

University of Montana

ScholarWorks at University of Montana

Graduate Student Theses, Dissertations, &
Professional Papers

Graduate School

1995

Design of a graphic user interface for a network management protocol

Xiaoan Hou

The University of Montana

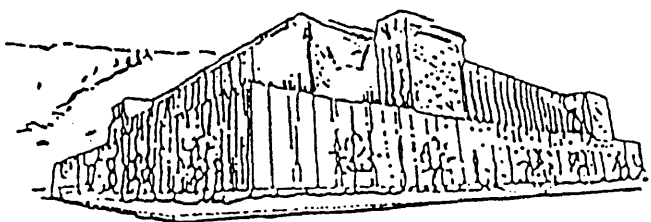
Follow this and additional works at: <https://scholarworks.umt.edu/etd>

Let us know how access to this document benefits you.

Recommended Citation

Hou, Xiaoan, "Design of a graphic user interface for a network management protocol" (1995). *Graduate Student Theses, Dissertations, & Professional Papers*. 5506.
<https://scholarworks.umt.edu/etd/5506>

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.



Maureen and Mike
MANSFIELD LIBRARY

The University of **MONTANA**

Permission is granted by the author to reproduce this material in its entirety,
provided that this material is used for scholarly purposes and is properly cited in
published works and reports.

*** Please check "Yes" or "No" and provide signature ***

Yes, I grant permission

X

No, I do not grant permission

Author's Signature

[Handwritten Signature]

Date

Dec 6, 95

Any copying for commercial purposes or financial gain may be undertaken only with
the author's explicit consent.

The Design of a Graphic User Interface for a Network Management Protocol

by

Xiaoan Hou

B.S. Beijing University of Iron and Steel Technology, 1983

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

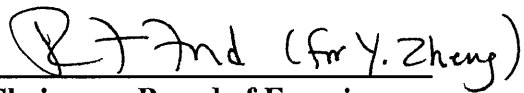
with a

major in Computer Science

University of Montana

December, 1995

Approved by


Chairman, Board of Examiners


Dean, Graduate School

DECEMBER 8, 1995
Date

UMI Number: EP40970

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI EP40970

Published by ProQuest LLC (2014). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Hou, Xiaolan, M.S., December 1995

Computer Science

The Design of Graphic User Interface for a Network Management Protocol (72 pp.)

Director: Youlu Zheng

QJ for YZ

The Simple Network Management Protocol (SNMP) was invented in the late 1980s to manage the TCP/IP based internet. The protocol has since become a de facto standard for network management. Network management is never simple--nor are the protocols that are used to implement it. An educational network management software, named SNMPview, has been developed using Xt/Motif user interface toolkit to provide vivid and playful graphic user interface to the command line interface of Massachusetts Institute of Technology's SNMP software package, *The SNMP Development Kit*. It displays the hosts in the local network using both a map and a sorted list. Inquiry of management information of a host can be done by simply clicking an MIB variable in the MIB-2 variable list. In the MIB mode, MIB variables are displayed in original tree hierarchy using a pushbutton to represent each node in the tree. The software can function both as a network manager and as an MIB browser. SNMPview is a good educational tool for those who are interested in exploring the current and novel techniques of network management.

ACKNOWLEDGMENTS

The author would like to express his deepest gratitude to his advisors, Dr. Youlu Zheng and Dr. Ray Ford, whose advice, constructive criticism and sincere interest made it possible to develop and complete this work.

The contributions of the committee members, Professor Alden Wright and Professor Darshan S. Kang are very much appreciated.

Special thanks are due to many other individuals, who helped in completing the educational process relating to this work.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES.....	vi
LIST OF TABLES	vii
CHAPTER 1 INTRODUCTION	1
GUI and X-Window System	1
SNMP.....	2
SNMP Model.....	3
Difference between SNMP and OSI CMIP	4
Structure of Management Information	5
CHAPTER 2 EVOLUTION OF NETWORK MANAGEMENT PROTOCOLS	9
History of SNMP	9
Simple Gateway Monitoring Protocol.....	12
Simple Network Management Protocol	12
Simple Network Management Protocol Security.....	15
Simple Network Management Protocol version 2.....	16
Major Enhancements of SNMPv2 over SNMPv1	18
CHAPTER 3 NETWORK MANAGEMENT SOFTWARE REVIEW	20
1. Cabletron SPECTRUM.....	21
2. Hewlett-Packard OpenView	22
3. IBM NetView/6000	24
4. WatchTower	25

5. NetWorks.....	27
6. LANsurveyor.....	28
7. MultiGate Manager.....	30
CHAPTER 4 DESIGN OVERVIEW OF <i>SNMPVIEW</i>.....	32
Syntax of MIT SNMP commands.....	32
Network Map and Host Selection.....	35
Toggle Button Implementation of SNMP commands.....	37
Procedures for Retrieving Management Information	38
Network Mode and MIB mode.....	39
Network File Management.....	41
Miscellaneous.....	43
CHAPTER 5 IMPLEMENTATION OF <i>SNMPVIEW</i>	45
Widget Hierarchy of SNMPview GUI	45
Host Search Algorithm	47
Pipe redirection of the snmpd response	47
Host selecting and testing scheme	48
CHAPTER 6 COMPARISON AND SUMMARY	49
REFERENCE.....	52
APPENDIX A Example C Code	54
main.c	55
get_host_list.c	64
get_snmp_response.c	66
NodeSelectCallback.c	68
BrowseCallback.c	70

LIST OF FIGURES

Figure 1	SNMP Model.....	3
Figure 2	MIB OID Tree for SNMP.....	7
Figure 3	Evolution of Management Protocols	11
Figure 4	SNMPview Graphical User Interface.....	36
Figure 5	Toggle Button Implementation of SNMP Commands.....	37
Figure 6	Prompt Dialog Widget Displays the Management Information	40
Figure 7	MIB OID Tree	42
Figure 8	Color Palette for SNMPview GUI Background.....	44
Figure 9	Widget Tree Hierarchy for SNMPview Graphical Interface	46

LIST OF TABLES

<u>Table 1 The Four Commands of SNMPv1</u>	4
<u>Table 2 MIT SNMP Commands and Their Syntax.....</u>	33

Chapter 1

INTRODUCTION

GUI and X-Window System

The graphical user interface (GUI) is one of the most revolutionary changes to occur in the evolution of modern computing systems. In the space of less than 15 years the expectation of what the interaction between human and computer should be like has changed from a terse, character-oriented exchange modeled on the teletypewriter to the familiar Windows, Icons, Menus and Pointing device (WIMP) interface [10]. This revolution has increased the accessibility and usability of computer systems to the general public. The X Window System (X for short) is widely recognized as the industry standard for network-based window systems [19]. X provides a powerful platform that allows programmers to develop sophisticated graphic user interfaces portable to any system that supports the X protocol [1]. Motif is a high-level user interface toolkit that makes it easier to write applications that use the X Window System. Programs based on Motif use three main libraries. The first is Xlib, which provides the lowest level interface to the facilities of the window system. The second is the Xt Intrinsics, which is a library that hides many details of Xlib and supports a higher-level programming model. Finally the Motif library builds on the Xt Intrinsics layer and provides visual components like buttons and scrollbars from which an application's user interface can be built. The interface components supported by Xt are called widgets. A widget is a complex data structure that combines an X window with a set of functions which manipulate that window [12].

A network management software, named *SNMPview*, was newly developed using Motif/Xt. *SNMPview* possesses a high portability to the large body of existing different UNIX platforms.

SNMP

In the IP world, simple network management protocol is synonymous with network management [16]. The word *simple* can be deceptive. The *simple* in simple network management protocol (SNMP), for example, might lead you to think it is a protocol, or a set of rules, for simple network management. In fact, network management is never simple nor are the protocols that are used to implement it. The oxymoronic SNMP is “simple” only in comparison to the OSI (Open Systems Interconnection) management model. In fact, SNMP was originally designed and implemented as an interim specification for communicating with network devices while the OSI specification was being finalized and being implemented during the late 80s. SNMP was supposed to fade away once OSI came on-line. But things have not worked out that way. By 1993, when OSI finally matured, SNMP had a three years head start and had already been implemented in hundreds of products. SNMP is now the de facto standard in network management.

SNMP Model

The SNMP model for managing networks is based on three pieces of software: *agents*, *MIBs* (management information bases), and *management stations* (or called *manager*). Figure 1 shows the schematic representation of SNMP model. *Agents* are pieces of software that run at each network device. They fetch information from a database called *management information base* which is also stored at the device. *Management stations* (*manager*) let you retrieve and display information gathered from a device's agent and MIB. *Management station* can also control (or *Set* in SNMP terms) those devices.

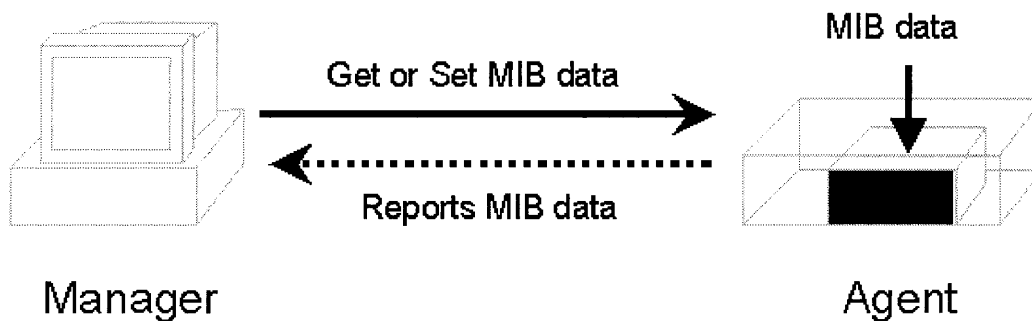


Figure 1 SNMP Model

SNMP takes an interesting alternative approach to network management [5]. Instead of defining a large set of commands, SNMP casts all options in a fetch-store paradigm. Conceptually, SNMP contains only two commands that allow a manager to fetch a value from a data item or store a value into a data item. All other operations are defined as side effects of these two operations. Table 1 shows the four operations offered by SNMP Version 1.

Table 1 The Four Commands of SNMPv1

Command	Meaning
get	Fetch a value from a specific variable
getnext	Fetch a value without knowing its exact name
set	Store a value in a specific variable
trap	Send notifications to the management station

Difference between SNMP and OSI CMIP

CMIP (Common Management Information Protocol) was developed by OSI, a part of the international standardization organization ISO that is involved with the standardization of the interconnection of open systems¹. This protocol is part of a protocol suite that aims to optimally offer the functionality of open system. CMIP includes the following operations:

<i>get</i>	retrieves specified information
<i>set</i>	changes the value of specified information
<i>action</i>	performs an imperative command, such as reset an interface;
<i>create</i>	forms a new instance of a managed objects;
<i>delete</i>	removes a specified object instance;
<i>event-report</i>	signals to a manager that an event of importance has occurred.

SNMP was developed by the joined effort of a couple of independent individuals within the Internet Society. The Internet Society is involved in the research and support of

¹ The term *open systems* refers to those (computer-)systems for which the application activities, or applications, are accessible (open) to applications of other systems.

networks based on a protocol suite of which the core is formed by TCP, UDP and IP.

SNMP includes the following operations:

<i>get</i>	same function as OSI get;
<i>getnext</i>	used for table row retrieval and for discovery of managed objects;
<i>set</i>	same function as OSI set;
<i>trap</i>	same function as OSI event-report;

The OSI *action*, *create*, and *delete* operations have no direct mapping to SNMP operations. However, the functionality of these operations can be implemented with SNMP get and set operations and proper design of SNMP MIB variables. It is simpler to implement SNMP than CMIP in hardware, which is the main reason why the SNMP is so popular today.

Structure of Management Information

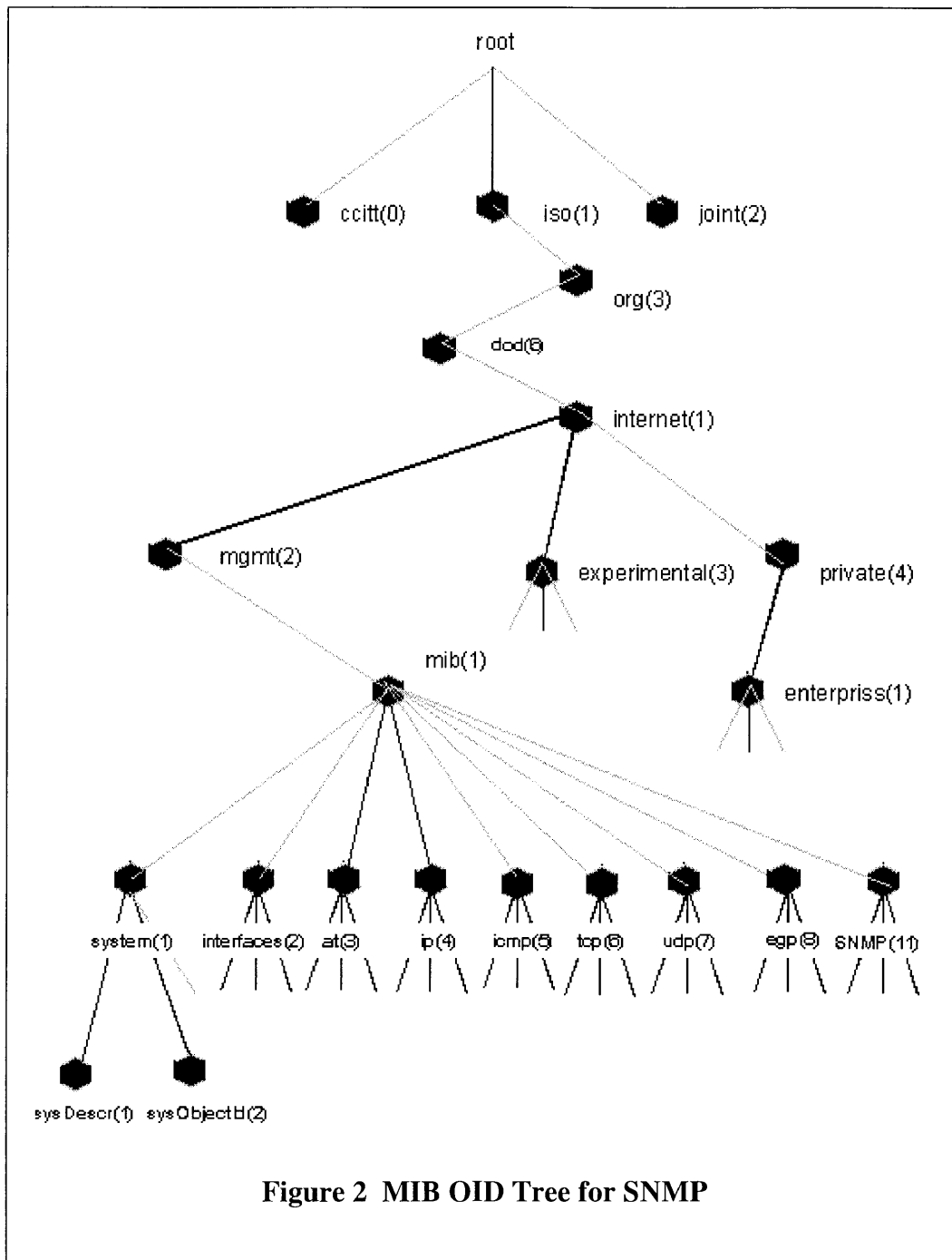
An MIB describes information that can be obtained and/or modified via a network management protocol [13]. This information enables systems on a network to be managed. Each entry in an MIB is called an MIB variable. For example, one common MIB variable is *sysDescr*, which describes system hardware and software. If you use a management station to retrieve the *sysDescr* variable from a Macintosh computer, you will get an answer like “Macintosh Quadra 800, System 7.1.”

In addition to the MIB standard, which specifies specific network management variables and their meanings, a separate standard specifies a set of rules used to define and identify

MIB variables. The rules are known as *the Structure of Management Information* (SMI). It defines the model of managed objects and the operations that can be performed on the objects, as well as data types that are permitted for the objects. Objects are unambiguously identified (or named) in SNMP by assigning them an object identifier (OID). Globally unique for all space and time, OIDs are a sequence of nonnegative integers organized hierarchically. For ease of use, a textual name is associated with each sequence element, or component, of an OID. The last component name is used by itself as a shorthand way of naming an object. All textual names of objects defined by IETF working groups are, by convention, made unique by using a different prefix for objects in each new MIB. SNMP uses an encoded form of the numeric value, not the textual name.

The root of the object identifier hierarchy is unnamed, but has three direct descendants managed by : ISO, CCITT, and jointly by ISO and CCITT. The descendants are assigned both short text strings and integers to identify them. The text strings are used when humans need to understand object names while computer uses the integers to form compact, encoded representation of the names. Figure 2 illustrates pertinent parts of the object identifier hierarchy and shows the positions of the node used by SNMP.

The name of an object in the hierarchy is the sequence of numeric labels on the nodes along a path from the root to the object. The sequence is written with periods separating the individual components. For example, the name 1.3.6.1 denotes the node labeled



internet. The MIB has been assigned a node under the *internet* management subtree with label *mib* and numeric value 1. Because all MIB variables fall under that node, they all

have names beginning with the prefix 1.3.6.1.2.1. For example, suppose one wanted to identify an instance of the variable *sysDescr*. The object class for *sysDescr* is:

iso	org	dod	internet	mgmt	mib	system	sysDescr
1	3	6	1	2	1	1	1

The object type would be 1.3.6.1.2.1.1.1 to which an instance sub-identifier of 0 is appended. That is, 1.3.6.1.2.1.1.1.0 identifies the one and only instance of *sysDescr*, which should be specified on a SNMP command line in order to get or set the system information about a remote host.

Chapter 2

EVOLUTION OF NETWORK MANAGEMENT PROTOCOLS

This chapter presents an account of the development of the Simple Network Management Protocol and also reviews the pertinent literatures or RFCs.

History of SNMP

Only a few years after the emergence of larger networks in the mid 1980's, it became apparent that a structured approach towards network management was needed to keep track of the vastly increasing number of management solutions offered by independent suppliers [15]. Each supplier of resources (e.g. hosts, gateways, bridges, routers, etc.) offered its own tools to manage its device(s). In an environment of interoperating devices from several vendors (a typical internet environment), this is clearly an undesired solution. This opinion was shared by some of the network gurus, both from research and industry, who gathered at a meeting in March 1987. The result of this meeting and already on-going research was the independent development of three network management protocols, namely:

- High-level Entity Management Protocol (HEMP);
- Simple Gateway Monitoring Protocol (SGMP);

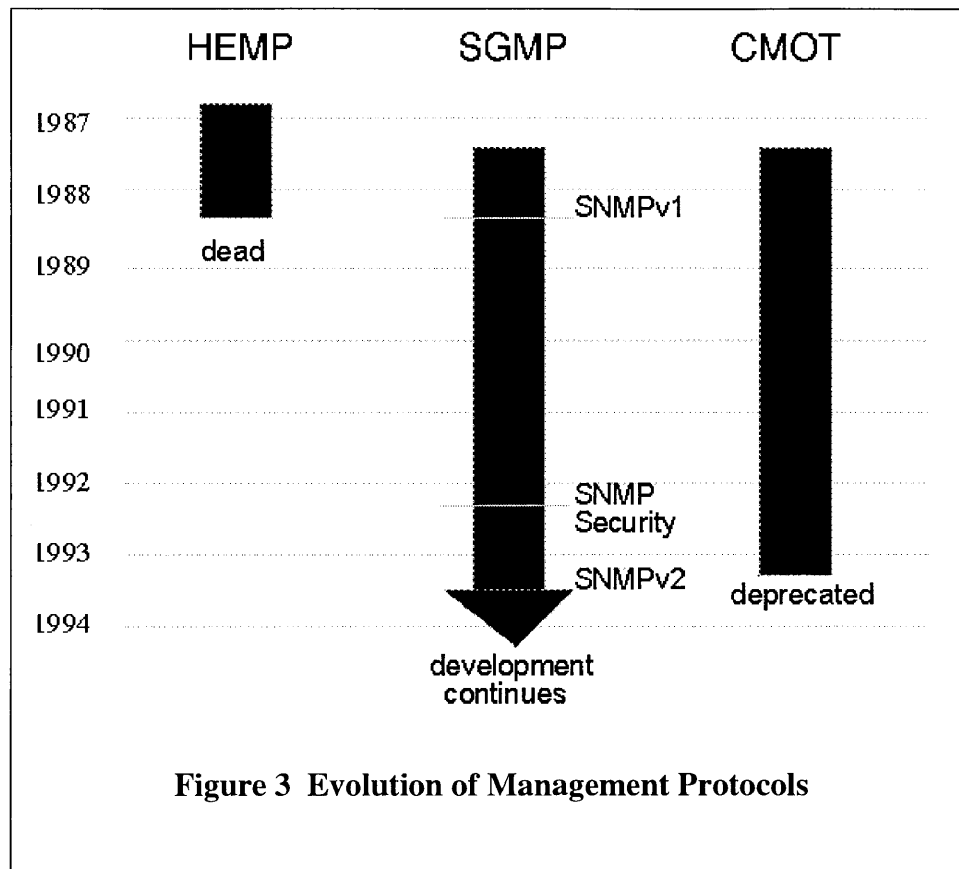
- Common Management Information Protocol over TCP (CMOT²).

Of these three protocols the first, which was developed in a project called HEMS, later died a sudden death at a meeting intended to discuss the evolution of these three protocols into one new protocol for the management of internets (see section: *SNMP*). The second and third options both proved viable, and selection between them was difficult. It was agreed that the evolution would be divided in a short-term solution and a long-term solution. This compromise was made since although SGMP was easy to implement and enjoyed a large acceptance, the functionality offered by CMIP was considered to be superior. However, at that time, CMIP enjoyed little support and needed further development.

So, for the short-term, the plan was to upgrade SGMP to reflect the experience of that time. While for the long-term, the OSI-based solution would be further developed and hopefully adopted. To make such a transition possible in the future, a common framework had to be developed. So, two working groups were formed, one to design the framework, and the other to design the successor to SGMP, which became SNMP.

But, even to this day, the Common Management Information Protocol (CMIP) still has a poor general acceptance³, so it seems fair to state that the proper evolution of SGMP is the one listed below:

² A mapping of the OSI network management framework on the Internet suite of protocols, through a light-weight presentation protocol.



- Simple Gateway Monitoring Protocol (SGMP);
- Simple Network Management Protocol (SNMP or SNMPv1);
- Simple Network Management Protocol Security (SNMP Security or Secure SNMP);
- Simple Network Management Protocol version 2 (SNMPv2).

The evolution of these management protocols took place over a relatively short period of time. Figure 3 shows the evolution in time.

³ In March 1993, the protocol standard for CMOT was deprecated to historic, with a not recommended status.

Since this report intends to cover the development towards SNMPv2, only the four protocols mentioned above will be explained in detail below.

Simple Gateway Monitoring Protocol

This was the first network management protocol designed for the management of TCP/IP-based internets. Actually, as its name indicates, the aim of this protocol was to provide the means to monitor the increasing number of gateways that make up an internet. The ideas for this protocol first surfaced in March 1987. This straightforward simple protocol was designed within two months, and within another two months it was implemented on various platforms. So by August of that same year, this protocol already enjoyed modest deployment.

Further information on this protocol can be found in the Internet standard (RFC-1028) and the review by Case, Davin, Fedor and Schoffstall [3].

Simple Network Management Protocol

In February 1988, the Internet Architecture Board (IAB) decided that the problems concerning the three coexisting protocols (HEMP, SGMP and CMOT) needed to be resolved. Therefore it created an ad hoc network management review group. At its first meeting (as mentioned in section: ***History of SNMP***), the consensus of the group was that

a new protocol had to be designed based on the SGMP, but that allowed a transition to CMIP in the future. The HEMP proposal was subsequently withdrawn at this meeting. So, to make the transition from SGMP to its successor possible, a common framework had to be developed.

The successor to SGMP was named the Simple Network Management Protocol. As the standardization process of the successor of SNMP developed, this protocol was renamed into SNMPv1.

Six months after the initial meeting of the ad hoc committee, the initial specification of the Internet-standard Network Management Framework⁴ was completed. It originally consisted of three RFCs (RFC-1065, RFC-1066 and RFC-1067), but has since then undergone various changes and extensions. The current version 1 of the Internet-standard Network Management Framework (i.e. SNMPv1) consists of the following three documents:

- RFC-1155, which defines the Structure of Management Information (SMI), the mechanisms used for describing and naming objects for the purpose of management;
- RFC-1213, which defines the Management Information Base (MIB II), the collection of objects available for the purpose of management;

⁴ Often simply referred to as: management framework.

- RFC-1157, which defines the Simple Network Management Protocol (SNMP), the protocol used for network access to the managed objects.

In addition to these documents, four other standards have been developed that could also be considered as belonging to (actually augmenting) the management framework. These are:

- RFC-1212, which defines a more concise description mechanism, which is wholly consistent with the SMI;
- RFC-1215 which defines a convention for defining traps for use with the SNMP;
- RFC-1303, which defines a convention for describing SNMP-based agents;
- RFC-1354 which defines a modified IP-forwarding-table MIB.

The framework has been, and continues to be, extended by numerous MIBs.

Since this is one of the most popular network management protocols at this moment, it enjoys a thorough and widespread documentation. Of these, the following are the most noticeable: the review by Rose [14], the review by Ben-Artzi et al [2], the review by Mazda [11] and the review by Case et al [4].

Simple Network Management Protocol Security

From the moment of its introduction on, the administrative framework of SNMP was limited to the use of a trivial authentication mechanism. This means that each SNMP message consists of a data-part and a community name. This community name is used to authenticate the sending party as being a member of a specific community. This allows various managers to belong to the same community. This community name however is placed in the clear (not encrypted) in the SNMP message. And the agent only checks the community name, not the address of the sending manager. Some agent implementation did not even (dare to) implement the set-request command because of this. Of course, this was not the correct way to cope with these deficiencies, since this resulted in non-conforming agents. The trivial security mechanism was a major bottleneck in the acceptance of the protocol. Therefore a solution needed to be developed, and the Internet Engineering Task Force (IETF) initiated a working group for this purpose.

Their goal was to develop a framework that would support various levels of security, with the option of expandability in the future. The final administrative framework consists of the following three documents:

- RFC-1351, which defines the Administrative Model, the model used to provide a unified conceptual basis for administering SNMP entities for the purpose of security;

- RFC-1352, which defines the Security Protocols that could be used in the administrative model;
- RFC-1353, which defines the Party MIB, the collection of objects describing the parties that are needed to support authentication, integrity, privacy and access control.

However, SNMP security has been more recently overtaken by other events. At the moment of its completion, a meeting of the IETF in July 1992, it was decided that the remaining deficiencies to the management framework would be resolved in a new version of the protocol, and that this successor would also include the new security extensions. Therefore the RFCs 1351, 1352 and 1353 are not to be used in actual implementations and became obsolete at the time the RFCs concerning SNMPv2 were released (April 1993).

Besides in the Internet standards, additional documentation can be found in the review by Davin, Galvin and McCloghrie [7].

Simple Network Management Protocol version 2

As the operational experience became greater, more deficiencies in the framework were encountered. So at the August 1991 IETF meeting, a special session was dedicated to the gathering of perceived deficiencies in the framework. And early 1992, a mail message was sent out inviting the submission of proposals that addressed the deficiencies in the SNMP

framework. At the July 1992 IETF meeting, besides the publishing of the security documents as RFCs, a proposal for the new framework was submitted and subsequently discussed. This proposal, called the Simple Management Protocol (SMP), ranged over 200 pages, and was accompanied by four independent, interoperable implementations. At that same meeting a deadline of September 10, 1992 was set for the submission of new proposals.

On the day of the deadline, still only one proposal had been submitted, the Simple Management Protocol. This proposal was consequently used as input for the standardization process that would eventually result in a new protocol, called the Simple Network Management Protocol version 2 (SNMPv2). This protocol is described in the following twelve documents:

- RFC-1441: Introduction to SNMPv2;
- RFC-1442: Structure of Management Information for SNMPv2;
- RFC-1443: Textual Conventions for SNMPv2;
- RFC-1444: Conformance Statements for SNMPv2;
- RFC-1445: SNMPv2 Administrative Model;
- RFC-1446: SNMPv2 Security Protocols;
- RFC-1447: SNMPv2 Party MIB;
- RFC-1448: Protocol Operations for SNMPv2;
- RFC-1449: Transport Mappings for SNMPv2;

- RFC-1450: Management Information Base for SNMPv2;
- RFC-1451: Manager to Manager MIB;
- RFC-1452: Coexistence between SNMPv1 and SNMPv2.

These RFCs were issued in April 1993 as proposed standards.

Note that all the RFCs described in the preceding sections make up the evolution path towards SNMPv2. All the ideas that they incorporate are now part of the SNMPv2.

For more information, see the review by Jander [9] and the review by Stallings [17].

Major Enhancements of SNMPv2 over SNMPv1

There are four major enhancements of SNMPv2 over SNMPv1:

I. SNMPv2 offers better security by adding request authentication.

With the SNMP v1 specification, you could not authenticate the source of a management message or prevent eavesdropping. Without authentication capability, SNMP was vulnerable to attacks that could modify or disable network configurations.

II. SNMPv2 provides better performance for multitable transfers.

The addition of a PDU (Protocol Data Unit) called GetBulk reduces the number of requests and replies and thereby improve the performance of retrieving entire MIB trees. Other PDUs were added as well.

III.SNMPv2 managers can share information by acting as agents

The issue of sharing manager information is solved by the party abstraction. A party can be either an agent or a manager. As an agent, a party is defined by its MIB. As a manager, it just reads and understands the same MIB.

IV.SNMPv2 opens up to other underlying protocols.

In the IP world, SNMP is synonymous with *network management* because the infrastructure was already in place for generating and transporting SNMP. By supporting other transports, SNMP has the opportunity to also become successful in other environments.

Simplicity is what made SNMP successful, but maturity in the form of SNMPv2, is what is designed to give it longevity.

Chapter 3

NETWORK MANAGEMENT SOFTWARE REVIEW

This chapter briefly reviews several major commercial network management systems (NMS). Following are major categories adopted in the review.

Vendor	Who developed the system or software ?
Price	What is the price of the system ?
Market Share	What is the estimated NMS market share ?
Installation	How easy is the installation procedure ?
Portability	What kind of platforms the product can run on ?
Event Notification	How are the network problem notified ?
Interface	What sort of interface is provided? Is the interface easy to use?
API	Is an Application Programming Interface available?
Dependencies Check	Does this product understand dependencies between network entities and have the ability to examine multiple network events and diagnose them as a single problem, concluding, for example, that several concurrent failures are actually the result of a problem with a single router?

Database System	What kind of database system is used ?
Autodiscovery	What ability is there to discover selected subnets, selected IP address range, etc ?
Polling	Is status and/or performance polling available ?
Network Support	What kind of network is supported (e.g., TCP/IP, AppleTalk, IPX) ?
MIB Browser	Is MIB Browser available ?

Seven commercial NMS products are evaluated: Cabletron *SPECTRUM*, HP *OpenView*, IBM *NetView/6000*,⁵³ *WatchTower*, *NetWorks*, *LANsurveyor* and *Multigate Manager*.

1. **Cabletron SPECTRUM**

Vendor	Cabletron
Price	\$7,500 for a single server and client
Market Share	7.4% in 1993
Installation	Not easy
Portability	Ultrix 4.3, AIX 3.2.2, Irix 4.0.1, SunOS 4.1 or 5.1
Event Notification	Alarm screen, sound, email, pagers
Interface	Both GUI and CUI available

⁵ On-line information about Cabletron *SPECTRUM*, HP *OpenView*, IBM *NetView/6000* is available from <http://smurfland.cit.buffalo.edu/NetMan/ProductReviews.html>

API	No
Dependencies Check	Understands dependencies
Database System	SQL interface to several DBMS (e.g., Oracle etc)
Autodiscovery	can discover selected subnets, selected IP address range etc.
Polling	Polling available
Network Support	AppleTalk, IPX, IP
MIB Browser	MIB Browser available, but awkward to use

The *SPECTRUM* is the only product which can handle dependencies between network entities. The *SPECTRUM* MIB Browser, called attribute walk, is complicated and awkward, requiring the user to specify instance IDs.

2. Hewlett-Packard OpenView

Vendor	Hewlett Packard
Price	End-user system: \$16,000, development package: additional \$13, 000
Market Share	22.5% in 1993
Installation	Not easy
Portability	HP and Sun workstations

Event Notification	?
Interface	GUI available
API	Yes, many third-party applications available
Dependencies Check	Doesn't understand dependencies
Database System	Commercial relational database system
Autodiscovery	?
Polling	Performance polling and status polling
Network Support	?
MIB Browser	MIB Browser available, but barely adequate

OpenView has gained considerable attention in the past year, and in terms of market share, has surpassed the previous market leader, *SunNet Manager*. There are now more third-party applications available for *OpenView* than any other NMS. IBM has licensed *OpenView* 3.1 source code from HP and used it as the basis for their *NetView/6000* product. Subsequently, DEC has licensed the *NetView/6000* source from IBM to make the DEC PolyCenter product. *OpenView* also makes up a component of OSF DME. In conclusion, *OpenView* and its derivatives appear to be the current industry leading approach to NMS.

The major shortcoming of HP *OpenView* is that it does not understand dependencies and can not perform any dependency heuristics before coloring a node or link red.

3. *IBM NetView/6000*

Vendor	IBM
Price	list price: \$15,000,
Market Share	?
Installation	Easy
Portability	IBM RS/6000
Event Notification	?
Interface	GUI available
API	Yes, many third-party applications available
Dependencies Check	Doesn't understand dependencies
Database System	Commercial relational database system
Autodiscovery	?
Polling	Performance polling and status polling
Network Support	?
MIB Browser	MIB Browser available, but barely adequate

IBM *NetView/6000* is a fairly new, comprehensive Network Management System (NMS).

It can be used by end users as an off-the-shelf, plug-and-play NMS system as well as a development platform for new network management applications. IBM licensed HP

OpenView 3.1 to use as the original base for *NetView/6000*. Subsequently, IBM has extended it considerably and integrated it with other software to create the *NetView/6000* product family. More recently, IBM has licensed *NetView/6000* to DEC.

IBM has improved substantially the installation procedures inherited from *OpenView*, making installation the easiest among the NMS products. The major shortcoming of *NetView/6000* is also that it can not understand dependencies.

4. WatchTower

Vendor	Intercon System Corporation
Price	Version 1.04 for \$2495
Market Share	?
Installation	?
Portability	Macintosh
Event Notification	?
Interface	GUI available
API	?
Dependencies Check	?
Database System	?

Autodiscovery	Yes, draw a map of network
Polling	No
Network Support	TCP/IP
MIB Browser	No

Intercon Systems Corporation's *WatchTower* is a bare-bones SNMP management station for the Macintosh. WatchTower supports SNMP only over TCP/IP.

WatchTower begins the way a good SNMP management station should, by letting the network manager draw a simple map of the network. But the program doesn't do much more than that. Network monitoring is severely restricted. No automatic device polling is available; if you want to see whether a device is up and running, you have to double-click on the device icon each time.

WatchTower does construct some graphs that show trends for TCP/IP nodes. But you can track only five variables(TCP and UDP input and output rates, as well as IP input rates).

WatchTower can also construct real time bar graphs showing some traffic, but the choices are extremely limited and not all that useful. Configuration is even more restricted. To configure a device using WatchTower, you must know the exact SNMP variable to change and its legal values.

5. *NetWorks*

Vendor	Caravelle Network Corporation
Price	Version 3.0.2 for \$995
Market Share	?
Installation	?
Portability	?
Event Notification	Dialog box, speak a message, pager, email
Interface	Device list and a list of notifications
API	?
Dependencies Check	?
Database System	?
Autodiscovery	No
Polling	Yes
Network Support	TCP/IP, IPX, DECnet, AppleTalk
MIB Browser	No

Caravelle Network Corporation's *NetWorks* knows about much more than just SNMP.

Originally designed to watch over AppleTalk network, the version 3.0 can monitor a network of computers and devices that talk AppleTalk, Novell NetWare's IPX, Digital Equipment Corporation's DECnet and TCP/IP. *NetWorks*, however, is only a monitoring

tool; it doesn't support any network configuration (or Set) functions.

The *NetWorks* approach to monitoring is based on a device list and a list of notifications.

NetWorks checks the devices on the list as often as you specify. If there is a problem,

NetWorks activates a notification from another list.

NetWorks can notify you of problems in a variety of ways, including displaying a dialog box on the Macintosh screen; using MacinTalk to speak a message; playing a recorded message or sound; sending a message to a pager; sending mail using QuickMail, Microsoft Mail, Simple Mail Transfer Protocol; etc.

6. LANsurveyor

Vendor	Neon Software
Price	Version 1.01 unlimited zones \$695
Market Share	?
Installation	?
Portability	?
Event Notification	Log file, dialog box, playing sound, pager, email
Interface	GUI
API	?
Dependencies Check	?

Database System	?
Autodiscovery	Yes
Polling	?
Network Support	AppleTalk
MIB Browser	No

Neon Software's *LANsurveyor* is an AppleTalk mapping and monitoring tool that can use SNMP over AppleTalk. *LANsurveyor*'s forte is map drawing. Set it loose on an AppleTalk network and *LANsurveyor* will find all of the routers, pick appropriate icons, and try to lay out into a logical map of the network. Masochists can ask *LANsurveyor* to find every AppleTalk node, not just routers.

Once you've built a map with *LANsurveyor*, getting information out of SNMP MIB is easy. Double click on a device icon and up comes a window that lets you look at that device's AppleTalk information, SNMP MIB information, and notes and comments that you may have added to the map.

LANsurveyor has fewer monitoring and notification options than *NetWorks*. *LANsurveyor* will watch a list of network devices and send notifications if a device becomes unavailable or if AppleTalk traffic error rates go above a threshold you define. You can send notifications in a variety of ways, including writing to a log file, displaying a dialog box, and making other visual changes to the map; playing sounds; sending pages; or sending

mail via QuickMail.

7. *MultiGate Manager*

Vendor	Network Resources Corporation
Price	Version 5.1 for \$2995
Market Share	?
Installation	?
Portability	Mac
Event Notification	Pager
Interface	GUI
API	?
Dependencies Check	?
Database System	?
Autodiscovery	No
Polling	Yes, very flexible
Network Support	TCP/IP
MIB Browser	No

The most bug-ridden of the products, Network Resources Corporation's *MultiGate Manager* still has the potential to be a powerful network monitor. Its good use of color

and graphics brings many of the features of larger SNMP management stations to the Mac.

The software does not draw network maps, but it lists TCP/IP network devices in tabular format, along with device interface status, network addresses, any device traps, and an alarm status field. It does not support SNMP over AppleTalk.

MultiGate Manager keeps an eye on your network in two ways. Devices in the network manager window are polled at whatever rate you specify. If a device becomes unavailable, you'll see that on the display. The display also shows the status of hubs and routers by displaying a series of dots by each device, one per interface. If the interface is up, the dot is green; if down, the dot is red. Trend/threshold windows let you continuously monitor the network through strip charts showing SNMP variables from one or more devices.

Chapter 4

DESIGN OVERVIEW OF *SNMPVIEW*

New SNMP support software, called SNMPview, has been developed to provide an aid for network managers or those who take interest in rapidly understanding current and novel techniques for network management. SNMPview provides a graphic user interface to the command line interface of *the MIT SNMP Development Kit*. [8]

Syntax of MIT SNMP commands

The SNMP development Kit is produced in the Advanced Network Architecture group at the MIT Laboratory for Computer Science. *The SNMP Development Kit* is a body of software designed to make the fabrication of network management applications as easy as possible. The foremost goal of *the SNMP Development Kit* is to foster serious as well as playful exploration of current and novel techniques for network management. This kit is intended as an aid for those who seek practical experience in both the use of network management information and the engineering issues involved in its acquisition and manipulation. The SNMP Development Kit offers four basic SNMP commands with syntax as shown in Table 2. The meaning of the options for SNMP commands in Table 2 are briefly described as follows:

If the `-h` flag is present, then the program will send its management information request to the IP address specified as *remoteHost*.

If the `-p` flag is present, then the program will send its management request to the UDP port specified as *Port*, instead of that assigned to the “snmp” service in the `/etc/service` database.

Table 2 MIT SNMP Commands and Their Syntax

```
snmpget [-h remoteHost] [-p Port] [-c community] [-t timeout] [-I requestID]
        [name] ...
snmpnext [-h remoteHost] [-p Port] [-c community] [-t timeout] [-I requestID]
        [name] ...
snmpset [-h remoteHost] [-p Port] [-c community] [-t timeout] [-I requestID]
        [name kind value] ...
snmptrap [-h remoteHost] [-p Port] [-c community] enterprise agent-address
        generic-trap specific-trap timestamp [name kind value] ...
```

If the `-c` flag is present, then the program will generate and accept management requests associated with the community name specified as *communityName*, instead of using the community name “public”.

If the `-I` flags present, then the program will identify its management request by the number specified as *requestID*, instead of using the value zero.

If the `-t` flag is present, then the program will terminate after a number of seconds specified as *timeout*, instead of waiting forever for a response from the remote agent.

The *snmpget* and *snmpnext* are the most often used commands by a network manager software. The *snmpget* command initiates a network management query to a remote management agent and later displays the response. The *snmpnext* command retrieves and displays specified subtree of the MIB. The *snmpset* command attempts to alter the items of management information to the value specified on the command line. The *snmptrap* command is for a network agent to send event notifications to the management station.

The following example shows how to use *snmpget* command:

```
snmpget -h selway 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0
```

This command has the effect of displaying the *sysDescr* and *sysObjectId* values for the host named *selway*. SNMP commands are lengthy and hard to use. The name of an MIB variable is a sequence of dot separated numbers, which can be long and hard to remember. Besides, using SNMP commands requires knowledge of RFC* 1155 (*Structure and Identification of management Information for TCP/IP-based Internets*), RFC1156 (*Management Information Base for Network Management of TCP/IP-based Internets*) and RFC1157 (*A Simple Network Management Protocol*).

* RFC stands for *Request for Comments*, which is the name of a series of notes that contains surveys, measurements, ideas, techniques, and observations, as well as proposed and accepted TCP/IP protocol standards. They are available on-line from the Network Information Center.

Network Map and Host Selection

SNMPview can perform an automated discovery of all the agents (computers, printers, etc) interconnected in a local network and then draw a map of the network in a two level tree. The root of the tree is the management station, which is the computer where the SNMPview software is installed. A SNMP command requires you to specify a remote host before you can query on the management information stored in that host.

SNMPview provides two ways to specify a remote host on the SNMP command line.

You can choose a host from either the network map or from the sorted host list. The host list is sorted by alphabetical order, which provides a fast way to identify a desired host to retrieve management information. Once a host is chosen, the status of the host is automatically tested using the *ping* command. If the host responds, the background of the host icon remains unchanged. If the chosen host is down at the time, the background of the host icon will be set to a dark color, which is an indication that the chosen host is not available for management information inquiry. In both cases, the IP address of the chosen host will be retrieved. There are two textfields on the GUI of SNMPview, one labeled *Selected Host* and the other *IP Address*. These two textfields are specially designed to display the chosen host at any time. Once a host is chosen, the *Selected Host* textfield shows the chosen host name and the *IP Address* textfield displays its IP address.

Figure 3 shows the network map and sorted list of all the hosts interconnected in our Computer Science Department subnetwork. A network node is represented by an icon, which is an XmRowColumn widget managing two widgets: an XmLabel widget to display the host name and an XmPushButton widget to display a pixmap of an IBM RISC

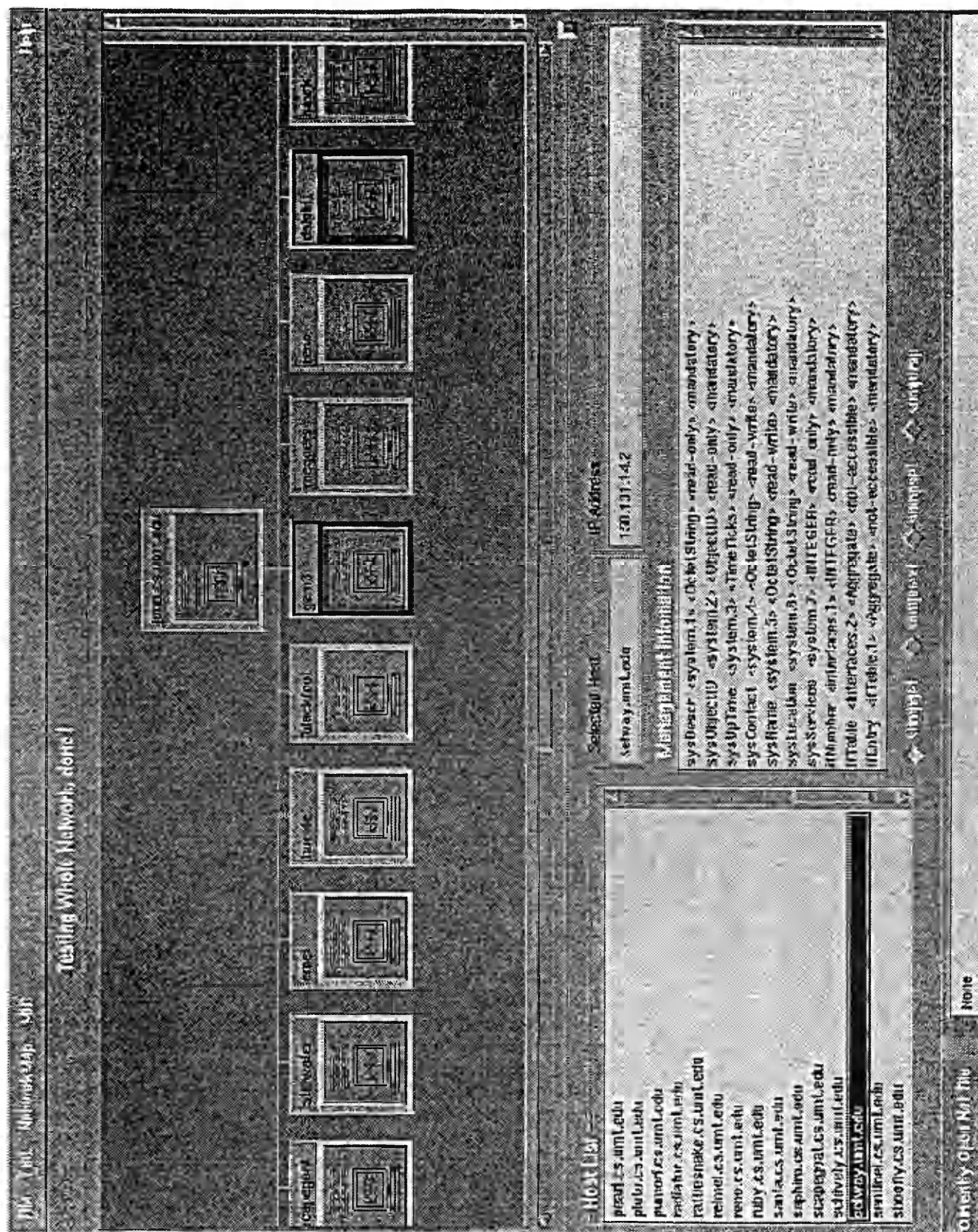


Figure 4 SNMPview Graphical User Interface

computer. The sorted host name list is displayed by an `XmlList` widget. From the figure, you can see the background of the host *gem3* and *dalphi* is dark, indicating these two hosts are down at the time of testing and are not available for management information retrieval.

Toggle Button Implementation of SNMP commands

The four basic SNMP commands are implemented as a `RadioBox`. That is, an `XmlRowColumn` widget is used to manage a set of four `XmToggleButton` widgets. A



Figure 5 Toggle Button Implementation of SNMP Commands

toggle button consists of an optional indicator and a label area. The labels for the toggle buttons are *snmpget*, *snmpnext*, *snmpset* and *snmptrap*, as shown in Figure 5. A diamond indicator means that the toggle button displays a “one-of-many” selection state. That is, only one `XmToggleButton` can be selected at one time. By default, the *snmpget* selection is set at the start of `SNMPview`. The center portion of the indicator is set to the specified color once the toggle button is selected. If a second button is selected afterwards, the first selected button will be unset. This feature of radio box provides a simple graphic way to select a SNMP command.

Procedures for Retrieving Management Information

The SNMPview GUI also includes an XmList widget displaying the MIB variable list, named *Management Information* as shown in Figure 4. To issue a SNMP command, as dictated by the syntax of SNMP commands, you need to first specify one SNMP command from the four available commands. This task is done by selecting one of the four toggle buttons as discussed in the above section. Second, you need to specify a remote host from which you are interested in retrieving management information. A remote host can be graphically chosen as discussed in the Section: *Network Map and Host Selection*. Third, you must specify the name or the OID of an MIB, which indicates what kind of information you want to retrieve. This job is accomplished by selecting an item from the *Management Information* list. Once you click an item in the *Management Information* list, the dot notation numeric name of the chosen MIB object is retrieved automatically through a search of the MIB-2 object database. Then a SNMP command is issued containing the specified host and the numeric name of the chosen MIB object. The dot notation numeric name of an MIB object is not shown in the *Management Information* list. This information is intentionally hidden from the user since the textual name of an MIB object is much more understandable and more likely to be useful when a user issues a SNMP command.

The timeout option is set to 5 seconds for every SNMP command issued. The other SNMP command options use their default value and are not necessary to be specified. If the selected remote host is not down and the remote host also has SNMP agent software

running at the time the SNMP command is issued, as shown in Figure 5, a prompt dialog widget will be popped up on top of the SNMPview GUI displaying the management information sent back by the chosen remote host. The prompt dialog widget in Figure 5 shows the general response format from a chosen remote host. The response includes an request id (0 by default), an error message showing if there is an error or not, an index indicating which variable in the command line name list caused the error, a count of variable bindings, a name to identify an instance of an MIB variable, the kind or data type of the MIB variable (OctetString) and finally the value bound to the MIB variable that is the desired management information . In this example, the remote host is specified as *selway* and the desired management information is *sysDescr* that was chosen from the *Management Information* list. The system information of *selway* is obtained as “selway.umt.edu: CPU???: ULTRIX V4.4 (Rev.69) system * 12”. Therefore, SNMPview can function like a network manager since it can monitor any management information associated with a specified remote host in the network.

Network Mode and MIB mode

If you choose the *Network Map* command from the *Network* pulldown menu or you open a network file, which stores a list of hosts, to draw a map of hosts in the underline subnetwork, the program automatically enters network mode. In network mode, the upper large scrolled window will show a map of hosts interconnected in the network; the lower left list widget will show a sorted list of hosts in the network. The lower right list widget will load the list of MIB-2 variables. You can perform the retrieval of network

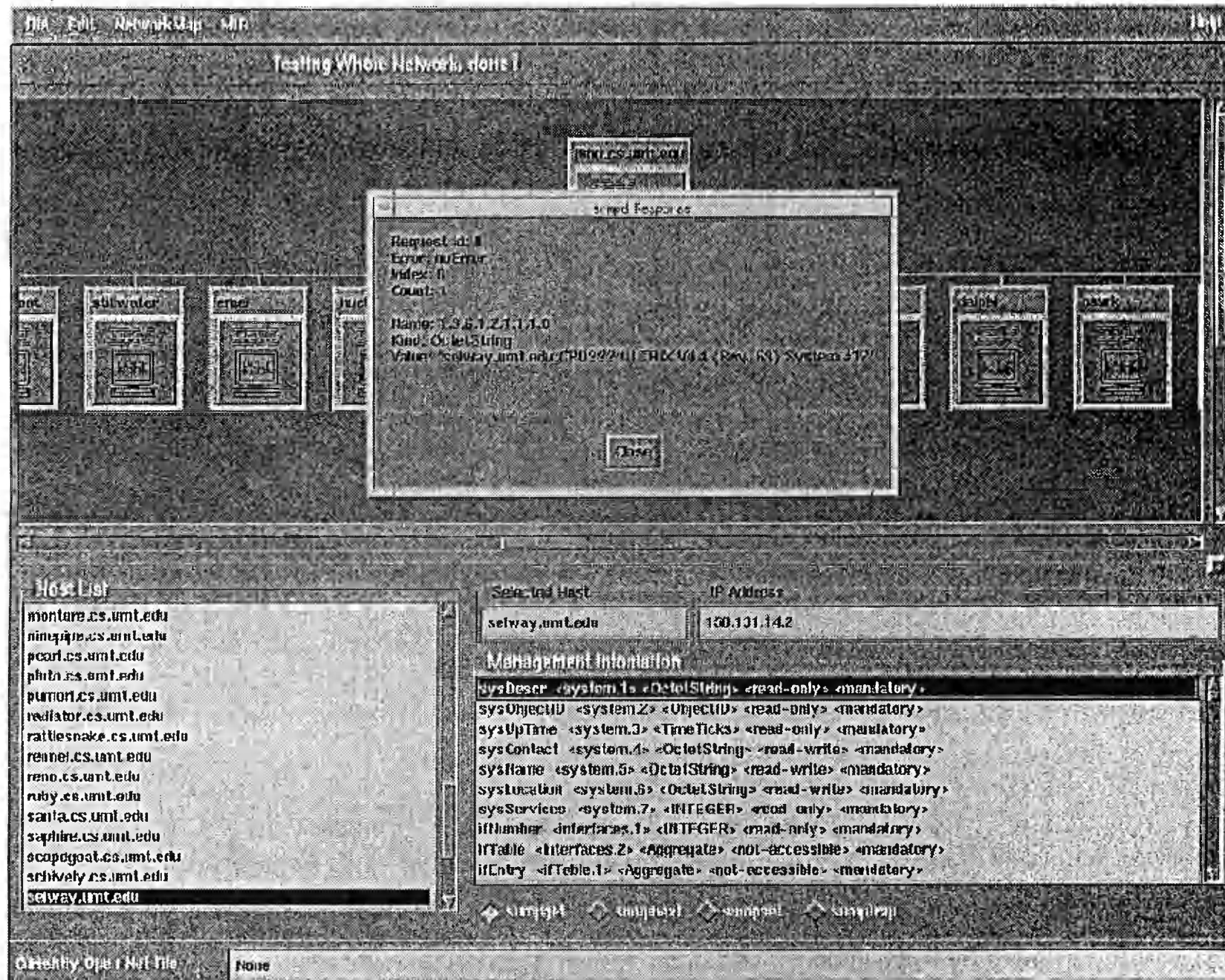


Figure 6 Prompt Dialog Widget Displays the Management Information

management information only in network mode. On the other hand, if you choose the *MIB* pulldown menu to display the MIB OID tree, the program automatically enters MIB mode. The upper large scrolled window now displays the MIB OID tree using a pushbutton labeled with MIB variable textual name representing each node in the tree. The lower left list widget will display the list of SNMP related MIB objects. The lower right list widget named *Management Information* remains unchanged. The dot notation numeric name of an MIB variable can be obtained by clicking the corresponding node in the tree or selecting the corresponding item in the *MIB OID* list. As shown in Figure 6, the textual name of the selected MIB variable is displayed in the *Selected MIB Object* textfield and the dot notation numeric name or OID in the *Object Identifier* textfield. Hence, SNMPview can also function as an MIB browser.

Network File Management

SNMPview can also assist in performing file management. It can open a network file, which stores a list of hosts, and draw a map of the hosts specified in that file. The host list is sorted first and then set into the *Host List* widget. It can also perform an automated search of all the hosts interconnected in the present local network and then draw a map of the network. The search algorithm is described in the next section. The *Delete* and *Add* commands under the *Edit* menu let you delete an unwanted host from the map, add a host into the network map, or save the modified host list as a new network file. This function provides the flexibility to remove those hosts that are always down and add those hosts that are new members of the network.

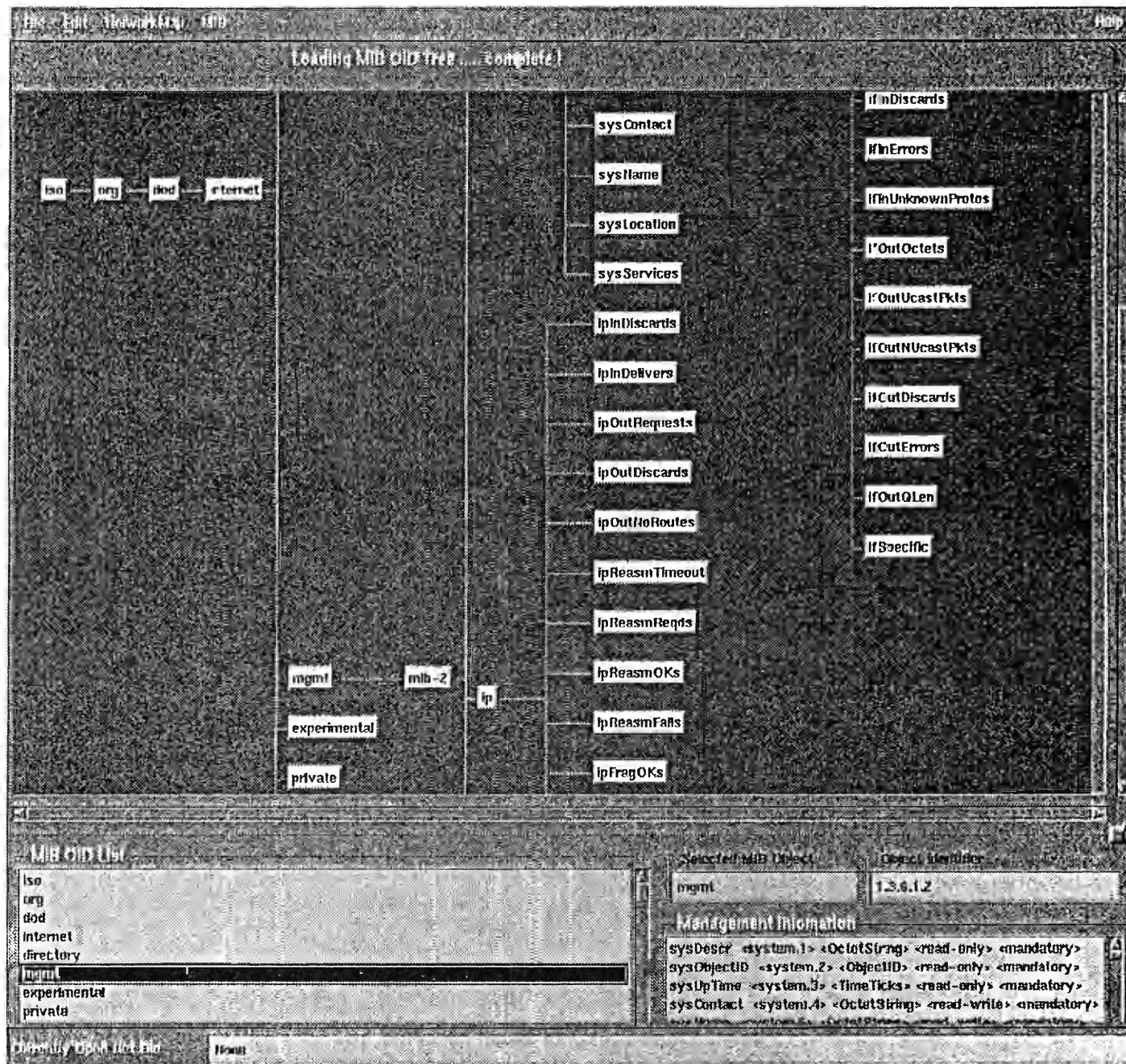


Figure 7 MIB OID Tree

Miscellaneous

Other features of SNMPview include paned windows, help facilities, and background adjustment. The XmPanedWindow widget class manages all its children in adjustable, vertical panes. Since the screen size is limited, the paned window provides a way to enlarge the network map area for a clearer view and at the same time hide the lower part host list, MIB variable list, and SNMP command toggle button from the user. The network map is displayed using XmScrolledWindow, which consists of a viewing area, a horizontal scrollbar and a vertical scroll bar. The scrolled window is especially good for displaying large maps. You can scroll to see the whole when the map is too big to fit into the computer screen area. Help facilities are available for users who do not know how to use the software. The *Help* menu in the menu bar provides help on how to use the software for a new user. On line help is always available when you save a file, add or delete a host. The background of the scrolled window, host list widget and MIB variable list can be adjusted to different color on one's own will. A popup menu is implemented to fulfill the color adjustment task. A menu is popped up when the cursor of the mouse is located in the scrolled window and the right button is clicked. The user can choose a entry to specify which background he/she wants to adjust. Then, as shown in Figure. 7, a color palette window is popped up for the user to choose a desired color. The chosen background will display the chosen color immediately afterwards.

Chapter 5

IMPLEMENTATION OF *SNMPVIEW*

Widget Hierarchy of SNMPview GUI

The `XmMainWindow` widget was used to provide a common layout of the SNMP graphic user interface. The `XmMainWindow` widget supports four distinct areas, each of which can be used to display a widget. These areas are the menu bar, the command area, the work area, and the message window. The menu bar area actually holds a `MenuBar` widget which contains *File*, *Edit*, *Network*, *MIB*, and *Help* five menus. The command area holds an `XmRowColumn` widget which contains two `XmLabel` widgets displaying the execution status of the program. The work area holds an `XmPanedWindow` widget which contains an `XmScrolledWindow` widget and an `XmRowColumn` widget. The message window holds an `XmRowColumn` widget which contains an `XmLabel` widget and an `XmTextField` widget. The detailed widget tree hierarchy is shown in Figure. 8. The C code, **main.c**, that implements this widget tree structure is given in *Appendix A*.

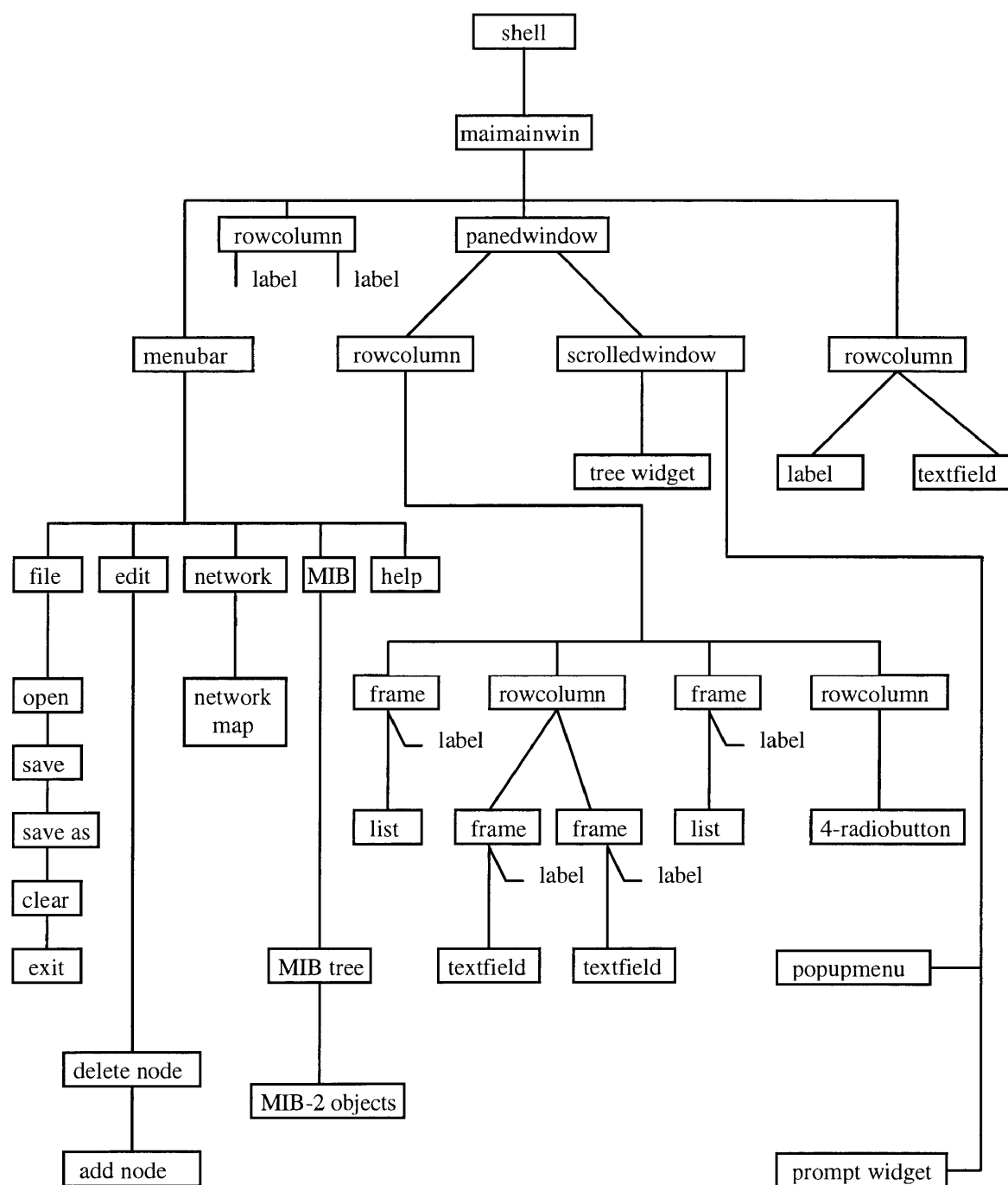


Figure 9 Widget Tree Hierarchy for SNMPview Graphical Interface

Host Search Algorithm

A host name cannot function as a network address. It must be converted to a network number. The relationships between host names and network numbers are stored in the text file “/etc/hosts.” The `gethostent()` subroutine was used to search the nodes in the network. The `gethostent()` function allows an application program to retrieve an entry from the “/etc/hosts” file. The `gethostent()` subroutine opens the “/etc/hosts” file and performs a sequential read of each line in the file starting from the beginning of the file. Each subsequent `gethostent()` subroutine call returns information for a different host. The `gethostent()` subroutine returns a pointer to a `hostent` structure, which contains the equivalent fields for a host description line in the “/etc/hosts” file. The subroutine, **`get_host_list.c`** in *Appendix A*, performs the network search task.

Pipe redirection of the snmpd response

Pipes are the oldest form of the UNIX IPC and provide a way to connect the output of one UNIX program to the input of another [6]. In *SNMPview*, pipes are used to direct the standard output of *snmpd* response to the input of a prompt widget. The code, **`get_snmp_response.c`** in *Appendix A*, creates one pipe and spawns a child process using the `fork()` system call. The child process executes a SNMP command. The parent process reads the *snmpd* daemon response from the standard output. The child process closes its standard output file descriptor and replaces it, using the `dup()` system call, with one end of the pipe [18]. The parent process reads data from the other end of the pipe. Then, the data is sent to the prompt widget and displayed in a prompt window.

Host selecting and testing scheme

Before the user can request the management information, he/she must choose a host first either from the network map or from the sorted host list. When a host is chosen, an automated test of the host status is carried out immediately using *ping* command. If the host responds, the background of the host icon remains unchanged. If the chosen host is down at the time, the background of the host icon will be set to dark color. In both cases, the *Selected Host* field and *IP address* will be set to the chosen host's. The subroutine **NodeSelectCallback.c** in *Appendix A* performs the test of a chosen host and sets the color to the background of that host. The `NodeSelectCallback()` function is invoked when the user presses and releases the mouse button of host icon while the pointer is in the `XmPushButton` widget. The next subroutine, **BrowseSelectCallback.c** in *Appendix A*, performs the same task as the previous one. The difference is that this function is invoked when a host is chosen from the sorted host list.

Chapter 6

COMPARISON AND SUMMARY

SNMPview was developed as an educational tool for those who want to rapidly learn about current network management techniques. This software possesses many advantages as an education tool.

SNMPview provides direct feel about how SNMP works. The user has a high flexibility in freely choosing what command to issue, what host to manage and what kind of management information to retrieve or set. To use this software requires little previous knowledge of SNMP. A user can play with the software using only a mouse and at the same time, gain a sense about how present networks are managed. In contrast, other available SNMP software packages are usually designed for efficient network management, not for education. They usually hide those details of SNMP command implementation from the user intentionally.

SNMP is puzzling for a learner. Extensive knowledge of MIB is a prerequisite in understanding Simple Network Management Protocol. SNMPview can function both as a network manager and as an MIB browser. Combining these two functions into one software package makes a good educational tool for understanding of SNMP. Though separate products containing either an MIB browser or a network manager are widely available, software packages that combine both an MIB browser and a network manager

into one are rare. For example, the popular software package HP OpenView , one of the major commercial network management systems, has an MIB browser. However, this package is expensive, its code is not in the public domain and thus cannot be modified by a potential user and is thus not ideal for educational use.

SNMPview provides two ways to select a host for the query of management information. The unique sorted list of hosts displayed by SNMPview enables a user to rapidly identify a desired host. The public domain SNMP software, *NetGuardian*⁶ developed by the University of Lisbon, provides a nice graphic user interface for SNMP. However, *NetGuardian* hides the implementation of SNMP commands and does not implement the *snmpset* operation. This kind of implementation restricts the software function to monitoring network. The software cannot be used to configure a network. Although *NetGuardian* draws nice map of a network to be managed, it does not provide the list of hosts in the network. It would be hard to find a desired host from the map when the number of hosts in the network becomes large.

As an educational tool, SNMPview is better than other available public domain packages, and more adaptable than commercial SNMP software. However, as a network management tool, SNMPview is insufficient. The SNMP command set is not efficiently implemented since only one piece of management information or one table of management information can be retrieved at one time. In order to make SNMPview a commercial

⁶ NetGuardian software is available in the public domain
<ftp://ftp.fc.ul.pt/pub/networking/snmp/>

product or even a more useful educational tool many network functions need to be implemented or enhanced. These include automated periodical device status polling, event (or problem) notification, hierarchical discovery of all the nodes in a wide area network, and many others.

REFERENCE

- [1] Andersen, David M., and Bruce A. Sherwood.[November, 1991], 'Portability and The GUI', *Byte*, November 1991
- [2] Ben-Artzi A., Chandna A., and Warriar U. [1990, July], 'Network Management of TCP/IP Networks: Present and Future,' *IEEE Network Magazine*, 4(4), 35-43
- [3] Case J. D., Davin J. R, Fedor M.S., and Schoffstall M. L. [1988, March], 'Introduction to the Simple Gateway Monitoring Protocol. *IEEE Network Magazine*, 4(4), 35-43.
- [4] Case J. D., Davin J. R, Fedor M.S., and Schoffstall M. L. [1989], 'Internet Management using the Simple Network Management Protocol' , *Proceedings of the 14th IEEE Conference on Local Computer Networks* (pp.156-199). Los Alamitos, CA: IEEE Computer Society Press.
- [5] Comer, Douglas E. [1991], *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, I , Englewood Cliffs: Printice Hall, Inc., pp. 403-20.
- [6] Curry, David A.[1991], *Using C on the UNIX System*, O'Reilly & Associates, Inc., Sebastopol, CA, pp. 148-51.
- [7] Davin J. R., Galvin J. M., and McCloghrie K. [1991], 'Secure Management of SNMP Networks', In I. Krishnan & W. Zimmer (Eds), *Proceedings of the IFIP TC6/WG 6.6 Second International Symposium on Integrated Network Management* (pp. 703-714). Amsterdam: North-Holland.
- [8] Davin, James R.[January, 1994], *The SNMP Development Kit Release Notes*, Laboratory of Computer Science, Massachusetts Institute of Technology, January 1994. (this references available in postscript file format from the ftp site: allspice.lcs.mit.edu)
- [9] Jander M. [1992b, November], 'SNMP2: Coming Soon to a Network Near You,' *Data Communications*, 21(16), 66-76
- [10] Mandelkern, Dave.[April, 1993], 'GUIs: The Next Generation', *Communication of the ACM*, 36(4), 37-39
- [11] Mazda F. [1991b, November], Convergence or Collision: SNMP and CMIP (Part II). *Telecommunications*, 25(9), 142-150.
- [12] McMinds, Donald L.[1993], *Mastering OSF/Motif Widget*, 2d ed.; New York: Addison-Wesley Publishing Company, pp. 1-20.

- [13] Perkins, David T. [1993], *Understanding SNMP MIBs*, Revision 1.1.7, SynOptics Communications, Inc., pp. 1-5.
- [14] Rose M. T [1991], *The Simple Book-An Introduction of TCP/IP-based internets* (3rd ed.). Englewoods Cliffs, NJ: Prentice Hall.
- [15] Schekkerman, E. J. [1993], '*An Analysis of the Simple Network Management Protocol version 2*', Masters' Thesis, 1993, Department of Computer Science, University of Twente, the Netherlands
- [16] Snyder, Joel.[August, 1994], 'SNMP revealed', *MacWorld*, 28, (8), 182-186..
- [17] Stallings W. [1993], *SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management Standards* (1st ed.), Reading, MA: Addison Wesley.
- [18] Stevens, W. Richard.[1992], *Advanced Programming in the UNIX Environment*, New York: Addison-Wesley Publishing Company, pp. 60-63
- [19] Young, Douglas A.[1994], *The X Window System Programming and Applications with Xt*, 2d ed.; Englewood Cliffs: Printice Hall, Inc., pp. 1-50.

APPENDIX A

Example C Codes

MAIN.C

```

/*
 * main() constructs the widget layout tree of SNMPviewer graphic
 * interface
 */
void main ( int argc, char ** argv )
{
    Widget tmpw;
    XtAppContext app;
    char    * parent = NULL;
    Pixel tmp, tmp1;
    Arg args(10);
    Cardinal nargs;
    int i,n;

    strcpy ( command, "snmpget" );

    /*
     * Initialize Xt and create an XmMainWindow widget to handle
     * SNMPview layout
     */
    shell = XtAppInitialize ( &app, "MainWin", NULL, 0,
                             &argc, argv, NULL, NULL, 0 );

    /*
     * Set the size of the mainwindow
     */
    nargs = 0;
    XtSetArg (args(nargs), XmNgeometry, "1050x830"); nargs++;
    XtSetValues ( shell, args, nargs );

    /*

```

```

* Create a mainwindow widget
*/
mainwindow = XtVaCreateManagedWidget ( "mainwindow",
                                         xmMainWindowWidgetClass,
                                         shell, XmNshowSeparator,
                                         TRUE, NULL );

/* Create the menu bar */
menu = CreateMenuBar ( mainwindow );
XtManageChild ( menu );

/*
* rowcolumn widget "rowcol2" to hold two labels showing the
* status information
*/
showinfo = XtVaCreateManagedWidget ( "rowcol2",
                                       xmRowColumnWidgetClass,
                                       mainwindow,
                                       XmNpacking, XmPACK_COLUMN,
                                       XmNnumColumns, 2,
                                       NULL);

status = XtCreateManagedWidget ( "Status: ",
                                  xmLabelWidgetClass,
                                  showinfo,
                                  NULL, 0 );

tmp = GetPixelByName ( status, "green" );
XtVaSetValues ( status, XmNforeground, tmp, NULL );

status_text = XtVaCreateManagedWidget ( "idle",
                                         xmLabelWidgetClass,
                                         showinfo,

```



```

rowcol1,
XmNwidth, 300,
XmNpacking, XmPACK_TIGHT,
NULL );

```

```

lframe = XtVaCreateManagedWidget ( "left", xmFrameWidgetClass,
rowcol, NULL );

```

```

llabel = XtVaCreateManagedWidget ( "Host List",
xmLabelWidgetClass,
lframe,
XmNchildType, XmFRAME_TITLE_CHILD,
NULL );

```

```

tmp = GetPixelByName ( llabel, "seashell" );
XtVaSetValues ( llabel, XmNforeground, tmp, NULL );

```

```

llist = XmCreateScrolledList ( lframe, "list", NULL, 0 );
XtManageChild ( llist );

```

```

tmp = GetPixelByName ( llist, "azure3" );
XtVaSetValues ( llist, XmNvisibleItemCount, 15,
XmNwidth, 300,
XmNbackground, tmp, NULL );

```

```

XtAddCallback ( llist, XmNbrowseSelectionCallback,
BrowseCallback, NULL );

```

```

rowcol = XtVaCreateManagedWidget ( "rowcol",
xmRowColumnWidgetClass,
rowcol1,
XmNpacking, XmPACK_TIGHT,
NULL );

```

```

row1 = XtVaCreateManagedWidget ( "row1",
                                xmRowColumnWidgetClass,
                                rowcol,
                                XmNpacking, XmPACK_COLUMN,
                                XmNnumColumns, 2,
                                NULL );

frame1 = XtCreateManagedWidget ( "frame1",
                                xmFrameWidgetClass,
                                row1, NULL, 0 );

host_name = XtVaCreateManagedWidget ( "Selected Host:",
                                xmLabelWidgetClass,
                                frame1,
                                XmNchildType, XmFRAME_TITLE_CHILD,
                                NULL);

name_label = XtVaCreateManagedWidget ( "host label",
                                xmTextFieldWidgetClass,
                                frame1,
                                XmNeditable, FALSE,
                                XmNcursorPositionVisible, FALSE,
                                NULL );

frame2 = XtCreateManagedWidget ( "frame2",
                                xmFrameWidgetClass,
                                row1, NULL, 0 );

host_addr = XtVaCreateManagedWidget ( "IP Address:",
                                xmLabelWidgetClass,
                                frame2,
                                XmNchildType, XmFRAME_TITLE_CHILD,
                                NULL);

```

```

addr_label = XtVaCreateManagedWidget ( "host label",
                                         xmTextFieldWidgetClass,
                                         frame2,
                                         XmNeditable, FALSE,
                                         XmNcursorPositionVisible, FALSE,
                                         NULL );

rframe = XtCreateManagedWidget ( "right", xmFrameWidgetClass,
                                   rowcol, NULL, 0 );

rlabel = XtVaCreateManagedWidget ( "Management Infomation",
                                     xmLabelWidgetClass,
                                     rframe,
                                     XmNchildType, XmFRAME_TITLE_CHILD,
                                     NULL );

tmp = GetPixelByName ( rlabel, "ivory" );
XtVaSetValues ( rlabel, XmNforeground, tmp, NULL );

rlist = XmCreateScrolledList ( rframe, "list", NULL, 0 );
tmp = GetPixelByName ( rlist, "LavenderBlush3" );
XtVaSetValues ( rlist, XmNvisibleItemCount, 10,
                XmNbackground, tmp, NULL );
XtManageChild ( rlist );
XtAddCallback ( rlist, XmNbrowseSelectionCallback,
                RBrowseCallback, NULL );

XtVaSetValues ( name_label, XmNbackground, tmp, NULL );

XtVaSetValues ( addr_label, XmNbackground, tmp, NULL );

/*
 * Create an XmRowColumn widget configured as a radio box.
 */

```

```

radiobutton = XmCreateRadioBox ( rowcol, "rowcol", NULL, 0 );
XtVaSetValues (radiobutton, XmNpacking, XmPACK_COLUMN,
                XmNnumColumns, 4, NULL );
XtManageChild ( radiobutton );

/*
 * Create the children of the XmRowColumn widget.
 */
tmp = GetPixelByName ( rlist, "red" );
tmp1 = GetPixelByName ( rlist, "yellow" );

for ( i = 0; i < XtNumber ( buttons ); i++ )
{
    Widget toggle = XtVaCreateManagedWidget ( buttons(i),
        xmToggleButtonWidgetClass,
        radiobutton,
        XmNselectColor, tmp,
        XmNforeground, tmp1,
        XmNindicatorSize, 20,
        NULL, 0 );

    if ( i == 0 ) XtVaSetValues ( toggle, XmNset, TRUE, NULL );
    XtAddCallback ( toggle, XmNvalueChangedCallback,
        ValueChangedCallback, NULL );
}

footnote = XtVaCreateManagedWidget ( "rowcol2",
        xmRowColumnWidgetClass,
        mainwindow,
        XmNpacking, XmPACK_COLUMN,
        XmNnumColumns, 2,
        NULL);

```

```

fileopen = XtCreateManagedWidget ( "Currently Open Net File",
                                   xmLabelWidgetClass,
                                   footnote,
                                   NULL, 0 );

```

```

tmp = GetPixelByName ( fileopen, "yellow" );
XtVaSetValues ( fileopen, XmNforeground, tmp, NULL );

```

```

filetext = XtVaCreateManagedWidget ("File Text:",
                                     xmTextFieldWidgetClass,
                                     footnote,
                                     XmNeditable, FALSE,
                                     XmNcursorPositionVisible, FALSE,
                                     NULL );

```

```

tmp = GetPixelByName ( filetext, "gray80" );
XtVaSetValues ( filetext, XmNbackground, tmp,
               XmNvalue, "None", NULL );

```

```

CreatePopupMenu ( shell );

```

```

/* declare variables and set the font */

```

```

{
    XFontStruct *font=NULL;
    XmFontList fontlist=NULL;
    char *namestring=NULL;

```

```

    namestring="-*-helvetica-*-*--14-*";
    font=XLoadQueryFont(XtDisplay(status),namestring);
    fontlist=XmFontListCreate(font,XmSTRING_DEFAULT_CHARSET);
    XtVaSetValues(status,XmNfontList,fontlist,NULL);

```

```

    font=XLoadQueryFont(XtDisplay(llabel),namestring);
    fontlist=XmFontListCreate(font,XmSTRING_DEFAULT_CHARSET);

```



```
XtVaSetValues(llabel,XmNfontList,fontlist,NULL);
```

```
font=XLoadQueryFont(XtDisplay(rlabel),namestring);
```

```
fontlist=XmFontListCreate(font,XmSTRING_DEFAULT_CHARSET);
```

```
XtVaSetValues(rlabel,XmNfontList,fontlist,NULL);
```

```
font=XLoadQueryFont(XtDisplay(status_text),namestring);
```

```
fontlist=XmFontListCreate(font,XmSTRING_DEFAULT_CHARSET);
```

```
XtVaSetValues(status_text,XmNfontList,fontlist,NULL);
```

```
}
```

```
XtVaSetValues ( mainwindow,
```

```
    XmNmenuBar,  menu,
```

```
    XmNcommandWindow, showinfo,
```

```
    XmNworkWindow, pane,
```

```
    XmNmessageWindow, footnote,
```

```
    NULL );
```

```
XtRealizeWidget ( shell );
```

```
XtAppMainLoop ( app );
```

```
}
```

GET_HOST_LIST.C

```

/*
 * The get_host_list() function search the whole network and
 * find all the nodes and then store the host name in the array hlist().
 */
void get_host_list ( char ** hlist, int* hcount )
{
    struct hostent *ahost;    /* temporarily store a host information */
    char mhost(NAMELENGTH);
                               /* store the local host name as network manager */

    char *tp1 = "localhost";
                               /* for removing "localhost" from the host list */
    char *tp2 = "loopback";
                               /* for removing "loopback" from the host list */
    int i;                     /* temporary variable */

    if ((i = gethostname(mhost, NAMELENGTH)) < 0 ) {
        /* get local host name */
        perror("gethostname");
        exit(0);
    }
    hlist(*hcount) = malloc(sizeof(char)*(strlen(mhost)+1));
    strcpy(hlist(*hcount), mhost);    /* store local host name in hlist(0) */
    strcpy(tmplist(*hcount), hlist(*hcount));

    ++(*hcount);
    if (*hcount > MAX_HOSTS) {
        perror("Too many hosts to manage");
        exit(0);
    }
}

```

```

while ( (ahost = gethostent()) != NULL ) {
    /* read /etc/hosts file for network node */
    if (!strcmp(ahost->h_name, mhost)) continue;
    if (!strcmp(ahost->h_name, tp1) || !strcmp(ahost->h_name, tp2))
        continue;
    /* remove "local host" and " loopback " */
    hlist(*hcount) = malloc(sizeof(char)*(strlen(ahost->h_name)+1));
    strcpy(hlist(*hcount), ahost->h_name );
                                /* store the host name into the list */
    strcpy(tmplist(*hcount), hlist(*hcount));
    ++(*hcount);
}
host_file_read = 1;
    /* indicating that the completion of network search */
}

```

GET_SNMP_RESPONSE.C

```

/*
 * The get_snmp_response subroutine creates a pipe and spawns a
 * child process. The child process executes a SNMP command and
 * directs snmpd response into one end of the pipe. The parent
 * process reads the response from the other end of the pipe.
 */
static void get_snmp_response ( char *poid )
{
    char tmp(200);           /* Hold a SNMP command */
    char buf(BUFSIZ);        /* Hold the response data from the snmpd */
    int fd(2);               /* Argument for pipe() system call */
    int n = 0;               /* Record how any bytes being read from the pipe */
    pid_t pid;               /* Record a process ID */

    /* A host must be chosen first before a SNMP command can be
       executed */
    if (! hostchosen ) {
        display_info("Error Message", "No host was chosen yet for SNMP info");
        return;
    }
    sprintf(tmp, "%s -h %s -t 10 %s", command, presentHost, poid );

    if (pipe(fd) < 0)
        perror("pipe error.\n");
    if ((pid = fork()) < 0 ) {
        perror("fork()");
    }
    if (pid == 0 ) {         /* In the child process */
        close(1);            /* Close the standard output */
        dup(fd(1));          /* duplicate the standard output to fd(1) */
        close(fd(1));        /* Close the write end of the pipe */

```

```

        close(fd(0));          /* Close the read end of the pipe */
        system(tmp);          /* Execute a SNMP command */
        exit(0);              /* Terminate the child process */
    }
    /* Parent process */
    n = read(fd(0), buf, BUFSIZ);
        /* Read the snmpd response from the pipe */
    close(fd(0));              /* Close the read end */
    close(fd(1));              /* Close the write end */
    while (wait((int *) 0) != pid);
        /* Wait on the child process to terminate */
    display_info ("snmpd Response", buf );
        /* Display the data using a prompt widget */
}

```

NODESELECTCALLBACK.C

```

/*
 * NodeSelectCallback() will be invoked when a button of a host icon
 * is being pressed and released. This function calls set_ip_addr() to
 * test the status of the chosen host and set the host name and IP
 * address to the Selected Host and IP Address fields. And then the
 * background of the chosen host is either unchanged or set to black *
 * base on the test results. If nlist(i).indicator == 0, means the system is
 * down and set background to black.
 */
static void NodeSelectCallback ( Widget w, XtPointer clientData,
                               XtPointer callData )
{
    char *ptr = (char *) clientData; /* Get the chosen host name */
    int i;                          /* Temporary variable */
    Pixel tmp;                       /* Store the color pixel */

    DisplayBusyCursor ( mainwindow );
        /* Display a busy cursor in mainwindow widget
        while the testing and setting of background
        is in progress */

    printf("You selected %s\n", ptr ); /* For debug purpose */
    set_ip_addr ( ptr );

        /* Test the host status and set the hostname and
        IP address field */

    i = find_node ( ptr, count );

        /* find the host position in the host structure list */

    if (nlist(i).indicator == 0) {
        /* The system is down and set the background to black */
        tmp = GetPixelByName ( XtParent(w), "black" );

```

```
XtVaSetValues ( XtParent(w), XmNbackground, tmp, NULL );  
tmp = GetPixelByName ( w, "black" );  
XtVaSetValues ( w, XmNbackground, tmp,  
                XmNshadowType, XmSHADOW_IN, NULL );  
}  
}
```

BROWSECALLBACK.C

```

/*
 * BrowseCallback() will be invoked when the user selects an entry in
 * the sorted host list. This function calls set_ip_addr() to test the status
 * of the chosen host and set the host name and IP address to the
 * Selected Host and IP Address fields. And then the background of
 * the chosen host is either unchanged or set to black base on the test
 * results. If nlist(i).indicator == 0, means the system is down and set
 * background to black.
 */
static void BrowseCallback ( Widget w,
                           XtPointer clientData,
                           XtPointer callData)
{
    XmListCallbackStruct *cbs = ( XmListCallbackStruct * ) callData;
    /* In order to extract the host name from the callData */

    Boolean result;
    /* Result = True when setting of background is successful */

    char *text;          /* Store the chosen host name */
    Widget w1, w2, w3;    /* Temporary variables */
    char name(5);         /* Store a widget name */
    Pixel tmp;            /* Store the color pixel */
    int node_num, i, widget_name, tp; /* Temporary variable */

    DisplayBusyCursor ( mainwindow );
    /* Display a busy cursor in mainwindow */

    /*
 * Retrieve the character data from the compound string
 */

```



```

if ( ( result = XmStringGetLtoR ( cbs->item,
                                XmFONTLIST_DEFAULT_TAG,
                                &text ) ) == TRUE )
{

    if ( signal ) {                                /* In network mode */
        set_ip_addr ( text );
        /* Test the chosen host and set the Selected host and IP
           address field */

        {
            /* This block of code is to identify the widget to set
               background */
            node_num = find_node ( text, count );
            for ( i = 0; i < count; i++)
                if ( !strcmp(text, timplist(i))) widget_name = i;
            sprintf(name, "%d", widget_name);
            w1 = XtNameToWidget( tree, name );
            w2 = XtNameToWidget( w1, "unit" );
            w3 = XtNameToWidget( w2, "button" );
        }

        if ( nlist(node_num).indicator == 0 ) {
            /* The system is down and set the background to black */
            tmp = GetPixelByName ( w2, "black" );
            XtVaSetValues ( w2, XmNbackground, tmp, NULL );
            tmp = GetPixelByName ( w3, "black" );
            XtVaSetValues ( w3, XmNbackground, tmp,
                           XmNshadowType, XmSHADOW_IN, NULL );
        }
    }
    else {                                /* In MIB mode */
        for ( i = 0; i < itemcnt; i++ ) {
            if ( ! strcmp(text, oidnamelist(i)) ) {

```

```
        tp = i;
        break;
    }
}

/* Set the MIB Obj Name and MIB OID fields */
XtVaSetValues ( name_label, XmNvalue, oidnamelist(tp), NULL );
XtVaSetValues ( addr_label, XmNvalue, oidnumlist(tp), NULL );
}
}
}
```
