

Network Monitoring System

For

P-1109

InfoTech Global (India) Ltd.

Project Report

Submitted in partial fulfillment of the requirements for the award of the degree of

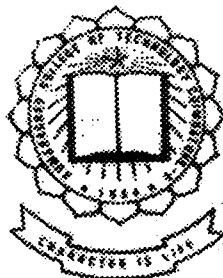
M.Sc. Applied Science (Software Engineering)
Bharathiar University,
Coimbatore.

Submitted By

B.C. SANTHOSH KUMAR
Reg. No. 9937S0088

Guided By

Mr. G. Baskar, (External Guide)
Miss. C.S. Sowmya M.E., (Internal Guide)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE – 641 006

CERTIFICATE

This is to certify that this project work entitled

“Network Monitoring System”

Submitted to

KUMARAGURU COLLEGE OF TECHNOLOGY

In partial fulfillment of the requirements for the award of the degree

Of

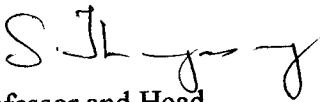
M.Sc. APPLIED SCIENCE (Software Engineering)

The record work done by

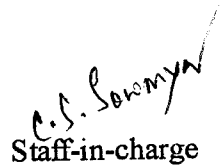
B.C. SANTHOSH KUMAR

Reg. No. 9937S0088

During his period of study in the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore – 641 006, under my supervision and guidance and this project work has not formed the basis for the award of any guidance and this project work has not formed the basis for the award of any degree/ Diploma/ Associate ship/ Followed or similar title to any candidate of any university.

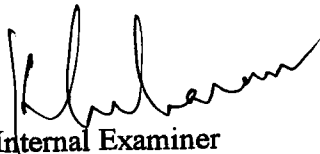


Professor and Head

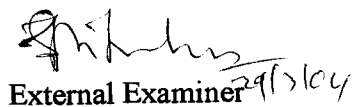


Staff-in-charge

Submitted for University Examinations held on 30/03/2004



Internal Examiner



External Examiner

Date : 04-03-2004

To Whomsoever It May Concern

This is to certify that Mr. B. C. Santhosh Kumar M.Sc (Software Engineering), Kumaraguru College of Technology, Coimbatore has successfully completed the project "Network Monitoring System" in our organization from **December 2003 to March 2004.**

During the period the conduct of the student was good.

Thanking You

Yours truly
For Infotech Global (India) Ltd

G. Ruban

G. Ruban
Director - Executions

DECLARATION

I hereby declare that the project work entitled

Network Monitoring System

Done at

InfoTech Global (India) Ltd.

and submitted to

Kumaraguru College of Technology

In partial fulfillment of the requirements for the award of the degree

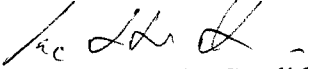
M.Sc. APPLIED SCIENCE (Software Engineering)

Is a report of work done by me during my period of study in Kumaraguru College of
Technology, Coimbatore – 641 006

Under the supervision of

**Mr. K.R. Baskaran B.E., M.S.,
Assistant Professor, Dept of Computer science & Engineering.
Kumaraguru College of Technology, Coimbatore.**

Place : Coimbatore
Date :


Signature of the Candidate
(B.C. Santhosh Kumar)

Staff-in-charge

**Mr. K.R. Baskaran B.E., M.S.,
Assistant Professor, Dept of Computer Science & Engineering,
Kumaraguru College of Technology, Coimbatore.**

ACKNOWLEDGEMENT

To add meaning to the perception, it is my indebtedness to honor a few who had helped me in this endeavor, by placing them on record.

With profound gratitude, I am extremely thankful to **Dr.K.K.Padmanaban B.Sc. (Eng), M.tech, Ph.D.**, Principal, Kumaraguru College of Technology, Coimbatore for providing me an opportunity to undergo the MSc - APPLIED SCIENCE SOFTWARE ENGINEERING course and thereby this project work also.

I extend my heartfelt thanks to my Computer Science & Engineering Department head, **Prof.Dr.S.Thangasamy B.E (Hons), Ph.D.**, for his kind advice and encouragement to complete this project successfully.

It's my privilege to express my deep sense of gratitude and profound thanks to **Mr.G.Baskar, NMS Project Manager**, InfoTech Global (India) Ltd, Bangalore for having allowed me to do my project work in his esteemed team and for helping me in all means in successful completion of this project work.

Gratitude will find least meaning without thanking my course coordinator **Mr.K.R.Baskaran B.E, M.S.**, Assistant Professor, Dept of Computer Science & Engineering and guide **Ms. C.S. Sowmya M.E.** for the valuable guidance and support throughout my project.

Words are boundless for me to express my deep sense of gratitude and profound thanks to **Mr.V.Deepak, Mr.Hegde** and all my associates at IGI, for all their kind guidance and encouragement towards my project work.

My gratitude is due to all staff members of CSE department, my parents and all my friends for their moral support and encouragement for successful completion of my project.

B.C. Santhosh Kumar.

Project Abstract

The project is done at InfoTech Global (India) Ltd, Bangalore; the main purpose of this project is to develop a Network Monitoring System to supervise the network by collecting the information from diverse parts of the network. This Project entitled “**Distributed Network Monitoring in Client/Server Systems**” is developed using ANSI C and QT for Interface Designing.

Emerging Network applications will push the limits of available network bandwidth. Two critical services are required to guarantee maximum efficient use of the network resources. The first is a unit for monitoring the performance of each component within the system. The second is a system for monitoring current network characteristics and providing this information to network-aware applications, which can effectively adapt to the current network conditions. Both capabilities require an adaptive monitoring infrastructure: ***a monitor data publishing mechanism and a monitor data analysis tools (logging engine)***. This project will develop a service that will provide both of these capabilities.

The NMS is composed of two parts. The first part is Data Analyzer, the *information gatherer*, called the **logging engine** that resides on all the computers making up the virtual environment. The **logging engine** is designed, so that it runs for all users. The second part of the system is the data publisher called **dashboard** that loads according to the user privileges. A **dashboard** contains a complete customizable interface where the user can customize the functionality of the NMS to ones needs.

Contents

| | |
|---|----|
| 1. INTRODUCTION | |
| 1.1 Organization Profile..... | 1 |
| 1.2 Problem Definition | 2 |
| 1.2.1 Objective..... | 2 |
| 1.3 Existing System..... | 3 |
| 1.4 Proposed System..... | 4 |
| 2. SYSTEM ANALYSIS | |
| 2.1 System Requirements..... | 7 |
| 2.2 Scope of the System | 8 |
| 2.3 System Environment | 8 |
| 2.4 System Modeling | 10 |
| 2.5 Justification of the Development Methodology..... | 12 |
| 3. SYSTEM DESIGN | |
| 3.1 Module Design | 14 |
| 3.2 Process View..... | 20 |
| 3.3 Implementation View..... | 25 |
| 4. SYSTEM IMPLEMENTATION | |
| 4.1 Testing and Test Plan..... | 27 |
| 4.2 Testing Methods..... | 28 |
| 4.3 Installation of the System..... | 32 |
| 5. FUTURE ENHANCEMENTS..... | 33 |
| 6. CONCLUSION | 34 |
| 7. REFERENCES | |
| 8. APPENDIX A | |
| 9. APPENDIX B | |

INTRODUCTION

1.1. Organization Profile

InfoTech Global (India) Ltd is a unit of the renowned IGI Group, which in past 10 years has established in Southern India in the field of engineering education and research, committed to provide support to small, medium and large corporations in the development and management of software essential to their needs over the entire life cycle of a system, providing expert services and support for "change management" in software systems allowing your organization to focus on its core business, offering the expertise of experienced individual software consultants, as well as an off-shore facility with a state-of-the-art information technology infrastructure, software services in Web Based Applications and e-commerce, Client-Server (Two-Three and N-Tier Technology), Group Ware and Workflow, Multimedia and Computer Graphics, CAD,CAM.

InfoTech Global (India) Ltd is one of the fastest growing software companies. It has many branches and strategic alliances located all over the world; their world wide registered Global head office is located at Bangalore. InfoTech Global (India) Ltd is currently having more than 250 strong, well-qualified team of Software and project management team, with well proven track record in handling large multinational projects and developing products for the global market place, with latest technology & tools, proven on-time delivery records, cost effectiveness and best of quality.

1.2. Problem Definition

The network has to be managed in order to optimize the utilization. It is very difficult to monitor the network manually. There is a need for a system to manage the network automatically, so that the performance of the network can be improved, the resource utilization can be reduced and intruders can be detected.

A stealth-monitoring utility is needed which provides the PCs equivalent of the security camera (and much more), causing no disruption to working practices or draining network resources.

Any user should be able to know the status of the network and the state of the individual PCs in the network sitting at any point.

Identifying performance bottlenecks and tuning clusters requires accurate load and status information of all parts of the system. An ideal monitor should be easy to configure and provide centralized graphical, real-time feedback without interfering with the system it measures. Such a tool must provide timely, accurate information even for systems under extreme load without causing an overhead to the network by itself.

The NMS attempts to meet these goals.

1.2.1. NMS Objective

The NMS is used to supervise the network by collecting the information from diverse parts of the network. It performs various types of monitoring like performance monitoring, account monitoring, system auditing and network traffic. NMS also

provides utilities for the user to make the resource utilization easier.

NMS

- Notifies the status of the memory, hard disks.
- Maintains secured log files.
- Clears and updates the log files on periodic basis.
- Generates query-based reports as per the user needs.
- Does client auditing
- Does network traffic analysis
- Lists workstations on the network
- Displays notification banners
- Provides utilities for resource sharing and information exchanging.

1.3. Existing System

Several utilities like **XStat**, **PMgr** and **NSpy** provide most of the monitoring features with one serious omission, continuous information under extreme load. These tools typically employ a user-level daemon process or sometimes a remote procedure call to monitor system activities.

On a system under heavy load or one spending long amounts of time inside the kernel, these tools are arbitrarily delayed since it take its own time to execute itself to gather information, manipulate information and transmit the information to the requested client. These tools act as having a server component & a client component interacting at different

levels. So this makes the monitoring always possible from only one location mainly the server.

This leads to a drawback like not being able to obtain the history of the network at different levels, because any failure or tampering of recorded information in the server leads to loss of information.

There are no possible ways to secure the log files. There are no encrypted log files. So anyone can view the logs and can modify it. Password protected log files are therefore a must. There are no built-in customizable messages to notify the PC user that the working environment is monitored with a monitoring program.

In general the existing systems

- Are not distributed
- Detects network traffic.
- Doesn't maintain log files with security.
- Doesn't do client side auditing.
- Measures the performance of all nodes connected in the network.

1.4. Proposed System

The proposed system is **distributed network monitoring in client/server systems**. The approach is placing the information gatherer (logging engine) inside the operating system, invoked at regular intervals using the system's soft clock interrupt handler, independent of scheduling disciplines which were applied for the application programs and the OS initiated processes.

The logging engine constructs and queues a simple network packet containing the current system status, updates it in a state table and broadcast it to other clients. The information gatherers constitute a network service, convened with the data publisher (dashboard) at user level.

NMS allows you to log all keystrokes, user names, passwords, path names, access times and window titles, then send the log file by broadcasting invisibly.

The NMS's dashboard is provided with high user interface where the user is able to set the log file size, file name, passwords, setting time delays, built-in customizable message to notify the PC user regarding the monitoring program since one should be indicated regarding the surveillance. The user can customize the text of the popup messages or banners according to the user's needs.

There are possible controls for the user to start or stop the logging engine whenever needed. The user can also customize it to load along with the OS.

Backing up the log files, resource sharing, mapping to virtual drive (thru NFS), a chat panel, and file related activities for set of file, for individual files, are the key feature of the NMS.

The NMS

- Detects network traffic and generates notification on network overload.
- Audits multiple clients dynamically.
- Maintains secured log files and generates query based reports.
- Provides higher level of user interaction
- Notify the CPU breakdown (user, system, intruders, and idle.)
- Notify the device status (read/write bytes, busy percentage)
- Notify the Memory usages
- Notify the file activities
- Provide file related utilities and communication utilities for the user.

The NMS will be a heavy base for monitoring stand alone as well as PCs in a network.

SYSTEM ANALYSIS

2.1. System Requirements

❖ Hardware Requirements

- Network enabled Workstations with Intel 80X86(or its clones) capable of running Linux.
- Bus/Star based interconnection network

❖ Software Requirements

- Linux (Kernel - 2.2.16 or higher)
- ANSI C/C++ compiler for Linux platform
- QT for Interface Designing

❖ Functionality Requirements

- **State Table:** The key mechanism in the NMS is the state table, which is maintained at every client and updated at intervals. Analyzing the state table the information of the network at a particular instance can be gathered
- **Task Control Mechanisms:** The NMS provides mechanisms for spawning a task on a local/remote host, killing a task, signaling a task apart from other message passing routines.

- **Message Passing Mechanisms:** Various message-passing primitives are implemented in this NMS. These routines follow an asynchronous mode of communication.
- **Group Communication:** NMS has facilities to address a group of tasks together. Various group operations like broadcast, multicast are implemented.

2.2. Scope of the System

The scope of this system is to provide a network monitoring solution. The system can be used by any PC User, a network administrator for any kind of network utilization like a file operation, a chat or a surveillance tool.

The boundary of the system is limited to Intranet at present and the future enhancement takes it to the Internet.

2.3. System Environment

❖ Development Environment

The NMS has been developed using C/C++ language & QT on Linux platform. GNU C/C++ compilers were used for the development of the system.

❖ Implementation Environment

It is mandatory that each workstation be configured for NFS (Network File System) for file sharing between the nodes in the

cluster. Also as every user requires identical logins in each of the workstations, the presence of NFS in the network is assumed.

❖ **Network Monitoring**

Network Monitoring is closely watching the network and recording the activities in the network called auditing. A network is audited. This auditing applies to each and every client available in the network.

❖ **Network Monitoring Tool**

Software which does Network Monitoring is known as Network Monitoring Tool. It acts like a video camera for a network.

❖ **Distributed Network Monitoring**

Distributed Network Monitoring is monitoring the network from different places and placing the reports in a distributed fashion. The main issue to be considered is consistency of data. The reports should be consistent. At any terminal the report generated should be same.

❖ **About QT**

Qt is a cross-platform C++ GUI application framework. It provides application developers with all the functionality needed to build state-of-the-art graphical user interfaces. Qt is fully object-oriented, easily extensible, and allows true component programming.

Since its commercial introduction in early 1996, Qt has formed the basis of many thousands of successful applications worldwide. Qt is also the basis of the popular KDE Linux desktop environment, a standard component of all major Linux distributions.

Qt is released in different editions. Qt Enterprise Edition and Qt Professional Edition for commercial software development.

2.4 System Modeling

The NMS is composed of two parts. The first part is Data Analyzer, the *information gatherer*, called the **logging engine** that resides on all the computers making up the virtual environment. The **logging engine** is designed, so that it runs for all users. The second part of the system is the data publisher called **dashboard** that loads according to the user privileges. A **dashboard** contains a complete customizable interface where the user can customize the functionality of the NMS to ones needs.

An exemplary diagram of the NMS computing model is shown in Figure 2.1. An architectural view of the NMS is shown in Figure 2.2.

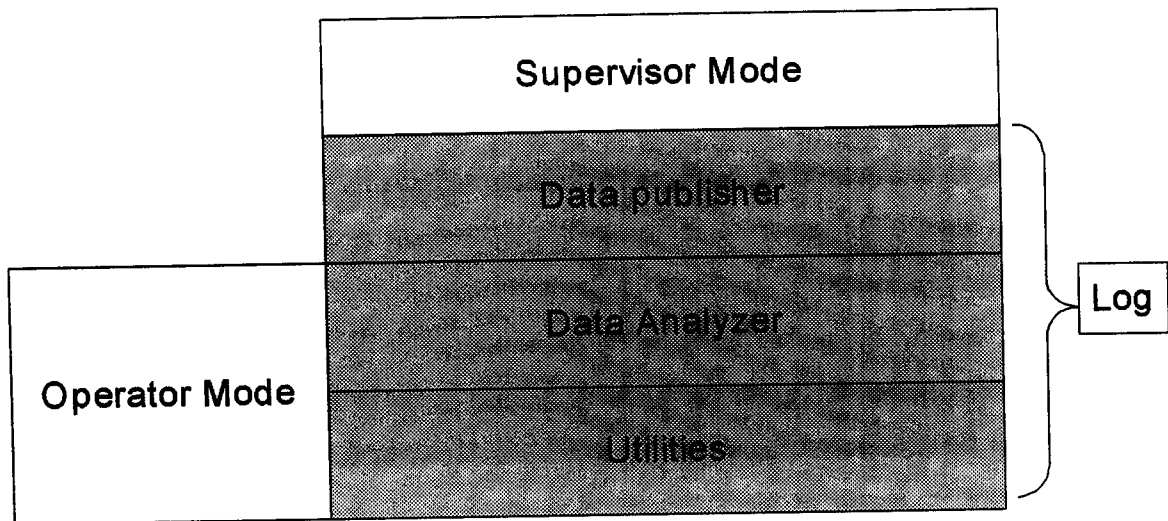


Figure 2.1 NMS Computational Model

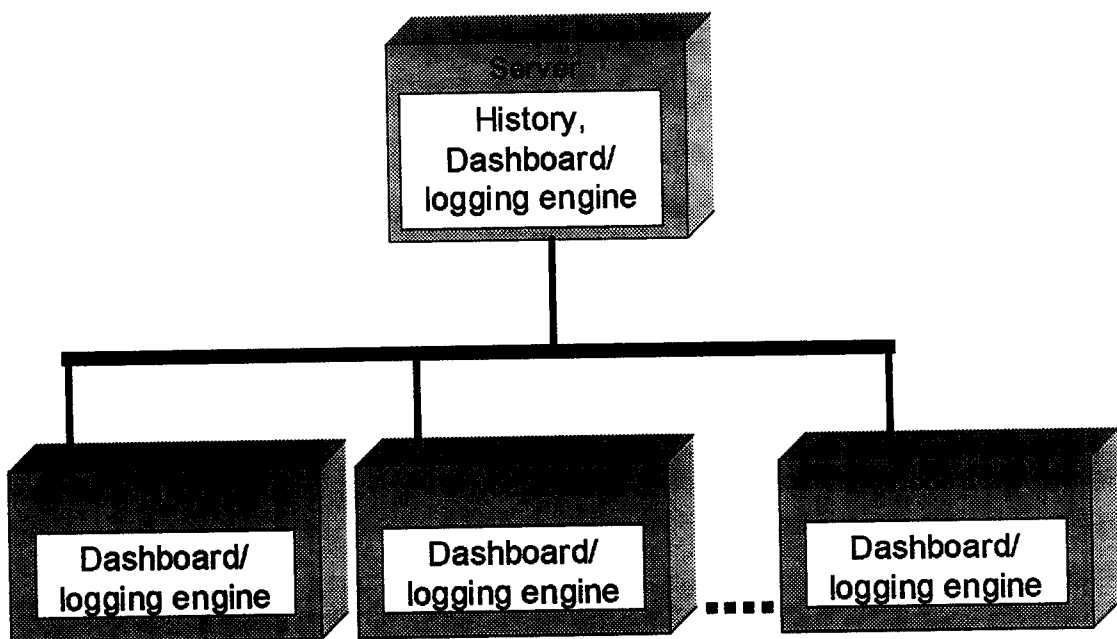


Figure 2.2 NMS Architectural Overview

The C language procedures for the NMS user interface are implemented as functions, following the general conventions used by most C systems, including Unix-like operating systems. The Graphical User Interface (GUI) is given for the **dashboard** using QT, a cross-platform GUI framework for Linux where the C language procedures are given a handle.

The NMS messages are uniquely identified by the Message Transmission Manager and the Message Recipient Manager available in the logging engine.

2.5 Justification of the Development Methodology

Incremental Model is the software process model used for the development of this system. The whole project was decomposed into different phases based on the requirements, and a step-by-step implementation of each phase was carried out. As this system is radically different from other software systems, and the software process was not very clear during the commencement of the project, evolutionary software process models seemed to be the ideal ones. Unlike *spiral model*, the incremental model does not require risk assessment expertise, and hence seemed to be the apt strategy to be adopted for the development of the system.

The NMS is built using C/C++ and QT on Linux platform. As Linux is the most preferred operating system for compute intensive, real-time and scientific computations, it serves as felicitous platform for making a Distributed NMS. The system uses internal data-structures *queues* and *linear lists*. C++ was preferred over C to implement these data-structures as it provides object-oriented features, which is necessary to manage the complexity of the system. Apart from the implementation of these data-structures, the rest of the system is built using ANSI C constructs. The internal communication between various processes in the system is realized through synchronous sockets.

3.1 Module Design

❖ The Main Module – NMS

The Prime modules in NMS are

- Data Analyzer (logging engine)
- Date Publisher (dashboard)
- Utility
- Log Generator

Figure 3.1 shows how NMS acts to different user privileges.

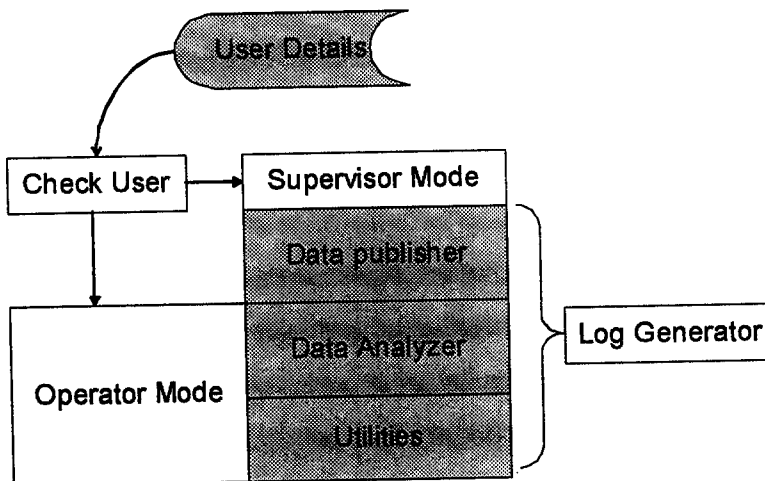


Figure 3.1 Loading NMS

The NMS operates in two modes namely

- Supervisor Mode
- Operator Mode

The *Supervisor mode* is one where the user having supervisor privileges can access the full functionality of the NMS. All available prime modules are initialized.

The *Operator mode* is one where the end-user is logged, the data publisher is not initiated in this mode.

❖ **Data Analyzer**

The Data Analyzer is responsible for information gathering and logging the information collected in files and making them available in a distributed manner. The Data Analyzer is a collection of sub modules interacting among themselves.

Each and every sub modules available in the Data Analyzer is made to interact with the Log Generator, a separate module for the information logging operations.

The modules available in the Data Analyzer are listed below

- System Analyzer
- System Auditor
- Fault Analyzer
- Network Auditor
- Resource Analyzer
- Manager-Transmission
- Manager- Reception
- State Table Generator
- State Table Recorder

The *System Analyzer* is responsible for checking the hardware present in individual systems available in the network.

The *System Auditor* is responsible for the user activities in each system. The user activity is recorded at regular intervals and then broadcasted. The state table and the log files are updated.

The *Fault Analyzer* is responsible for checking the network faults, i.e. break in the path (no physical link between two clients)

The *Network Auditor* is responsible for tracking the traffic in the network. It takes care of the network load and updates the logs at regular interval. The information from the network auditor is used for remedies.

The *Resource Analyzer* is used to check whether the system is available in the network. The analyzer not only checks the

clients but also finds the peripherals like printers, scanners available in the network.

The *Manager-Transmission* is responsible for getting the message from different sub modules and framing the appropriate headers to the messages and sending the message.

The *Manager-Reception* is responsible for receiving the messages from the co-modules available in the parent client as well as the co-clients. It takes care of deframing the framed messages received and distributing it to appropriate co-modules.

The *State Table Generator* is responsible for generating a state table and updating it in other clients.

The *State Table Recorder* writes the current state table to the physical medium.

❖ **Data Publisher**

A data publisher contains a complete customizable interface where the user can customize the functionality of the NMS to ones needs. The user with supervisor privileges can only access the data publisher.

The data publisher is provided with a graphical use interface (GUI) from which the user can easily customize his setting for X-window Systems. The data publisher also works with a CUI environment.

The prime modules of the data publisher are

- Customize Manager
- Report Generator

The *Customize Manager* can be used to set the intervals for logging, refreshing the physical medium, placing the history at the server, clearing all logs after backing-up, number of days a log can retain, the size of the log files, naming the log files, place of distribution, start/stop logging engine along with OS, setting up the user controls.

The *Report Generator* is the user level interface which is used to extract the information from the logs maintained, from the state tables using various requirement parameters like date and time ranges, user names and system names from the user.

❖ **Utility**

The utility module is the set of utilities for the users to make use of the available resources to higher extents.

The sub-modules of the utility module are

- Chat
- Resources Tracker
- Information Tracker
- Information Manipulator
- Virtual Drive Mapping

The *Chat* is a simple communication utility for the users to exchange messages. There is possibility for one-one, one-many and many-one communication.

The *Resources Tracker* is used to find the computer in the network. This is similar to a find computer utility available in the network neighborhood application of Microsoft.

The *Information Tracker* is used to find the files available in the network. Checking is done for multiple instances at multiple locations.

The *Information Manipulator* takes care of file copy, file move, file delete. Batch processing is possible, i.e. copying or moving or deleting set of files.

The *Virtual Drive Mapping* is directing a drive virtually created to another client's directory or a drive. The fstab system file's information is modified to make this mapping. The mapped drive can be accessed as a local drive.

❖ **Log Generator**

The Log Generator generates log on all activities related to the NMS and the information obtained through NMS. This is an individual module, which always interacts with all other modules in the NMS. The log generator acts independent of the modules and creates log for the NMS related services also.

3.2 Process View

❖ Utilities

➤ Chat

The *Chat* is a simple communication utility for the users to exchange messages. There is possibility for one-one, one-many and many-one communication.

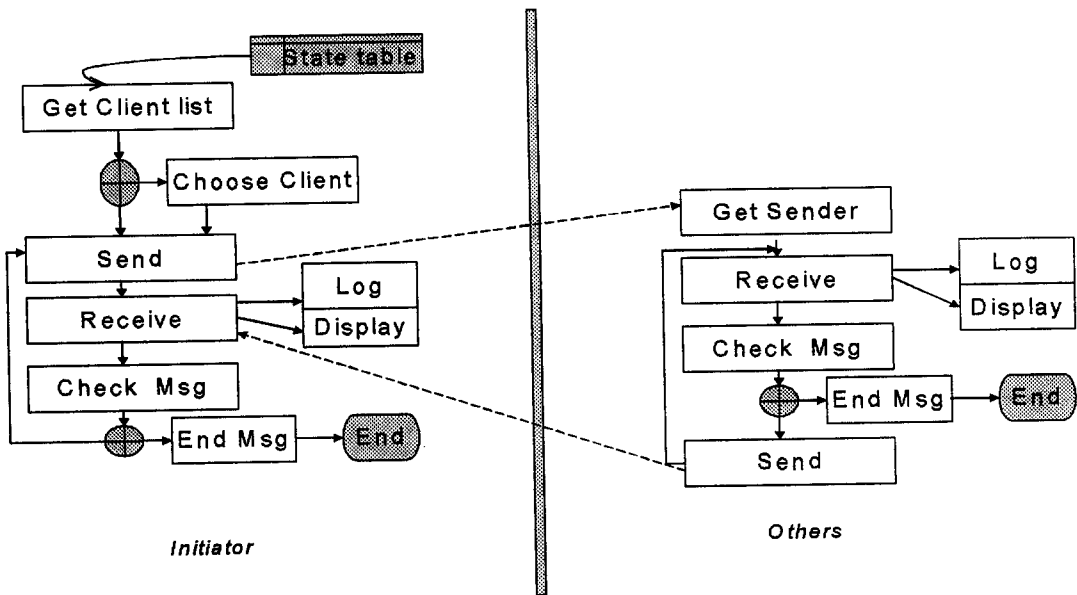
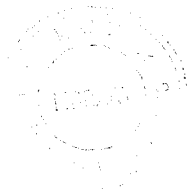


Figure 3.2.1(A) Chat



➤ Resource Tracker

The *Resources Tracker* is used to find the computer in the network. This is similar to a find computer utility available in the network neighborhood application of Microsoft.

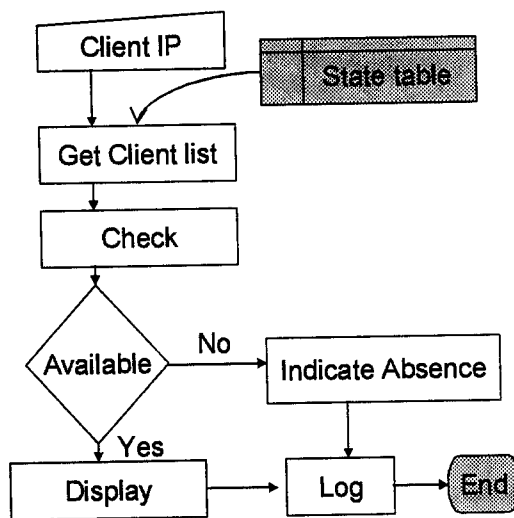


Figure 3.2.1(B) Resource Tracker

➤ **Information Tracker**

The *Information Tracker* is used to find the files available in the network. Checking is done for multiple instances at multiple locations.

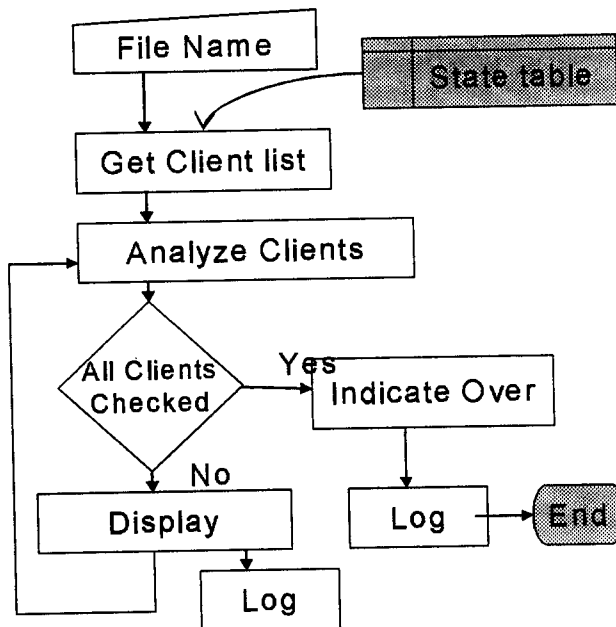


Figure 3.2.1(C) Information Tracker

➤ Information Manipulator

The *Information Tracker* is used to find the files available in the network. Checking is done for multiple instances at multiple locations.

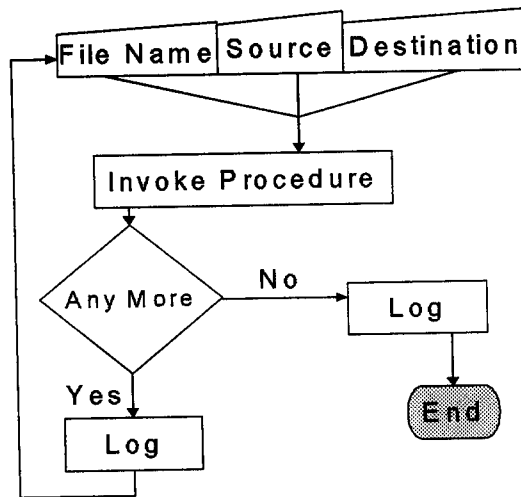


Figure 3.2.1(D) Information Manipulator

- ✓ Choose an operation from the list, a file copy or move or delete or rename.
- ✓ Choose the files, source destinations and required parameters.
- ✓ Invoke appropriate procedures and log events.

➤ **Virtual Drive Mapping**

The *Virtual Drive Mapping* is directing a drive virtually created to another client's directory or a drive. The fstab system file's information is modified to make this mapping. The mapped drive can be accessed as a local drive.

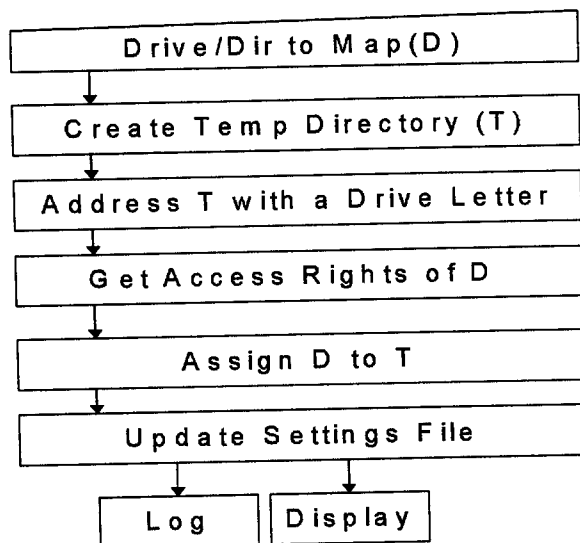


Figure 3.2.1(E) Virtual Drive Mapping

3.3 Implementation View

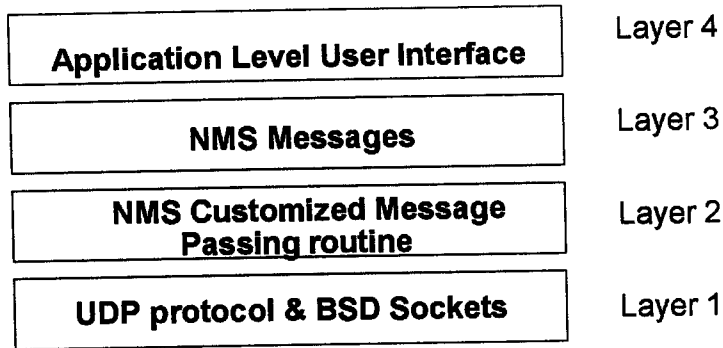


Figure 3.3 Implementation Model

❖ Overview

The implementation model consists of four layers and is depicted in the figure 3.3. They are

- UDP protocol & BSD Sockets
- NMS Customized Message Passing routines
- NMS Messages
- Application Level User Interface

❖ **Layers**

➤ **UDP Protocol & BSD Sockets**

NMS internally uses BSD sockets and UDP Protocol for communicating with various processes/modules through the network. It is to be noted here that UDP provides an unreliable, connectionless packet delivery service.

➤ **NMS Customized Message Passing Routines**

The responsibility of this layer is to provide a reliable, connectionless communication mechanism, which can be used to transfer data/messages through the network by the NMS.

➤ **NMS Messages**

Various modules of the NMS interact with each other through pre-defined messages.

➤ **Application Level User Interface**

The User Level Interface is given to customize the NMS and make use of the system to the higher extent.

SYSTEM IMPLEMENTATION

The unit testing has been done separately for each of the modules. Both white box and black box testing techniques were employed. The testing was done with maximum number of test cases. The test cases were carefully designed during the process of coding and design. The test cases covered almost every part of the code to make the modules fault proof. The errors uncovered were rectified, and the modules were re-tested with all the test cases to ensure that the corrective measures taken did not cause any inadvertent effects. Some samples are shown in the table below.

4.1 Testing and Test Plan

The modules, their functionality and interaction between modules are tested in the integrated subsystems. Testing was also carried out to investigate the changes made in the individual modules, to check and find the weakness in them.

Various testing strategies like unit testing, integration testing, validation testing and system testing were applied. The testing techniques like white-box testing, basis path testing, control structure testing, and black box testing ensured successful testing of the system.

4.2 Testing Methods

❖ Unit Testing

The unit testing has been done separately for each of the modules. Both white box and black box testing techniques were employed. The testing was done with maximum number of test cases. The test cases were carefully designed during the process of coding and design. The test cases covered almost every part of the code to make the modules fault proof. The errors uncovered were rectified, and the modules were re-tested with all the test cases to ensure that the corrective measures taken did not cause any inadvertent effects. Some samples are shown in Table 4.1.

| Seq. No. | Test case | Condition being checked | Expected output |
|-----------------|------------------------|---|---|
| 1 | Delete PST | Deleting the Log file externally | Creates & updates logs and continues |
| 2 | Delete Hstry | Deleting the Hstry record externally | Redistributes, updates local logs and continues |
| 3 | Network disconnected | Try to find the stability | Updates HLT and continues with local host |
| 4 | Kill logging engine | Externally killing the logging engine module | Indicates error and automatically dashboard exits |
| 5 | Log file size overflow | The log file size increases than the user specified size | Warn the user and wait |
| 6 | Invalid IP force | Try to connect to a offline machine | Print message and stop |
| 7 | No disk space | Gathered information could not be stored due to disk out of space | Print message and stop |
| 8 | Date/Time out of range | Giving invalid date and time for reports | Warns the user and proceeds with valid data |
| 9 | Invalid log file size | Setting up negative log file sizes | Warn the user and wait |
| 10 | Pause logger | Stopping the logger for some time | Updates local logs and resumes |

Table 4.1 Test Cases

❖ **Integration Testing**

The integration testing was done when various modules of the system were integrated together. Top-Down approach was followed during the process of integration. Further, *depth first* integration was carried out. The black box testing technique was employed throughout this phase. Stubs were created as required to ensure an incremental approach, and later, on completion of each set of tests, stubs were appropriately replaced with real modules. Finally, Regression testing was conducted to ensure that the errors rectified were successfully working, and they do not cause any adverse effects.

During testing, each interface provided by the system was supplied with variety of arguments respective to it. The results were as expected for all combinations of the arguments.

❖ **Validation Testing**

The Validation testing was carried out at the culmination of the integration testing. Validation was done against the *validation criteria* specified in the software requirements specification. It was done as a series of black-box tests that demonstrated conformity with requirements. It ensured that all functional requirements are satisfied; all performance requirements are achieved; documentation is correct and human-engineered; and other requirements are met.

❖ **Performance & Stress Testing**

As a part of system testing, performance tests were conducted along with stress testing. The stress and performance of the system was calculated based on the traffic on the network, system load and other system resources. It was found that the system was working fine with no stress and slowed down in small fractions of time when the traffic and CPU load was increased. The variation of other available system resources like memory, disk-space did not make any significant performance degradation in the system.

❖ **Recovery Testing**

Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed. Many different test cases were designed and tested, for example killing the *dashboard* explicitly, forcing one of the workstation involved to crash, dismantling the network connections abruptly, flooding the network. In all the cases the system responded as expected, sending appropriate notifications and error messages.

❖ **Platform Relevant Testing**

The system has been tested on two flavors of Linux, namely Red Hat and SuSE. The kernels that involved in testing were versions 2.2.16 and 2.4.2. It was found that these were completely compatible with the NMS.

4.3 Installation of the System

Installation of the NMS is very simple, and can be done by any user with a valid account on the host.

Before Installation the system, it's forced in to many server-testing phases. After the system clears all the tests, it's released for Installation. After the data has been initially set, the system is ready for use. The Installation type or the change over technique from the existing system is a step by process.

First a module in the part of the system is Installation and checked for suitability and the efficiency. If the end user related to the particular module is satisfied, the next step of Installation is processed with. That's modules related to the previous module are Installation.

FUTURE ENHANCEMENTS

This session briefs about the possible enhancements that can be done to improve the scope and functionality of the system.

❖ Support for Heterogeneous Architecture

Though Linux is the only operating system supported by NMS in its first version, other UNIX clones like System V, SCO Unix, Free BSD, AIX, and Solaris could easily be accommodated. Future versions may also support other radically different platforms like Windows and Mach.

The scope can be extended to **Internet**.

CONCLUSION

“NMS” has been developed keeping in mind the entire network user roles satisfaction. The system has been developed in an interactive manner and has been targeted for use in Intranet.

The tool takes care of the performance of the network, maintenance of log files, resources available in the network, system auditing and traffic in the network. Extending the scope for the Internet can enhance the system. The tool is designed to get adapted to the changes that occur in future.

The key feature of the tool is that NMS is distributed, providing possible path for recovery in case of failures.

REFERENCES

Books:

W. Richard Stevens, "*Unix Network Programming*", Volume 1, Second Edition, Addison Wesley, 1999

Pradeep K. Sinha, "*Distributed Operating Systems*", IEEE Computer Society Press, 1998

Andrew S. Tanenbaum, "*Modern Operating Systems*", Prentice Hall of India, 1996

Douglas E. Comer, "*Internetworking with TCP/IP*", Volume 1, (Principles, Protocols and Architecture), Third Edition, Prentice Hall of India, 2001

W. Richard Stevens, "*Unix Network Programming*", Prentice Hall of India, 2000

V. Honavar, "*Distributed Knowledge Networks*", IEEE Computer Society Press, 1999

G. Helmer, J. Wong, V. Honavar, and L. Miller, "*Intelligent Agents for Intrusion Detection*", IEEE Computer Society Press, 1998

J. Yang, R. Havaladar, V. Honavar, L. Miller, and J. Wong, "*Coordination and Control of Distributed Knowledge Networks*", IEEE Computer Society Press, 1998

Web Pages

- General Linux Advocacy – December 2003
 1. <http://oloon.student.utwente.nl/linux/LDP/HOWTO/mini/Advocacy.html>
The Linux Advocacy mini-HOWTO.
 2. <http://www.10mb.com/linux/>

- User Interface design pages – January 2003
 1. <http://www.cis.ohio-state.edu/~perlman/CIS516/uidesign.html>
Intelligent User Interface design -- a book length site
 2. <http://www4.ncsu.edu/~aklikins/gnome/style.html>
The GNOME (GNU Network Object Model Environment) homepage for UI design

- Linux – December 2003
 1. <http://www.linux.com>
 2. <http://www.linux.net>

- Performance Monitor – December 2003
 1. <http://www.usenix.org/students/research.html>
 2. <http://www.cs.iastate.edu/~honavar/aigroup.html>

- Forums/ Groups – December 2003
 1. <http://www.yahogroups.com/linux>
 2. <http://www.linuxforum.com>

APPENDIX A

Project Dictionary

A.1 Process State Table Structure (PST)

| Date | Time | Duration | Process Id | Title | Path | Text | Status |
|------|------|----------|------------|-------|------|------|--------|
|------|------|----------|------------|-------|------|------|--------|

This state table contains the list of all process happening in a single client. This table is recorded to the log file in each and every client.

A.2 Host List State Table Structure (HLT)

| System IP | Status |
|-----------|--------|
|-----------|--------|

This state table is dynamic, holding the clients available in the network and the status of the client. The status is a Boolean data holding 0 or 1 as values.

The status value 0 indicates shutdown and 1 indicates presence. When a state table is created the current system updates the value by 1 and any normal termination leads to a value 0.

This action is also recorded. This state table is unique in all clients.

A.3 Distributed History File Structure (Hstry)

| |
|------|
| Logs |
|------|

Hstry file is updated at all clients at regular intervals specified by the user, the file is created and information from all PST is appended. The PST of all clients is cleared.

This Hstry file is distributed at all available clients and the data publisher can access it for reports at any time.

At a user specified interval, the server is updated with only one copy of the Hstry and all other copies are cleared.

There is also option for customizing the NMS to have Hstry permanently in various systems.

A.4 NMS Message Format

| | | |
|--------------------|---------------|---------|
| NMS Message Header | Module Header | Message |
|--------------------|---------------|---------|

The NMS Message format is a combination of NMS Message Header, Module Header and Actual Message.

The *NMS Message Header* will indicate that the message is for NMS.

The *Module Header* indicates to which module the message belongs to.

A.4.1 Message Header List

1. NMS Message Header – “<N><M>”

2. Data Analyzer

| Modules | Headers |
|--------------------------|----------------|
| System Analyzer | <SA>Z> |
| System Auditor | <SA>D> |
| Fault Analyzer | <FA>Z> |
| Network Auditor | <NA>D> |
| Resource Analyzer | <RA>Z> |
| Transmission Manager | <TM>R> |
| Reception Manager | <RM>R> |
| State Table Generator | <ST>G> |
| State Table Recorder | <ST>R> |

3. Data Publisher

| Modules | Headers |
|-------------------|----------------|
| Customize Manager | <CM>R> |
| Report Generator | <RP>R> |

4. Log Generator – “<LG>G>”

5. Utilities

| Modules | Headers |
|----------------------------|----------------|
| Chat | <CH>U> |
| Resource Tracker | <RT>U> |
| Information Tracker | <IT>U> |
| Information Manipulator | <IM>U> |
| Virtual Drive Mapping | <VD>U> |

Appendix B

Glossary of Terms

Distributed Computing Environment (DCE)

The OSF Distributed Computing Environment is a comprehensive, integrated set of services that supports the development, use and maintenance of distributed applications. It provides a uniform set of services, anywhere in the network, enabling applications to utilize the power of a heterogeneous network of computers.

Interconnection network

It is the system of logic and conductors that connects the processors in a parallel computer system. Some examples are bus, mesh, hypercube and Omega networks

Latency

The time taken to service a request or deliver a message which is independent of the size or nature of the operation. The latency of a message passing system is the minimum time to deliver a message, even one of zero length, which does not have to leave the source processor.

Single Point of Failure

This is where one part of a system will make the whole system fail.

State Table

The key mechanism in the NMS is the state table which is maintained at every client and updated at intervals. Analyzing the state table the information of the network at a particular instance can be gathered

Sockets

Also commonly known as Unix Berkeley Sockets, these were developed in the early 1980s as a means of providing application writers a portable means of accessing the communications hardware of the network. Since sockets allow point-to-point communications between processes, it is used in most of the networked workstation implementations of message passing libraries.