A Web-based Network Management Tool using Simple Network Management Protocol (SNMP)

GRADUATE PROJECT

Submitted to the faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Visrunkala S. Kasba
Fall 2003

Committee Members

Dr. Dulal C. Kar
Committee Chairperson

Dr. R. Stephen Dannelly
Committee Member

Dr. John Fernandez
Committee Member
ABSTRACT

As networks grow bigger, inter-network management becomes a challenging task for the network manager. A Network Management Tool that shows the system information, routing information, Transmission Control Protocol / User Datagram protocol (TCP/UDP) connection information of network devices like routers, servers, printers and workstations is of great help for the network manager. These devices hold a virtual information database called Management Information Base (MIB), which contains information about objects like hardware parameters, configuration parameters, and performance statistics. Simple Network Management Protocol (SNMP) is an application layer protocol that allows accessing, retrieving, or modifying the information contained in the MIBs of the devices. The retrieved information from these devices can be used to monitor the network. A Web-based application using SNMP that will query MIBs of network devices and display the retrieved information is developed. The tool also allows the manager to set (manipulate) the values of read-writable parameters.
# TABLE OF CONTENTS

Abstract............................................................................................................................................ii

Table of Contents...............................................................................................................................iii

List of Figures........................................................................................................................................vi

List of Tables.......................................................................................................................................vii

1. Introduction and Background .......................................................................................................1
   1.1 Introduction..............................................................................................................................1
   1.2 Background and Rationale.......................................................................................................1
   1.3 This Document.........................................................................................................................2

2. The Web-based Network Management Tool using SNMP......................................................3
   2.1 About the Network Management Tool...................................................................................3
   2.2 User Interface Description.......................................................................................................3
      2.2.1 System Information Tab.....................................................................................................4
      2.2.2 Interface Information Tab................................................................................................5
      2.2.3 Port Scan Tab.....................................................................................................................5
      2.2.4 MIB Browser Tab...............................................................................................................6
      2.2.5 Help Page..........................................................................................................................7
   2.3 Concepts related to Network Management Tool.......................................................................8
      2.3.1 MIB..................................................................................................................................8
      2.3.2 SNMP.............................................................................................................................12
      2.3.3 Components of SNMP.....................................................................................................12
      2.3.4 SNMP in Detail...............................................................................................................13
3. Project Environment and System Design

3.1 Requirements of the Project

3.2 Analysis of Requirements

3.3 Choice of Programming Languages

3.3.1 Security of Network Information

3.4 System Requirements

3.5 Design of the Tool

3.5.1 Why Client-server Architecture

3.5.2 Conceptual Design of the Tool

3.5.3 Data Flow Diagrams of Level One of the Software Design

4. Implementation

4.1 Client Implementation

4.1.1 Class NMS

4.1.2 Class NMSClient

4.1.3 Class Session

4.1.4 Class SIPanel

4.1.5 Class EIPanel

4.1.6 Class PSPanel

4.1.7 Class MIBPanel

4.2 Server Implementation

4.2.1 Class Server

4.2.2 Class NMSServer

4.2.3 SNMP Package
4.2.4 Class SysInfo ......................................................... 40
4.2.5 Class EtherInfo .................................................... 42
4.2.6 Class PortScan ................................................... 45
4.2.7 Class MIBTreeInfo ............................................... 46

5. Testing and Evaluation ................................................ 49
   5.1 Software Testing .................................................. 49
   5.2 Usability Testing .................................................. 49

6. Results and Conclusions .............................................. 52
   6.1 Results .............................................................. 52
   6.2 Conclusions ........................................................ 53

7. Future Work .......................................................... 54

8. Acknowledgements ................................................... 55

9. Bibliography and References ........................................ 56

10. Appendix A: Questionnaire for Testing the Usability of the Tool ........................................ 58
11. Appendix B: Digital Media of the Project .......................... 59
LIST OF FIGURES

1. Figure 2.1 A Screen shot of the Applet– System Information Tab……………………4
2. Figure 2.2 A Screen shot of the Applet – Interface Information Tab…………………5
3. Figure 2.3 A Screen shot of the Applet – Port Scan Tab…………………………6
4. Figure 2.4 A Screen shot of the Applet – MIB Browser Tab………………………7
5. Figure 2.5 A Screen shot of the Tool - Help page……………………………………8
6. Figure 2.6 OID Tree…………………………………………………………………11
7. Figure 2.7 Framework of SNMP Managed Network………………………………13
8. Figure 2.8 PDU Formats………………………………………………………………15
9. Figure 3.1 Client-Server Architecture………………………………………………18
10. Figure 3.2 Conceptual Level Diagram of the System……………………………20
11. Figure 3.3 Level One Data Flow Diagram for System Information Tab………21
12. Figure 3.4 Level One Data Flow Diagram for Interface Information Tab………22
13. Figure 3.5 Level One Data Flow Diagram for Port Scan Tab…………………22
14. Figure3.6 Level One Data Flow Diagram for MIB-Browser Tab………………23
15. Figure 4.1 Interactions between Client, Server and SNMP Enabled Devices……36
LIST OF TABLES

4.1 Method Summary for Class NMSClient......................................................26
4.2 Method Summary for Class Session..........................................................27
4.3 Method Summary for Class NMSServer.....................................................38
4.4 Method Summary for Class SysInfo..........................................................41
4.5 Ether Info related OIDs and their location on MIB Tree..............................43
4.6 Method Summary for Class EtherInfo.......................................................44
4.7 Method Summary for Class Port Scan......................................................46
4.8 Method Summary for Class MibTreeInfo..................................................47
1. INTRODUCTION AND BACKGROUND

1.1 Introduction

Inter-network management becomes more challenging, as the network of an organization grows bigger. Managing a heterogeneous network with multiple vendor devices adds to the challenge faced by the network manager. Network Management can be described as: “Operations, administration, maintenance, and provisioning of network and services.” [Mani 2000]. The goal of network management is to ensure that the users of a network receive the services with the quality that they expect. This translates to a constant, close, and effective monitoring of network and trouble-shooting the network devices, which is quite a serious and tedious job for the network manager. A number of management tools have been developed by multiple vendors for solving a myriad network issues ranging from common management services like node discovery to intelligent auto-discovery using Simple Network Management Protocol (SNMP). In this project a Network Management Tool has been developed to monitor and manage the network at the College of Business (CoB) using SNMP protocol.

1.2 Background and Rationale

The network manager at CoB at Texas A&M University - Corpus Christi (TAMU-CC) has a LAN that is supported by 8 servers viz., 5 Windows 2000 servers, 2 Windows NT 4.0 servers and One Linux RedHat 8.0 server. The manager uses network monitoring tools like ping, traceroute and portscan to check the network parameters like status of nodes and links, and also monitors Web and email servers from time to time. The network manager at CoB has expressed a strong need to have a Web-based Network
Management Tool, that not only will show the uptime and routing information to monitor the status of nodes and links, but also provide the facility to manage the network and be able to manipulate some of the network parameters. After discussing with network manager a list of detailed requirements was prepared. Based on these requirements software design was developed. The next five chapters of this document describe the design and implementation of the whole project in a detailed way.

1.3 This document

This document is organized into six chapters to describe the project. Chapter one gives a brief background and overview of the problem and describes the need for its implementation. Chapter two describes the basic concepts and the graphical user interface. Chapter three describes the software design with data flow diagrams. Chapter four describes the implementation of the design and the way project is coded with excerpts from key classes and methods. Chapter five describes the usability testing and evaluation of the Web-based Network Management Tool. Chapter six brings out the results of this project and describes the scope of future work.
2. THE WEB-BASED NETWORK MANAGEMENT TOOL USING SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

2.1 About the Network Management Tool

The Network Management tool is a Web-based tool. It is a java applet that retrieves information for the target IP address and SNMP Community name provided by the user. The applet is a tabbed interface that has separate tabs to display system information, Ethernet information, Port Scan and MIB tree. System information includes system name, system contact, system location, system uptime and system description. Interface information includes number of interfaces, type of the interface, IP-address, physical address, and Maximum Transmission Unit (MTU). The tool also performs port-scan for popular ports like HTTP, FTP, TELNET, POP3, SMTP and PRINT. The tool provides a MIB browser to view MIBs on any SNMP enabled devices like router, server, and workstation. With the set operation, the manager can set the values of some of the parameters.

2.2 User Interface Description

The user interface consists of a java applet with tabbed JPanel as interface. With the use of JTabbed Pane feature of Java [Naughton 2000], different tabs are used to display information for each of the four features namely System Information, Interface Information, Port Scan, and MIB Browser. Yet, all features appear on the same applet at one time and manager can choose to use the feature he wishes. Just above the tabbed interface, manager is provided with two input boxes to input IP Address and SNMP community name.
2.2.1 System Information Tab

Manager can enter IP Address and SNMP community name in the two input boxes provided, just above the tabs. With valid IP Address and valid SNMP Community name in the input boxes, the manager can press the get System Information button on System Information Tab. If the IP Address is not valid or if the community name is not valid the authentication fails and appropriate error message is triggered. If the values entered are correct, the system information is successfully displayed. Figure 2.1 depicts System Information Tab of the Applet.

![System Information Tab](image)

Figure 2.1 A Screen Shot of the Applet – System Information Tab
2.2.2 Interface Information Tab

The second tab on the applet is the one that displays Interface Information of the network device. Information such as number of Ethernet interfaces present in the device like 1 or 2, type of each interfaces for example, loop-back or Motorola surf-board, the IP-address of the interface; 127.0.0.1 or 66.69.249.123, the physical address of the device and the Maximum Transmission Unit (MTU), as shown in Figure 2.2.

![Figure 2.2 A Screen Shot of the Applet – Interface Information Tab](image)

2.2.3 Port Scan Tab

The third tab displays information about the major active ports within the device at that point of time. Popular ports like HTTP, TCP ECHO, FTP, TELNET, POP3,
SMTP, SNMP, and PRINT are scanned. A check box is provided for each port, which is checked if the port is reachable. Figure 2.3 provides a screen shot of the tool.

![Figure 2.3 A Screen Shot of the Applet – Port-Scan Tab](Image)

2.2.4 MIB Browser Tab

MIB browser is of much importance for the Network Manager at the College of Business, as this would allow the manager to browse the MIBs and manipulate the values of some of the parameters, as required by the manager. The MIB information of the device is retrieved and displayed hierarchically. All the OIDs along with the parameters the OIDs represent are displayed. Figure 2.4 provides a screen shot of the tool.
2.2.5 Help Page

A Tutorial / Help Page is provided to give a quick information about the Network management Tool, Standard MIB as well as other MIBs and SNMP versions. Figure 2.5 shows a screen shot of the Help Page.
The requirements of the client, analysis of requirements, design of the user interface, and the choice of programming language for the project is provided in the section for Project Environment and System Design.

2.3 Concepts related to the Network Management Tool

2.3.1 MIB

The network devices like routers, hubs, bridges, and servers have managed objects inside them and they store information about themselves in Management Information Base (MIB). These objects are defined and identified with Object Identification Numbers (OID) and are organized in a hierarchical tree structure,
technically called Management Information Base (MIBs). Vendors build MIBs when they manufacture network devices.

Internet Activities Board (IAB), has designated Simple Network Management Protocol (SNMP), Structure and Identification of Management Information (SMI) and Management Information Base (MIB) as full Standard Protocols with recommended status. With this all the vendors who manufacture TCP/IP compliant network devices should follow and implement the above three protocols.

“Management information is viewed as a collection of managed objects, residing in a virtual information store, termed the Management Information Base (MIB). Collections of related objects are defined in MIB modules. These modules are written using a subset of Open System Interconnect (OSI)’s, Abstract Syntax Notation One (ASN.1), termed the Structure of Management Information (SMI)” [RFC1907 1996].

The agent inside the managed devices collects and stores management information and makes this information available to Network Management Systems (NMSs) when NMSs query the agents. The complete dialogue between the NMS and an agent takes place through SNMP. These agents are responsible to provide local information about number of objects within the network device such as system objects, interface objects, tcp, udp, icmp and snmp activities. MIB information is conceptualized in a tree format for the convenience of the users or network managers. These MIBs are accessed using a network management protocol like SNMP. The agent, present inside a managed device, helps NMS to read the information contained in the MIBs through SNMP protocol.
There are two types of managed objects namely scalar and tabular. Scalar objects define a single object instance. Tabular objects define multiple related object instances that are grouped in an MIB table. An Object Identification Number is used to uniquely identify a managed object within a managed device. For example: the OID for object sysDescription is 1.3.6.1.2.1.1.1. What does this number mean? Reading from left to right 1 represents ISO (Figure 2.6), 3 represents Identified Organizations, 6 represents Department of Defence, 1 represents Internet, 2 represents Management, 1 represents MIB-II, 1 represents System (object) and 1 represents system description.

The MIB hierarchy can be depicted as a tree with a nameless root, levels of which are assigned by different organizations as depicted in Figure 2.6. The top level MIB Object identifiers are organization by International Standards Organization (ISO). Lower level MIBs are allocated by the respective parent organization. An MIB is extensible by virtue of its experimental and private level branches. All the vendors come under private branches. MIBs that are not standardized are typically placed and categorized under experimental branch.
Figure 2.6 OID Tree
2.3.2 SNMP

Simple Network Management Protocol (SNMP) is a vendor-independent protocol for transporting management data between networked devices and applications and the systems that control and monitor those devices and applications.

SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth. Three versions of SNMP have evolved over time namely, SNMP version 1 (SNMPv1), SNMP version 2 (SNMPv2) and SNMP version 3 (SNMPv3). SNMP (v1) is basic and is not in much use anymore, SNMPv2 offers additional protocol operations over SNMPv1, and SNMPv3 offers additional security, though; all three versions have a number of features in common.

2.3.3 Components of SNMP Managed Network

An SNMP managed network consists of three main components namely, managed devices, agents, and Network Management Systems (NMSs) as illustrated in Figure 2.7.

A managed device or a network element is a network node like computer hosts, printers, routers, bridges etc., which contains an SNMP agent. The agents inside the managed devices collect and store management information and make this information available to NMSs using SNMP.

An agent is a network-management software module that resides inside a managed device. An agent has local knowledge of management information like system information, interface information like interface number, interface description and translates that information into a form compatible with SNMP.
A **NMS** executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs exist on any managed network.

![Framework of SNMP Managed Network](image)

**Figure 2.7 Framework of SNMP Managed Network**

### 2.3.4 SNMP in Detail

SNMP provides basic commands for protocol operations and security, defines message formats and data types in order to monitor and control the managed devices. SNMP v1 and SNMPv2 differ in message formats, data types, protocol operations and security. SNMPv1 [RFC2089 1997] offers operations like

- **Get** – Retrieves an object instance from the agent.
Get-next – Retrieves the next object instance from a table or list within an agent.

Set – Sets object instances within an agent.

Trap – Informs asynchronously the NMS of some event. Trap does not elicit a response from the receiver.

SNMPv2 has included two new protocol operations namely, GetBulk and Inform.

GetBulk - to retrieve large blocks of data, such as multiple rows in a table.

Inform - allows one NMS to send trap information to another NMS and then to receive a response.

SNMP uses a subset of Abstract Syntax Notation.1 (ASN.1) to accommodate communication between diverse systems. The standard ASN.1 data types are Integer, Octect, String, and Object Identifier.

SNMPv2 was published in RFC 1901. SNMPv2 offers enhanced features over SNMPv1. The SNMPv2 made enhancements to the existing SNMPv1 Structure of Management Information (SMI) data types. New SMI data types like 

Bit Strings, Network Addresses and Counters were added [RFC1901 1996], along with Integer, Octet, String, and Object Identifier. SNMPv2 provides for the exchange of messages, which convey management information between the agents and the management stations.

The form of these messages is a message "wrapper" which encapsulates a Protocol Data Unit (PDU) [RFC1901 1996]. The structure of PDU is depicted in the Figure 2.8. The Message consists of Message Header and a PDU. Message header consists of Version Number and Community Name. The PDU consists of Error Status and Error Index fields for Get, GetNext, Inform, Set and Trap. The PDU for GetBulk consists of Non-repeaters and Max repetitions.
Since SNMPv1 did not implement authentication, many vendors did not implement set operation, thereby reducing SNMP to a monitoring facility [Cisco 2002]. But in SNMPv2, the authentication, authorization, and access control are incorporated.

The operation of the set-request PDU is improved in several ways over SNMPv1 [Kunapareddy 1998]. In this project, The Network Management Tool is built using SNMPv2 data types and operations.
3. PROJECT ENVIRONMENT AND SYSTEM DESIGN

3.1 Requirements of the Project

The Network Manager for CoB, Mr. James Davis was interviewed. The existing needs as well as the future requirements for monitoring the network of CoB were discussed and prioritized. The following requirements were identified.

1. Be able to monitor system parameters like system name, contact, location, OID, description, and uptime.
2. Be able to know the number of active interfaces, their IP addresses, sub-net masks, hardware addresses, their speeds, and MTUs.
3. Be able to browse the MIB Tree of SNMP enabled devices to check the values of various parameters like TCP traffic, UDP traffic, connection errors, and user statistics.
4. Be able to retrieve the values of desired parameters.
5. Be able to scan ports to check what major ports are active.
6. To check the link status on a network, by trace-route.

Based on the discussion, a Web-based system that would provide flexibility in monitoring the devices was preferred to software that would run on a standalone workstation/client. Hence, Web-based system is considered for the Network Management Tool. The requirements influenced the choice of the programming languages as well as the design of the Network Management Tool.

3.2 Analysis of the Requirements

A careful analysis of the requirements and a thorough review of the literature revealed that, the first four requirements could be achieved by retrieving information
about the desired parameters from MIBs. Thus the major portion of the project is dependent on mining the MIBs. The sixth and seventh requirements namely port scan and trace route can be achieved through coding the special functions.

3.3 Choice of Programming Languages

A programming language that supports a Web-based application and also has object oriented features to hide the details of implementation of complex data retrieval as well as provide reusability of classes is ideal for the project. Java Programming Language aptly fits the requirements. API for Java 2 Platform Standard Edition v1.4.1 and Java SNMP APIs developed by Jonathan Sevy [Sevy 2000] for java based SNMP queries and other implementations were used for the construction of the Network Management Tool. Java tabbed panel was used to display all the information retrieved from MIBs. Hyper Text Markup Language (HTML) was used for scripting the Web pages.

3.3.1 Security of Network Information

For security of the sensitive network information OpenSSL is used. The Web pages related to Network Management System are SSL enabled.

3.4 System Requirements

The system was developed on an IBM-compatible personal computer that had 128 MB RAM and a 7 GB hard drive to sufficiently support the system. The system was developed and tested on standard Web-browsers like Netscape, Internet-Explorer, and Mozilla, so that the network manager can use the system from any standard Web-browser.
3.5 The Design of the Tool

The design of the tool adopts Client-Server architecture as depicted in Fig 3.1. The client is the Network Management Tool applet (from here on ‘the applet’) on the Web browser. The server program is called Network Management Server (NMS) and it runs on any machine that supports Java 2 Standard Edition (J2SE) version 1.4.1 and above. The applet initiates the connection and communicates with the server via a TCP socket on which NMS is listening. The NMS listens for connections on a particular TCP port, in this case port 6789, and obtains a socket to use for the communication. Once the connection is established between the applet and NMS, the applet sends a request to the NMS to retrieve particular data. NMS in turn sends the request to SNMP enabled device through UDP port 161 and waits for the device to respond. The SNMP agent inside SNMP enabled device sends the requested information back to NMS. NMS sends the information back to the applet through TCP port. The client displays the information on the applet. This process is illustrated through Figure 3.1

![Figure 3.1 Client–Server Architecture](image-url)
3.5.1 Why Client–Server Architecture?

Java applets are run on any browser that supports Java and are run from the user’s machine from anywhere in the world. Since it is run on the user’s machine, there is a greater possibility for a malicious programmer to write applets that might destroy local system files or retrieve sensitive information from the remote user’s machine. To avoid such security breaches, there are some restrictions on capabilities of an applet. The following are the limitations of an applet. [Sun 2003]:

1. Applets cannot load libraries or define native methods.
2. An applet cannot ordinarily read or write files on the host that is executing it.
3. An applet cannot make network connections except to the host that it came from.
4. An applet cannot start any program on the host that is executing it.
5. An applet cannot read certain system properties.

With these restrictions, it is not possible for an applet to request SNMP agents in SNMP enabled devices and retrieve MIB information.

To get around this problem Policy Tool feature of Java was explored. Policy Tool feature can provide a trusted applet with network permissions, which enables the applet to accept, connect, listen, and resolve on a host. But, it is practically not possible to implement Policy Tool Feature on all the SNMP enabled devices like printers, routers, hubs and such, on a network. Hence, the concept of using Policy Tool feature of Java was eliminated.

It is clear that an applet alone is not sufficient to retrieve security sensitive network information from various network devices. But to keep up with the requirement of a
‘Web-based Tool’, having an applet is imperative. At this point, client-server architecture is considered as a feasible alternative. The applet is used as a client that would only send TCP/IP requests to a server and receives TCP/IP replies from the server. The server program constantly running on a machine would listen to applet’s requests and in turn query the SNMP enabled device and send the information back to the client (applet), which would just display the information. Ultimately, the path of client-server architecture proved successful.

3.5.2 The Conceptual Design of the Tool

The NMS tool can be described at the conceptual level as a tool that accepts input in the form of an IP address from the manager and retrieves required information from a device using SNMP and outputs on the screen. Figure 3.2 shows the Conceptual Level Diagram of the tool.

![Conceptual Level Diagram of the System](image)

**Figure 3.2 Conceptual Level Diagram of the System**

3.5.3 Data Flow Diagrams of Level One of the Software Design

*The main page with Network Management Tool applet*

The main page is the one that contains Network Management Tool applet. The applet uses a JTabbedPane to accommodate all the features of Network Management Tool. On the first tab that is named “Sys Info”, the manager is provided with two input boxes – one to enter the IP address of the destination device, and the other for community
name of the SNMP enabled device. (If there is no community name, the default community name is ‘public’). The input from the applet goes to the NMS, which in turn queries the SNMP enabled device and validates if the input provided is correct or incorrect. There are two outcomes for this event, success or failure. If the input is valid, the values for System Info are retrieved by NMS and returned to the client. If input is invalid, the NMS sends back the failure message. This gives another opportunity for the manager to enter valid IPAddress and/or valid SNMP community name. Upon valid input, the server processes the rest of the query to retrieve ‘System Information’ from the SNMP enabled device and send it back to the client. Figure 3.3 illustrates the mechanism in detail.

![Figure 3.3 Level One Data Flow Diagram for System Information Tab](image)

*Figure 3.3 Level One Data Flow Diagram for System Information Tab*

*Interface Information Tab*

When the manager chooses Interface Info tab on the applet, and presses the get information button, the IP Address of the target device provided in the System Information tab is used. The request to retrieve interface information is sent to the NMS, which further sends the request to the device. Figure 3.4 shows the DFD of Interface Information Page.
Port-Scan Tab

The port scan function does not use SNMP protocol. On Port Scan tab, when manager presses scan ports button, the target SNMP device is scanned for 10 major ports. The request is sent through TCP/IP layer and reach the target device and retrieves status of the port—whether active or passive and displays the response on the Port Scan Page for all major ports. DFD provided in Figure 3.5 explains the scenario in detail.

MIB-Browser Page

The initial IP address entered by the manager on System Information Page is stored in a temporary variable and the same IP is used in the MIB-Browser Page. The MIB tree structure is retrieved from the host and displayed on MIB-Browser Page. This Screen also allows the manager to browse the MIB tree and check the values of various
parameters and also to modify the values of some parameters. Some parameters cannot be changed and they hold default values. Some values can be changed by `setRequest` function of SNMP. Figure 3.6 shows the DFD for MIB-Browser Page.

![Figure 3.6 Level One Data Flow Diagrams for MIB-Browser Tab](image)

**Help Page**

Help Page is an HTML page that serves as a resource for MIB related questions and other FAQs. It includes Uniform Resource Locators (URLs) to most popular Web sites such as Request For Comments (RFCs) related to SNMP, MIB-2 and www.faqs.org and many other knowledge based Websites. It also provides help on how to use the MIB information in Network monitoring.

Thus the design of the tool follows the DFDs mentioned in this chapter. The code implementation is described in the next chapter.
4. IMPLEMENTATION

This chapter narrates the project implementation with code excerpts. The project follows a client–server architecture. The project includes 6 server-side and 8 client-side programs. Client programs include:

1. NMS.java (NMS: Network Management System)
2. NMSClient.java (NMSClient: Network Management System Client)
3. Session.java
4. SIPanel.java (SIPanel: System Information Panel)
5. EIPanel.java (EIPanel: Ethernet Information Panel)
6. PSPanel.java (PSPanel: Port Scan Panel)
7. MIBPanel.java (MIBPanel: Management Information Base Panel)
8. mib_dat.java

Server programs include:

1. Server.java
2. NMSServerjJava
3. SysInfo.java
4. EtherInfo.java
5. PortScan.java
6. MibTreeInfo.java

These Java programs when compiled produce classes, which are the actual executables. In this document, from now on the Java programs are referred as classes.
4.1 Client Implementation

The client programs are Java classes that have specific methods in them to perform various tasks.

4.1.1 Class NMS

Class NMS is the driver program on the client side. Its purpose is to:

1. Provide the basic tabbed panel interface for the Network Management Tool applet.
2. Instantiate individual class constructors namely, SIPanel, EIPanel, PSPanel and MIBPanel.
3. Instantiate Session object, pass session object to SIPanel, EIPanel, PSPanel and MIBPanel, through their respective constructors.

These individual classes or ‘panel’ classes provide the features and functionalities within each tab of the applet interface. Each of these classes define a Java panel that holds objects like Java Textboxes, Java Buttons, Java Scroll Panes, Java Tree, Java Checkboxes, Java Dropdown menu etc., which serve different functions on the applet. These objects are interactive, they can display information, accept information from user/manager and some of these objects are controlled by mouse click.

4.1.2 Class NMSClient

Class NMSClient has 4 methods that are common to a client program. As illustrated in Table 4.1, these methods are responsible to open connection with the server program, send request to the server, accept information from the server, and close connection with server program.
Since *SIPanel, EIPanel, PSPanel*, and *MIBPanel* act as individual clients, each of these Panels needs to perform the functions of

1. Accept request from user,
2. Open connection to server,
3. Send the user request to server, wait for server response and
4. Get the information from server back to client and display it on the applet.

The aforementioned functions are common for all the panels. Hence, they are defined within one class called *NMSClient* and each of the panels instantiate *NMSClient* to perform these functions.

**Table 4.1 Method Summary for Class NMSClient**

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean openConnection( )</td>
</tr>
<tr>
<td>Opens socket to the specified server at the specified port</td>
</tr>
<tr>
<td>String outToServer(String request)</td>
</tr>
<tr>
<td>Sends data stream to the server.</td>
</tr>
<tr>
<td>String infromServer( )</td>
</tr>
<tr>
<td>Reads the input stream from the server.</td>
</tr>
<tr>
<td>void closeConnection( )</td>
</tr>
<tr>
<td>Closes the socket connection</td>
</tr>
</tbody>
</table>
4.1.3 Class Session

*SIPanel* is the first tab that appears on the applet, and therefore forms the first interface with which the user interacts. It has text boxes where the user/network manager can enter IPAddress and the community name of the SNMP enabled device. This information given by the user is needed by all other panels to retrieve the information from *NMS* server. Hence IPAddress and community name are stored in a class called *Session* from where it can be retrieved by other panels namely, *EIPanel, PSPanel* and *MIBPanel*, when necessary. Class *Session* has four methods as shown in Table 4.2

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void setCommunity(String comm)</td>
<td>Sets the community name</td>
</tr>
<tr>
<td>void SetIP(String IP)</td>
<td>Sets the IPAddress</td>
</tr>
<tr>
<td>String getCommunity()</td>
<td>Gets community name</td>
</tr>
<tr>
<td>String getIP()</td>
<td>Gets the IPAddress</td>
</tr>
</tbody>
</table>

4.1.4 Class SIPanel

*SIPanel* provides the first tabbed interface that the user would interact. It has few text boxes and a button.
1. **SIPanel** accepts user information: IP address and SNMP community name and passes the information to the object **Session** with the help of **Session** methods **setIP** and **setCommunity**.

2. **SIPanel** validates the user input. It is done in two steps:
   a. Checks for null or malformed IP address. If the IP is not null or malformed, then it opens connection with server and sends IP and community to NMSServer.
   b. NMSServer in turn checks with the SNMP enabled device and confirms the validity of the input back to SIPanel.

3. At the click of the button, ‘Get System Info’, it acts as a client to the server program. It instantiates **NMSClient** - opens socket connection with NMSServer, if connection is established successfully, it identifies itself as ‘SysInfo’ to the server and then sends requests to the server, waits for server response, retrieves information from server and closes connection with server.

The following code snippet shows IP validation process within SIPanel.

Textfield1 and Textfiled2 are where the user enters IP address and community name.

```java
input = txtfld1.getText();
if(input.equals("")){
    JOptionPane.showMessageDialog(null, "Empty String in IPAddress field");
    return;
}
try{
    InetAddress hostAddress =
    InetAddress.getByName(input);
} catch(Exception ue) {
    JOptionPane.showMessageDialog(null, "Malformed IP Address " + input );
    return;
}
```
The code snippet below shows how SIPanel opens connection with NMServer program and the error message it throws when it is not able to connect to NMServer:

```java
NMSClient nmscli = new NMSClient();
if(!nmscli.openConnection()){
    JOptionPane.showMessageDialog(null, "Server Error:
    Could not connect to Server" );
    return;
}

input = "SysInfo";
nmscli.outToServer(input);
reply = nmscli.inFromServer();

nmscli.outToServer("OK");
reply = nmscli.inFromServer();

if(reply.equals("NoConnection")){
    JOptionPane.showMessageDialog(null, "Could not
    establish SNMP connection with given IPAddrress and
    SNMP community name");
    return;
}

//else of connection is established system information is
//retrieved with the following code

nmscli.outToServer(input);
reply = nmscli.inFromServer();
....
....
...
//After retrieval of information connection with server is
//closed
try {
    nmscli.closeConnection();
} catch (Exception e) {}
4.1.5 Class EIPanel:

Class EIPanel corresponds is the second tab on the applet user interface. The class EIPanel provides,

1. A tabular interface in which interface (Ethernet) information is displayed.
2. At the click of the button, ‘Get Ether Info’, it acts as a client to the server program. It instantiates NMSClient -
   a. Opens socket connection with the server, if unable to open connection
      throws an exception as shown in following code:
      ```java
      if(!nmscli.openConnection()){
          JOptionPane.showMessageDialog(null, "Server Error: Could not connect to Server");
          return;
      }
      ```
   b. Retrieves the IPAddress and Community name from Session object.
      ```java
      input = ses.getIP();
      input = ses.getCommunity();
      ```
   c. Checks for null and malformed IPAddress.
   d. Then sends IPAddress and Community name to NMS server. Also identifies itself as EtherInfo to server.
      ```java
      input = "EtherInfo";
      nmscli.outToServer(input);
      reply = nmscli.inFromServer();
      txt = reply;

      input = ses.getIP();
      nmscli.outToServer(input);
      reply = nmscli.inFromServer();

      input = ses.getCommunity();
      nmscli.outToServer(input);
      reply = nmscli.inFromServer();
      ```
   e. If the information provided is valid, server responds back.
      ```java
      nmscli.outToServer("OK");
      reply = nmscli.inFromServer();
      ```
EIPanel further requests interface information, waits and retrieves information.

```java
while(!(reply.equals("EIDone"))){
    if(i == 2){
        j++;
        i = 0;
    }
    eiData[i][j] = reply.toString();
    i++;
    nmscli.outToServer(reply);
    reply = nmscli.inFromServer();
}
for(i = 0; i < 2; i++)tableModel.addRow(new Object[]{"","","","","","",""});
for(i = 0; i < 2; i++){
    for(j = 0; j < 8; j++){
        txt = txt + eiData[i][j] + "" + '\t';
        jt.setValueAt(eiData[i][j], i, j);
    }
    txt = txt + "" + '\n';
}
```

f. Once NMSServer finishes with giving information to EIPanel, connection is closed with the NMSServer.

```java
try{
    nmscli.closeConnection();
} catch (Exception e) {};
```

4.1.6 Class PSPanel:

PSPanel provides the user interface for the third tab on the applet. It has Java check boxes against port names, and a ‘PortScan’ button.

a. It instantiates NMSClient when ‘PortScan’ button is clicked by the user, thereby opening connection with the NMS server.

b. It retrieves IP Address and Community name provided by the user. From the Session object, PSPanel opens connection with the server and identifies
itself as PortScan.

```java
    input = "PortScan";
    nmscli.outToServer(input);
    reply = nmscli.inFromServer();
```

c. Once connection is established and the IPAddress and Community name is valid, it scans for nine major ports on the SNMP enabled device. For the ports that are open on the SNMP enabled device the check boxes are checked.

d. After completion of scanning ports, the connection is closed with the server.

The following code excerpt illustrates the process.

```java
{
    ..
    ..
      //Telnet
      nmscli.outToServer("OK");
      reply = nmscli.inFromServer();
      if(reply.equals("1"))
        jcb3.setSelected(true);

      //SMTP
      nmscli.outToServer("OK");
      reply = nmscli.inFromServer();
      if(reply.equals("1"))
        jcb4.setSelected(true);

      //HTTP
      nmscli.outToServer("OK");
      reply = nmscli.inFromServer();
      if(reply.equals("1"))
        jcb5.setSelected(true);

    ..
    ..
    ..
    ..
}
```
4.1.7 Class MIBPanel

Class MIBPanel is the largest class and it provides interface for MIBInformation Tab. MIBPanel holds many objects like MIBTree structure, Java text boxes, Java text areas, Java drop-down menu and Java buttons. It also retrieves all the information for the objects of MIBTree such as OID, syntax, access, status, description and value.

a. MIBTree nodes are controlled by mouse click, when the nodes are clicked; they expand to show the sub-tree. It is a Java Tree feature.

b. When a leaf is clicked the entire path of the leaf within the MIB-tree is displayed, along with its OID, access, syntax, status and brief description about the parameter. A two-dimensional array consisting of 6 columns and around 450 rows holds path of each variable of the MIBTree, its OID, syntax, status, and description. At the mouse click on a particular node of MIBTree, this array is stepped through and information is displayed. The following is the code that handles this process.

```java
void doMouseClicked(MouseEvent me) {
    TreePath tp = mibTree.getPathForLocation(me.getX(), me.getY());
    if (tp != null)
        jtf.setText(tp.toString());
    else
        jtf.setText(" ");

    String input, reply;
    String[] mibTreeInfo = {"", "", "", "", "", ""};

    mib_dat mibdat = new mib_dat();
    for (int i = 0; i < 451; i++) {
        if (mibdat.mibdata[i][0].equals(tp.toString())) {
            oid.setText(mibdat.mibdata[i][1]);
            syntax.setText(mibdat.mibdata[i][2]);
            access.setText(mibdat.mibdata[i][3]);
            if (access.getText().equals("read-write")) {
                setOID.setEnabled(true);
            }
        }
    }
    Display.showInputBox("input", &input);
    Display.showReplyBox("reply", &reply);
}
```
setOID.setActionCommand("setOID");
setOID.addActionListener(this);
}
else
setOID.setEnabled(false);
status.setText(mibdat.mibdata[i][4]);
descrip.setText(mibdat.mibdata[i][5]);
}
}

c. With the leaf selected, if GetOID button is clicked it displays the value contained by the parameter within the SNMP enabled device. The following code snippet shows how GetOID works.

```java
if(command == "getOID"){
    if(!nmscli.openConnection()){
        JOptionPane.showMessageDialog(null, "Server Error: Could not connect to Server");
        return;
    }
    input = "MibOIDInfo";
nmscli.outToServer(input);
    reply = nmscli.inFromServer();
    txt = reply;
    nmscli.outToServer(oid.getText());
    reply = nmscli.inFromServer();
    StringTokenizer st =
    new StringTokenizer(reply, "|");
    while (st.hasMoreTokens()){
        mibInfo.append(st.nextToken());
        mibInfo.append("+\n");
    }
    try {
        nmscli.closeConnection();
    } catch (Exception e) {}
}
```

d. Some of the parameters for which the access is read/write, their values can be set to the value desired by the user. First, the leaf for which the access is read-write is to be selected by clicking once on the parameter. Then the appropriate new value is to be inputted in the text box provided. Also appropriate data type
from the dropdown menu is to be selected. Finally clicking the ‘SetOID’ button will set the new value.

4.2 Server Implementation

The server programs are Java classes that have specific methods in them to perform various tasks.

4.2.1 Class Server

Class Server is the driver program on the server side. Its purpose is to instantiate the NMSServer class.

4.2.2 Class NMSServer

Class NMSServer is the most important program. It acts as a liaison between the client and SNMP agent. NMSServer listens for socket connection from client. When a client tries to open connection through a socket, NMSServer accepts the connection. Then it checks for the clientSentence i.e., if the client is SIPanel or EIPanel or PSPanel or MIBPanel. Each of these panel classes while establishing connection with server also identifies itself as SysInfo, EtherInfo, PortScan or MibOIDInfo. Hence, the server knows which client or panel has established connection and the server would further make instantiate appropriate class that would retrieve specific information from SNMP agent, as illustrated in Figure 4.1.
As shown in Figure 4.1 when client (SIPanel or other panels) makes a successful connection with NMSServer it identifies itself uniquely to the server, for example, SIPanel identifies itself as ‘SysInfo’. NMSServer further calls methods of other classes like SysInfo by a method processSysInfoRequest. Instance of class SysInfo with its methods interacts with SNMP Agent and retrieves the system information requested by the client SIPanel and passes it to NMSServer. NMSServer in turn sends it to SIPanel. For each of the clients/panels this process is repeated as shown in the following code snippet.

```java
if(clientSentence.equals("SysInfo")) {
    returnSentence = "Processing SysInfo" + \n';
    outToClient.writeBytes(returnSentence);
    processSysInfoRequest();
}
else if(clientSentence.equals("PortScan")) {
    returnSentence = "Processing PortScan" + \n';
    outToClient.writeBytes(returnSentence);
    processPortScanRequest();
}
else if(clientSentence.equals("EtherInfo")) {
    returnSentence = "Processing EtherInfo" + \n';
    outToClient.writeBytes(returnSentence);
    System.out.println(clientSentence);
    processEtherInfoRequest();
}
```
else if(clientSentence.equals("MibTreeInfo")){
    returnSentence = "Processing MibTreeInfo" + '\n';
    outToClient.writeBytes(returnSentence);
    System.out.println(clientSentence);
    processMibTreeInfoRequest();
}

There are six methods in class NMSServer to process requests from clients and to make calls to methods of other classes to retrieve information from SNMP agents. These methods are summarized in Table 4.3. Method processSysInfoRequest listens to SIPanel/client request, reads the IP and Community name from client, and passes those values as parameters while it calls methods of class SysInfo as shown in the following code snippet, after Table 4.3.
<table>
<thead>
<tr>
<th>Method Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void processSysInfoRequest()</code></td>
<td>instantiates class <code>SysInfo</code> and passes IP, Community name</td>
</tr>
<tr>
<td></td>
<td>calls methods of <code>SysInfo</code>, namely: <code>getSysName</code>,</td>
</tr>
<tr>
<td></td>
<td><code>getSysContact</code>, <code>getSysLocation</code>, <code>getSysDescr</code>, <code>getSysUptime</code></td>
</tr>
<tr>
<td><code>void processEtherInfoRequest()</code></td>
<td>instantiates class <code>EtherInfo</code> and forwards requests received</td>
</tr>
<tr>
<td></td>
<td>from class <code>EIPanel</code>, retrieves information from <code>EtherInfo</code> by</td>
</tr>
<tr>
<td></td>
<td>calling <code>EtherInfo</code> methods</td>
</tr>
<tr>
<td><code>void processPortScanRequest()</code></td>
<td>instantiates class <code>PortScan</code> and forwards requests received</td>
</tr>
<tr>
<td></td>
<td>from class <code>PSPanel</code>, retrieves information from <code>PortScan</code> by</td>
</tr>
<tr>
<td></td>
<td>calling <code>PortScan</code> methods</td>
</tr>
<tr>
<td><code>void processMibTreeInfoRequest()</code></td>
<td>instantiates class <code>MibTreeInfo</code> and forwards requests received</td>
</tr>
<tr>
<td></td>
<td>from class <code>MIBPanel</code>, retrieves information from <code>MibTreeInfo</code></td>
</tr>
<tr>
<td></td>
<td>by calling <code>MIBInfo</code> methods</td>
</tr>
<tr>
<td><code>void processMibOIDInfoRequest()</code></td>
<td>instantiates class <code>SysInfo</code> and forwards requests received</td>
</tr>
<tr>
<td></td>
<td>from class <code>SIPanel</code>, retrieves information from <code>SysInfo</code> by</td>
</tr>
<tr>
<td></td>
<td>calling <code>SysInfo</code> methods</td>
</tr>
</tbody>
</table>

If `SysInfo` is able to connect to the SNMP device successfully, then the method `processSysInfoRequest` sends out individual requests by calling individual methods of `SysInfo` class to retrieve system name, location, contact, description, and uptime as shown in the following code snippet.

```java
systemInfo = new SysInfo(IP, comm);
//read client request
clientSentence = inFromClient.readLine();
```
//process client requests, if SysInfo establishes SNMP
//connection with the device successfully

if(systemInfo.snmpConnect()){
    //call to method getSysName of SysInfo class
    returnSentence = systemInfo.getSysName() + '\n';
    outToClient.writeBytes(returnSentence);

    //call to method getSysContact of SysInfo class
    clientSentence = inFromClient.readLine();
    returnSentence = systemInfo.getSysContact() + '\n';
    outToClient.writeBytes(returnSentence);

    ...
    ...
    ...
}

Similarly, the other methods namely, process etherInfoRequest,
processPortScanRequest, processMibTreeInfoRequest, and
processMibOIDInfoRequest read from clients, pass IP Address and Community
name as parameters to classes EtherInfo, PortScan, MIBTreeInfo and once successful
connection is established by these classes with SNMP enabled device, then calls specific
methods within these classes to retrieve information from SNMP enabled device.

4.2.3 SNMP Package

Classes SysInfo, EtherInfo, PortScan, and MibTreeInfo communicate between
NMSServer and SNMP agent of the SNMP enabled device. These classes use SNMP
protocol for all its communications. They connect to the SNMP enabled device through
port 161. Dr. Jonathan Sevy of Drexel University developed the SNMP Java API that
facilitates the communication with SNMP agents using SNMP protocol, in 1997. This
API was used for this project. Classes SysInfo, EtherInfo, PortScan and MIBTreeInfo
import package SNMP.
A number of classes in the Java SNMP package represent the standard SNMP data types. Each class represents a data type like Integer, Uinteger32, Gauge, Counter32, Counter64, TimeTicks, Octet string, IPAddress, NSAPAddress, Object identifier, Sequence, VarBind, VarBindList and Null.

The class SNMPv1CommunicationInterface provides much of the user functionality. The constructor for this class opens a datagram socket to a specified host for SNMP communication on the standard port (192). This constructor takes three parameters: an Integer giving the SNMP version in use (0 for SNMPv1 and 1 for SNMPv2), the remote device IPAddress and a string giving the SNMP community to be used for the operations. Several methods of this class such as getMIBEntry(), getNextMIBEntry(), setMIBEntry(), retrieveMIBTable(), provide a means to retrieve and set the values of SNMP variables on the remote device.

4.2.4 Class SysInfo

Class SysInfo has eight methods through which it communicates with the SNMP agent of the SNMP enabled device. These methods are summarized in Table 4.4.
Table 4.4 Method Summary for class SysInfo

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
</tr>
<tr>
<td>snmpConnect()</td>
</tr>
<tr>
<td>makes connection with comInterface of SNMP agent</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>snmpClose()</td>
</tr>
<tr>
<td>closes connection with comInterface of SNMP agent</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>getSysName()</td>
</tr>
<tr>
<td>calls the method getMIBEntry and passes itemID</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>getSysContact()</td>
</tr>
<tr>
<td>calls the method getMIBEntry and passes itemID</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>getSysLocation()</td>
</tr>
<tr>
<td>calls the method getMIBEntry and passes itemID</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>getSysUptime()</td>
</tr>
<tr>
<td>calls the method getMIBEntry and passes itemID</td>
</tr>
<tr>
<td>String</td>
</tr>
<tr>
<td>getSysDescr()</td>
</tr>
<tr>
<td>calls the method getMIBEntry and passes itemID</td>
</tr>
<tr>
<td>void</td>
</tr>
<tr>
<td>getTable(String id)</td>
</tr>
<tr>
<td>calls the method getMIBEntry and passes itemID</td>
</tr>
</tbody>
</table>

In order to communicate with SNMP agent, class SysInfo imports SNMP package.

The following are some of the SNMP object declarations:

```java
SNMPv1CommunicationInterface comInterface;
SNMPVarBindList newVars;
SNMPSequence pair;
SNMPObjectIdentifier snmpOID;
SNMPObj ect snmpValue;
```

Class SysInfo creates a communications interface with a remote SNMP-capable device. For this, version of SNMP, hostIPAddress and community name needs to be passed as parameters. The following is the code snippet to create new comInterface.
comInterface = new SNMPv1CommunicationInterface(version, hostAddress, community);

Once comInterface is created, SNMPVarbindlist object binds comInterface with the method getMIBEntry for the given item id. SNMPSequence retrieves the value. The following is the code snippet to illustrate this process.

```java
String itemID = "1.3.6.1.2.1.1.5.0";
newVars = comInterface.getMIBEntry(itemID);
pair = (SNMPSequence)(newVars.getSNMPObjAt(0));
```

The method getMIBEntry is called by passing itemID through it. Thus it other methods getSysName, getSysContact and so on listed in the Table 4.4

### 4.2.5 Class EtherInfo

Class *EtherInfo* has four methods, which help in retrieving Ethernet interface information from SNMP agent. The default constructor of *EtherInfo* accepts IPAddress and Community name as parameters. The methods of class *EtherInfo* are constructed based on the different OIDs to be retrieved from the MIBTree. In *EIPanel*, the tabular interface is expected to display InterfaceNo. Type, Description, MTU, Speed, Physical Address, IPAddress and Sub-net Mask. All these parameters have different OIDs and are located under different nodes on MIBTree. The parameters actual name and location on MIB Tree is shown in the Table 4.5.
### Table 4.5 EtherInfo related OIDs and their location on MIBTree

<table>
<thead>
<tr>
<th>Item CommonName</th>
<th>Actual Name on MIBTree</th>
<th>OID of the Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface No.</td>
<td>ifIndex</td>
<td>1.3.6.1.2.1.2.2.1.1</td>
</tr>
<tr>
<td></td>
<td>(located under ifTable)</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>IfDescr</td>
<td>1.3.6.1.2.1.2.2.1.2</td>
</tr>
<tr>
<td></td>
<td>(located under ifTable)</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>IfType</td>
<td>1.3.6.1.2.1.2.2.1.3</td>
</tr>
<tr>
<td></td>
<td>(located under ifTable)</td>
<td></td>
</tr>
<tr>
<td>MTU</td>
<td>IfMtu</td>
<td>1.3.6.1.2.1.2.2.1.4</td>
</tr>
<tr>
<td></td>
<td>(located under ifTable)</td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>IfSpeed</td>
<td>1.3.6.1.2.1.2.2.1.5</td>
</tr>
<tr>
<td></td>
<td>(located under ifTable)</td>
<td></td>
</tr>
<tr>
<td>Physical Address</td>
<td>IfPhysAddress</td>
<td>1.3.6.1.2.1.2.2.1.6</td>
</tr>
<tr>
<td></td>
<td>(located under ifTable)</td>
<td></td>
</tr>
<tr>
<td>IPAddress</td>
<td>IpAdEntAddr</td>
<td>1.3.6.1.2.1.4.20.1.1</td>
</tr>
<tr>
<td></td>
<td>(located under ipAddrTable)</td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td>IpAdEntNetMask</td>
<td>1.3.6.1.2.1.4.20.1.3</td>
</tr>
<tr>
<td></td>
<td>(located under ipAddrTable)</td>
<td></td>
</tr>
</tbody>
</table>

The methods of class *EtherInfo* meet the challenge of getting all these information located at different places and arranging them into a tabular format and display them.

Method `getTable` retrieves information under *ifTable* of the MIBTree. Method `getIPTable` retrieves information from *ipAddressTable* of the MIBTree. Both these methods first open a comInterface with SNMP device to start the communication process. Method `getEtherInfo` retrieves the MIB information. It passes OID and a vector as parameters and calls the method `getTable`, and `getIPTable` which in turn retrieves
the information. The code for method getEtherInfo illustrates clearly the process of retrieval of Ethernet information from SNMP device.

```java
public Vector getEtherInfo()
{
    Vector v = new Vector();

    //the following OID corresponds to
    getTable("1.3.6.1.2.1.2.2.1.1", v);
    getTable("1.3.6.1.2.1.2.2.1.2", v);
    getTable("1.3.6.1.2.1.2.2.1.3", v);
    getTable("1.3.6.1.2.1.2.2.1.4", v);
    getTable("1.3.6.1.2.1.2.2.1.5", v);
    getTable("1.3.6.1.2.1.2.2.1.6", v);
    getIPTable("1.3.6.1.2.1.4.20.1.1", v);
    getIPTable("1.3.6.1.2.1.4.20.1.3", v);
    return v;
}
```

Method printVector prints the contents of the vector one after the other with the help of an iterator.

**Table 4.6 Method summary for class EtherInfo**

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void getTable(String id, Vector v)</td>
<td>Gets the ifTable information from MIBTree</td>
</tr>
<tr>
<td>void getIPTable(String id, Vector v)</td>
<td>Gets the ipTable information from MIBTree</td>
</tr>
<tr>
<td>Vector getEtherInfo()</td>
<td>Calls getTable and getIPTable methods many times by sending an ItemID and a vector to hold the retrieved information, each time.</td>
</tr>
<tr>
<td>void printVector(Vector v)</td>
<td>Prints the contents of the vector through an iterator</td>
</tr>
</tbody>
</table>
4.2.6 Class PortScan

PortScan is instantiated by NMSServer. PortScan provides individual methods to scan the major ports. Each method opens a socket on a particular port number. Once a socket successfully opened on a particular port, the socket is closed and integer 1 is returned to NMSServer, which indicates the port is open. If socket was not opened on the port, the method throws an exception, returning 0 to the NMSServer. Thus NMSServer sends the signal to check the checkbox on the PSPanel for the ports that are open. Table 4.7 illustrates the method summary of the class PortScan

```java
public int FTPScan (){
    try{
        socket = new Socket(hostAddress, 21);
        socket.close();
        System.out.println( hostAddress.getHostName() );
        return 1;
    }catch(Exception e){
        return 0;
    }
}
```
Table 4.7 Method Summary of Class PortScan

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>int FTPScan()</td>
</tr>
<tr>
<td>Opens connection on port 21 and return 1 or 0. Closes connection</td>
</tr>
<tr>
<td>int TelnetScan()</td>
</tr>
<tr>
<td>Opens connection on port 23 and return 1 or 0. Closes connection</td>
</tr>
<tr>
<td>int SMTPScan()</td>
</tr>
<tr>
<td>Opens connection on port 25 and return 1 or 0. Closes connection</td>
</tr>
<tr>
<td>int HTTPScan()</td>
</tr>
<tr>
<td>Opens connection on port 80 and return 1 or 0. Closes connection</td>
</tr>
<tr>
<td>int POP3Scan()</td>
</tr>
<tr>
<td>Opens connection on port 110 and return 1 or 0. Closes connection</td>
</tr>
<tr>
<td>int NNTPScan()</td>
</tr>
<tr>
<td>Opens connection on port 119 and return 1 or 0. Closes connection</td>
</tr>
<tr>
<td>int SNMPScan()</td>
</tr>
<tr>
<td>Opens connection on port 161 and return 1 or 0. Closes connection</td>
</tr>
<tr>
<td>int SSLScan()</td>
</tr>
<tr>
<td>Opens connection on port 443 and return 1 or 0. Closes connection</td>
</tr>
<tr>
<td>int PrintScan()</td>
</tr>
<tr>
<td>Opens connection on port 515 and return 1 or 0. Closes connection</td>
</tr>
</tbody>
</table>

4.2.7 Class MibTreeInfo

Class MibTreeInfo is instantiated by class NMSServer to display and set the MIBTree node values or leaf values. There are two methods as summarized in Table 4.8.
Table 4.8 Method Summary of Class MibTreeInfo

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getOID(String id)</td>
</tr>
<tr>
<td>String setOID(String id, String value, String valueType)</td>
</tr>
</tbody>
</table>

Method `getOID` accepts a String variable, which is an OID. It opens a `comInterface` with the SNMP agent and retrieves the OID value from the MIBTree. The following code snippet shows the process.

```
newVars = comInterface.retrieveMIBTable(itemID);
for (int i = 0; i < newVars.size(); i++) {
    pair = (SNMPSequence)(newVars.getSNMPObjectAt(i));
    snmpOID =
        (SNMPObjectIdentifier)pair.getSNMPObjectAt(0);
    snmpValue = pair.getSNMPObjectAt(1);
    String typeString = snmpValue.getClass().getName();
    if(i > 0)
        snmpString = snmpString + "|";
    snmpString = snmpString + snmpOID + ": " +
        snmpValue.toString();
}
```

This method throws two types of exception, ‘interrupted during retrieval’ and ‘exception during retrieval’. This method returns SNMP string that is retrieved.

Method `setOID` accepts three values OID itemid, the value to be set and the datatype of the value to be set. It opens a SNMP `comInterface` and sets the value to the given itemid. The following code snippet illustrates the process.

```
comInterface = new SNMPv1CommunicationInterface(version,
    hostAddress, community);
```
newVars = comInterface.retrieveMIBTable(itemID);
pair = (SNMPSequence)(newVars.getSNMPObjAt(0));
snmpOID = (SNMPObjIdentifier)pair.getSNMPObjAt(0);

SNMPObj itemValue;
valueTypeString = "snmp." + valueTypeString;
Class valueClass = Class.forName(valueTypeString);
itemValue = (SNMPObj)valueClass.newInstance();
itemValue.setValue(valueString);

newVars = comInterface.setMIBEntry(snmpOID.toString(),
                                            itemValue);

..  

Thus, the Client programs support the interface and information retrieval from
NMSServer. Among server programs *NMSClient* alone interacts with client programs
and rest of the server programs. It acts as a liaison between client and server programs.
The server programs *SysInfo, EhterInfo, PortScan and MibTreeInfo* interact with SNMP
agent of the SNMP-enabled device and retrieve information located in MIBTree and send
it to *NMSServer*. *NMSServer* in turn passes the information to the client.
5. TESTING AND EVALUATION

The Network Management Tool involved both software testing as well as Usability Testing. The Website containing the Network Management Tool is tested for proper usability.

5.1 Software Testing

The tool was tested with different test cases to check if it performed all the functions as stated in the requirements. The tool was tested while it was developed. Initially a client program and a server program were developed. Later the client that supported tabbed interface with other features and a server that would retrieve information form SNMP enabled device was developed. Initially small test messages and print statements were used to interact between client and server. At later stage the server and client programs were coded to handle complex tasks by having many specific classes and methods. At that point each class was tested with a driver program. Several distinct test cases were evolved for each unit. As all the units were developed, test cases were evolved for the entire tool. Proper documentation of the code was followed for ease of inspection and maintenance.

5.2 Usability Testing

The Web pages containing the tool was tested and evaluated after the tool is developed. After the tool was completely developed, the primary user, the Network Manager of CoB, tested the tool. The Web pages were tested on two platforms - IBM compatible PC and Macintosh machine. The Web pages were also tested using different kinds of browsers like Netscape Navigator, Internet Explorer and Mozilla.
Prototypical users acting as network managers consisting of eight people were asked to evaluate usability of the tool. A questionnaire was designed to get the feedback and evaluators were asked to explore and test the tool. A copy of the questionnaire is attached to the Appendix A of this document. The questionnaire consisted of questions relating to following usability measures:

a. Ease of finding information
b. Legibility or Ease of reading the content.
c. Completeness with which site’s subject is treated
d. Appearance of the site
e. Consistency in Layout, Menus, Visual Cues
f. Help on how to use site

The feedback thus obtained was used to fix some of the glitches and to improve the usability of the tool. The questionnaire was scored using Likert scale. The results of the evaluator’s comments were used to improve the Usability of the Web site.

Dr. John Fernandez, a usability expert, performed a heuristic evaluation, against the general guidelines [Rosson 2000] listed below.

a. Use of simple and natural dialog
b. Speak the user’s language
c. Minimize the memory load
d. Be consistent
e. Provide feedback
f. Provide clearly marked exits
g. Provide short cuts
h. Provide good error messages

i. Prevent Errors

j. Include good help and documentation

Dr. Fernandez suggested few changes to the user interface which were adopted to improve the usability of the interface.
6. RESULTS AND CONCLUSIONS

6.1 Results

The primary objective of building a Web-based Network Management Tool is achieved. The tool has been tested and found satisfactory by College of Business network manager, Mr. James Davis.

The design of the tool has been changed to client-server architecture to keep up with the most important requirements ‘Web-based’ feature and ‘security’. The tool has all the features that were initially proposed, except for trace-route. Trace-route was not implemented for two reasons.

1. Trace route is an icmp-based tool and there are no Java APIs available to implement icmp.

2. Trace route traces the route of the packets that are sent from a host to the destination IP Address. With the adoption of client-server architecture for the Network Management Tool, the client program, which is the applet, cannot create packets and send to the user-desired destination IPAddress. Only server program that resides on a remote machine can do that. But the route will be traced from server machine but not from user/client machine, which does not server the purpose. Hence, trace-route was not implemented.

The tool has other nice features like it can accommodate other private MIBs. A separate tab can be added to the existing interface, private MIBs can be downloaded and made into a Java file with a two dimensional array that holds the information. This file can be used to retrieve and display the information.
6.2 Conclusions

The project has been taken up to address the problem of having a comprehensive Web-based Network Management Tool using SNMP for College of Business (CoB) at TAMU-CC. The requirements of network manager at CoB have been procured and the software design was developed based on those requirements and also considering data security. The programming language Java was chosen as it suits to the complex programming needs and to keep up with the most important requirement – to have a Web-based tool. All the features initially proposed were implemented and the tool has been successfully tested for usability. The feedback obtained from usability evaluation was implemented. The network manager of CoB is satisfied with the tool. There is lot of scope for future work; most importantly the tool could be further extended to use/include private MIBs.
7. FUTURE WORK

There is a lot of scope for future work with this project. Small, medium and large
projects could be evolved. Some of the ideas are listed here under:

1. Upgrade Dr. Jonathan Sevy’s Java SNMP API

   The main reason to upgrade the SNMP API is to explore all the feature of SNMP
   in network management and monitoring. The existing API works for SNMP version 1
   and version 2, but does not actually implement all the features of SNMP version 2. The
   existing API can be re-coded and upgraded to accommodate SNMP version 3, which is
   more sophisticated.

2. Feature to browse private MIBs or hardware specific MIBs like Cisco MIB, Hewlett-
   Packard MIB etc can be included. Private MIBs can be downloaded and made into a dat
   file and be included along with the standard MIB-II, thus, increasing the capabilities of
   the tool.

4. Similar tool could be developed using a different programming language.

5. A feature to draw a graph on a separate panel for the selected parameters on MIBTree
   can be developed.

6. More ideas and current information about SNMP can be obtained from URLs
   www.simpleweb.org and www.snmp.com
I would like to render my sincere acknowledgements to the Major Advisor, of my graduate project Dr. Dulal Chandra Kar, for his constant support, valuable insights and encouraging words through two semesters, from the proposal stage to the completion of the project. I also thank the members of my graduate project committee, Dr. Fernandez and Dr. Dannelly for their support, valuable insights at various stages of the project, feedback about the design and usability of the tool and quick e-mail responses while I was 150 miles away from Corpus Christi.

I sincerely thank and acknowledge the network manager at College of Business, Mr. James Davis, for encouraging my thought of developing Network Management Tool using SNMP, for being very co-operative by enabling SNMP service on one of his servers and providing access to the server for my use, all through the development and testing stages of the tool. I greatly appreciate his valuable feedback about the tool and improving its usability.

I thank my colleagues and juniors for participating in the usability testing of the tool. I appreciate them for sparing their time and providing their valuable feedback..

I deeply appreciate my husband Mr. Maruthi Dantu’s enduring support, patience and encouragement all through the intensive days of my project.

Above all, I thank the Almighty for giving me the strength and will to finish the undertaken task, successfully.
BIBLIOGRAPHY AND REFERENCES

(visited Jan. 27, 2003).

[Kunapareddy 1998] Kunapareddy, P. On The Performance management of
Heterogeneous Networks Using SNMP. Thesis. Texas A&M University, College
Station 1998.

[Lemay 1999] Lemay, Laura and Cadenhead, Rogers. Teach Yourself Java 2 Platform in
21 Days. SAMS, New Delhi, 1999.


(visited on Feb. 03 2003).

Protocol (SNMP) v2 Available at www.faqs.org/rfcs/rfc1907.html


[Rosson 2002] Rosson, Mary Beth and Carroll, John M, Usability Engineering. Morgan
[Sevy 2000] Sevy, J. Package SNMP. Available from

http://edge.mcs.drexel.edu/GICL/people/sevy/snmp/docs/index.html


http://Java.sun.com/docs/books/tutorial/applet/practical/security.html

APPENDIX A

Questionnaire for Testing Usability of the Web site Containing Network Management Tool

This is a survey to improve the usability of Web pages containing Network Management Tool. Please respond to the following questions by circling the opinion that best corresponds to your own.

1. At the first glance of the Website is it evident what information can be obtained from the Web site?

   Strongly Disagree  Disagree  Neutral  Agree   StronglyAgree

2. Is the information presented in the Web site legible and easy to read?

   Strongly Disagree  Disagree  Neutral  Agree   StronglyAgree

3. Does the key words on tabs (on applet) make sense and represent the information contained/displayed under a particular tab.

   Strongly Disagree  Disagree  Neutral  Agree   StronglyAgree

4. Is there consistency in layout of the applet?

   Strongly Disagree  Disagree  Neutral  Agree   StronglyAgree

5. How do you rate the appearance (look & feel, colors used) of the site?

   Strongly Disagree  Disagree  Neutral  Agree   StronglyAgree

7. Are the visual clues helpful?

   Strongly Disagree  Disagree  Neutral  Agree   StronglyAgree

8. Is the ‘help’ feature provided informative?

   Strongly Disagree  Disagree  Neutral  Agree   StronglyAgree

Please use the space below and at the back of the questionnaire to provide your input/comments that would help to improve the Web site/applet.
APPENDIX B

Digital Media of the Project