NETWORK HANDLING KIT

A PROJECT REPORT

Submitted by

P.ARUN VENKATESH BABU	**71201205008**
R.JEGATHEESH	**71201205020**
A.SAKTHIVEL	**71201205050**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Kumaraguru College of Technology, Coimbatore

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2005

BONAFIDE CERTIFICATE

Certified that this project report **"NETWORK HANDLING KIT"** is the bonafide work of **"P.ARUN VENKATESH BABU, R.JEGATHEESH, A.SAKTHIVEL"** who carried out the project work under my supervision.

SIGNATURE

Prof. Dr. S.Thangaswamy

HEAD OF THE DEPARTMENT

Computer Science and Engineering
Kumaraguru College of Technology
Coimbatore - 641006.

SIGNATURE

Mrs. N. Chitra devi

SUPERVISOR

Information Technology
Kumaraguru College of Technology
Coimbatore - 641006.

The candidates with University Register Nos. **71201205008, 71201205020 71201205050, were** examined by us in the project viva-voce examination held on .....18:04:05.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We

Arun venkatesh babu P.	71201205008
Jegatheesh R.	71201205020
Sakthivel A.	71201205050

Declare that the project entitled "**NETWORK HANDLING KIT**", submitted in partial fulfillment to Anna University as the project work of Bachelor Of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of **Mrs.N.Chitra Devi M.E.,** Senior Lecturer, Department of Information Technology , Kumaraguru College Of Technology, Coimbatore.

Place: Coimbatore

Date : 18.04.05

[Arun venkatesh babu P.]

[Jegatheesh R.]

[Sakthivel A.]

Project Guided by

[Mrs.N.Chitra Devi M.E.,]

ACKNOWLEGEMENT

The exhilaration achieved upon the successful completion of any task should be defiantly shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of my college for having provided us excellent facilities to work with. we express my deep gratitude to our Principal **Dr.K.K.Padmanabhan B.Sc(Engg), M.Tech, Ph.D.,** for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head of the department, **Prof.S.Thangasamy B.E(Hons), Ph.D.,** whose consistent support and enthusiastic involvement helped us great deal.

We are greatly indebted to our beloved guide **Mrs.N.Chitra Devi M.E.,** Senior Lecturer, Department Information Technology for her excellent guidance and timely support during the course of this project. As a token of my esteem and gratitude, we honor her for the assistance towards this cause.

We also thank my project coordinator **Mr.K.R.Baskaran M.S.,** for his invaluable assistance. we also feel elated in manifesting my deep sense of gratitude to all the staff and the lab technicians in the Department of Computer science and Engineering.

We feel proud to pay my respectful thanks to my parents for their enthusiasm and encouragement and also we thank my friends who have associated themselves to bring out this project successfully.

ABSTRACT

The project network handling kit is a collection of tools to manage a network. The kit does the function of mapping, monitoring, alerting and proactively troubleshooting the network. The kit gives an edge over various tools available for network handling as it supports a web based interface. The kit seamlessly integrates various tools available for networking and adds various functionalities to support web based interface.

The mapping module gives the IP address of the host connected and the number of hosts that are connected in the network. The monitoring module gives the details about the data transfer between the hosts and the server in the network. Monitoring also does the function of finding the anomalies of the network and fixing the problems in the network.

The alerting function does the process of giving the notification to the network administrator about the problems created by the host. The alerting is done visually by giving alert flags to the administrator. The troubleshooting function does the process of fixing the problem on the basics of time interval. Various indications at a particular time interval will help the administrator to trouble shoot the network.

The user friendly interface provided by this system eases the work of the administrator and helps the end user to know the performance of the network

LIST OF FIGURES

S.No	Figure-name	Page no.
1.	NETWORK HANDLING KIT	40
2	MAPPING	41
3.	MONITORING	42
4.	ALERTING	44
5.	TROUBLE SHOOTING	46
6.	TCP DUMP	47
7.	SYSTEM SERVICES	48

LIST OF ABBREVIATIONS

1.	NHK	Network Handling Kit
2.	OpenNMS	Open Network Monitoring Station
3.	LibPCap	Library Packet Capture
4.	IP	Internet Protocol
5.	LAN	Local Area Network
6.	HTTP	Hyper Text Transfer Protocol
7.	XML	extended Mark-up Language

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGENO
	ABSTRACT	**v**
	LIST OF FIGURES	**vi**
	LIST OF ABBREVIATIONS	**vii**
1.	**INTRODUCTION**	1
	1.1 Existing system	2
	1.2 Proposed system	3
2.	**SOFTWARE REQUIREMENT ANALYSIS**	4
	2.1 Project Definition	4
	2. 2 Project Plan	4
3.	**SOFTWARE REQUIREMENT SPECIFICATION**	6
4.	**SYSTEM STUDY**	11
	4.1 Introduction to NHK	11
	4.2 Introduction to C	11
	4.3 Html	12
	4.4 XML	15
5.	**SOFTWARE IMPLEMENTATION MODEL**	18
	5.1 Module description	18
	5.1.1 Mapping	18
	5.1.2 Monitoring	19
	5.1.3 Alerting	21

5.1.4 Trouble Shooting 22

6. PRODUCT TESTING 23

7. FUTURE ENHANCEMENTS 26

8. CONCLUSION 27

9. APPENDIX 28

9. 1 Sample Code 28

9. 2 Sample Output 40

10. REFERENCES 49

1. INTRODUCTION

The main aim of this project is to create a graphical user interface for two existing Linux based tools that helps the network administrator to easily recognize the connected hosts in the network and the kind of traffic generated by them, the interface can be accessed from any client terminal across the network. This project is divided into four modules namely as mapping, monitoring, alerting and troubleshooting.

PROBLEM DESCRIPTION:

This project tries to provide a graphical user interface which would help the administrator to understand the network and help him to take actions regarding the anomalies in the network. The graphical user interface is designed for two Linux based tools namely OpenNMS and Nagios, these tools serve the purpose of mapping, monitoring and alerting. Currently these tools provide information regarding the network which is not easily recognizable by the administrator and they don't provide a graphical user interface.

Network handling kit tries to integrate these two tools and provide a user interface which would provide information generated by them in a graphical form. The project classifies the interface into various sections by which the administrator can get information like the hosts currently connected, the traffic generated by them, number of hosts that produce risk in the network and various information like the total network load etc.,

The administrator can access this information from any client terminal provided that he connects to the server with the help of a port number of the application. The connection can be made through a

browser which supports graphical information, independent of the operating system.

The kit is developed in such a way that the information regarding the network changes dynamically at a certain time interval. The change in the network is got by sniffing the packets in the network. There is some notification about the risk generated in the network, by which the administrator can take some necessary action to reduce it.

1.1 Existing System and Limitations:

Existing network handling tools for Linux like openNMS and Nagios are not integrated and they don't provide a web interface for the end-users. It has several limitations.

OpenNMS:

OpenNMS is a tool used for discovering the hosts in a network which efficiently maps all the host connected to the network using the IP and MAC addresses. Currently OpenNMS doesn't hold any user interface. Lack of user interface makes the work of the administrator hard.

Nagios:

Nagios is a system and network monitoring application. It watches hosts and services that are currently connected in the network, alerting when things go bad and when they get better. In short Nagios is a tool used for monitoring and alerting. Monitoring is done for various protocols which produce a result that gives an output of traffic of various protocols.

Limitations:

- No proper integrated tools for network handling.

Tools like OpenNMS and Nagios are independent tools and

- Lack of web interface for these tools.

OpenNMS and Nagios don't have web interface, understanding of their output is not subjected to a naive user.

- Cannot be accessed at client terminals.

Details generated by these tools can be accessed only at the server where these tools are installed.

1.2 Proposed System and Advantages:

Our intention is to develop a network handling kit which would do the functions of openNMS and Nagios as total. Develop a web interface which would help system administrator and naïve users understand and manage a network. Provide information regarding the packets transferred and the protocol transfer as a whole.

Advantages:

- Easy understanding of mapped and monitored data
- Can be accessed at all client terminals.
- Graphical user Interface for network handling.

2. SYSTEM REQUIREMENTS ANALYSIS

System study is an activity that encompasses most of the tasks that we have collectively called computer system engineering. System study is conducted with the following objectives.

- Identify the needs.
- Evaluate the system concept for feasibility.
- Perform economic and technical analysis.
- Allocate function to hardware, software, people and other system elements.
- Create system definition that forms the foundation for all the subsequent engineering works.

2.1 PRODUCT DEFINITION:

The processing of creating the web interface for the Linux based network provides the traffic information about the hosts connected in the network. This will also gives network load and the throughput like data sent and data received through the network. The web interface gives the information about the host configuration and the protocols that are used in the host.

2.2 PROJECT PLAN:

The requirement phase deals with the sequence of activities in producing the Software Requirement Specification (SRS) document. It must be ensured that all the requirements of the software are elicited and analyzed. In other words the needs of the system are identified.

In the problem analysis phase, the current system is analyzed by study of the existing materials, and the changes to be made in the proposed system are decided upon. A clear understanding of the needs of the system must be framed. Analysis leads to actual specification.

Designing aims at how to satisfy the needs of the system. The different modules of the system and the interaction between these modules to produce the desired functionality are identified. During detailed design, the internal logic of each module and their algorithmic design is specified. The major and important decisions are made in this phase.

Coding is the process of translating the design into code. The code developed must be easy to understand. Well-written code can reduce testing and maintenance effort.

Testing is done to check if the requirements of the user are met. It also involves in checking the functionality of each module as per the design document.

3. SOFTWARE REQUIREMENT SPECIFICATIONS

Contents:

3.1 Introduction

- 3.1.1 Purpose
- 3.1.2 Scope
- 3.1.3 Definition
- 3.1.4 Abbreviation
- 3.1.5 References

3.2 General Description

- 3.2.1 Product Overview
- 3.2.2 User Characteristics
- 3.2.3 General Constraints
- 3.2.4 General assumptions

3.3 Specific Requirements

- 3.3.1 Inputs and Outputs
- 3.3.2 Functional requirements
- 3.3.3 System Requirements
- 3.3.4 Performance constraints
- 3.3.5 Software Constraints

3.1 Introduction

3.1.1 Purpose:

The purpose of this document is to specify the requirements of project "Network handling Kit". It describes the interfaces for the system. The document also bridges the communication gap between the customer and the analyst.

3.1.2 Scope:

The project aims at developing a user interface for two existing tools OpenNMS and Nagios. This tool does the process of mapping and monitoring and provides information which is not graphical. The user interface developed would help the administrator easily understand the network and take effective measures for the anomalies found.

3.1.3 Definition:

Customer:

A person or organization, internal or external to the producing organization, who takes financial responsibility for the system. In a large system this may not be the end user. The customer is the ultimate recipient of the developed product and its artifacts.

User :

A person who will use the system that is developed.

Analyst:

The Analyst details the specification of the system's functionality by describing the requirements aspect and other supporting software requirements

3.1.4 Abbreviation:

SRS	: Software Requirement Specification.
NHK	: Network Handling Kit.
OpenNMS	: Open Network Monitoring station.
LibPCap	: Library Packet Capture.
IP	: Internet Protocol.
LAN	: Local Area Network.
GUI	: Graphical User Interface.
GIF	: Graphical Interchange Format.

3.2 General Description

3.2.1 Product Overview:

This system aims to provide the administrator to manage the entire network through a web interface. This helps the administrator get the details of all active hosts in the network and the amount of traffic generated by them. The administrator is been provide with various options like Network load, Throughput, Traffic generated by various protocols by which the administrator can take necessary actions to control the anomalies of the network.

3.2.2 User Characteristics:

The main user of this system is the administrator who is in charge of the network. He is expected to have a basic knowledge of networking and its operations.

3.2.3 General Constraints:

The application is programmed to run in a Linux server. The interface can be accessed from any platform by using HTTP

3.2.4 General assumptions:

The application should be continuously run in a server and a browser installed in a client machine to access the GUI of NHK.

3.3 Specific Requirements

3.3.1 Inputs and Outputs:

The main input of the system is the IP address of the server along with the port number in a browser at any client in the network. All functions are pre-defined and do not require any inputs from the administrator. The administrator just has to select from any of the pre-defined functions available to him.

3.3.2Functional requirements:

i. The system should be able to accept the port no that is entered by the administrator.

ii. The system should allow the network administrator to perform the desired functions without any problems.

3.3.3 System Requirements:

3.3.3.1 Hardware Requirements:

Processor	:	Pentium III
RAM size	:	128 MB RAM
Hard disk capacity	:	20GB

3.3.3.2 Software Requirements:

Operating System	:	PCQ Linux 2004
Language	:	C, HTML, XML.

3.3.4 Performance constraints:

The system should be connected to the server to access the port number of the application by using the browser with the limited time period.

3.3.5 Software Constraints:

The clients run on any platform with any browser. The server requires that OpenNMS, Nagios and Tomcat 4.1 to be installed and running in the server system.

4. SYSTEM STUDY

4.1 Introduction to NHK:

NHK is the tool to manage the network efficiently with the graphical user interface. It is the integration of the two existing tools named OpenNMS and Nagios. It makes the work of an administrator easier to find the traffic in the network than the existing tool. The interface is designed to update dynamically at a certain time interval which would keep the administrator updated with the current network information.

4.2 Introduction to C:

C is a general purpose, structured programming language. Its instructions consist of terms hat resemble algebraic expressions, augmented by certain English keywords like if, else, for, do and while. In this respect c resembles other high level structured programming languages such as Pascal and FORTRAN. C also contains certain additional features, however, that allow it to be used at a lower level, thus bridging the gap between machine language and the more conventional high-level languages. This flexibility allows C to be used for system programming as well as for application programming.

C is characterized by the ability to write concise source programs, due in part to large number of operators include within the language. It has relatively small instruction set, though actual implementations include extensive library functions which enhance the basic instructions. Thus the features and capabilities of the languages can be easily be extended by the user.

STRUCTURE OF C PROGRAM

Every c program consists of one or more module called as functions. One of the functions must be called as main. The program will always begin by executing main function, which may access other functions.

Each function must contain:

- A function heading, which consist of the function name, followed by an optional list of arguments, enclosed in parenthesis.
- A list of argument declarations, if arguments are included in the heading.
- A compound statement, which comprises the remainder of the function.

C PROGRAM CHARECTERSTICS

Integrity

- Refers to the accuracy of calculations, all other program enhancements will be meaningless if the calculations are not carried out correctly

Clarity

- Refers to the overall readability of the program, with particular emphasis on its underlying logic

Simplicity

- The clarity and accuracy of the program are enhanced by keeping things as simple as possible, consistent with overall program objectives.

Efficiency

- It is considered with speed and effective memory utilization.

Modularity

- Breaking down of program into a series of identifiable tasks. Each sub tasks are implemented as separate program module.

Generality

- Programming in c is as general as possible; this can be obtained with certain effort in it.

4.3 HTML:

HTML stands for Hyper Text Markup language. It is a language used to create hypertext documents that have hyperlinks embedded in them. It is only a formatting language and not a programming language. Hyperlinks are underlined or emphasized words or locations in the screen that leads to other documents. WWW is a global, interactive, dynamic, cross platform, distributed, graphical hypertext information system.

The idea behind the hypertext is instead of reading the text in a linear rigid structure, the information can be navigated according to interest and preferences. HTML pages with audio and video files are called as hypermedia.

HTML is a language for describing structured documents. HTML describes the structures as lists, headings, paragraphs etc. elements of the web document are labeled through the use of HTML tags.

ADVANTAGES

A HTML document is small and hence easy to send over the net. It is small because it does not include format information

HTML documents are cross platform compatible and device independent. Only a HTML readable browser is needed to view them. Font names location are not needed.

HTML supports various features like

- Centered and right aligned text
- Tables
- Math equations
- Text and image alignment

VARIOUS TERMS RELATED TO HTML

WEB PAGE:

- A single file on the disk of the server which is retrieved formatted and displayed by a web browser.

HOME PAGE:

- A home page is the first in the set of documents. It is the entry point for a particular WWW site.

WEB PRESENTATION:

- A web presentation is a set of inter-linked web pages or documents.

WEB SERVER:

- The server feeds requested documents to the clients, this is achieved by sending request information to the server from the client terminal

BASIC HTML TAGS

Every HTML document starts with a <HTML> tag.

The prologue of the document is put into <HEAD> ……..</HEAD> part.

The title of the document is put between <TITLE>……. </TITLE> part.

The rest of the HTML document is enclosed in <BODY>…… <BODY>tags.

4.4 XML:

XML stands for extensible Markup Language; it defines a text format for structured documents and data. Any kind of structured data can be represented in XML (thus the term "extensible"): spreadsheets, texts, phone book entries, etc...

Like HTML, XML is a Markup language, i.e. it uses tags (words between '<' and '>') and attributes (in the form 'name=value'). Unlike HTML, XML uses tags to delimit chunks of data, without any interpretation of the data (HTML adds semantics to the tags, e.g. <B> </B> means that the text should be shown in **bold** font). XML leaves the interpretation of tags (their meaning) to the application that reads/writes the data. Unlike HTML, XML lets you create your own tags for use with your application.

XML Basics:

1. XML is a Markup language: it uses tags (words between '<' and '>') and attributes (in the form 'name=value')
2. XML files are text files: it allows experts to debug applications easily
3. XML documents should begin with an XML declaration which specifies the version of XML being used, e.g. **<? Xml version="1.0"?>**
4. Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags (e.g. **<MESSAGE>**) and end-tags (e.g. **</MESSAGE>**)
5. Empty elements are defined by an empty-element tag, e.g. **<READ/>**
6. The ampersand character (&) and the left angle bracket (<) may appear in their literal form only when used as markup delimiters. Otherwise, they must be escaped using either numeric character references (e.g. **&** for ampersand and **<** for left angle bracket) or the strings "**&**" and "**<**" respectively. The right angle bracket (>) may be represented using the string "**>**"
7. XML comments are delimited by <!-- and -->. For compatibility, the string "--" (double-hyphen) must not occur within comments
8. Attributes are used to associate name-value pairs with

defined in XML. Attribute values are defined between single-quotes (') or double-quotes (")

9. Attribute values can contain quote and double-quotes: they must be escaped by "**'**" for the apostrophe or single-quote character (') and by "**"**" for the double-quote character (")
10. Special attribute **xml:space**, when set to **preserve**, means that white space should be preserved by applications processing the XML document.

5. SOFTWARE IMPLEMENTATION MODEL

5.1 Module Description:

There are 4 modules in this system. They are:

- Mapping
- Monitoring
- Alerting
- Troubleshooting

5.1.1 Mapping:

The mapping module gives information about the hosts that are connected in the network. The information about the hosts in the network is shown by means of tabular form. The table contains the following fields:

- Domain Name

  Because it is hard to remember the string of numbers that make up an IP address, and because IP addresses sometimes need to change, all servers on the Internet also have human-readable names, called **domain names**. For example, it is easier for most of us to remember www.linwin2k.4t.com than it is to remember 216.27.61.137.

- IP address

  Each machine on the Internet is assigned a unique address called an **IP address**. IP addresses are 32-bit numbers, normally expressed as four "octets" in a "dotted decimal number." A typical IP address looks like this: 216.27.61.137.

- MAC address.
- Name of the Network interface card vendor.
- Hop distance and host contacts and
- Age inactivity.

5.1.2 Monitoring:

Monitoring of the network is done to analyze the traffic of the network and find the anomalies of the network. The process of the network monitoring is done by setting the network card in the promiscus mode, setting of the card in the promiscus mode makes the card sniff all the packets passing through the network.

This process is done by using a module named LibPCap which would set the Ethernet card of the server in which the application is installed to the promiscus mode, the data collected by LibPCap is directed to a file from which

network is obtained and given in a tabular form. The monitoring module provides information regarding traffic generated by various protocols used in the network. Clear classification between the traffic generated locally and remotely is done.

Information regarding the data sent and received from a particular host either remote or local is also provided. Throughput of data from a particular host is given in terms of packets per second and bits per second.

- PROTOCOLS - The **protocol** is the pre-defined way that someone who wants to use a service talks with that service. The "someone" could be a person, but more often it is a computer program like a Web browser. Protocols are often text, and simply describe how the client and server will have their conversation. The **http** in the Web's protocol. Some common protocols that are used to transfer data in a network are,

  - **IP** (Internet Protocol) - the main delivery system for information over the Internet
  - **TCP** (Transmission Control Protocol) - used to break apart and rebuild information that travels over the Internet
  - **HTTP** (Hyper Text Transfer Protocol) - used for Web pages
  - **FTP** (File Transfer Protocol) - used to download and upload files
  - **UDP** (User Datagram Protocol) - used for information that

- **ICMP** (Internet Control Message Protocol) - used by a router to exchange the information with other routers
- **SMTP** (Simple Mail Transport Protocol) - used to send text-based information (e-mail)
- **SNMP** (Simple Network Management Protocol) - used to collect system information from a remote computer
- **Telnet** - used to perform commands on a remote computer

5.1.3 Alerting :

Alerting is the process of giving risk indications of traffic generated to the administrator. This process is done here using risk flags, the risk flags are attached with the host name in each table. The risk is indicated using three colors in flags namely red, green and yellow. Red indicates high risk, yellow indicates medium risk whereas green indicates low risk. The risk is calculated using the packets transferred from a particular host.

Information regarding the network load is given in a graphical form with respect to time and data transferred would give a quick alert to the administrator regarding the traffic of the network.

The following fields would act as an alert to the administrator.

- Unhealthy hosts.
- Risk flags.

5.1.4 Troubleshooting:

The troubleshooting function does the process of fixing the problem on the basics of time interval. Various indications at a particular time interval will help the administrator to trouble shoot the network.

6. PRODUCT TESTING

Testing is done to detect the errors in the software. This implies not only to the coding phase but to uncover errors introduced in all the previous phases.

Unit Testing: Each and every module is tested separately to check if its intended functionality is met. Some unit testing performed are,

- Checking the proper working of the system information and process list retrieval modules in the server application.
- Checking for proper connectivity between server and clients in the network.
- Ensuring that the application runs in the server does not interfere with other functions.
- Verifying the application interface and ensuring correctness of data transmission in the network.

Mapping:

- Verifying the working of Tomcat 4.0 at the server.
- Checking the proper working of the mapping tool OpenNMS.
- Checking of dynamic updation of the host if configured.

Monitoring:

- Checking the proper working of the monitoring tool Nagios.
- Checking of dependencies of the tool with the configuration.

Integration Testing: It is the testing performed to detect errors on interconnection between modules. Here, all the modules integrated to the server system are combined to form the server applications and tested to ensure that they work in synchronization and without interference from each other.

- The dependencies needed to integrate OpenNMS and Nagios are installed.
- LibPCap is installed to integrate the two tools.

Validation Testing: Validation testing can be defined in many ways but a simple definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the users.

- Checking the IP address of the hosts listed in the table.
- Verifying that the operating system and the server name listed in the interface are correct.

Output Testing: After performing the validation testing the next step is output testing of the proposed system since no system is useful if it does not produce the required output in the specific format. The outputs generated are displayed by the system under consideration are tested by asking the users about the formats required by them. The following test cases are generated:

INPUT	DESIRED OUTPUT	ACTUAL OUTPUT	COMMENTS
Port number with the IP address.	Display of the interface at the client side.	Display of the interface at the client side.	Verified and found to be correct.
Clicking the link host information.	Display of all currently active hosts with their details.	Display of all currently active hosts with their details.	Verified and found to be correct.
Clicking the link throughput information.	Display the throughput information of the host.	Display the throughput information of the host.	Verified and found to be correct.
Clicking the link network activity.	Display the activity of the network at a	Display the activity of the network at a	Verified and found to be correct.

System Testing: The system is tested against the system requirements to see if all the requirements are met and if the system performs as per the specified requirements. The system is tested as a whole to check for its functionality.

The interaction of the tool with other applications is checked. Connections to the server from client are also seen so that the client interacts with the server properly.

7. FUTURE ENHANCEMENTS

The project Network Handling Kit is designed in such a way that future enhancements can be made in an easy manner. The project is quite flexible and can be easily customized according to the user's requirements.

One possible enhancement to the project is the addition of details regarding mapping.

- Graphical representation of the mapped network can be added for easy reference for the administrator and even for the naïve user.

Enhancement in the area of alerting can be done using

- SNMP alerting from any client terminal can be given to the server regarding anomalies so that the administrator can take necessary action for the problem.

Features in troubleshooting can be added as

- Auto indication to the client regarding the system traffic generation can be done where the client can take necessary action for the problem.

8. CONCLUSION

The project Network Handling Kit offers the following advantages:

- Easy understanding of mapped and monitored data is provided by the GUI.
- Quite flexible in nature allowing addition of newer functions more easily
- The interface can be accessed from any client machine thus reducing the load on the server and reducing time taken.
- The application is run on the server side thus provide high security and preventing misuse of the system
- User friendly interface which requires no training

9. APPENDIX

9.1 SAMPLE CODE

MAPPING

UPDATING TRAFFIC

```
static void updateRoutedTraffic(HostTraffic *router) {
 if(router != NULL) {
  if(router->routedTraffic == NULL) {
   int mallocLen = sizeof(RoutingCounter);

   router->routedTraffic = (RoutingCounter*)malloc(mallocLen);
   if(router->routedTraffic == NULL) return;
   memset(router->routedTraffic, 0, mallocLen);
  }

  if(router->routedTraffic != NULL) { /* malloc() didn't fail */
   incrementTrafficCounter(&router->routedTraffic->routedPkts, 1);
   incrementTrafficCounter(&router->routedTraffic->routedBytes,
                           (Counter)(h_save->len - sizeof(struct
ether_header)));
  }
 }
}
```

ADDING CONTACTED PEERS

```
static void addContactedPeers(HostTraffic *sender, HostAddr *srcAddr,
                              HostTraffic *receiver, HostAddr *dstAddr,
                              int actualDeviceId) {
 if((sender == NULL) || (receiver == NULL) || (sender == receiver)) {
```

```
  if ((sender != NULL) && (sender->l2Family ==
FLAG_HOST_TRAFFIC_AF_FC) &&
      (strncasecmp (sender->fcCounters->hostNumFcAddress,
FC_FAB_CTLR_ADDR,
                    strlen (FC_FAB_CTLR_ADDR)) == 0)) {
    /* This is normal. Return without warning */
    return;
  }
  traceEvent(CONST_TRACE_ERROR, "Sanity check failed @
addContactedPeers (%p, %p)",
             sender, receiver);
  return;
 }

 if((sender != myGlobals.otherHostEntry) && (receiver !=
myGlobals.otherHostEntry)) {
  /* The statements below have no effect if the serial has been already
computed */
  setHostSerial(sender); setHostSerial(receiver);

  sender->totContactedSentPeers +=
    incrementUsageCounter(&sender->contactedSentPeers, receiver,
actualDeviceId);
  receiver->totContactedRcvdPeers +=
    incrementUsageCounter(&receiver->contactedRcvdPeers, sender,
actualDeviceId);
 }
}
```

```
#ifdef FRAGMENT_DEBUG
static void dumpFragmentData(IpFragment *fragment) {
 printf("FRAGMENT_DEBUG: IPFragment: (%p)\n", fragment);
 printf("                %s:%d->%s:%d\n",
      fragment->src->hostResolvedName, fragment->sport,
      fragment->dest->hostResolvedName, fragment->dport);
 printf("                FragmentId=%d\n", fragment->fragmentId);
 printf("                lastOffset=%d, totalPacketLength=%d\n",
      fragment->lastOffset, fragment->totalPacketLength);
 printf("                 totalDataLength=%d,
expectedDataLength=%d\n",
      fragment->totalDataLength, fragment->expectedDataLength);
 fflush(stdout);
}
#endif

HOST INFORMATION
Static void updateHostList (char *fileName, u_char upDownloadMode,
HostTraffic *theRemHost) {
  if(fileName != NULL) {
    FileList *list, *lastPtr = NULL;
    int numEntries = 0;

    if(theRemHost->protocolInfo == NULL) theRemHost->protocolInfo =
 calloc(1, sizeof(ProtocolInfo));
    list = theRemHost->protocolInfo->fileList;

 #ifdef DEBUG
```

```
            fileName, theRemHost->hostNumIpAddress);
#endif
   while(list != NULL) {
    if(strcmp(list->fileName, fileName) == 0) {
       FD_SET(upDownloadMode, &list->fileFlags);
       return;
    } else {
       lastPtr = list;
       list = list->next;
       numEntries++;
    }
   }
   if(list == NULL) {
    list = (FileList*)malloc(sizeof(FileList));
    list->fileName = strdup(fileName);
    FD_ZERO(&list->fileFlags);
    FD_SET(upDownloadMode, &list->fileFlags);
    list->next = NULL;

    if(numEntries >= MAX_NUM_LIST_ENTRIES) {
       FileList *ptr = theRemHost->protocolInfo->fileList->next;

       lastPtr->next = list; /* Append */
       /* Free the first entry */
       free(theRemHost->protocolInfo->fileList->fileName);
       free(theRemHost->protocolInfo->fileList);
       /* The first ptr points to the second element */
       theRemHost->protocolInfo->fileList = ptr;
```

```
        list->next = theRemHost->protocolInfo->fileList;
        theRemHost->protocolInfo->fileList = list;
      }
    }
  }
}
```

MONITORING

VARIOUS PROTOCOL USAGE CALCULATIONS

```
void printIpProtocolUsage(void) {
  HostTraffic **hosts, *el;
  u_short clientPorts[MAX_ASSIGNED_IP_PORTS],
serverPorts[MAX_ASSIGNED_IP_PORTS];
  u_int j, idx1, hostsNum=0, numPorts=0, maxHosts;
  char buf[LEN_GENERAL_WORK_BUFFER], portBuf[32],
hostLinkBuf[LEN_GENERAL_WORK_BUFFER];

  printHTMLheader("TCP/UDP: Local Protocol Usage", NULL, 0);

  memset(clientPorts, 0, sizeof(clientPorts));
  memset(serverPorts, 0, sizeof(serverPorts));
  hosts =
(HostTraffic**)mallocAndInitWithReportWarn(myGlobals.device[myGl
obals.actualReportDeviceId].
                                             hostsno*sizeof(HostTraffic*),
                                             "printIpProtocolUsage");
  if(hosts == NULL)
    return;
```

```
 maxHosts =
myGlobals.device[myGlobals.actualReportDeviceId].hostsno;
 for(el=getFirstHost(myGlobals.actualReportDeviceId);
    el != NULL; el = getNextHost(myGlobals.actualReportDeviceId, el))
{
   if(subnetPseudoLocalHost(el) && (el->hostNumIpAddress[0] != '\0'))
{
     hosts[hostsNum++] = el;
     if(el->portsUsage != NULL) {
        PortUsage *ports = el->portsUsage;
        while(ports) {
          clientPorts[j] += ports->clientUses, serverPorts[j] += ports-
>serverUses;
          numPorts++, ports = ports->next;
        }
     }
   }
   if(hostsNum >= maxHosts) break;
  } /* for */
  if(numPorts == 0) {
   printNoDataYet();
   free(hosts);
   return;
  }
  /* Hosts are now in a contiguous structure (hosts[])... */

  sendString("<CENTER>\n");
```

```
 sendString(""TABLE_ON"<TABLE BORDER=1
"TABLE_DEFAULTS"><TR "TR_ON" "DARK_BG"><TH "TH_BG"
COLSPAN=2>Service</TH>"
          "<TH "TH_BG">Clients</TH><TH
"TH_BG">Servers</TH>\n");

  for(j=0; j<MAX_ASSIGNED_IP_PORTS; j++)
    if((clientPorts[j] > 0) || (serverPorts[j] > 0)) {
      safe_snprintf(__FILE__, __LINE__, buf, sizeof(buf), "<TR
 "TR_ON" %s><TH "TH_BG" ALIGN=LEFT
 "DARK_BG">%s</TH><TD "TD_BG" ALIGN=CENTER>%d</TD>"
                  "<TD "TD_BG">\n", getRowColor(),
                  getAllPortByNum(j, portBuf, sizeof(portBuf)), j);
      sendString(buf);

      if(clientPorts[j] > 0) {
        PortUsage *ports = getPortsUsage(hosts[idx1], j, 0);

        sendString("<UL>");
        for(idx1=0; idx1<hostsNum; idx1++)
          if((hosts[idx1]->portsUsage != NULL)
            && ports && (ports->clientUses > 0)) {
            safe_snprintf(__FILE__, __LINE__, buf, sizeof(buf),
  "<li>%s\n",
                          makeHostLink(hosts[idx1],
  FLAG_HOSTLINK_TEXT_FORMAT, 1, 0, hostLinkBuf,
  sizeof(hostLinkBuf)));
            sendString(buf);
```

```
    sendString("</UL>");
  } else
    sendString(" ");
  sendString("</TD><TD "TD_BG">");

  if(serverPorts[j] > 0) {
    PortUsage *ports = getPortsUsage(hosts[idx1], j, 0);

    sendString("<UL>");
    for(idx1=0; idx1<hostsNum; idx1++)
      if((hosts[idx1]->portsUsage != NULL)
        && ports && (ports->serverUses > 0)) {
        safe_snprintf(__FILE__, __LINE__, buf, sizeof(buf),
"<li>%s\n",
                      makeHostLink(hosts[idx1],
FLAG_HOSTLINK_TEXT_FORMAT, 1, 0, hostLinkBuf,
sizeof(hostLinkBuf)));
        sendString(buf);
      }
    sendString("</UL>");
  } else
    sendString(" ");

  sendString("</TD></TR>");
  } /* for */

  sendString("</TABLE>"TABLE_OFF"<P>\n");
  sendString("</CENTER>\n");
```

```
printHostColorCode(FALSE, 0);

printFooterHostLink();
free(hosts);

THROUGHPUT
Void printThptStats(int sortedColumn _UNUSED_) {
 char tmpBuf[128], formatBuf[32], formatBuf1[32];

 printHTMLheader("Network Load Statistics", NULL, 0);

 /*
 if(myGlobals.device[myGlobals.actualReportDeviceId].dummyDevice)
{
   printFlagedWarning("<I>Network load statistics are not available for
virtual interfaces</I>");
   return;
  }
  */

if(myGlobals.device[myGlobals.actualReportDeviceId].numThptSamples
== 0) {
   printNoDataYet();
   return;
  }

  sendString("<CENTER>\n");
```

```
 sendString("<A HREF=\"" CONST_THPT_STATS_MATRIX_HTML
"?col=1\" BORDER=0 BGCOLOR=white>"
            "<IMG SRC=\"" CONST_THROUGHPUT_GRAPH
CHART_FORMAT "?col=1\" alt=\"Current Hour throughput
chart\"></A><BR>\n");
#endif

  safe_snprintf(__FILE__, __LINE__, tmpBuf, sizeof(tmpBuf),
"<H4>Time [ %s through %s]</H4>",
            formatTimeStamp(0, 0, 60, formatBuf, sizeof(formatBuf)),
            formatTimeStamp(0, 0,  0, formatBuf1, sizeof(formatBuf1)));

  sendString(tmpBuf);


 if(myGlobals.device[myGlobals.actualReportDeviceId].numThptSamples
 > 60) {
 #ifndef EMBEDDED
    sendString("<P><A HREF=\""
 CONST_THPT_STATS_MATRIX_HTML "?col=2\" BORDER=0
 BGCOLOR=white>"
             "<IMG SRC=\"" CONST_THROUGHPUT_GRAPH
 CHART_FORMAT "?col=2\" alt=\"Current Day throughput
 chart\"></A><BR>\n");
 #endif
    safe_snprintf(__FILE__, __LINE__, tmpBuf, sizeof(tmpBuf),
 "<H4>Time [ %s through %s]</H4>",
                formatTimeStamp(0, 24, 0, formatBuf, sizeof(formatBuf)),
```

```
                formatTimeStamp(0,  0, 0, formatBuf1,
sizeof(formatBuf1)));
   sendString(tmpBuf);

#ifndef EMBEDDED
if(myGlobals.device[myGlobals.actualReportDeviceId].numThptSamples
> 1440 /* 60 * 24 */) {
     sendString("<P><IMG SRC=\"" CONST_THROUGHPUT_GRAPH
CHART_FORMAT "?col=3\" alt=\"Current 30day throughput
chart\"><BR>\n");
     safe_snprintf(__FILE__, __LINE__, tmpBuf, sizeof(tmpBuf),
"<H4>Time [ %s through %s]</H4>",
                formatTimeStamp(30, 0, 0, formatBuf, sizeof(formatBuf)),
                formatTimeStamp( 0, 0, 0, formatBuf1,
sizeof(formatBuf1)));
     sendString(tmpBuf);
    }
#endif
  }

  sendString("</CENTER>\n");
 }
```

HTML

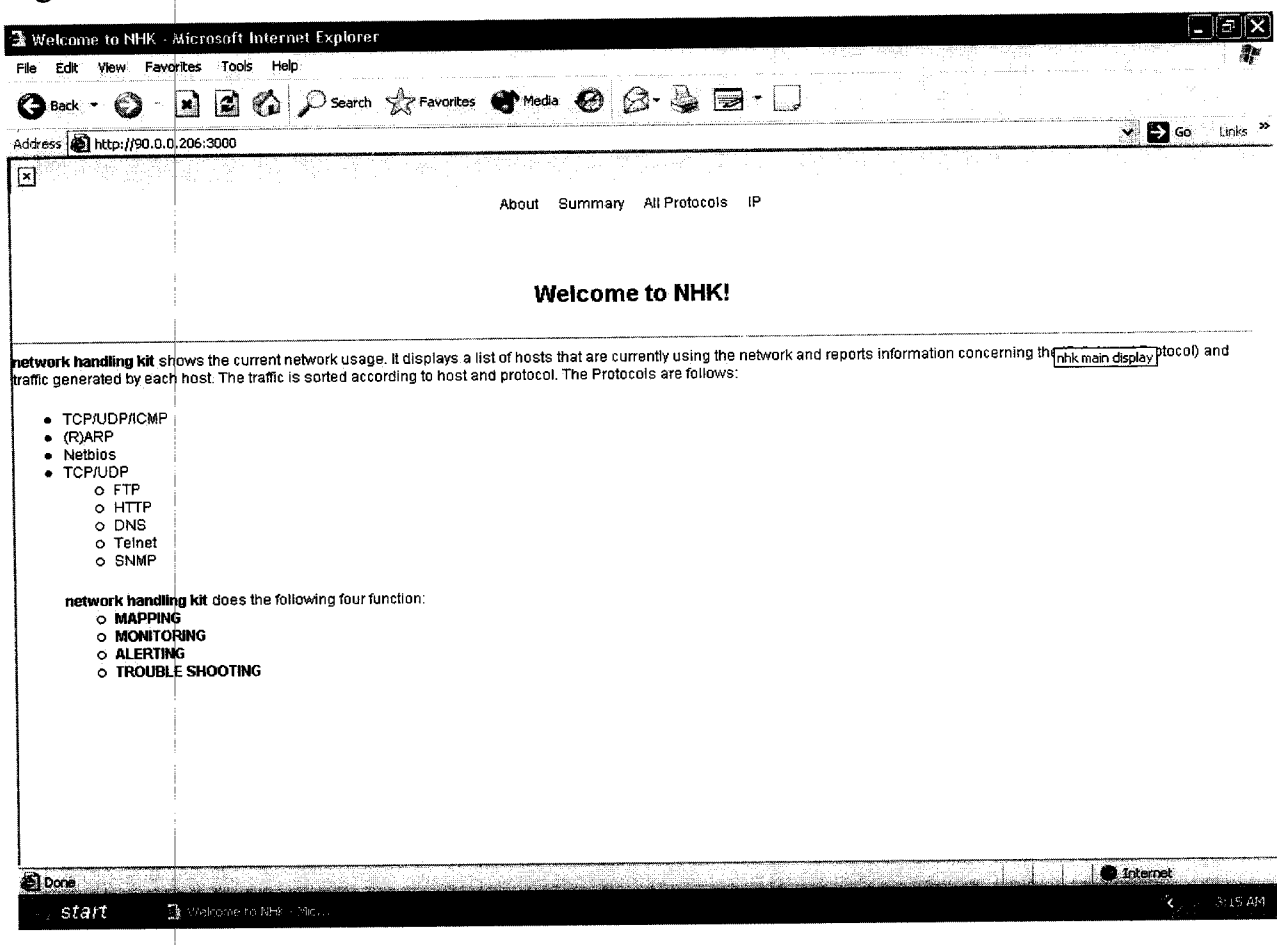
MENU DEFINITION

```
<html>
<head>
< Validates as HTML 4.01 Transitional with the following tag >
<meta http-equiv="Content-Type" content="/html; charset=utf-8">
<meta http-equiv="Expires" content="0">
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Content-Style-Type" content="text/css">
<meta http-equiv="Window-target" content="top">
<meta name="ROBOTS" content="NOINDEX,NOFOLLOW">
<meta name="description" content=" network handling kit.">
<meta name="author" content="nhk">
<meta name="generator" content="nhk">
<link rel="stylesheet" type="text/css" href="style.css">
<title>ntop summary menu (normally a frame)</title>
</head>
<body background="menuline.png">  
<span class="leftmenuitem">  About us</span>
<span class="leftmenuitem"> <a href=" Summary:"
target="area">Traffic</a></span> <b> | </b>
<span class="leftmenuitem"> <a href="All protocols"
target="area">Hosts</a></span> <b> | </b>
<span class="leftmenuitem"> <a href="IP traffic"
target="area">Network Load</a></span> <b> | </b>
</body>
</html>
```

9.1SAMPLE OUTPUTS

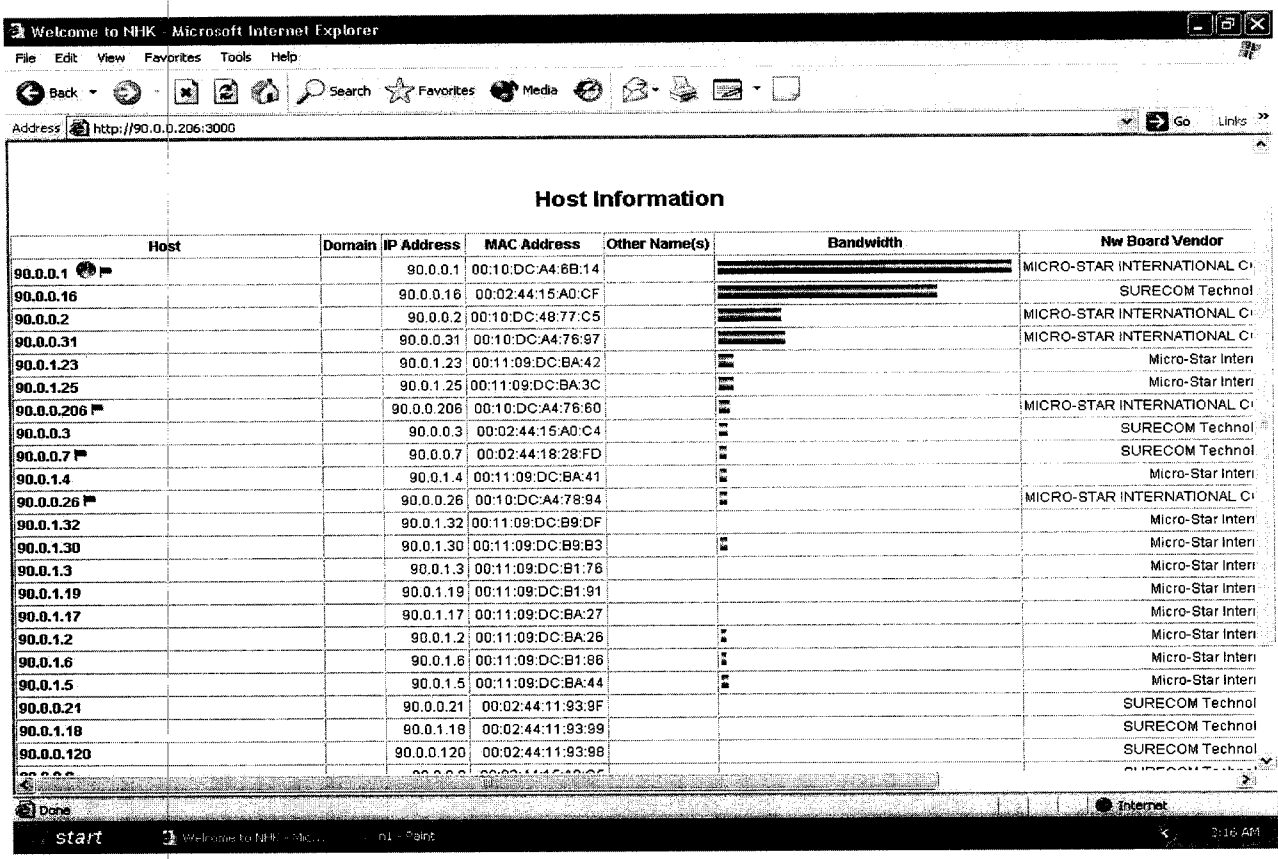
NETWORK HANDLING KIT

Figure 1



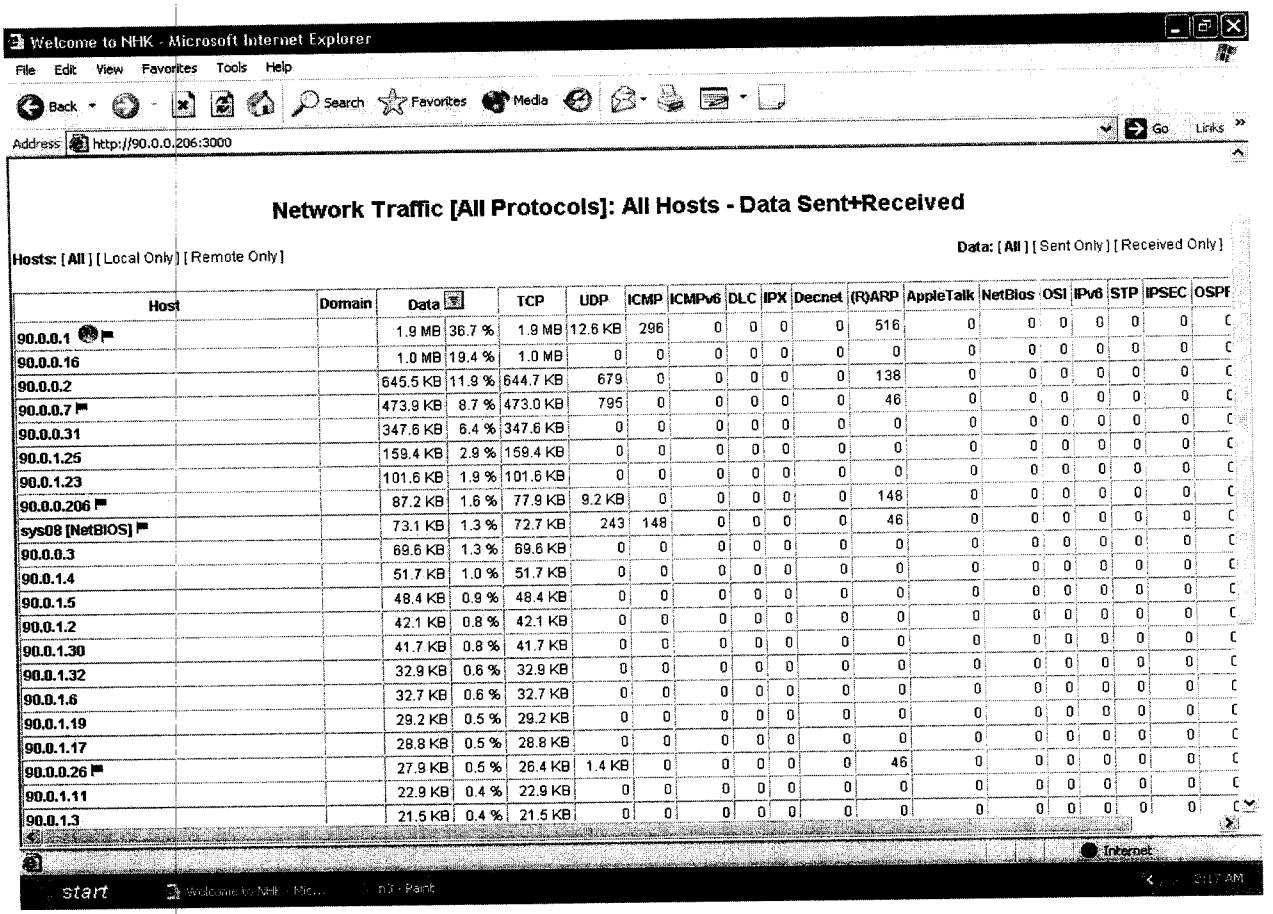
MAPPING

Figure 2:



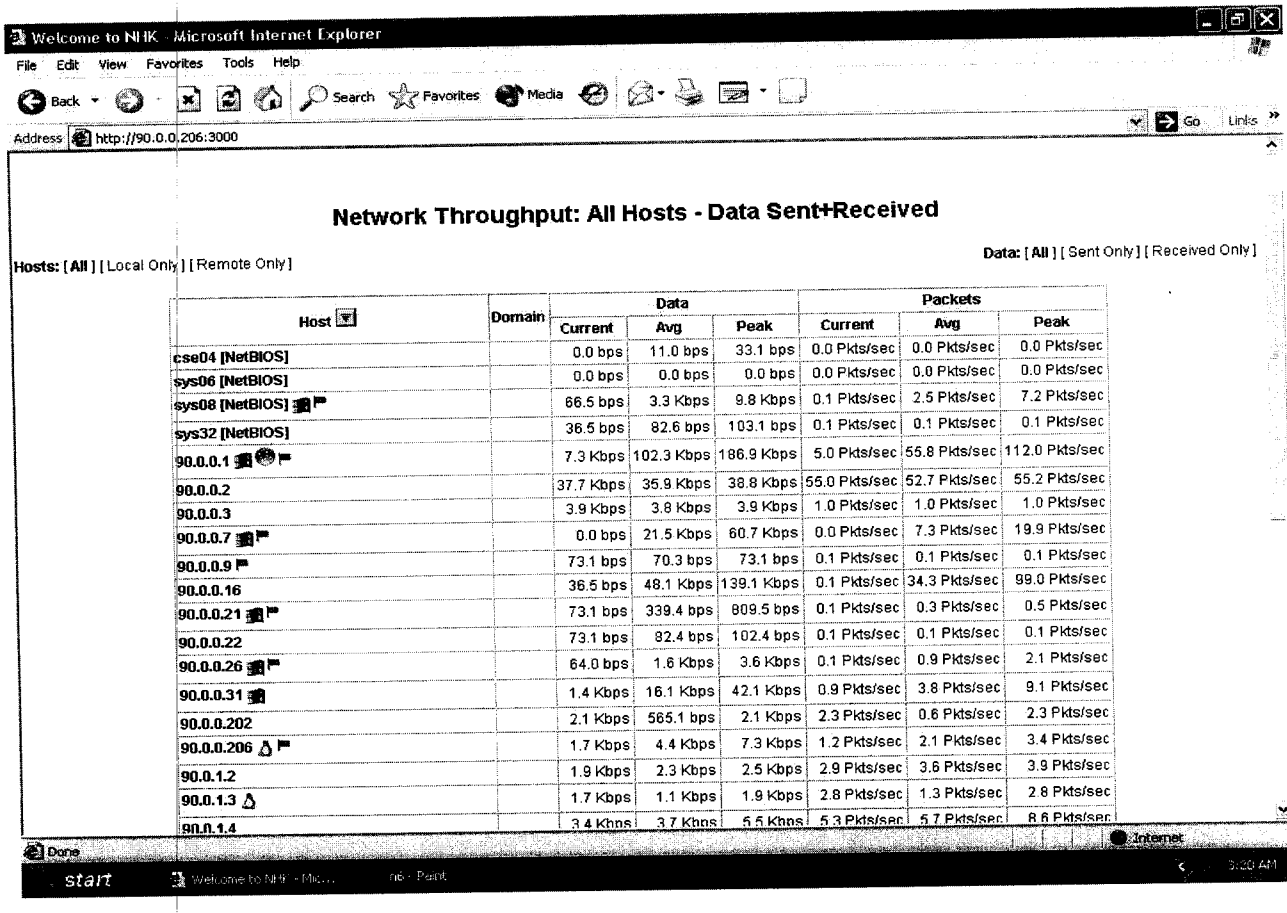
MONITORING

Figure 3:



Host	Domain	Data		TCP	UDP	ICMP	ICMPv6	DLC	IPX	Decnet	(R)ARP	AppleTalk	NetBios	OSI	IPv6	STP	IPSEC	OSPF
90.0.0.1		1.9 MB	36.7 %	1.9 MB	12.6 KB	296	0	0	0	0	516	0	0	0	0	0	0	C
90.0.0.16		1.0 MB	19.4 %	1.0 MB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.0.2		645.5 KB	11.9 %	644.7 KB	679	0	0	0	0	0	138	0	0	0	0	0	0	C
90.0.0.7		473.9 KB	8.7 %	473.0 KB	795	0	0	0	0	0	46	0	0	0	0	0	0	C
90.0.0.31		347.6 KB	6.4 %	347.6 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.25		159.4 KB	2.9 %	159.4 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.23		101.6 KB	1.9 %	101.6 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.0.206		87.2 KB	1.6 %	77.9 KB	9.2 KB	0	0	0	0	0	148	0	0	0	0	0	0	C
sys08 [NetBIOS]		73.1 KB	1.3 %	72.7 KB	243	148	0	0	0	0	46	0	0	0	0	0	0	C
90.0.0.3		69.6 KB	1.3 %	69.6 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.4		51.7 KB	1.0 %	51.7 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.5		48.4 KB	0.9 %	48.4 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.2		42.1 KB	0.8 %	42.1 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.30		41.7 KB	0.8 %	41.7 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.32		32.9 KB	0.6 %	32.9 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.6		32.7 KB	0.6 %	32.7 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.19		29.2 KB	0.5 %	29.2 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.17		28.8 KB	0.5 %	28.8 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.0.26		27.9 KB	0.5 %	26.4 KB	1.4 KB	0	0	0	0	0	46	0	0	0	0	0	0	C
90.0.1.11		22.9 KB	0.4 %	22.9 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C
90.0.1.3		21.5 KB	0.4 %	21.5 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	C

Figure 4:



Host	Domain	Data			Packets		
		Current	Avg	Peak	Current	Avg	Peak
cse04 [NetBIOS]		0.0 bps	11.0 bps	33.1 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.0 Pkts/sec
sys06 [NetBIOS]		0.0 bps	0.0 bps	0.0 bps	0.0 Pkts/sec	0.0 Pkts/sec	0.0 Pkts/sec
sys08 [NetBIOS]		66.5 bps	3.3 Kbps	9.8 Kbps	0.1 Pkts/sec	2.5 Pkts/sec	7.2 Pkts/sec
sys32 [NetBIOS]		36.5 bps	82.6 bps	103.1 bps	0.1 Pkts/sec	0.1 Pkts/sec	0.1 Pkts/sec
90.0.0.1		7.3 Kbps	102.3 Kbps	186.9 Kbps	5.0 Pkts/sec	55.8 Pkts/sec	112.0 Pkts/sec
90.0.0.2		37.7 Kbps	35.9 Kbps	38.8 Kbps	55.0 Pkts/sec	52.7 Pkts/sec	55.2 Pkts/sec
90.0.0.3		3.9 Kbps	3.8 Kbps	3.9 Kbps	1.0 Pkts/sec	1.0 Pkts/sec	1.0 Pkts/sec
90.0.0.7		0.0 bps	21.5 Kbps	60.7 Kbps	0.0 Pkts/sec	7.3 Pkts/sec	19.9 Pkts/sec
90.0.0.9		73.1 bps	70.3 bps	73.1 bps	0.1 Pkts/sec	0.1 Pkts/sec	0.1 Pkts/sec
90.0.0.16		36.5 bps	48.1 Kbps	139.1 Kbps	0.1 Pkts/sec	34.3 Pkts/sec	99.0 Pkts/sec
90.0.0.21		73.1 bps	339.4 bps	809.5 bps	0.1 Pkts/sec	0.3 Pkts/sec	0.5 Pkts/sec
90.0.0.22		73.1 bps	82.4 bps	102.4 bps	0.1 Pkts/sec	0.1 Pkts/sec	0.1 Pkts/sec
90.0.0.26		64.0 bps	1.6 Kbps	3.6 Kbps	0.1 Pkts/sec	0.9 Pkts/sec	2.1 Pkts/sec
90.0.0.31		1.4 Kbps	16.1 Kbps	42.1 Kbps	0.9 Pkts/sec	3.8 Pkts/sec	9.1 Pkts/sec
90.0.0.202		2.1 Kbps	565.1 bps	2.1 Kbps	2.3 Pkts/sec	0.6 Pkts/sec	2.3 Pkts/sec
90.0.0.206		1.7 Kbps	4.4 Kbps	7.3 Kbps	1.2 Pkts/sec	2.1 Pkts/sec	3.4 Pkts/sec
90.0.1.2		1.9 Kbps	2.3 Kbps	2.5 Kbps	2.9 Pkts/sec	3.6 Pkts/sec	3.9 Pkts/sec
90.0.1.3		1.7 Kbps	1.1 Kbps	1.9 Kbps	2.8 Pkts/sec	1.3 Pkts/sec	2.8 Pkts/sec
90.0.1.4		3.4 Kbps	3.7 Kbps	5.5 Kbps	5.3 Pkts/sec	5.7 Pkts/sec	8.6 Pkts/sec

ALERTING

Figure 5:

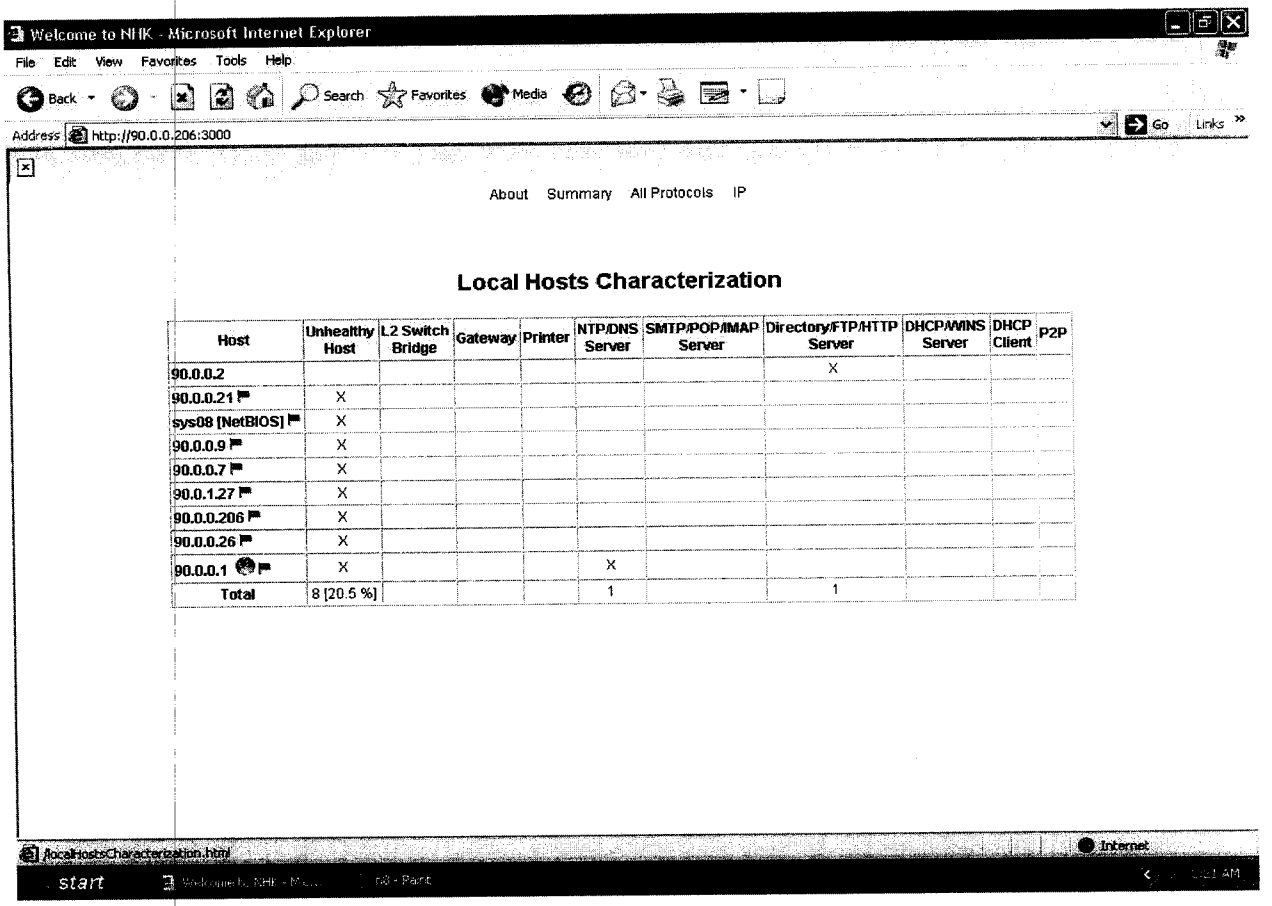
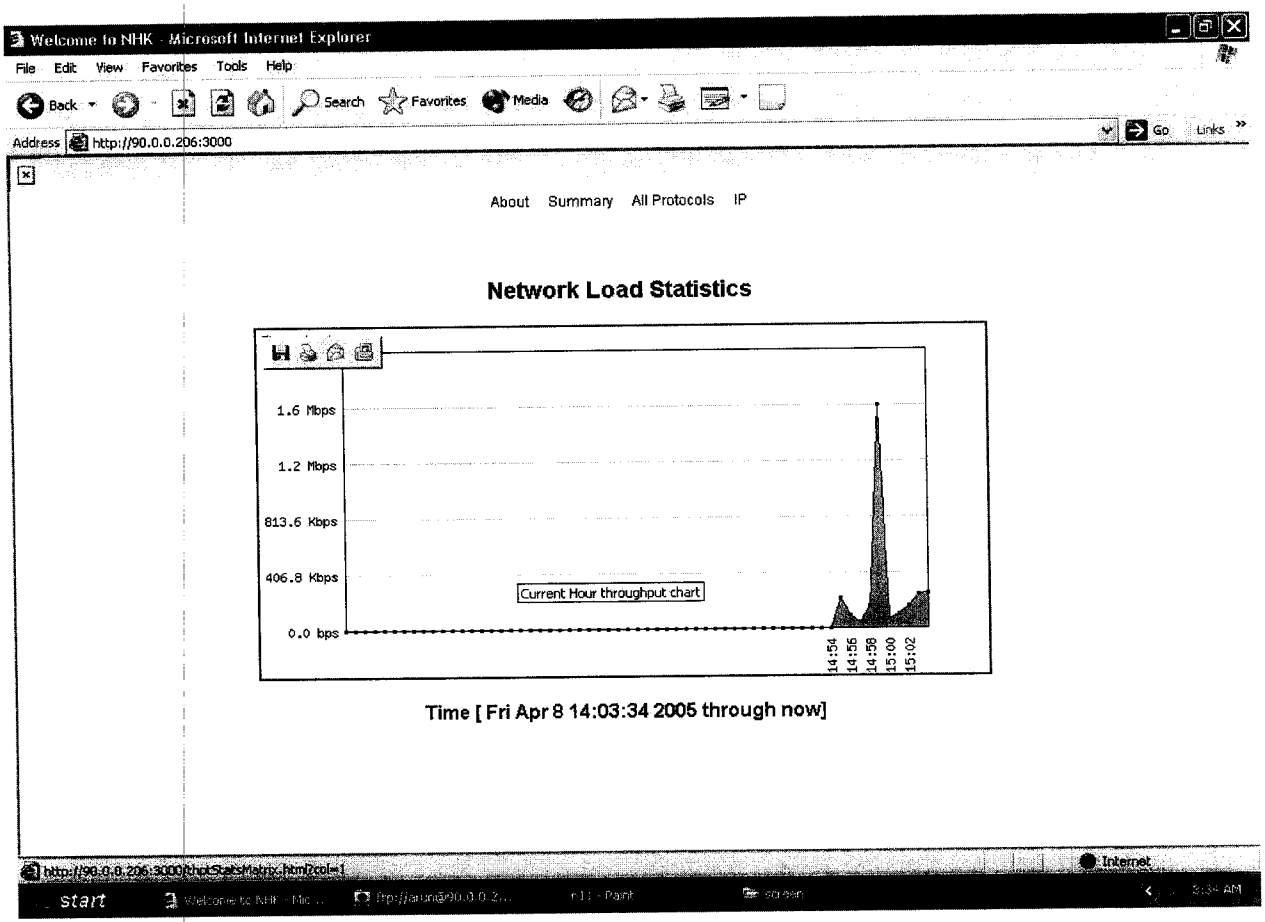
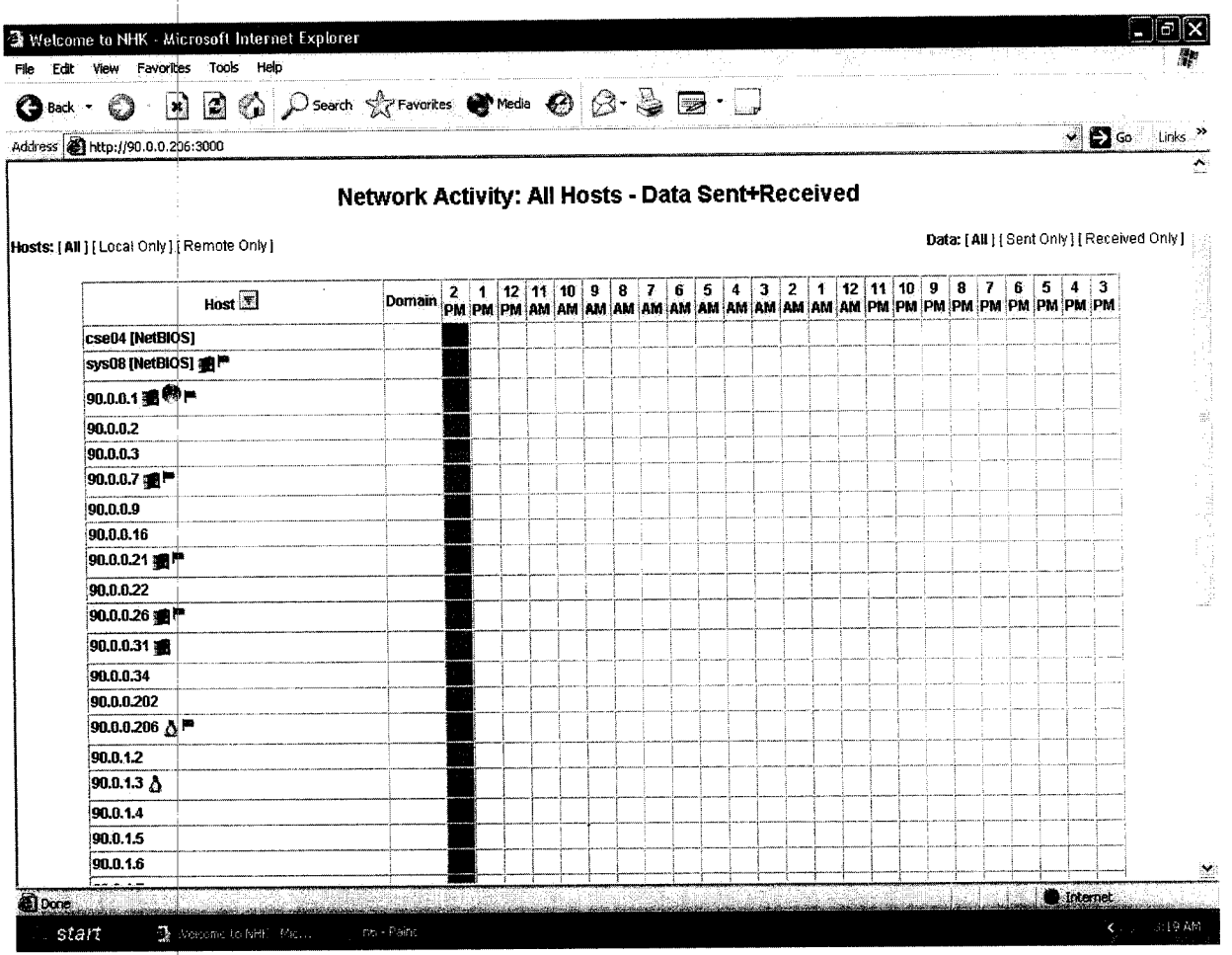


Figure 5:


Welcome to NHK - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://90.0.0.206:3000
Go Links
About Summary All Protocols IP
Network Load Statistics
1.6 Mbps
1.2 Mbps
813.6 Kbps
406.8 Kbps
0.0 bps
Current Hour throughput chart
14:54
14:56
14:58
15:00
15:02
Time [Fri Apr 8 14:03:34 2005 through now]
Internet
start

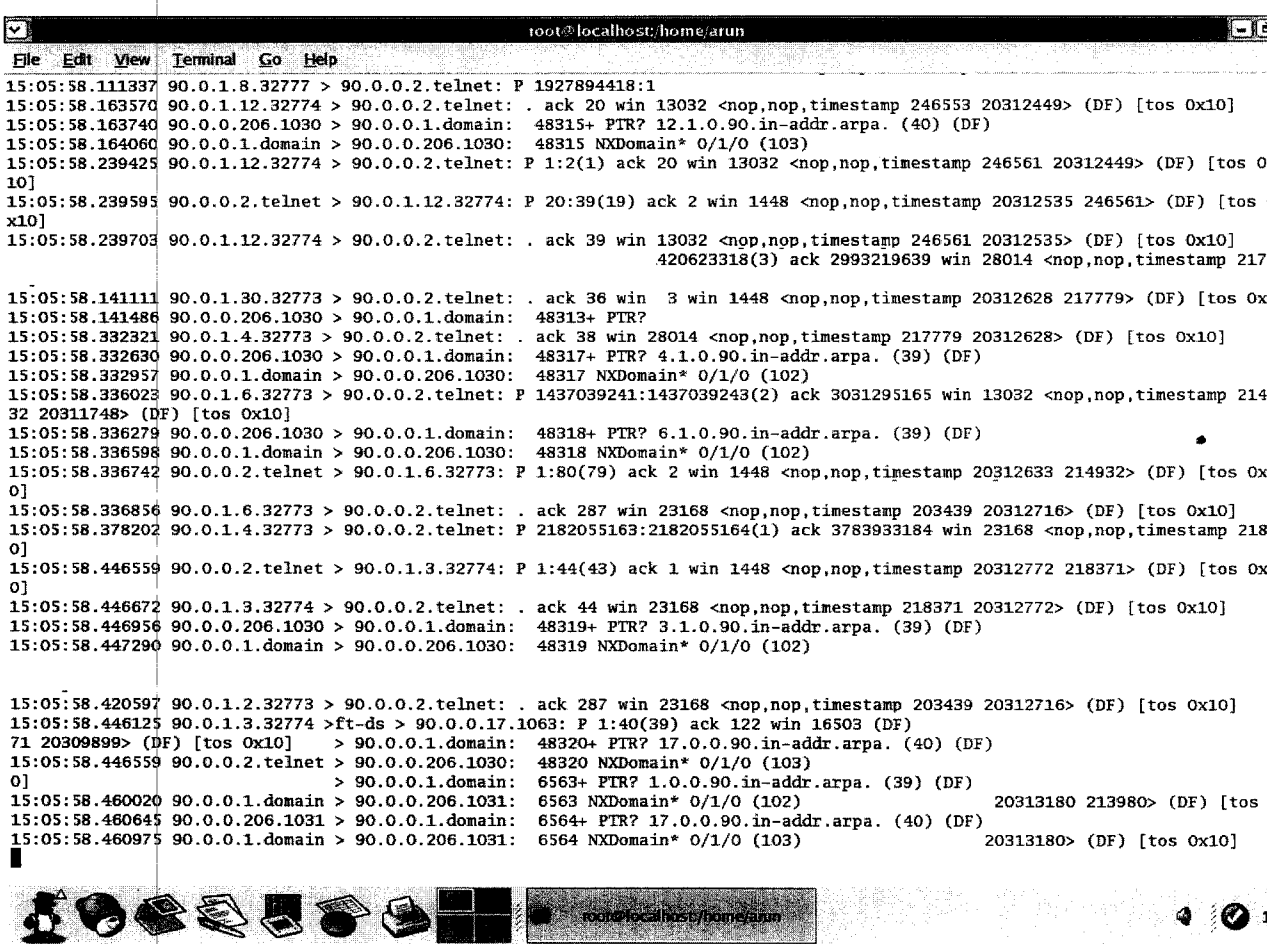
TROUBLESHOOTING

Figure 6:



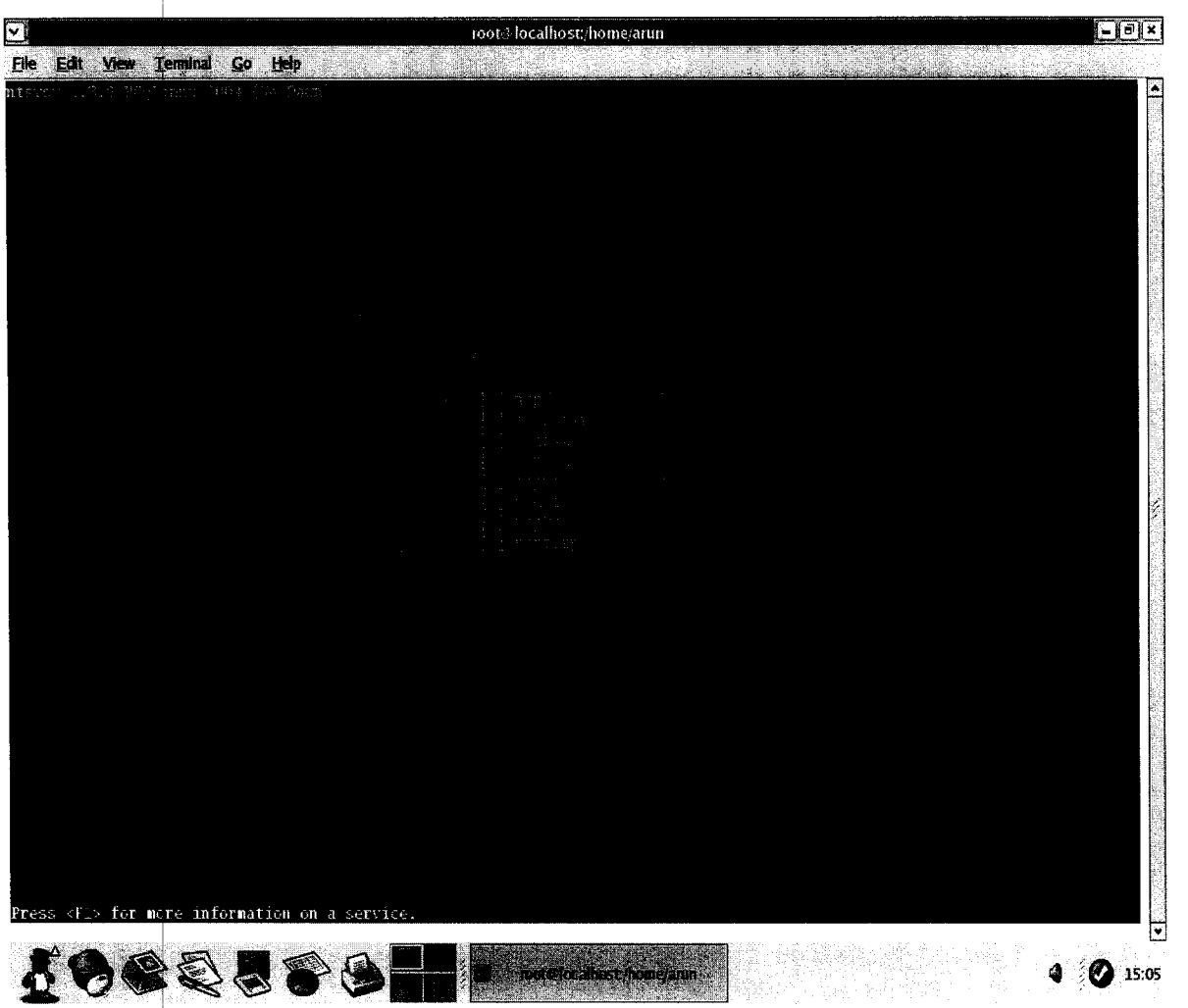
TCP DUMP

Figure 7:



SYSTEM SERVICES

Figure 8:



10. REFERENCES

1. Douglas E. Comer (2003) "Internetworking with TCP/IP principles, protocols, and architectures" Prentice – Hall of India, Inc. Volume 1.
2. Byron Gottfried (1998) "Programming with C" Tata McGraw – Hill Publishing group, I Edition.
3. www.linux.org
4. www.linuxdoc.org
5. www.netfilter.org
6. www.opennms.org
7. www.nagios.org