in almost any type of computer and operating system without making changes. C++ is one of the most used and ported to different platforms programming language

- **Brevity**

Code written in C++ is very short in comparison with other languages, since the use of special characters is preferred before key words it is effort saving.

- **Modular programming**

An application's body in C++ can be made up of several source code files that are compiled separately and then linked together. Saving time since it is not needed to recompile the complete application when making a single change but only the file that contains it. In addition, this characteristic allows to link C++ code with code produced in other languages like Assembler or C.

- **C Compatibility**

Any code written in C can easily be included in a C++ program without making changes.

- **Speed**

The resulting code from a C++ compilation is very efficient, due indeed to its duality as high-level and low-level language and to the reduced size of the language itself.

2.4.2 Open Source Tool

There are many open source tool available on the internet. However, for this project, I choose to explain more on WxWidgets.

WxWidgets

WxWidgets is a set of libraries that allows C++ applications to compile and run on several different types of computer, with minimal source code changes. It provides functionality for accessing some commonly-used operating system facilities, from copying and deleting files to socket and thread support. Its main target however is to produce a Graphical User Interface.

CHAPTER 3

METHODOLOGY

CHAPTER 3

METHODOLOGY

System development methodology describes how the development will be conducted. It includes the methods, procedures, and techniques that are used to collect vast information and analyze them. Mainly the system development process model involves system requirements as the input; system development and evaluation as the process and a finished software product as the output.

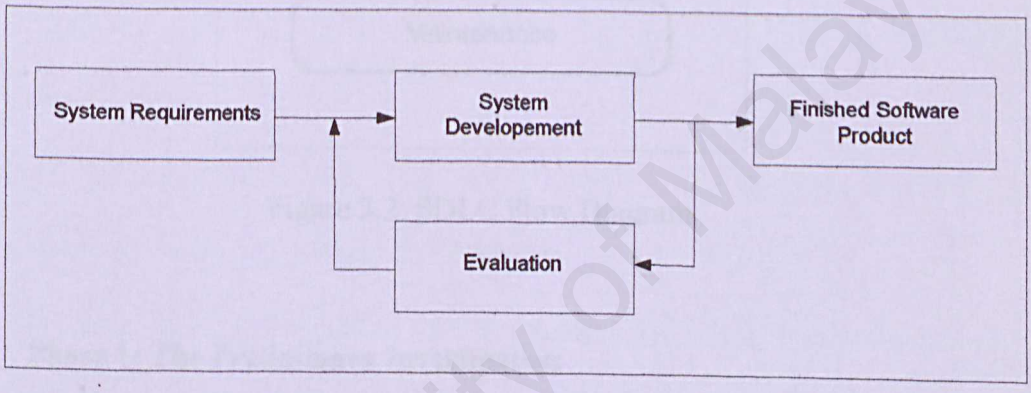

Figure 3.1: System Development Process Model

This chapter will further explain the proposed life cycle model that will be used in the system's development.

3.1 SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

System Development Life Cycle (SDLC) is the process of developing information systems through investigations, analysis, design, implementation and maintenance. SDLC are usually made up of six phases:

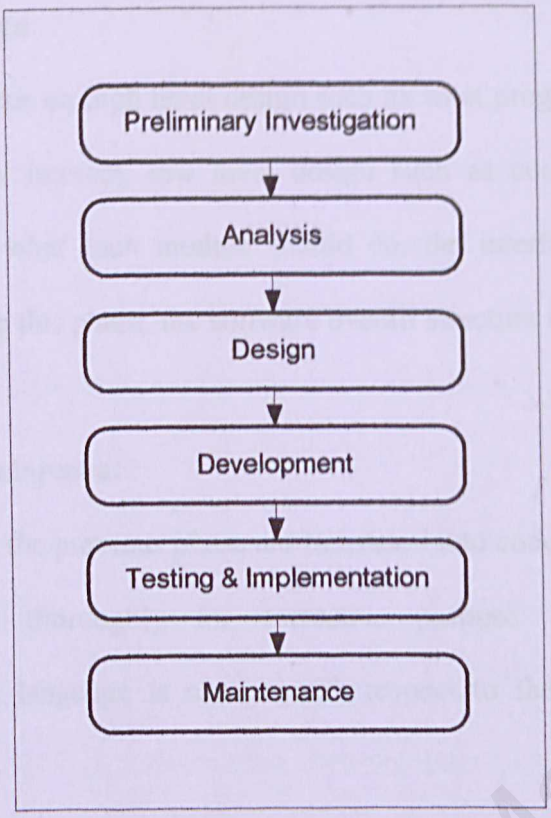

Figure 3.2: SDLC Flow Diagram

Phase 1: The Preliminary Investigation

This phase is also known as the feasibility study phase. The objective of this phase is to verify problems and define the project constraint in terms of time, technical and budgetary. This phase will decide whether the project will proceed as planned or be cancelled due to constraints.

Phase 2: Analysis

This phase gathers the requirements specification from a detailed studies on information acquired during the investigation phase. The requirements will help provide details on what the program or software should do.

Phase 3: Design

This phase focus on high level design such as what programs are needed and how will they interact, low level design such as construction of project modules and what each module should do, the interface design and data design. During this phase, the software overall structure is defined.

Phase 4: Development

Designs from the previous phase are translated into codes. Every requirement is examined thoroughly for corrective purpose. This is when the programming language is chosen with respect to the type of application developed.

Phase 5: Testing and Implementation

Programs that are written in several modules are tested separately and then combined to be test as a whole system. There are three main tests for the newly developed software:

i) integration testing – to ensure interfaces between module work

ii) volume testing – to ensure the system works on the intended platform and with expected volume of data

iii) acceptance testing – to ensure that the system does what the user required

After all the tests have been done on the software, it is then installed on specific hardware and delivered. Users of the system will be trained during this phase.

Phase 6: Maintenance

The system will need maintenance gradually to prevent it from encountering errors from unexpected inputs and such. During this phase, the software could be enhanced to meet the changes of requirements if there are any.

3.2 PROPOSED LIFE CYCLE MODEL

3.2.1 Project Feasibility

After going through a lot of research and reviews, I found out that the Network Scanner project needs me to understand the concepts of computer networking, concepts of network scanning, and the structure of a network scanner and learn to use an open source tool for its development. The project constraints that I am facing now are time constraint and technical constraint.

Time is needed for me to learn using the development tool and increase my understanding in the mentioned subjects before. However, there are a lot of tutorials on open source tools on the internet and there is also a lot of information on network scanning. What I need is a systematic method that allows me to learn and understand phase by phase.

With the knowledge and skills I have in C++ programming and the wxWidget application, I am facing difficulties to develop the Network Scanner without following sequential steps that helps me learn with try-and-error method from the start.

For the Network Scanner development project, I propose the combination of Iterative and Incremental Life Cycle with the Waterfall Life Cycle.

3.2.2 Iterative and Incremental Life Cycle

Iterative is the basics of software development process. When an artifact is produced, we tend to revise it and then produce a second version which is better than the first.

Incremental process is the process of handling a large amount of information in order of current importance, where the most important aspects will be concentrated on first while postponing other aspects that are less important until eventually every aspect of the information will be covered.

In Iterative and Incremental Life Cycle model, iteration and incrementation are used in conjunction with one another. There is no single requirement phase or design phase. Instead, there are multiple instances of each phase in the life cycle (Schach, 2002).

Strengths

i) There are multiple opportunities to make sure that the product software is correct.

ii) The robustness of the architecture can be determined early in the life cycle.

iii) Risks can be resolved early.

iv) A working version of software or its module is available from the start. In this project case, the source codes for network scanning that can be downloaded from the internet.

3.2.3 The Waterfall Life Cycle

The Waterfall Life Cycle model is a sequential development phase. It requires every phase to be complete before starting on another phase. However, if there

is a modification on one of the phase, the products of the earlier phase have to be modified to as a consequence of following a feedback loop. That earlier phase is only considered complete only when the documentation for the phase has been modified and checked by the software quality assurance (SQA) group (Schach, 2004).

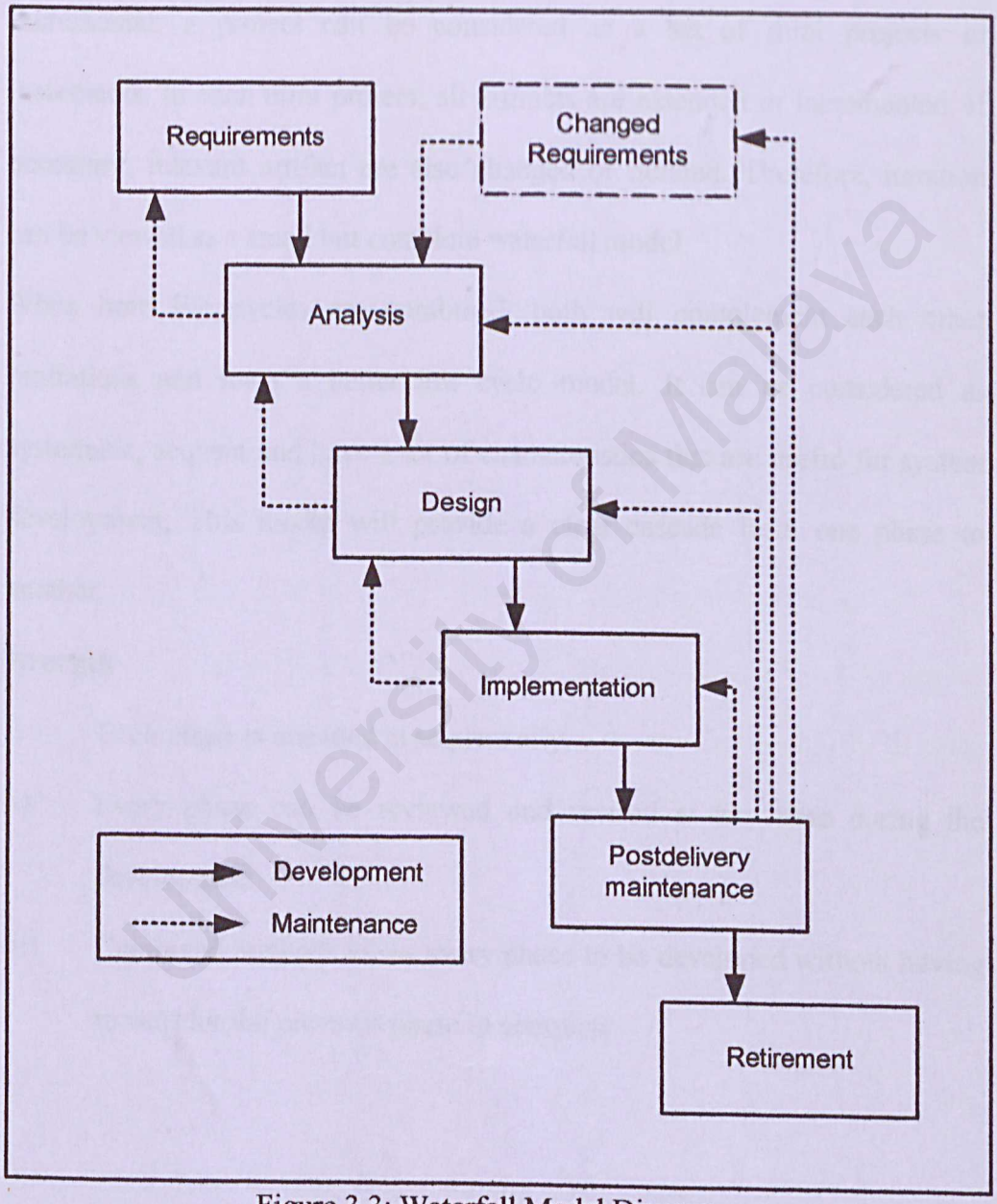

Figure 3.3: Waterfall Model Diagram

Strength

i) Enforce a disciplined approach.

ii) Easy to follow and understand the development phases.

iii) Easy to maintain.

3.2.4 Combination: Iterative and Incremental Waterfall Life Cycle

I combined both life cycles because of their similarity. In Iterative and Incremental, a project can be considered as a set of mini projects or increments. In each mini project, all artifacts are extended or incremented. If necessary, relevant artifact are also changed or iterated. Therefore, iteration can be viewed as a small but complete waterfall model.

When both life cycles are combined, both will complement each other limitations and form a better life cycle model. It can be considered as systematic, sequent and have a lot of characteristics that are useful for system development. This model will provide a clear cascade from one phase to another.

Strength

i) Each phase is attended to sequentially.

ii) Every phase can be reviewed and revised at any time during the development.

iii) Combined methods allow every phase to be developed without having to wait for the previous phase to complete.

3.3 WHY CHOOSE ITERATIVE AND INCREMENTAL WATERFALL?

i) The waterfall model is the most popular model used for system development. Because it is sequential, it made it easy to understand the workflow in the system design.

ii) The combination of iterative and incremental model with waterfall model forms a systematic model where every step in each phase is sequential so that even the workflow in each phase is easy to understand.

iii) The development process for this modified waterfall model is not just simple linear but it also involve a sequence of iteration of development activities. This is one feature that makes it possible to help me cope with the learning curve of using open source tool in this project.

iv) Achievements of every step in this model can be viewed frequently, therefore giving more chance to discover errors and mistakes easily.

v) The material that will be used for iteration and incremental process is the source codes that are available over the internet. I hope that I will be able to study, understand and revise the code from time to time using the iteration and incremental process.

CHAPTER 4

ANALYSIS

CHAPTER 4

ANALYSIS

Analysis is done on the information collected previously in order to extract important and specific information for the Network Scanner development. In this chapter analysis is used to gather the requirements of the Network Scanner. This chapter will describe the techniques that were used to define the requirements of the software development and describe details of the Network Scanner requirements.

4.1 TECHNIQUES USED TO DEFINE REQUIREMENTS

There are several techniques that I have used to define the requirements of the Network Scanner project. Although the techniques that I used are common techniques, but the input from those techniques really help me on defining the requirements of this project.

4.1.1 Internet Research

A big amount of the information I acquired is found on the internet. There are actually a lot of websites that offer information on network scanning. Source codes for network scanning functions and tutorials for open source tool that I use in the project development are widely found over the internet. I also read forum threads on network scanning tool development to gather more information on Network Scanner requirements.

4.1.2 Library Research

I like browsing the online library services offered by the university library. There are quite a number of books on network security however it is hard to find any publication on network scanning. I also went to the faculty library in

search of those books. However I cannot find any books on network scanning there. But the information I got from the network security also give me the rough ideas on what network scanner requirements should be.

4.1.3 Previous Thesis Research

The seniors thesis that are kept in our faculty library has been a great help for me not only in defining my project's requirements from information on other similar projects but they also give me the idea of doing a guideline for my thesis .

4.2 REQUIREMENTS SPECIFICATION

The process of requirements analysis objective is to produce a software specification definition – the requirements specification. It is an abstract description of the services which the system should provide and constraints under which the system must operate. There are to main requirements that will be discussed here; functional requirements and non-functional requirements. To ensure the quality of the system developed, both requirements are important.

4.2.1 Functional Requirements

A functional requirement is a description of activities and services a system should provide (Whitten *et al*, 2003). These requirements are needed in order to achieve the purpose and objective of the system. The functional requirements that I have defined for the network scanner project are:

- The software must be able to send request packets to target hosts on the network.

  The software should be able to use network services like ping and trace-routes in order to gain information about the network.

- The software must also be able to receive reply packets from the hosts and extract the data in the packet for information.

  This is important so that the packet can be analyzed and information such as IP address and physical address can be extracted.
- The software must be able to use and differentiate different kinds of network protocol.

  This is because different kinds of protocol give different data for different information.
- The system with the software must have physical connection to a network in order to perform successful scans.

  If the software is not online, request packet could not be sent.

4.2.2 Non-functional Requirements

Non-functional requirements are descriptions of other features, characteristics and constraints that define a satisfactory system (Whitten *et al*, 2003). They are the factors that must also be taken into consideration when developing the system. Sometimes they play important role to ensure system robustness and success.

- Reliability

  To ensure that system performs its functions with require precision and accuracy.
- Efficiency

  Efficiency is understood as the ability of a process procedure to be called or accessed unlimitedly to produce similar performance outcomes at an

acceptable or credible speed (Sommerwille, 1995). It is measured based on time performance, report generation speed, etc.

- User friendly

  The interface should be easy to use and understand. Information and instructions should be made clear. Among the criteria of a good user friendly interface are consistency of the design, able to show error messages and puts the user in control of the system.

- Manageability

  The system should be easy to maintain, adaptive to the surrounding technology and evolutionary, which is the ability to be enhanced in the future should there be a requirement to do so.

- Stability

  The system should be able to maintain its performance in any situation and able to cope with a heavy flow of work.

4.3 SYSTEM DEVELOPMENT TECHNOLOGY

This section will discuss about hardware and software that will be used in the system development.

4.3.1 Hardware

The hardware needed for the development of the Network Scanner project are:

- Personal computer with Pentium processor
- Input and output devices such as mouse, keyboard and monitor.
- A network interface card for physical connection to a network.

4.3.2 Software

The software needed for the development of the Network Scanner project :

- Windows 98 or later operating systems, currently Windows XP Professional.
- Open source tool, WxWidgets or MS-Windows based application tool.
- Microsoft Visual C++ compiler or Borland C++ any other C++ compiler that is supported by WxWidgets.

CHAPTER 5

SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

System design is the specification or construction of a technical, computer based solution for the requirements identified during the analysis phase (Whitten *et al*, 2003). System design is the first of the three technical activities – design, coding, and testing – that are required to build and verify the software. For the network scanner project, the design process produces functionality design and user interface design.

5.1 FUNCTIONALITY DESIGN

The functionality design deals with the purpose and collaboration of each module to achieve the overall system functionality specification.

5.1.1 Structure Chart

Below is the structure chart of the Network Scanner. Overall, the Network Scanner has two main modules; the Scan Module and the Report Module.

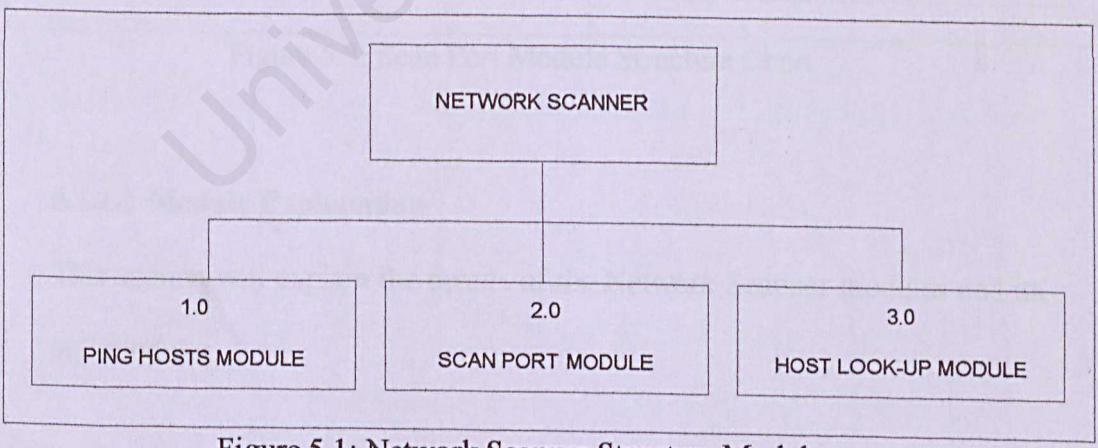

Figure 5.1: Network Scanner Structure Module

The Ping Hosts Module can be divided to three sub-modules; Ping Hosts Configuration module, Ping Engine module and Ping Logging module. While

the Scan Port Module can be divided to three sub-modules, Scan Port Configuration module, Scan Port Engine module and Scan Port Logging module.

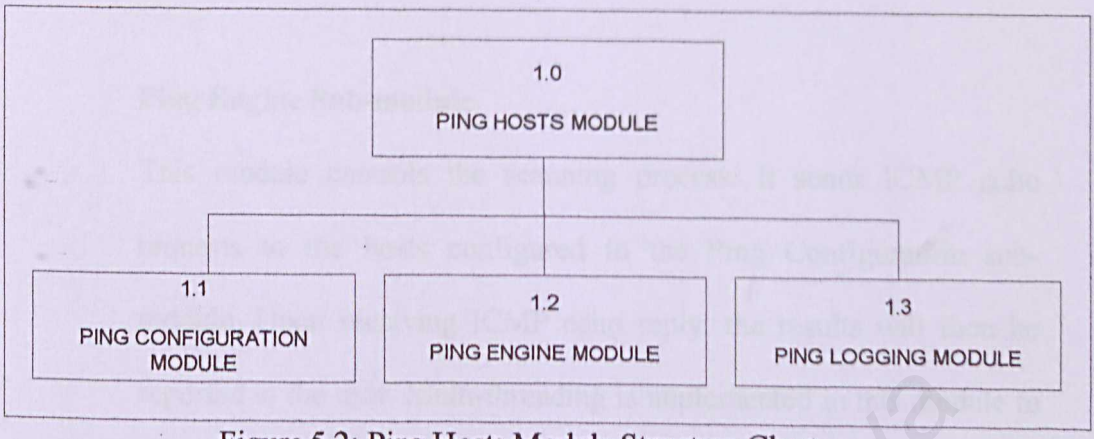

Figure 5.2: Ping Hosts Module Structure Chart

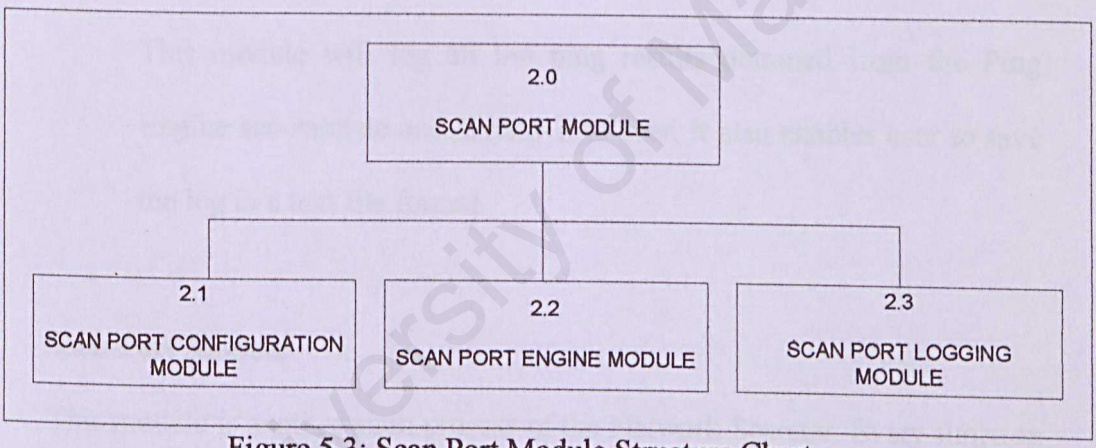

Figure 5.3: Scan Port Module Structure Chart

5.1.1.1 Module Explanation

This section will explain the details of the Network Scanner modules and its sub-modules.

Ping Hosts Module

This is one of the main processes of the Network Scanner system. Every function in this module revolves around the ping host process.

Ping Configuration Sub-module

Its purpose is to set up the initial or required parameters before the commencement of the Ping Engine's operation.

Ping Engine Sub-module

This module controls the scanning process. It sends ICMP echo requests to the hosts configured in the Ping Configuration sub-module. Upon receiving ICMP echo reply, the results will then be reported to the user. Multi-threading is implemented in this module to speed up the ping process.

Ping Logging Sub-Module

This module will log all the ping results obtained from the Ping Engine sub-module and display it to user. It also enables user to save the log in a text file format.

Scan Port Module

This module is another main process of the Network Scanner. Every function of this module revolves around port scanning process in the Network Scanner.

Scan Port Configuration Sub-module

This module initialize all the parameters need by the Scan Port Engine to run the port scan. The parameters include target IP address and range of ports.

Scan Port Engine Sub-Module

This module commences the port scanning process with the parameters configured in the Scan Port Configuration sub-module. Multi-threading is implemented in this module to speed up the scanning process.

Scan Port Logging Sub-Module

This module will log all the scan results obtained from the Scan Port Engine sub-module and display it to user. It also enables user to save the log in a text file format.

Host Look-Up Module

This module is only an extra function for looking up host names in the network and in the internet.

5.1.2 Data Flow Diagram (DFD)

DFD is a process model used to depict the flow of data through a system and the work or processing performed by the system (Whitten *et al*, 2003).

The figure below presents the context diagram of the Network Scanner.

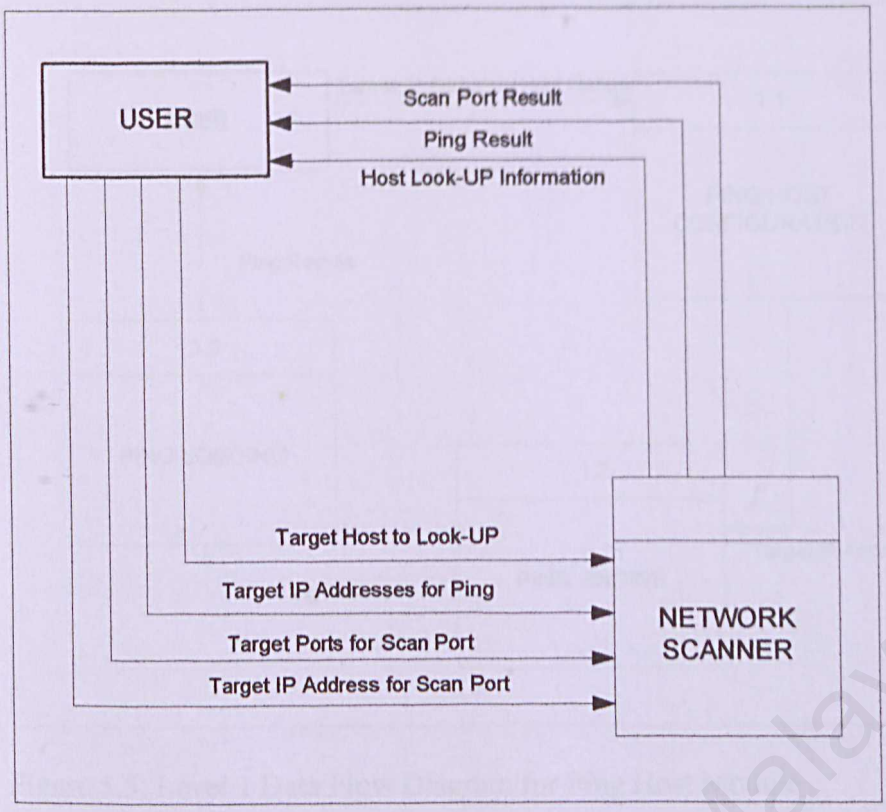

Figure 5.4: Context Diagram (Level 0 Data Flow Diagram)

Figure shows the Level 1 Data Flow Diagram for Ping Host Module, Scan Port and Host Look-UP Module.

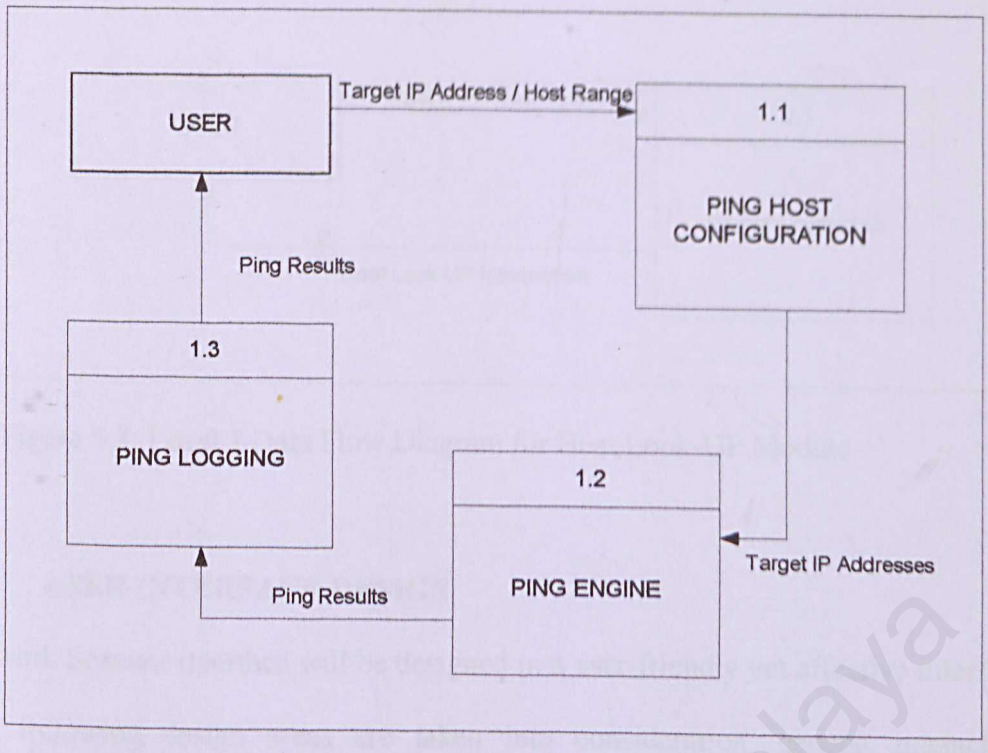

Figure 5.5: Level 1 Data Flow Diagram for Ping Host Module

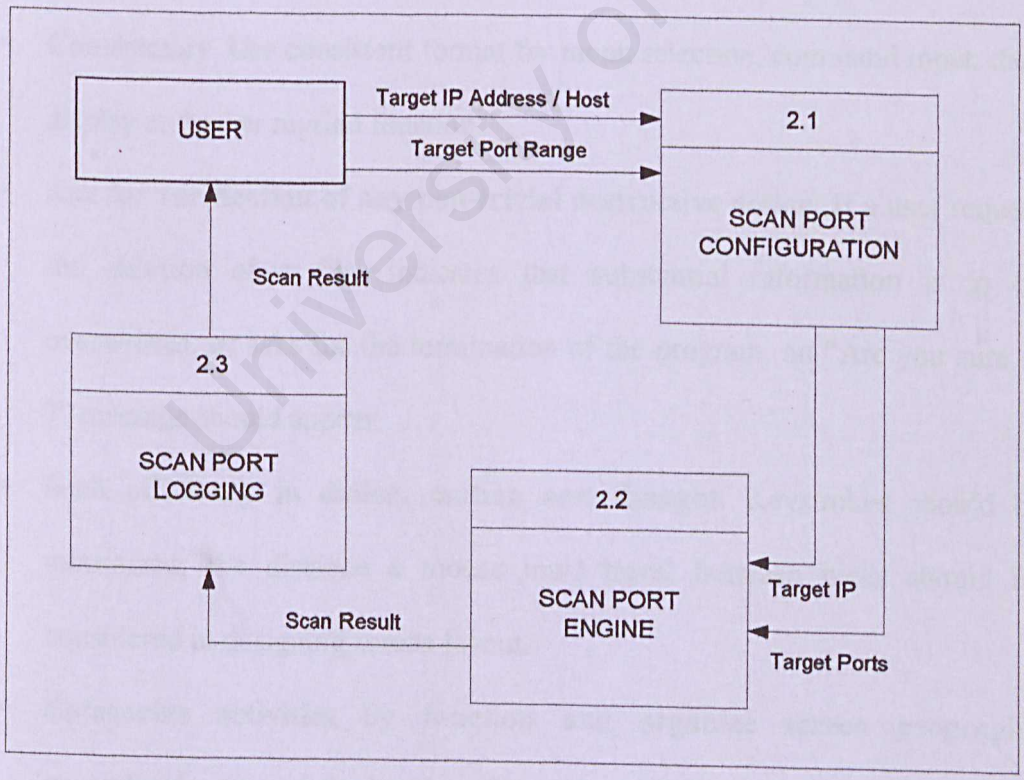

Figure 5.6: Level 1 Data Flow Diagram for Scan Port Module

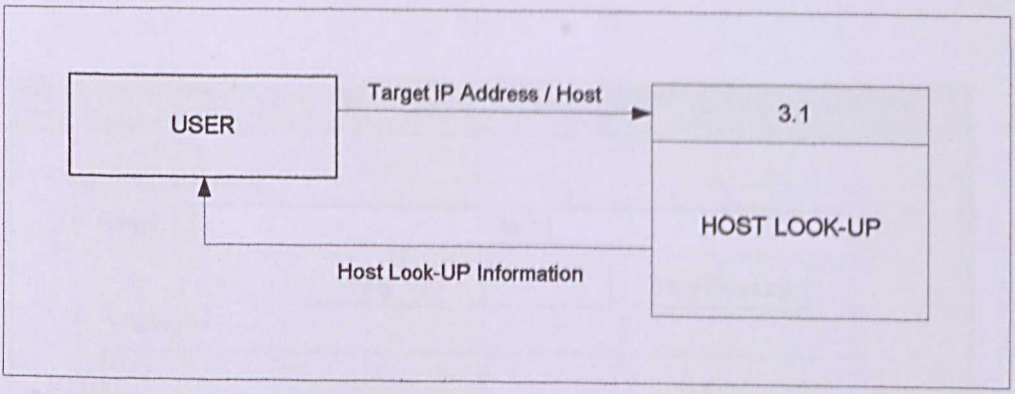

Figure 5.7: Level 1 Data Flow Diagram for Host Look-UP Module

5.2 USER INTERFACE DESIGN

Network Scanner interface will be designed in a user-friendly yet affective interface. The following design areas are taken into consideration: general interaction, information display and data entry. The guidelines below will also be adopted in designing the interface for Network Scanner.

- **Consistency**. Use consistent format for menu selection, command input, data display and other myriad function.
- **Ask for verification of any non-trivial destructive action**. If a user request the deletion of a file, indicates that substantial information is to be overwritten, or asks for the termination of the program, an "Are you sure… ?" message should appear.
- **Seek efficiency in dialog, motion and thought**. Keystrokes should be minimized; the distance a mouse must travel between picks should be considered in designing screen layout.
- **Categorize activities by function and organize screen geography accordingly**. One of the key benefits of the pull-down menu is the ability to organize commands by type. Proper placement of commands and action is recommended.

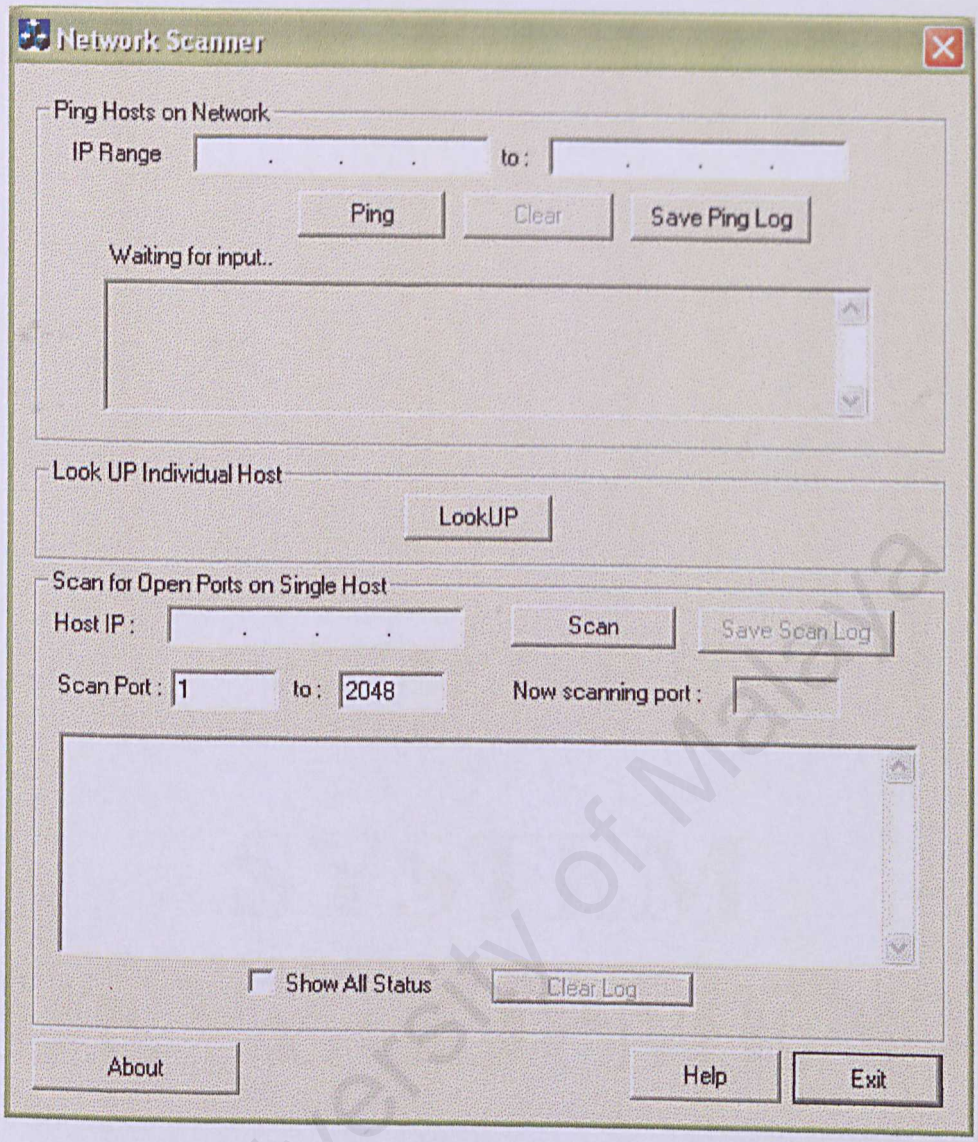

Figure 5.6: User Interface Design

The figure above is the layout that I have designed for my Network Scanner user interface.

CHAPTER 6

SYSTEM DEVELOPMENT

CHAPTER 6

SYSTEM DEVELOPMENT

The system development phase is where all the design requirements are coded into computer language. In this phase is where all the development process occurs with the use of deliverables and automated tools to realize and continuously improve the system design.

6.1 HARDWARE REQUIREMENTS

During this phase, the Network Scanner program is developed on:

- Windows XP Service Pack 2 platform
- 1.8Ghz Pentium IV Processor
- 512MB RAM
- Storage space up to 1.5GB for development and testing
- 15" monitor with resolution at least 800x600
- Input devices – standard mouse and keyboard
- D-Link DFE-538TX 10/100MBps network adapter

In this project development, the network adapter is the most important hardware for the system as it provides the connection to the network and it is a hardware that has the ability to listen, send and receive packet from other computers on the network.

6.2 SOFTWARE REQUIREMENT

Another most important tool used to implement this system is the software. The software enables a programmer to code the whole program and compile it to the

language that the machine understands. Most of the tools I used to develop this program are very familiar in our faculty.

The system software requirements are summarized in the table below.

Software	Requirement for	Role
Microsoft Windows XP Service Pack 2	System Requirements System Environment System Development	Operating system platform in development.
Microsoft Visual Studio 6.0 (Microsoft Visual C++ 6.0)	System Development	Programming Language Compiler and User Interface Design.
Windows XP SDK Include files and Libraries	System Development	Compiler assistant
Microsoft Foundation Classes	System Development	User Interface Design

Table 6.1: System Software Requirements

At first, I proposed to build the system with an open source tool which is WxWidget. However at the last moment I realized I took a very long time to study how to use the tool, so I converted to Microsoft Foundation Classes in Microsoft Visual C++ 6.0.

6.2.1 Microsoft Foundation Classes

Microsoft Foundation Class (MFC) provides a framework on which applications can be developed for MS-Windows. It is a class hierarchy a programmer can use to build Windows application quickly and easily. Typically MFC is used by programmers to create Windows programs with Graphical User Interfaces (GUIs). Users interact with GUI by *clicking the mouse, pressing the button, pressing a key*, etc. when a GUI event occurs, the windows operating system sends a message to the program. Programming the functions that respond to these messages is called *event-driven programming*. With event-driven programs, the user, not the programmer, dictates the order

of program execution by interacting with the GUI. Instead of the program "driving" the user, the user "drives" the program. With the user in control, programs become more user-friendly (Deitel & Deitel, 2000).

6.3 PROGRAM DEVELOPMENT AND CODING

There are several steps to be taken into account during the early development and implementation phase to ensure that the phase will run smoothly with little or reduced difficulties.

6.3.1 Review of the Program Documentations

Documentations that were made along the way during the earlier phases, system analysis and system design, are all reanalyzed to extract the information on what and how the program should be developed. From the documentations, information like system design, its architectural view and concepts, flow diagrams and sample layout of the program is gathered together to provide guidance and understanding of the works that needs to be done in the coding process.

6.3.2 Designing the Program

This step is to determine how the program should work and can accomplish the features and functions described in the program documentations reviewed earlier. In this process, a clearer view of the system structure is obtained.

6.3.3 Coding Approaches

Generally there are two types of coding approaches, top-down approach and bottom-up approach. Top-down approach is the method of designing the interface or the upper level modules and then designing the lower level modules. While bottom-up approach is the method of designing the lower level modules wile designing the upper level modules.

In this project development, I used both top-down and bottom-up approaches at the same time. While designing the interface, I built the lower level modules for the interface I designed concurrently. Then each module is built and tested before fully integrating them to the interface designed.

When building each of the modules, I spent much of my time looking for codes on the internet that have similarities with my project. I studied the codes thoroughly to understand network programming and tried to develop my project first using the codes from the internet. After that I tried experimenting with every code by modifying them so that I can understand the codes when building the program and integrating it with the user interface. I have also searched for information for every object class used in my project and studied them. I have also put my own comments in the code for future reference and better understanding.

6.3.4 Coding Style

I tried to write easy to understand codes and heavily commented the codes to provide better understanding and for future reference. I also used standard naming convention, which is naming the variables according to what the

variables represent to reduce confusion to those who will try to understand the codes later.

6.4 MODULE IMPLEMENTATION

As described in the system design phase, my program is made up of three main modules. Here I will explain deeper the module's functionality and how the module is implemented.

6.4.1 Ping Host Module

The Ping Hosts module function is to ping every host within the IP address range specified by the user. The module will send ICMP Echo Request and listen for ICMP Echo Request Reply to determine whether the hosts on the network are alive or not.

The main functions used in this module are

- `void CNetworkScannerDlg::OnBtnPing()`
  - This function runs when the user clicks on the "Ping" button after entering the IP address range. The function will validate the IP address range, build the host loop and call the next function which is a worker thread, `UINT runPing (LPVOID pParam)`, to run the ping.
- `UINT runPing (LPVOID pParam)`
  - This thread will ping the host it is assigned to in the host loop using these functions;
    - `int SendEchoRequest(SOCKET s,LPSOCKADDR_IN lpstToAddr),`

- `int WaitForEchoReply(SOCKET s),`
- `DWORD RecvEchoReply(SOCKET s), LPSOCKADDR_IN lpsaFrom, u_char *pTTL)` and
- `u_short in_cksum(u_short *addr, int len).`

It will then calculate the TTL and return the ping result to display to the user.

- `int SendEchoRequest(SOCKET s,LPSOCKADDR_IN lpstToAddr)`
  - This function fills in the Echo Request header packet and sends it to the destination host.
- `int WaitForEchoReply(SOCKET s)`
  - This function uses the `select()` function from `winsock.h` header file to determine when the data is waiting to be read.
- `DWORD RecvEchoReply(SOCKET s, LPSOCKADDR_IN lpsaFrom, u_char *pTTL)`
  - This function receives incoming data and parses out the data fields.
- `u_short in_cksum(u_short *addr, int len)`
  - This program is used to compute the checksum for data in the packet headers.

- `void CNetworkScannerDlg::OnTimer (UINT nIDEvent)`
  - This is the timer function that is used by the runping thread to send the results to be displayed to user.
- `void CNetworkScannerDlg::OnClearping()`
  - This functions runs when the user clicks on the "Clear" button to clear the ping log.
- `void CNetworkScannerDlg::OnBtnSaveping()`
  - When the user wants to save the ping log by clicking on the "Save Ping" button, this function will save the ping log in a text file.

6.4.2 Scan Port Module

The Scan Port module's function is to scan for open ports and determine port connection status on a target host specified by user. The module will attempt to connect to every port in the range specified by user on the target host and return the connection status results.

The main functions used in this module are

- `void CNetworkScannerDlg::OnBtnScan()`
  - This function runs when the user clicks on the "Scan" button after entering the target host IP address and the target port range. It calls the function `void CNetworkScannerDlg::scan()`.
- `void CNetworkScannerDlg::scan()`
  - This function will build the port loop and call the `UINT runScan (LPVOID pParam)` thread to execute the port scan.

- `UINT runScan (LPVOID pParam)`
  - This function will do the port scanning and return the port connection status.
- `void CNetworkScannerDlg::OnTimer(UINT nIDEvent)`
  - This timer function is used by the `runscan` thread to send scan result to be displayed to the user.
- `void CNetworkScannerDlg::OnBtnPausescan()`
  - When the user clicks on the "Pause" button, this function will stop the scan momentarily.
- `void CNetworkScannerDlg::OnBtnContscan()`
  - When the user clicks on the "Continue Scan" button the function will resume the scan on the port where it left at.
- `void CNetworkScannerDlg::OnBtnClearlog()`
  - This function runs when the user wants to clear the scan log.
- `void CNetworkScannerDlg::OnBtnSavescan()`
  - This function will save the scan log in a text file.

6.4.3 Host Look-UP Module

The Host Look-UP module's function is to look up the host name and the host's IP address of the host name or IP address given by user.

The main functions used in this module are

- `void CLookUPHostDlg::OnBtnLookuph()`
  - This function will run when user clicks on the "Look-UP" button. This function will return the information of the looked-up host.

- `void CLookUPHostDlg::OnBtnClearlu()`
  - This function will clear all the fields in the Host Look-UP section dialog.

There are also other supporting modules such as Help and About. However these modules do not play vital parts in the project. The Help module provides the user the information on how to use the Network Scanner program. While the About module provides information about the Network Scanner Program.

CHAPTER 7

SYSTEM TESTING

CHAPTER 7

SYSTEM TESTING

In this project, system testing is done through out the project development, not just at the end of the project development. Each time the program is tested to demonstrate the existence of faults. Testing is not done to ensure the program runs correctly, but it is done to verify whether there are faults in the program or not.

7.1 TYPES OF FAULTS

Generally there are three types of faults.

7.1.1 Algorithmic Faults

Algorithmic faults are more like logical fault where sometimes when the algorithm is first developed; there are some logical steps and information missed by the developer that might cause problems when running the algorithm later.

7.1.2 Syntax Fault

This can happen due to typing errors and mistakenly misplaced or forgetting important symbols or variables during the coding process.

7.1.3 Documentation Fault

Documentation fault happens when the documentation provided earlier does not really respond to the project requirements, thus making many

questionable actions arise, for example, whether to apply *this* technique or *that* technique?

During the project development, the most common faults I faced were algorithmic faults and sometimes, syntax faults. The algorithmic faults occurred because I did not fully understand about the functions I want to implement in the project especially the multithreading functions. It works well in the Scan Port module but it has some bugs when applied to the Ping Host module. While the syntax faults usually occurred because I forgot to type some characters and symbols in some places, and there are times when I forgot to declare a variable.

7.2 TESTING PLANNING

There are several steps to be followed when planning a system test.

7.2.1 Establish Test Objective

First we must determine what to test of the system. Always remember that the purpose of system testing is to demonstrate any existence of faults. This project test objective is to find any errors that will affect the results that will be delivered by the system, for example, error when pinging a host that is not online.

7.2.2 Designing Test Cases

Test cases are designed to demonstrate a situation where a fault could occur. For this project, example test cases are

- What if the user entered an invalid IP address range?

- What happens if the target host is not alive or not online?

7.2.3 Writing Test Cases

This step is to put the test cases into documentation to be delivered during the test phase.

7.2.4 Testing Test Cases

Every test cases is studied and tested logically to assume whether the fault will exist or not during the test execution.

7.2.5 Executing Text Cases

Every test case will be executed on the program. If any error or fault occurs, it means that the test have achieved its objectives.

7.3 THE TESTING PROCESS

In the project testing process, there are several step-by-step testing processes that follow. The test process is dhuhuhone through out the project development starting from the first unit that was built. The figure below shows the processes involved in system testing.

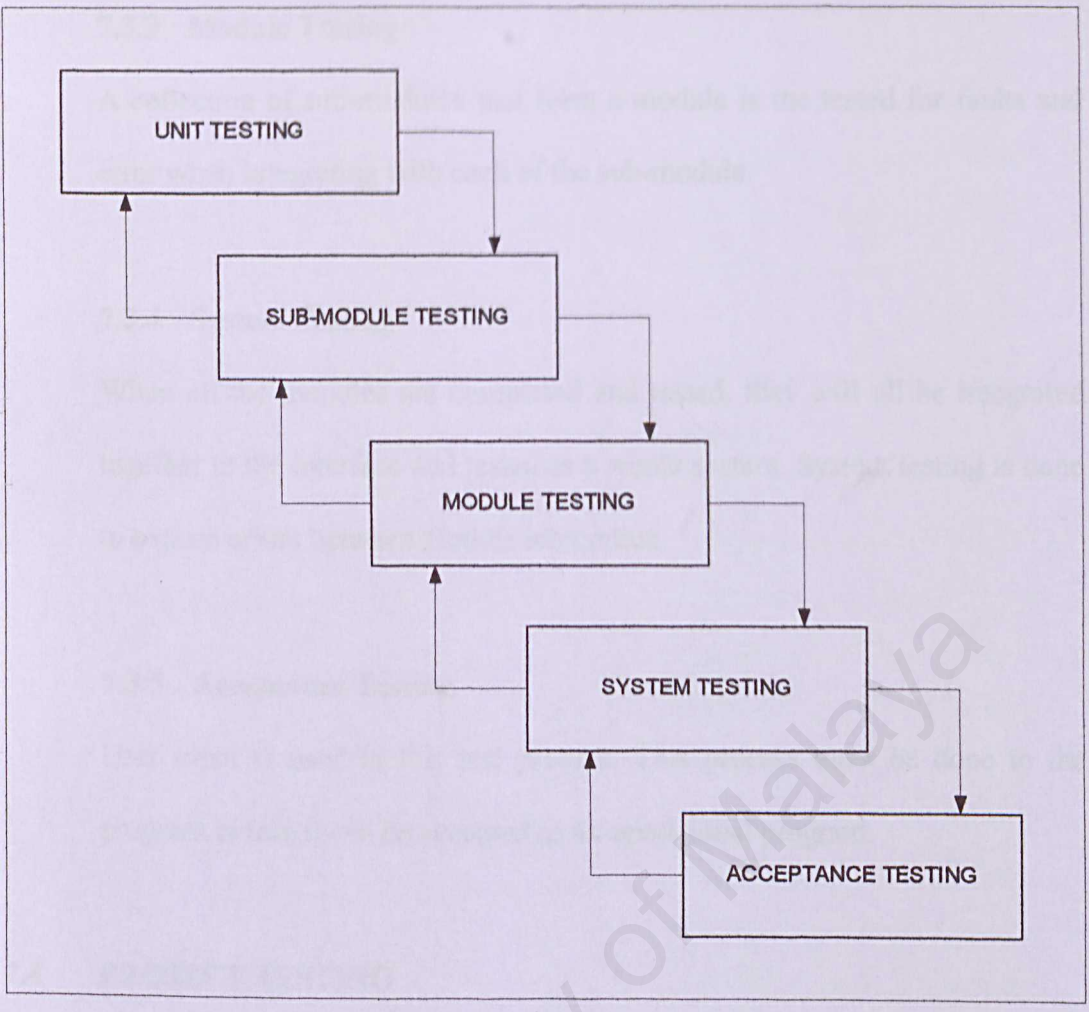

Figure 7.1: Testing Process Flow Chart

7.3.1 Unit Testing

Every unit, every function in the program is tested independently for syntax error and faults.

7.3.2 Sub-Module Testing

A collection of functions that form a sub-module is tested to determine whether fault exists or not between the functions.

7.3.3 Module Testing

A collection of sub-modules that form a module is the tested for faults and error when integrating with each of the sub-module.

7.3.4 System Testing

When all the modules are completed and tested, they will all be integrated together to the interface and tested as a whole system. System testing is done to expose errors between module integration.

7.3.5 Acceptance Testing

User input is used in this test process. This process must be done to the program before it can be accepted as an operational program.

7.4 PROJECT TESTING

Since the Network Scanner program is a network-based program, in the first testing process, I used the localhost, the computer I built the Network Scanner on as the target host. Then for the live data test, I only have the option to test the Network Scanner thoroughly on a fully functional network. The network that I have chosen to be the test bed is the network at the 10^{th} Residential College.

7.4.1 Ping Host Module Testing

The table below depicts a sample of the test that has been done to the Ping Host module after completion of the Network Scanner.

Table 7.1: Ping Host Module Testing

Test Case	Expected Result	Real Result
User enters IP address range from 192.68.2.201 to 192.168.3.230	An error message will appear and ask the user to enter the valid IP address range.	Error message appeared and the user is asked to enter a valid IP address range.
One or several target host in the range is not online or not alive.	The offline hosts will return 'Request timed out' while the online hosts will return ping reply message.	The online host did return the ping reply message and the offline host did return the 'Request timed out'. But if the first host is not online, the ping log result will show the ping reply of the next online host in the target range. This kind of output does not happen when using ordinary for loop. But the thing is, using for loop takes much more time when the target range is wide. **This is the multithreading bug that I cannot resolve during the project development.

7.4.2 Scan Port Module Testing

The table below depicts a sample of the test that has been done to the Scan Port module after completion of the Network Scanner.

Table 7.2: Scan Port Module Testing

Test Case	Expected Result	Real result
If the target address is not on the network.	An error message will appear in the scan log telling the user of some unknown error when connecting to the port.	The error message never came out. There is a high possibility that it scanned the ports of a host on the internet, not on the local network.

7.4.3 Look-UP Host Module Testing

The table below depicts a sample of the test that has been done to the Look-UP Host module after completion of the Network Scanner.

Table 7.3: Look-UP Host Module Testing

Test Case	Expected Result	Real Result
The target host entered is not online or not alive.	An error message will pop up and tells the user that the host is not valid or not online.	An error message popped up and tells the user that the host is not valid or not online.

After the testing on the whole Network Scanner is done, all the bugs are debugged even though there are still a few bugs left, they are the bugs that cannot be resolved until now.

CHAPTER 8

SYSTEM EVALUATION

CHAPTER 8

SYSTEM EVALUATION

System evaluation is done after the project is completed to asses how much the project has accomplished its goal and features. The results of system evaluation are used to improve the system design and features in the future. This topic describes the evaluation that I have done to the Network Scanner project.

8.1 PROBLEMS ENCOUNTERED AND SOLUTIONS

1. No experience with the tools used for the systems development

I experienced an enormous learning curve in understanding how the files and libraries in the WxWidgets for creating user interface works. I tried very hard to find documentation to aid me in understanding the tool but it came to no avail and time was getting shorter and shorter.

In the last moment I changed my development tool from WxWidgets to Microsoft Foundation Classes (MFC) using Microsoft Visual Studio 6.0. Although I am new to MFC too, the information on MFC can be found widely on the internet and MFC is very easy to use and understand. I could master the usage of MFC in just a few days compared to WxWidgets that took me almost a month.

2. No experience in network programming

I have never had the experience in building a networking tool using C++. At first look, I can hardly understand the network program codes I downloaded from the internet.

I searched the internet for information of every class that is associated with network programming and studied the classes to understand its usage in network programming and how to use network variables in C++.

3. No experience in building a multithreaded program

I have never build programs that integrates multithreading before and it was really challenging to understand about multithreading.

I tried using the multithreading code I downloaded from the internet. Although it did not turn out very well, the module worked.

4. Lack of time to concentrate on the project

Under the pressure of completing my degree in the minimum 6 semester's time, most of the course I took during this semester has coursework to be submitted before the examinations, usually around the 12th week. Most of the coursework involves system development which stole most of my time from concentrating on the thesis project.

8.2 PROJECT STRENGTHS

1. Windows-based application

Microsoft Windows operating systems is the most widely used operating system. Since the Network Scanner is developed as a Windows-based application, it should have no problems running in Windows environment.

2. Simple User Interface

The user interface of this application is very simple and very easy to understand.

3. Easy reference

An easy-to-understand user manual to aid the user when using the program is always available at the click on the "Help" button.

8.3 SYSTEM LIMITATIONS

1. The system uses multithreading to execute ping and scan ports, however the thread does not display the ping and scan result directly. Instead, the results are sent to the timer function and displayed sequentially according to IP addresses or port numbers.

2. The Scan Port Module can only scan for open ports and port connection status on a single host at one time.

3. The Scan Port module uses the TCP Connect port scanning technique only and does not provide other port scanning options. It also does not provide fingerprinting information for each of the port scanned.

4. The Ping Host module uses the ICMP Sweeps technique in Ping Sweeps and does not provide other ping functions.

5. Simple interface that does not allow result manipulation.

8.4 FUTURE ENHANCEMENTS

1. The results display function will be integrated in the thread and the worker thread will be integrated in the object class.

2. The Scan Port module will be enhanced to allow port scanning on many different hosts at the same time using multithreading.

3. Fingerprinting module will be added to the Scan Port module to get information on the scanned ports. Other port scanning techniques such as UDP Scan will also be added as an option in port scanning.

4. Other Ping Sweeps techniques such as Broadcast ICMP will be added to the Ping Host module as a ping option.

5. The interface will be enhanced to allow the manipulation of the results data when users click on the results to get information or do another operation.

CONCLUSION

CONCLUSION

The Network Scanner can be considered as a successful project minus the multithreading bugs and project limitations. The Network Scanner can ping every host in the IP address range given by the user successfully, the fault only arises hen it comes to the ping result display. It can also execute port scanning without error and provide a variable of results regarding the port connection status grabbed from the network programming error handlers. The Network Scanner can look-up most of the target hosts given especially well known targets on the internet.

From this project, I have learned the basics of network programming. I finally know which class to use, which header files to include and what functions to call from my attempts to code the Network Scanner.

I have also learned how to use the WxWidgets library and MFC to design a simple user interface. I hope I can design much better interface in the future. Apart from that I have also learned to be patient and that hard work and concentration is the key of building a successful application.

Lastly, completing this project gives me the mast valuable experience in doing research and development of computer science subjects that can never be gained much in any other way, without hard work and knowledge.

APPENDIX

APPENDIX A:

USER MANUAL

THE NETWORK SCANNER

USER MANUAL

Table of Contents

List of Figures

STARTING UP

The Network Scanner user interface consists of three parts.

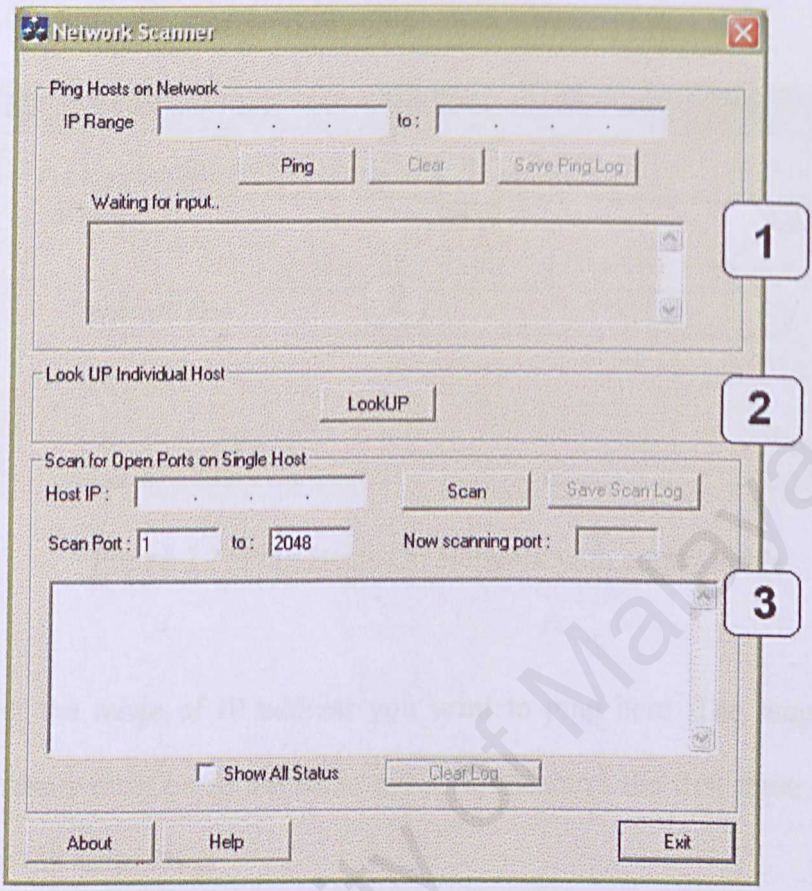

Figure 1.1: The Network Scanner Main Interface

1. Ping Hosts On Network
2. Look UP Individual Host
3. Scan for Open Ports on Single Host

Next we will look at every section of the Network Scanner user interface.

PING HOSTS ON NETWORK

This is the Ping Hosts on Network section.

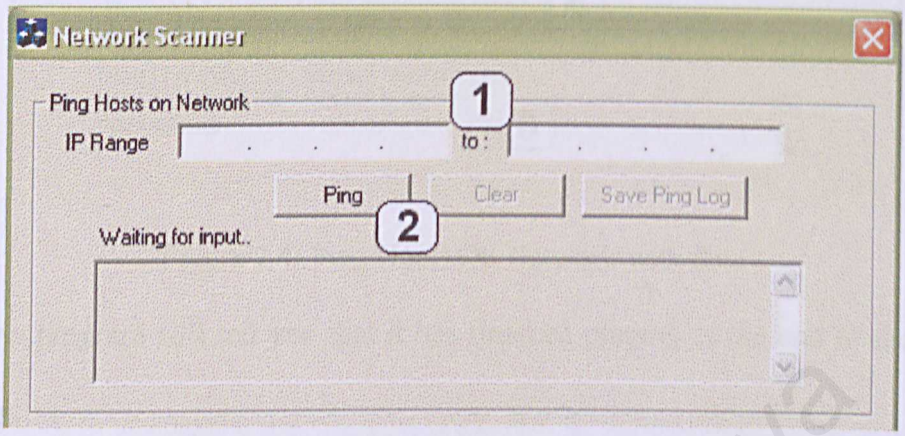

Figure 2.1: Ping Hosts On Network with empty fields

1. Enter the range of IP address you want to ping here. The range of the IP addresses must be in the same network, meaning the first three octets must have the same value.

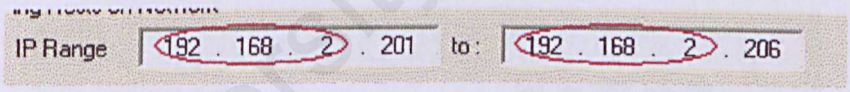

Figure 2.2: Valid IP Range

If not, an error message will appear.

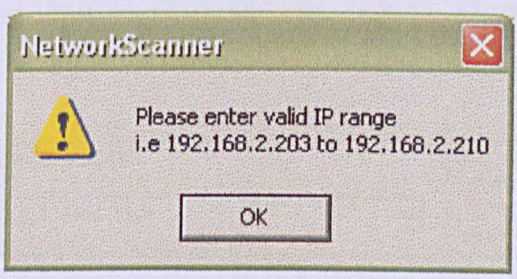

Figure 2.3: Invalid IP Range error message

2. Click on the "Ping" button to run the ping. Perhaps you must wait for a while.

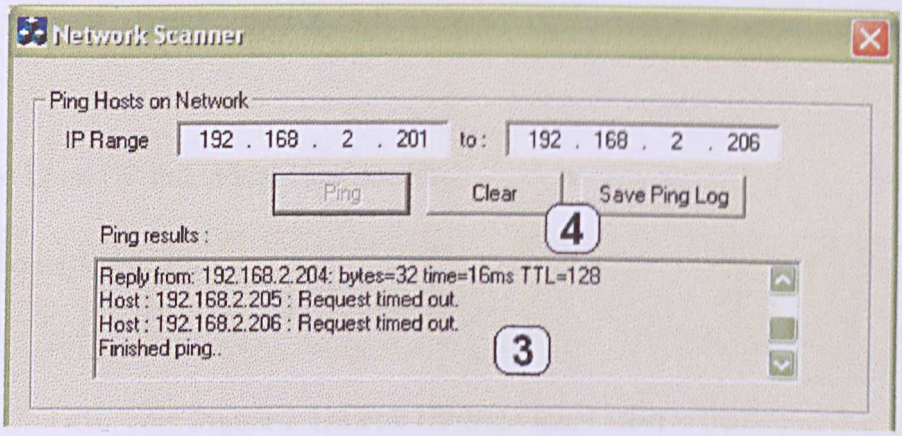

Figure 2.4: Ping Hosts On Network with data

3. The program will tell you that it has finished pinging at the end of the ping log.

4. After the ping result is displayed, you can either click on the "Clear" button to clear the ping log or click on the "Save Ping Log" button to save the ping log in a text file. A popup window will appear to allow you to name the file and choose a storage location.

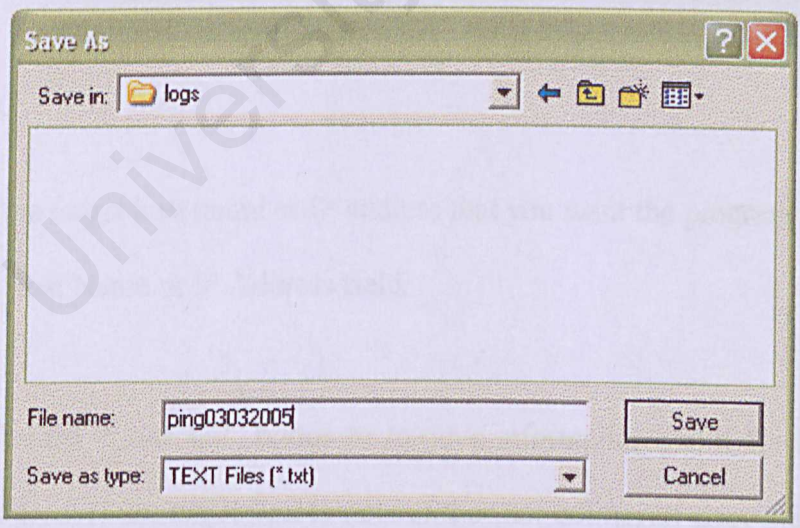

Figure 2.5: Save Ping Log

LOOK UP INDIVIDUAL HOST

This is the Look UP Individual Host section.

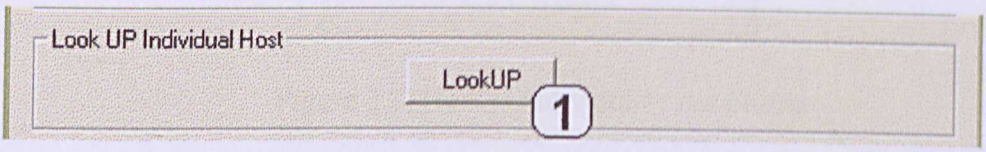

Figure 3.1: Look UP Individual Host

1. Click on the "Look UP" button to open the Look UP Interface.

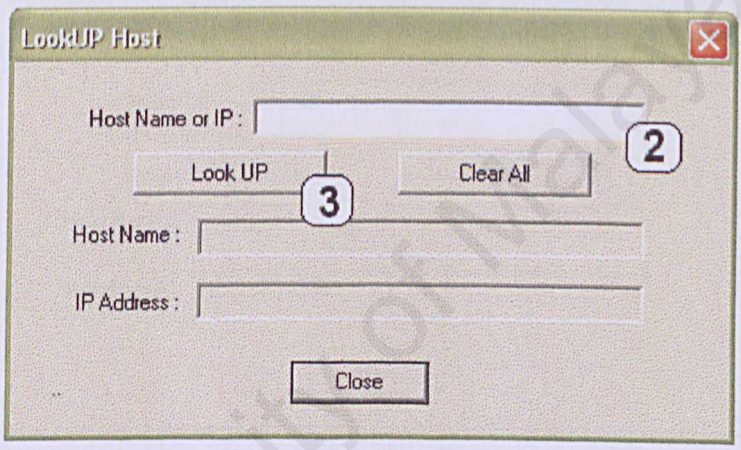

Figure 3.2: Look UP Interface

2. Enter the target host name or IP address that you want the program to look up in the Host Name or IP Address field.

3. Click on the "Look UP" button to retrieve information about the target host you entered. If the target host is not valid or not online, an error message will appear telling you that the host is not valid or not online.

Figure 3.3: Host is not valid or not online

4. If the host is valid and online, the host name of the host will be displayed in the Host Name field.

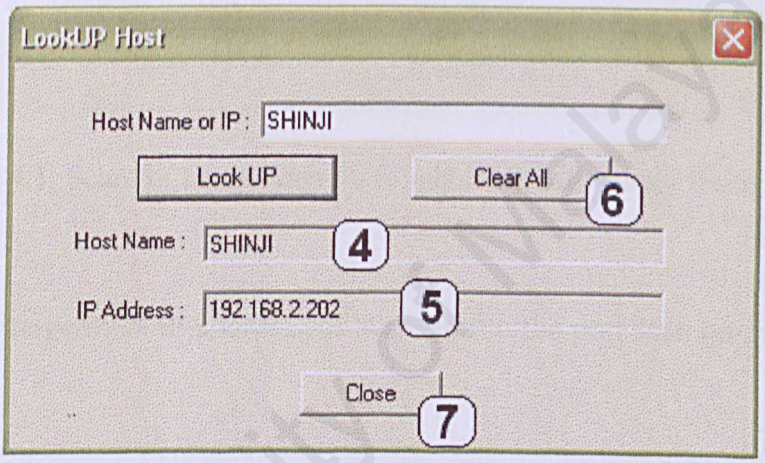

Figure 3.4: LookUP Host Interface with data and results

5. And the IP Address of the host will be displayed in the IP Address field.

6. Click on the "Clear All" button to clear all the fields in the LookUP Host interface.

7. Click on the "Close" button to exit the LookUP Host interface.

SCAN FOR OPEN PORTS ON SINGLE HOST

This is the Scan for Open Ports on Single Host section.

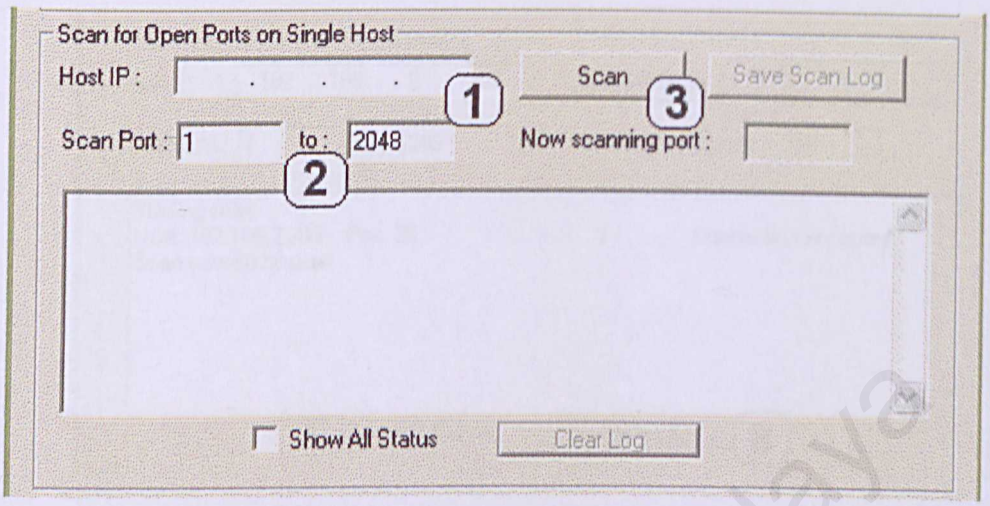

Figure 4.1: Scan for Open Ports on Single Host with empty fields

1. Enter the target host IP address you want to run the port scan on in the Host IP field.
2. Enter the range of ports you want to scan on the target host in the Scan Port field.
3. Click on the "Scan" button to start the port scanning.

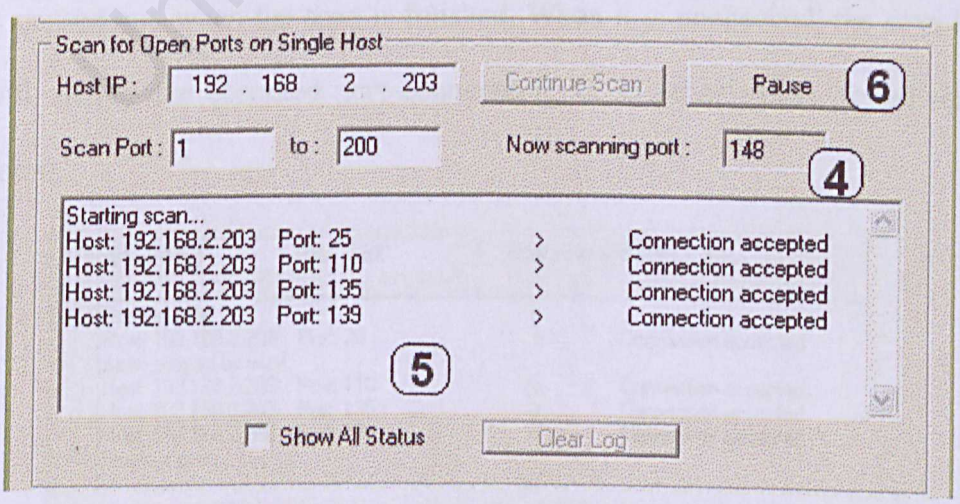

Figure 4.2: Port Scanning in action

4. The Now Scanning Port field will show you the current port being scanned.

5. The port scan result will be displayed in the scan log.
6. While the Network Scanner is still scanning, you can pause the scan by clicking on the "Pause" button.

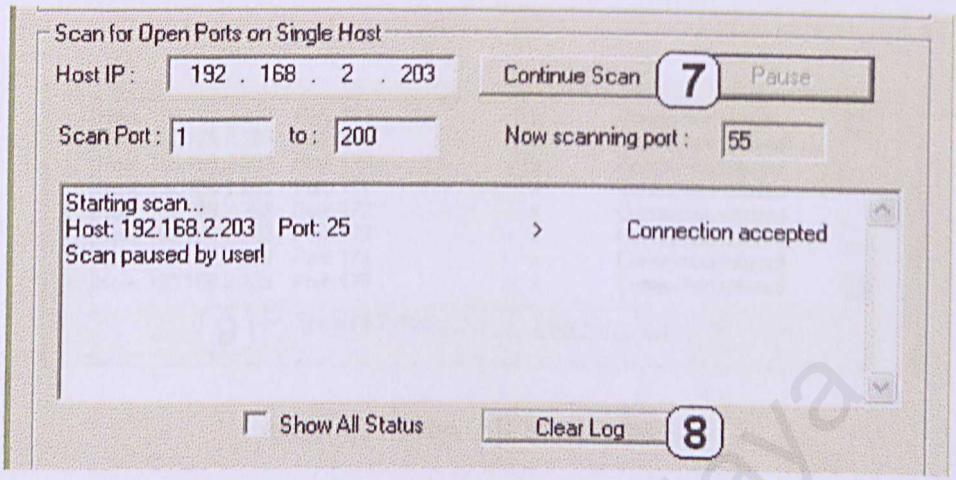

Figure 4.3: Scan paused

7. To continue the scan, simply click the "Continue Scan" button.
8. The "Clear Log" button is available when you paused the scan and after the scan is finished. You can click on the "Clear Log" button to clear the scan log at either time.
9. The "Show all status" checkbox is available before the scan, during the scan is paused and when the scan is finished. When it is unchecked, the scan log will display the accepted port connections, or in other words, found open ports.

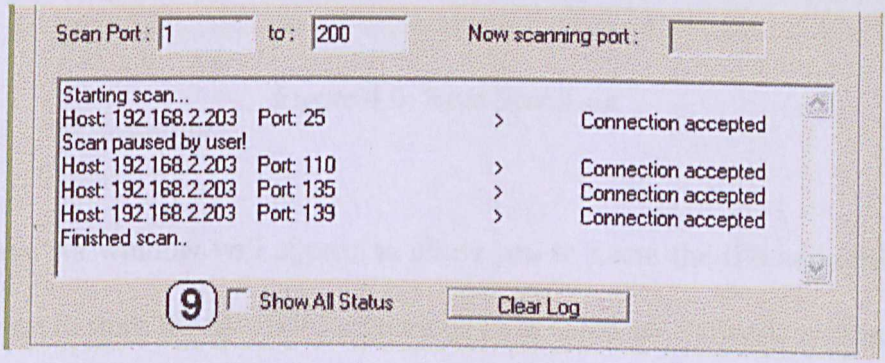

Figure 4.4: Show All Status unchecked

If you want the scan log to display all the port connection status, check on the "Show all status" checkbox before starting the scan, during the scan is paused or when starting a new scan.

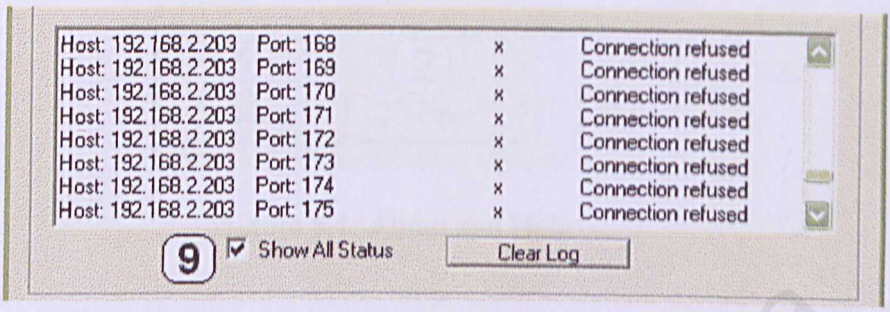

Figure 4.5: Show All Status checked

10. After the scan is finished, the scan log will inform you that it has finished scanning. The "Save Scan Log" button will be available and you can click it to save the scan log in a text file.

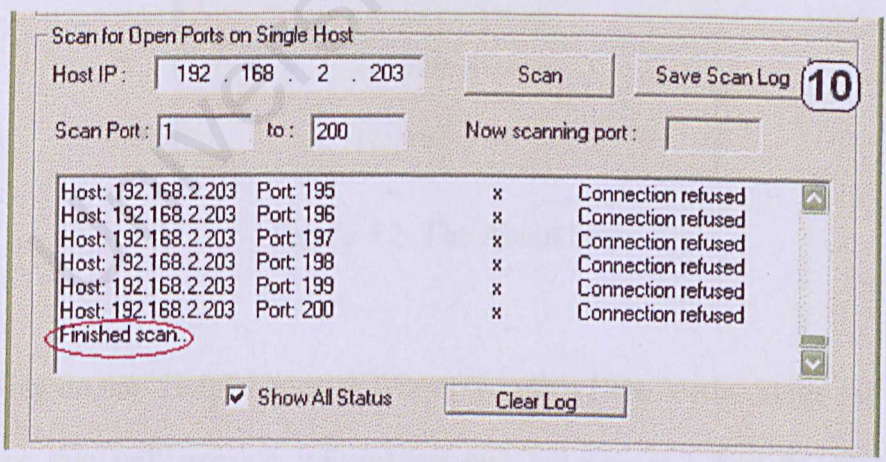

Figure 4.6: Save Scan Log

A Save As window will appear to allow you to name the file and choose a location to store the file like the previous Ping Hosts on Network section.

SUPPORTING FEATURES

At the bottom left of the main Network Scanner interface, there are another two buttons.

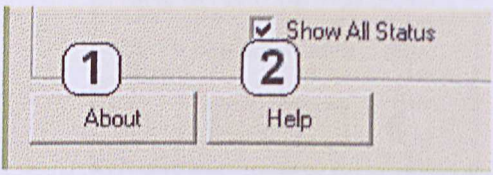

Figure 5.1: About and Help

1. When you click on the "About" button, the about box that display brief information about the Network Scanner will appear. Click "OK" to close the window.

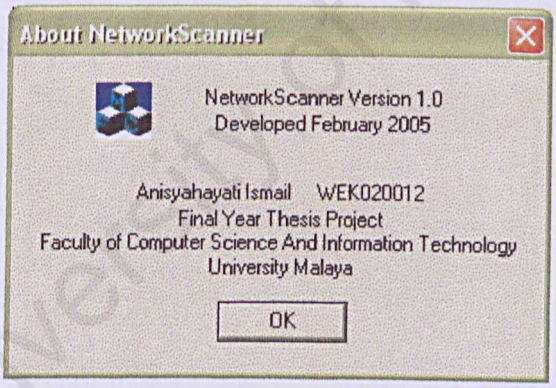

Figure 5.2: The About box

2. Clicking on the "Help" button will bring you to a Help Menu. There are three buttons that will provide information and aid you in using the Network Scanner program.

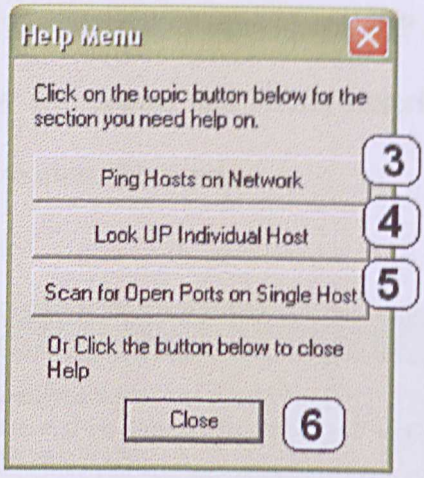

Figure 5.3: Help Menu

3. When you click on the first button, "Ping Hosts on Network", the section's help window will appear and briefly describe about the section's interface. Click "OK" to close the window.

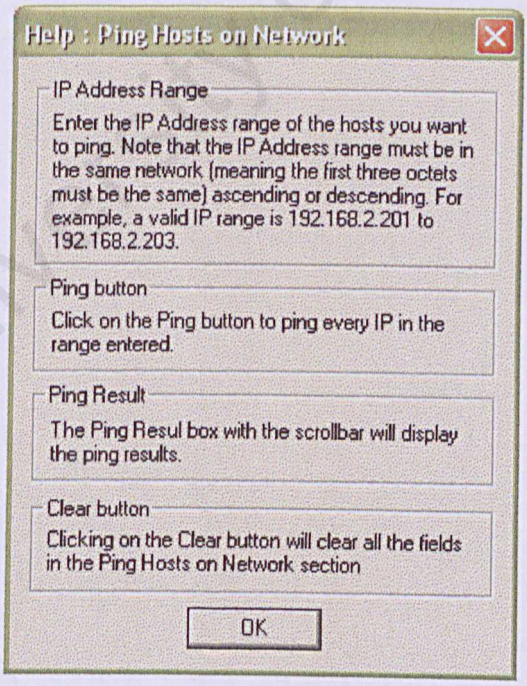

Figure 5.4: Help: Ping Hosts on Network

4. When you click on the second button, "Look UP Individual Host", the section's help window will appear and briefly describe about the section's interface. Click "OK" to close the window.

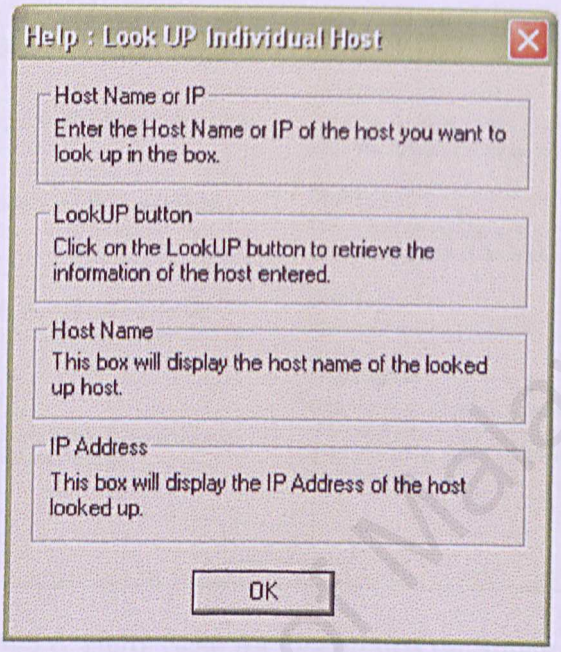

Figure 5.5: Help: Look UP Individual Host

5. When you click on the first button, "Scan for Open Ports on Single Host", the section's help window will appear and briefly describe about the section's interface. Click "OK" to close the window.

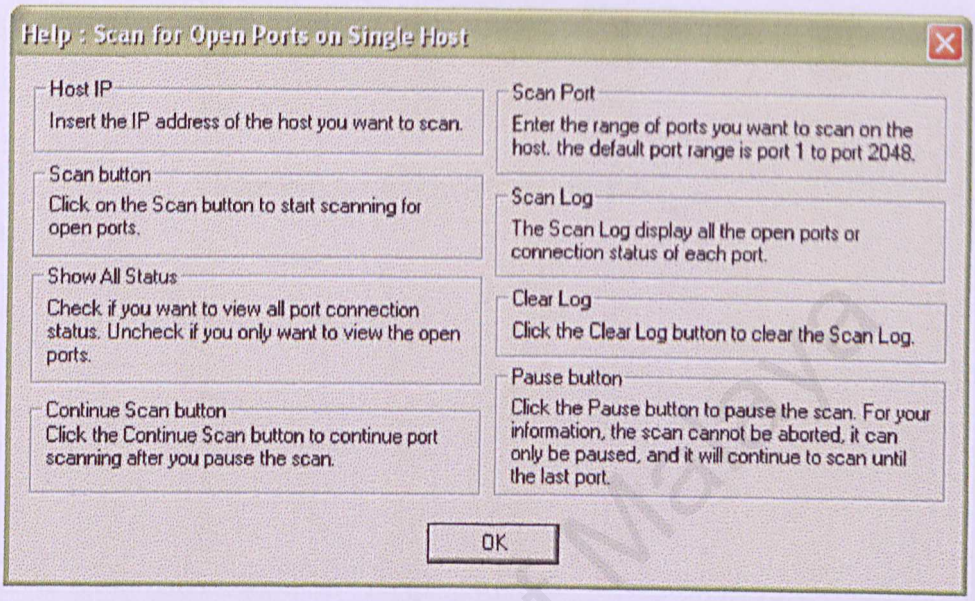

Figure 5.6: Help: Scan for Open Ports on Single Host

6. Click on the "Close" button to exit the Help Menu.

EXITING THE APPLICATION

Simply click on the "Exit" button at the bottom left corner of the interface or the "X" symbol on the top right corner of the applications interface.

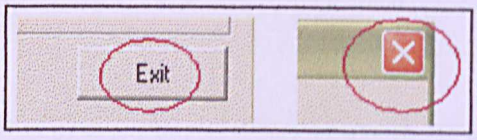

Figure 6.1: Exiting the Application

APPENDIX B:

PROJECT SOURCE CODE

THE NETWORK SCANNER

SOURCE CODES

Table of Contents

PING HEADER FILE

```
//
// Ping.h
//
#pragma pack(1)

#define ICMP_ECHOREPLY      0
#define ICMP_ECHOREQ  8

// IP Header -- RFC 791
typedef struct tagIPHDR
{
      u_char  VIHL;                 // Version and IHL
      u_char      TOS;              // Type Of Service
      short TotLen;                 // Total Length
      short ID;                     // Identification
      short FlagOff;          // Flags and Fragment Offset
      u_char      TTL;              // Time To Live
      u_char      Protocol;         // Protocol
      u_short     Checksum;         // Checksum
      struct      in_addr iaSrc;    // Internet Address - Source
      struct      in_addr iaDst;    // Internet Address - Dest
}IPHDR, *PIPHDR;

// ICMP Header - RFC 792
typedef struct tagICMPHDR
{
      u_char      Type;             // Type
      u_char      Code;             // Code
      u_short     Checksum;         // Checksum
      u_short     ID;                     // Identification
      u_short     Seq;              // Sequence
      char  Data;             // Data
}ICMPHDR, *PICMPHDR;

#define REQ_DATASIZE 32           // Echo Request Data size

// ICMP Echo Request
typedef struct tagECHOREQUEST
{
      ICMPHDR icmpHdr;
      DWORD dwTime;
      char  cData[REQ_DATASIZE];
}ECHOREQUEST, *PECHOREQUEST;

// ICMP Echo Reply
typedef struct tagECHOREPLY
{
      IPHDR ipHdr;
      ECHOREQUEST     echoRequest;
      char    cFiller[256];
}ECHOREPLY, *PECHOREPLY;

#pragma pack()
```

CONNECTIONCHECK HEADER FILE USED BY PING AND SCAN PORT

```
// CConnectionChk.h : header file
//
// Class used to check and determine the connection status
// before Network Scanner start to scan for open ports.

class CConnectionChk
{
      public:

            CConnectionChk();
            CConnectionChk(CString _ip, UINT _port)
            {
                  ip = _ip;
                  port = _port;
                  scanlog = "";
                  found = false;
                  finished = false;
            }

            CConnectionChk(CString CurHost)
            {
                  curhost = CurHost;
                  pinglog = "";
                  pinged = false;
            }

            CString scanlog, pinglog;
            CString ip;
            CString curhost;
            UINT port;
            bool found;
            bool finished;
            bool alive;
            bool offline;
            bool pinged;
};
```

TIMER FUNCTION SHARED BY PING AND PORT SCAN

```
//this function is used by the worker thread to send scan and
// ping results to the scan log and ping log box
void CNetworkScannerDlg::OnTimer(UINT nIDEvent)
{
if (nIDEvent == status_timer_id) //initialize timer
{
if(PingClicked)
{
if(cPingChks.GetSize() > 0)
{
CConnectionChk* cp = (CConnectionChk*) cPingChks.ElementAt(0);
//check current

if (cp->pinged)
{
cPingChks.RemoveAt(0, 1); //remove pinged host from the array

//if host is alive or not
if (cp->alive || cp->offline)
{
pinglog += cp->pinglog;
SetDlgItemText(IDC_PINGRESULT, pinglog);

((CEdit*)GetDlgItem(IDC_PINGRESULT))->
LineScroll(((CEdit*)GetDlgItem(IDC_PINGRESULT))->
GetLineCount(), 0);

delete cp; //empty the array.
}
}
}
else
{
//if timer still running and ping finished
if (status_timer_id !=0)
{
KillTimer(status_timer_id); //stop timer
status_timer_id = 0;
}

//inform user ping is finished
pinglog += "Finished ping..\r\n";
SetDlgItemText(IDC_PINGRESULT, pinglog);

//enable Scan button
CButton *pBtnScan = ( CButton * )GetDlgItem( IDC_BTN_SCAN );
pBtnScan->EnableWindow( true );

//enable "LookUP" button
CButton *pBtnLookUP = ( CButton * )GetDlgItem( IDC_BTN_LOOKUP
);
pBtnLookUP->EnableWindow( true );
```

```
//enable "Save Ping Log" button
CButton *pBtnPingL = ( CButton * )GetDlgItem( IDC_BTN_SAVEPING
);
pBtnPingL->EnableWindow( true );
}
return;
}
else
{
if(cConnectionChks.GetSize() > 0)
{
CConnectionChk*          cc          =          (CConnectionChk*)
cConnectionChks.ElementAt(0); //check current element/port

if (cc->finished)
{
//output at "Now scanning port :"
SetDlgItemInt(IDC_CURPORT_SCANNED, cc->port);
cConnectionChks.RemoveAt(0, 1); //remove scanned port

//if port is found and the "Show All Status" is checked
if (cc->found || IsDlgButtonChecked(IDC_CHECK_SHOWSTATALL))
{
//display open port scan result/all status scan result
log += cc->scanlog;
SetDlgItemText(IDC_SCANLOG, log);

((CEdit*)GetDlgItem(IDC_SCANLOG))->
LineScroll(((CEdit*)GetDlgItem(IDC_SCANLOG))->GetLineCount(),
0);

delete cc; //empty the array.
}
}
}
else
{
//if timer still running and scan finished
if (status_timer_id !=0)
{
KillTimer(status_timer_id); //stop timer
status_timer_id = 0;
}

//enable Clear Log button
CButton *pClearLog = ( CButton * )GetDlgItem( IDC_BTN_CLEARLOG
);
pClearLog->EnableWindow( true );

// show "Scan" button
CButton *pBtnScan = ( CButton * )GetDlgItem( IDC_BTN_SCAN );
                    pBtnScan->ShowWindow(SW_SHOW);

// show and enable "Save Scan" button
```

```
CButton    *pBtnSaveScan    =    (    CButton    *    )GetDlgItem(
IDC_BTN_SAVESCAN );
pBtnSaveScan->ShowWindow(SW_SHOW);
pBtnSaveScan->EnableWindow( true );

// hide "Pause" and "Continue Scan" buttons
CButton    *pBtnPause    =    (    CButton    *    )GetDlgItem(
IDC_BTN_PAUSESCAN );
pBtnPause->ShowWindow(SW_HIDE);

CButton    *pBtnContScan    =    (    CButton    *    )GetDlgItem(
IDC_BTN_CONTSCAN );
pBtnContScan->ShowWindow(SW_HIDE);

// empty the "Now scanning port: "
SetDlgItemText(IDC_CURPORT_SCANNED, "");

//inform user scan is finished
log += "Finished scan..\r\n";
SetDlgItemText(IDC_SCANLOG, log);

((CEdit*)GetDlgItem(IDC_SCANLOG))->
LineScroll(((CEdit*)GetDlgItem(IDC_SCANLOG))->GetLineCount(),
0);

//enable "Ping" button
CButton *pBtnPing = ( CButton * )GetDlgItem( IDC_BTN_PING );
                    pBtnPing->EnableWindow( true );

//enable "LookUP" button
CButton *pBtnLookUP = ( CButton * )GetDlgItem( IDC_BTN_LOOKUP
);
pBtnLookUP->EnableWindow( true );

}
}
}

PingClicked = false;
CDialog::OnTimer(nIDEvent);
}
```

PING HOST ON NETWORK MAIN FUNCTIONS

Function: On click of the "Ping" button

```
void CNetworkScannerDlg::OnBtnPing()
{
PingClicked = true;
BtnPingClicked = true;

//disable Scan button
CButton *pBtnScan = ( CButton * )GetDlgItem( IDC_BTN_SCAN );
pBtnScan->EnableWindow( false );

//disable "LookUP" button
CButton *pBtnLookUP = ( CButton * )GetDlgItem( IDC_BTN_LOOKUP
);
pBtnLookUP->EnableWindow( false );

//disable "Ping" button
CButton *pBtnPing = ( CButton * )GetDlgItem( IDC_BTN_PING );
pBtnPing->EnableWindow( false );

LPHOSTENT lpHostEnd;
LPHOSTENT lpHostStart;
CString   cstrHost;
CString   cstrHostStart;
CString   cstrHostEnd;
CString   pingresultsuccess1;
CString   pingresultsuccess2;
CString      pingresult;
CString   PingStatus;
struct    in_addr *startptr;
struct    in_addr *endptr;

pingresult = "";
SetDlgItemText(IDC_PINGRESULT, pingresult);

GetDlgItemText(IDC_PINGDESTSTART, cstrHostStart);
GetDlgItemText(IDC_PINGDESTEND, cstrHostEnd);

//////Lookup start host
lpHostStart = gethostbyname(cstrHostStart);

if(lpHostStart == NULL)
{
     AfxMessageBox("Please enter a valid host!");
     return;
}

startptr = (struct in_addr *)lpHostStart->h_addr_list[0];
```

```
//get ip address by octet (break it to octets)
int aS = startptr->S_un.S_un_b.s_b1;
int bS = startptr->S_un.S_un_b.s_b2;
int cS = startptr->S_un.S_un_b.s_b3;
int dS = startptr->S_un.S_un_b.s_b4;

CString StartH, EndH;
StartH.Format("%d.%d.%d.%d", aS, bS, cS, dS);

//////Lookup end host
lpHostEnd = gethostbyname(cstrHostEnd);

if(lpHostStart == NULL && lpHostEnd == NULL)
{
      AfxMessageBox("Please enter a valid host!");
      return;
}

endptr = (struct in_addr *)lpHostEnd->h_addr_list[0];

//get ip address by octet (break it to octets)
int aE = endptr->S_un.S_un_b.s_b1;
int bE = endptr->S_un.S_un_b.s_b2;
int cE = endptr->S_un.S_un_b.s_b3;
int dE = endptr->S_un.S_un_b.s_b4;

EndH.Format("%d.%d.%d.%d", aE, bE, cE, dE);

//start of first three octet comparison
//first octet
if(aS != aE)
{
      AfxMessageBox("Please enter valid IP range\ni.e
      192.168.2.203 to 192.168.2.210");

      //enable "Ping" button
      CButton *pBtnPing = ( CButton * )GetDlgItem(
      IDC_BTN_PING );
      pBtnPing->EnableWindow( true );

      //reset status text
      PingStatus = "Waiting for input..";
      SetDlgItemText(IDC_STATUS1, PingStatus);
}
//second octet
else if(bS != bE)
{
      AfxMessageBox("Please enter valid IP range\ni.e
      192.168.2.203 to 192.168.2.210");

      //enable "Ping" button
      CButton *pBtnPing = ( CButton * )GetDlgItem(
      IDC_BTN_PING );
      pBtnPing->EnableWindow( true );

      //reset status text
```

```
        PingStatus = "Waiting for input..";
        SetDlgItemText(IDC_STATUS1, PingStatus);
}

//third octet
else if(cS != cE)
{
        AfxMessageBox("Please enter valid IP range\ni.e
        192.168.2.203 to 192.168.2.210");

        //enable "Ping" button
        CButton *pBtnPing = ( CButton * )GetDlgItem(
        IDC_BTN_PING );
        pBtnPing->EnableWindow( true );

        //reset status text
        PingStatus = "Waiting for input..";
        SetDlgItemText(IDC_STATUS1, PingStatus);
        }
        else
        {
                if(dS > dE)
                {
                        int tmp;

                        tmp = dS;
                        dS = dE;
                        dE = tmp;
                }

                //Change the status line
                PingStatus = "Ping results :";
                SetDlgItemText(IDC_STATUS1, PingStatus);

                //Ping Here!!
                CString Hosts = "";
                CString CurHost;

                if (status_timer_id == 0)
                {
                        //set the timer for worker thread
                        status_timer_id = SetTimer(1, 25, 0);
                }

                //start of host loop
                for(int i = dS; i <= dE; i++)
                {
                CurHost.Format("%d.%d.%d.%d", aS, bS, cS, i);

                //initialize
                CConnectionChk* cp = new CConnectionChk(CurHost);
                cPingChks.Add(cp); // increment cp for next host??
                //run the ping using the worker thread
                AfxBeginThread(runPing, (void*) cp,
                THREAD_PRIORITY_NORMAL); //run the scan using the
                worker thread
```

```
            }

        }//end of first three octet comparison

        //enable "Clear" button
        CButton *pBtnClear = ( CButton * )GetDlgItem(
        IDC_BTN_CLEARPING );
        pBtnClear->EnableWindow( true );

}
```

Function: Run Ping Thread

```
UINT runPing (LPVOID pParam)
{
      LPHOSTENT lpCurHost;
      struct    sockaddr_in saDest;
      struct    sockaddr_in saSrc;
      DWORD   dwTimeSent;
      DWORD   dwElapsed;
      u_char    cTTL;
      int       nLoop;
      int       nRet;

      //receive CConnectionChk parameter from the
      //CNetworkScannerDlg::OnBtnPing() function
      CConnectionChk* cp = (CConnectionChk*) pParam;

      CString tmp = cp->curhost;

      cp->alive = false;
      cp->offline = false;

      SOCKET        rawSocket;

      //Create raw socket
      rawSocket = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
      if(rawSocket == SOCKET_ERROR)
      {
            AfxMessageBox("Could not open socket.");
            return 0;
      }

      //LookUp CurHost
      lpCurHost = gethostbyname(tmp);

      if(lpCurHost == NULL)
      {
            AfxMessageBox("Please enter a valid host!");
            return 0;
      }
```

```
//Setup destination socket address
saDest.sin_addr.s_addr = *((u_long FAR *)(lpCurHost
->h_addr));
saDest.sin_family = AF_INET;
saDest.sin_port = 0;

//start of ping loop
//Ping once but use for loop to enable break
for(nLoop = 0; nLoop < 1; nLoop++)
{
      //Send ICMP echo request
      SendEchoRequest(rawSocket, &saDest);


      //Use function select() (from winsock) to wait for
      //data to be received
      nRet = WaitForEchoReply(rawSocket);

      if(nRet == SOCKET_ERROR)
      {
            CString pingerror;
            pingerror.Format("Error waiting for
            packet!");
            cp->pinglog += pingerror + "\r\n";
            break;
      }

      else if(!nRet)
      {
            CString pingfail;
            pingfail.Format("Host : %s : Request timed
            out.", tmp);
            cp->pinglog += pingfail + "\r\n";
            cp->offline = true;
            break;
      }

     else
     {
            //Receive reply
            dwTimeSent = RecvEchoReply(rawSocket, &saSrc,
            &cTTL);

            //Calculate elapsed time
            dwElapsed = GetTickCount() - dwTimeSent;

            //display ping result in the Ping Result box
            CString pingsuccess;
            pingsuccess.Format("Reply from: %s: bytes=%d
            time=%ldms TTL=%d",
            inet_ntoa(saSrc.sin_addr),
            REQ_DATASIZE, dwElapsed, cTTL);
            cp->pinglog += pingsuccess + " \r\n";
            cp->alive = true;
     }
```

```
    }

    nRet = closesocket(rawSocket);

    if(nRet == SOCKET_ERROR)
        AfxMessageBox("Cannot close socket");

    cp->pinged = true;
    return 0;
}
```

Function: Send Echo Request

```
//This function fills in the echo request header and send to
//destination
int SendEchoRequest(SOCKET s,LPSOCKADDR_IN lpstToAddr)
{
    static ECHOREQUEST echoReq;
    static nId = 1;
    static nSeq = 1;
    int nRet;

    // Fill in echo request
    echoReq.icmpHdr.Type        = ICMP_ECHOREQ;
    echoReq.icmpHdr.Code        = 0;
    echoReq.icmpHdr.Checksum    = 0;
    echoReq.icmpHdr.ID          = nId++;
    echoReq.icmpHdr.Seq         = nSeq++;

    // Fill in some data to send
    for (nRet = 0; nRet < REQ_DATASIZE; nRet++)
        echoReq.cData[nRet] = ' '+nRet;

    // Save tick count when request sent
    echoReq.dwTime   = GetTickCount();

    // Put data in packet and compute checksum
    echoReq.icmpHdr.Checksum = in_cksum((u_short *)&echoReq,
    sizeof(ECHOREQUEST));

    // Send the echo request

    nRet = sendto(s,                                    // socket
                  (LPSTR)&echoReq,                      // buffer
                  sizeof(ECHOREQUEST),
                  0,                                    // flags
                  (LPSOCKADDR)lpstToAddr, // destination
                  sizeof(SOCKADDR_IN));   // address length

    if (nRet == SOCKET_ERROR)
```

```
            AfxMessageBox("SendTo() error");

      return (nRet);
}
```

Function: Wait for Echo Reply

```
//This function uses the function select() (from winsock) to
//determine when data is waiting to be read
int WaitForEchoReply(SOCKET s)
{
      struct timeval Timeout;
      fd_set readfds;

      readfds.fd_count = 1;
      readfds.fd_array[0] = s;
      Timeout.tv_sec = 5;
      Timeout.tv_usec = 0;

      return(select(1, &readfds, NULL, NULL, &Timeout));
}
```

Function: Receive Echo Reply

```
//This function receives incoming data and parse out fields
DWORD RecvEchoReply(SOCKET s, LPSOCKADDR_IN lpsaFrom, u_char
*pTTL)
{
      ECHOREPLY echoReply;
      int nRet;
      int nAddrLen = sizeof(struct sockaddr_in);

      // Receive the echo reply
      nRet = recvfrom(s,              // socket
            (LPSTR)&echoReply,        // buffer
            sizeof(ECHOREPLY),        // size of buffer
            0,                        // flags
            (LPSOCKADDR)lpsaFrom,     // From address
            &nAddrLen);               // pointer to address len

      // Check return value
      if (nRet == SOCKET_ERROR)
            AfxMessageBox("Error receiving");
```

```
        // return time sent and IP TTL
        *pTTL = echoReply.ipHdr.TTL;
        return(echoReply.echoRequest.dwTime);
}
```

Function: Checksum Routine

```
////////////////////////////////////////////////////////////////
// Mike Muuss' in_cksum() function and his comments from the
// original ping program
//
//  Author -    Mike Muuss
//              U. S. Army Ballistic Research Laboratory
//              December, 1983
////////////////////////////////////////////////////////////////
/*
*                   I N _ C K S U M
* Checksum routine for Internet Protocol family headers (C
* Version)
*/
u_short in_cksum(u_short *addr, int len)
{
      register int nleft = len;
      register u_short *w = addr;
      register u_short answer;
      register int sum = 0;

//Our algorithm is simple, using a 32 bit accumulator (sum),
//we add sequential 16 bit words to it, and at the end, fold
//back all the carry bits from the top 16 bits into the
//lower 16 bits.
      while( nleft > 1 )  {
            sum += *w++;
            nleft -= 2;
      }

      /* mop up an odd byte, if necessary */
      if( nleft == 1 ) {
            u_short    u = 0;

            *(u_char *)(&u) = *(u_char *)w ;
            sum += u;
      }

// add back carry outs from top 16 bits to low 16 bits
      sum = (sum >> 16) + (sum & 0xffff);//add hi 16 to low 16
      sum += (sum >> 16);                  //add carry
      answer = ~sum;                       // truncate to 16 bits
      return (answer);
}
```

LOOK UP INDIVIDUAL HOST MAIN FUNCTION

Function: When user clicks on the "Look UP" button

```
void CLookUPHostDlg::OnBtnLookuph()
{
	CString cstrHost;
	LPHOSTENT lpHostName;
	LPHOSTENT lpHostAddr;
	char* hostName;
	struct	sockaddr_in hostAdd;
	CString hostAddress;

	unsigned long addr;

	GetDlgItemText(IDC_HOSTTOLOOK, cstrHost);
	addr = inet_addr(cstrHost);

	//Lookup host
	//get host address
	lpHostAddr = gethostbyname(cstrHost);

	if(lpHostAddr == NULL)
	{
		AfxMessageBox("Host is not valid or not online.");
		return;
	}

	hostAdd.sin_addr.s_addr = *((u_long FAR *)(lpHostAddr
	->h_addr));

	//display host address
	hostAddress.Format("%s", inet_ntoa(hostAdd.sin_addr));
	SetDlgItemText(IDC_IPADDLU, hostAddress);

	//get host name
	lpHostName = gethostbyaddr((char *) &addr, 4, AF_INET);

	if(lpHostAddr != NULL)
	{
		hostName = (lpHostAddr->h_name);
		//display host name
		SetDlgItemText(IDC_HOSTNAMELU, hostName);
		return;
	}

}
```

SCAN FOR OPEN PORTS ON SINGLE HOST MAIN FUNCTIONS

Function: The scan() function

```
void CNetworkScannerDlg::scan()
{
	//display "Starting scan.." in the scan log box
	log = "Starting scan...\r\n";
	SetDlgItemText(IDC_SCANLOG, log);

	//get the IP number entered by user.
	GetDlgItemText(IDC_SCAN_IP, IP);

	if (status_timer_id == 0)
	{
		//set the timer for worker thread
		status_timer_id = SetTimer(1, 25, 0);
	}

	UINT startPort, endPort, port;

	//get the port values entered by user
	startPort  = GetDlgItemInt(IDC_START_PORT);
	endPort    = GetDlgItemInt(IDC_END_PORT);

	for (port = startPort; port <= endPort; port++)
	{
		//initialize connection to individual port
		CConnectionChk* cc = new CConnectionChk(IP, port);
		cConnectionChks.Add(cc); // increment cc
		//run the scan using the worker thread
		AfxBeginThread(runScan, (void*) cc,
		THREAD_PRIORITY_NORMAL);
	}

	//enable "Pause" button
	CButton   *pBtnPause   =   (   CButton   *   )GetDlgItem(
	IDC_BTN_PAUSESCAN );
	pBtnPause->EnableWindow(true);

	//disable Clear Log button
	CButton   *pClearLog   =   (   CButton   *   )GetDlgItem(
	IDC_BTN_CLEARLOG );
	pClearLog->EnableWindow( false );

}
```

Function: The Run Scan thread

```
UINT runScan (LPVOID pParam)
{
	//receive	CConnectionChk	parameter	from	the
	//CNetworkScannerDlg::scan() function
	CConnectionChk* cc = (CConnectionChk*) pParam;

	cc->finished = false;
	CString tmp = "";
	CSocket s;

	//create Windows socket(STREAM SOCKET) and attach it to
	//the specified IP address
	s.Create(0, SOCK_STREAM, "");

	tmp.Format("%d", cc->port);

	//if the connection is successful
	if (s.Connect((LPCTSTR) cc->ip, cc->port))
	{
	cc->scanlog += "Host: " + cc->ip + "	Port: " + tmp +
	"\t\t>\tConnection accepted\r\n";
	cc->found = true;
	}
	else
	{
		switch(GetLastError())
		{
		case WSAECONNREFUSED://if connection refused
			cc->scanlog += "Host: " + cc->ip + "	Port:
			" + tmp + "\t\tx\tConnection refused\r\n";
			break;

		case WSAETIMEDOUT://if connection timed out
			cc->scanlog += "Host: " + cc->ip + "	Port:
			" + tmp + "\t\to\tConnection timed out\r\n";
			break;

		default://if connection status unknown
			cc->scanlog += "Host: " + cc->ip + "	Port:
			" + tmp + "\t\t!\tUnknown error when trying
			to connect\r\n";
			break;
		}
	}

	s.Close();
	cc->finished = true; // the port have been scanned

	return 0;

}
```

REFERENCES

Software Development and SDLC [online]. StartVBdotNet.com

Available from http://www.startvbdotnet.com/sdlc/sdlc.htm

The System Development Life Cycle [online]

Available from http://www.muskalipman.com/VBObjects/SDLC.pdf

Terms and Definitions [online]

Available from http://www.webopedia.com/TERM/

Terms and Definitions [online]

Available from http://www.wikipedia.com/

McNab, C. *Network Scanner Types* [online] O'Reilly. Available

http://www.oreilly.com/catalog/networksa/chapter/ch04.pdf

Forno, R,. and Kenneth R. van Wyk. *Ethereal and Nmap* [online] O'Reilly

OnLamp.com. Available from

http://www.onlamp.com/pub/a/onlamp/excerpt/incidentres_07/index.html

Johns, St. M. 1993. *Identification Protocol RFC1413* [online]. US Department of

Defense. Available from http://www.ietf.org/rfc/rfc1413.txt

Sanfilippo, S. 2004. *What's Hping* [online].

Available from http://wiki.hping.org/

Sharpe, R. *Ethereal User Guide* [online]. Ethereal.com

Available from http://ethereal.0ni0n.org/docs/user-guide/

2000. *Description of C++*. [online]. The C++ Resource Network

Available from http://www.cplusplus.com/info/description.html

Smart, J. *What is WxWidgets?* [online]. Anthemion Sofware.

http://www.wxwindows.org/

Examples of Network Monitoring Tools [online]

Available from http://www.ciac.org/ciac/ToolsDOSNetwork.html

February 2005. *Winsock Functions* [online]

Available from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/winsock_functions.asp

Chand, Mahesh. September 2000.*Multithreading using MFC in Plain English: Part I* [online]. Available from

http://www.mindcracker.com/mindcracker/c_cafe/mt/mtt1.asp

Cumming, M. *Multithreading Applications Using MFC* [online]. Available from http://www.murrayc.com/learning/windows/multithreading.shtml#ProcessesAndTheirThreads

Microsoft: Visual C++ FAQS [online]. Available from http://www.tek-tips.com/faqs.cfm?fid=5162

Microsoft Corporation. 2005. *Windows Sockets in MFC* [online]. Available from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vccore/html/_core_windows_sockets_in_mfc.asp

Microsoft Corporation. 2005. *CSocket* [online]. Available from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_wcemfc4/html/aflrfCSocket.asp

Wheeler, J. August 1998. *An ICMP Class for MFC* [online]. Available from http://www.codeguru.com/Cpp/I-N/internet/network/article.php/c3395/#Demo

Ellis, J.J.. 2002. *Creating Threads in MFC* [online]. Available from http://www.apostate.com/programming/mfc-threads.html

FunctionX. *Visual C++ Tutorial* [online]. Available from http://www.functionx.com/visualc/howto/menu.htm

Microsoft Corporation. 2005. *CDC Class* [online]. Available from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_mfc_cdc.asp

Lee Chin Han (2003/2004) *Network Traffic Monitoring System with Graphical Approach.* Degree of Computer Science, University of Malaya.

Tan Yu Jin (2002) *Internet Information Server Scanner: Scanning on URL Vulnerabilities.* Master Degree of Computer Science, University of Malaya.

Suhaila Shawal (1998/1999) *LAN Snanner.* Degree of Computer Science, University of Malaya.

Deitel H. M., Deitel P. J., Nieto T. R. & Strassberger E.T.. (2000). *Getting Started with Microsoft Visual C++ 6 with an Introduction to MFC.* New Jersey, A Pearson Education Company.

BIBLIOGRAPHY

Forouzan, B. A. (2002) *Data Communictaions and Networking.* New York, McGraw-Hill.

Stallings, W. (2003) *Network Security Essentials: Applications and Standards.* New Jersey, Prentice Hall.

Maiwald, E. (2002) *Network Security: A Beginners Guide.* Osborne, Brandon A. Nordin.

Schach, S. R. (2004) *Object Oriented and Classical Software Engineering.* New York, Elizabeth A. Jones.

Whitten, J. L. *et al.* (2003) *System Analysis and Design Methods.* New York, Brent Gordon.

Sommerwille, J. (1995) Software Engineering 5th Edition, Addison Wesley.

Deitel H. M., Deitel P. J., Nieto T. R. & Strassberger E.T.. (2000). *Getting Started with Microsoft Visual C++ 6 with an Introduction to MFC.* New Jersey, A Pearson Education Company.