Design and Development of SNMP based Network Monitoring Tool for SCADA

Thesis submitted in partial fulfillment of the requirements for the award of degree

Master of Engineering
in
Computer Science and Engineering

Submitted By
AMAN MAHAJAN
(Roll No. 800932002)

Under the supervision of

Mr. Haresh Joshi
Manager-Technology, Crompton Greaves Global R&D

Dr. A.K Verma
Assistant Professor, CSED, Thapar University

COMPUTER SCIENCE AND ENGINEERING
DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2011

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"DESIGN & DEVELOPMENT OF SNMP BASED NETWORK MONITORING TOOL FOR SCADA"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. A.K Verma and Mr.Haresh Joshi* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature: 01/06/11

(Aman Mahajan)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. A.K Verma)
Assistant Professor,
CSE Deptartment

01.06.11

(Mr. Haresh Joshi)
Manager, DARC
CG Global R&D

Crompton Greaves Ltd
Global R & D Centre
Kanjur, Mumbai - 400078
India

Countersigned by

15/6/11

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

ACKNOWLEDGEMENT

No volume of words is enough to express my gratitude towards my guide, **Dr. Anil Kumar Verma**, Assistant Professor, Computer Science and Engineering Department, Thapar University, Patiala and **Mr. Haresh Joshi**, Manager-Technology, DARC, Crompton Greaves Global R&D, Mumbai, who have been very concerned and have supervised the work presented in this thesis report. They helped me to explore this vast field in an organized manner and provided me with all the ideas on how to work towards a research oriented venture.

I am also thankful to **Dr. Maninder Singh**, Head of Department, **Mr. Karun Verma**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my friends who were always there in the need of the hour and provided with all the help and facilities, which I required, for the completion of my thesis.

I would also like to thank the organization i.e. Crompton Greaves Limited (CGL) for providing all the necessary resources and permission to carry out this research activity. I also wish to thank the employees of DARC department for their continuous help and support throughout out stay in CGL.

Most importantly, I would like to thank my **Parents, friends** and the **Almighty** for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Aman Mahajan

(800932002)

ABSTRACT

This work describes the design and development of architecture for automated network node discovery, monitoring and topology analysis, implemented as an integrated module for the CrompSCADA. The architecture includes functionality for flexible network model assessment, using a method for versatile comparison between off-line database models and real-world models. These models are populated by current node data collected by network sensors. The presented architecture supports – an efficient creation and synchronization of network topology information, accurate recognition of new/ replaced/upgraded nodes, including SCADA specific devices like Xcell RTU (Remote Terminal Unit), IED (Intelligent Electronic Device), and also provides a free replica of existing vendor specific enterprise network management and provisioning systems. An evaluation of the implementation shows evidence of accurate discovery and classification of unmatched hosts in a live network with over 400 nodes, and presents data on performance and scalability levels.

The work was carried out at Distribution Automation Research Centre (DARC), Crompton Greaves Global R&D, Mumbai as part of the MoU between Thapar University, Patiala and Crompton Greaves, Mumbai.

Keywords: ICMP, SNMP, IP, MIBs, OID.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS & ACRONYMS

API	Application Programming Interface
BER	Basic Encoding Rules
BGP	Border Gateway Protocol
BPS	Bits Per Second
CG	Crompton Greaves
CIL	Common Intermediate Language
CLI	Common Language Infrastructure
CLR	Common Language Runtime
DCOM	Distributed Component Object Model
DCS	Distributed Control System
DNS	Domain Name Server
GDI	Graphics Device Interface
GPL	General Public License
HTTP	HyperText Transfer Protocol
HMI	Human-Machine Interface
ICMP	Internet Control Message Protocol
IED	Intelligent Electronic Devices
IP	Internet Protocol
ISO	International Organization for Standardization
IMAP	Internet message access protocol
IEC	International Electrotechnical Commission
LAN	Local Area Network
MIB	Management Information Base
OID	Object Identifier
OPC	OLE for Process Control
PAC	Programmable Automation Controller

PC	Personal Computer
PLC	Programmable Logic Controller
POP3	Post Office Protocol 3
RFC	Request For Comment
RTT	Round Trip Time
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SDH	Synchronous Digital Hierarchy
SMTP	Simple Mail Transfer Protocol
SMS	Short Message Service
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TTL	Time-To-Live
UDP	User Datagram Protocol
WAN	Wide Area Network
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 SCADA

SCADA [1] stands for *supervisory control and data acquisition*. It generally refers to industrial control systems: computer systems that monitor and control industrial, infrastructure, or facility-based processes, as described below:

- Industrial processes include those of manufacturing, production, power generation, fabrication, and refining, and may run in continuous, batch, repetitive, or discrete modes.
- Infrastructure processes may be public or private, and include water treatment and distribution, wastewater collection and treatment, oil and gas pipelines, electrical power transmission and distribution, wind farms, civil defense siren systems, and large communication systems.
- Facility processes occur both in public facilities and private ones, including buildings, airports, ships, and space stations. They monitor and control HVAC, access, and energy consumption.
- SCADA is very important system for monitoring and management of electric stations with real time data logging and reporting functionality.

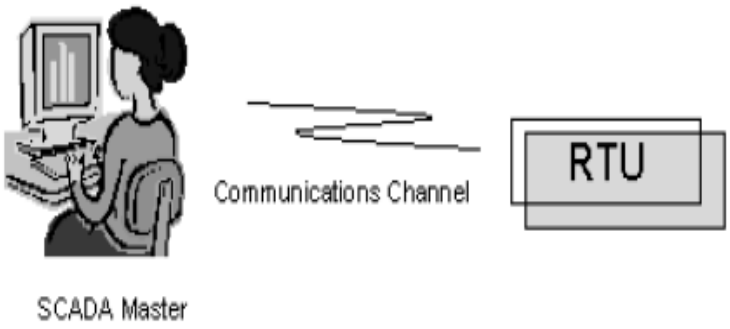


Figure 1: SCADA System.

1.2 Common system components

A SCADA System usually consists of the following subsystems:

- A Human-Machine Interface or HMI is the apparatus which presents process data to a human operator, and through this, the human operator monitors and controls the process.
- A supervisory (computer) system, gathering (acquiring) data on the process and sending commands (control) to the process.
- Remote Terminal Units [2] (RTUs) connecting to sensors in the process, converting sensor signals to digital data and sending digital data to the supervisory system.
- Programmable Logic Controller [3] (PLCs) used as field devices because they are more economical, versatile, flexible, and configurable than special-purpose RTUs.
- Communication infrastructure connecting the supervisory system to the Remote Terminal Units.

1.3 Supervision vs. control

There is, in several industries, considerable confusion over the differences between SCADA systems and distributed control systems (DCS). Generally speaking, a SCADA system always refers to a system that *coordinates*, but does not *control* processes in real time. The discussion on real-time control is muddied somewhat by newer telecommunications technology, enabling reliable, low latency, high speed communications over wide areas. Most differences between SCADA and DCS are culturally determined and can usually be ignored. As communication infrastructures with higher capacity become available, the difference between SCADA and DCS will fade.

1.4 Systems concepts

The term SCADA usually refers to centralized systems which monitor and control entire sites, or complexes of systems spread out over large areas (anything from an industrial plant to a nation). Most control actions are performed automatically by Remote Terminal Units ("RTUs") [2] or by programmable logic controllers ("PLCs") [3]. Host control functions are usually restricted to basic overriding or *supervisory*

level intervention. For example, a PLC may control the flow of cooling water through part of an industrial process, but the SCADA system may allow operators to change the set points for the flow, and enable alarm conditions, such as loss of flow and high temperature, to be displayed and recorded. The feedback control loop passes through the RTU or PLC, while the SCADA system monitors the overall performance of the loop.

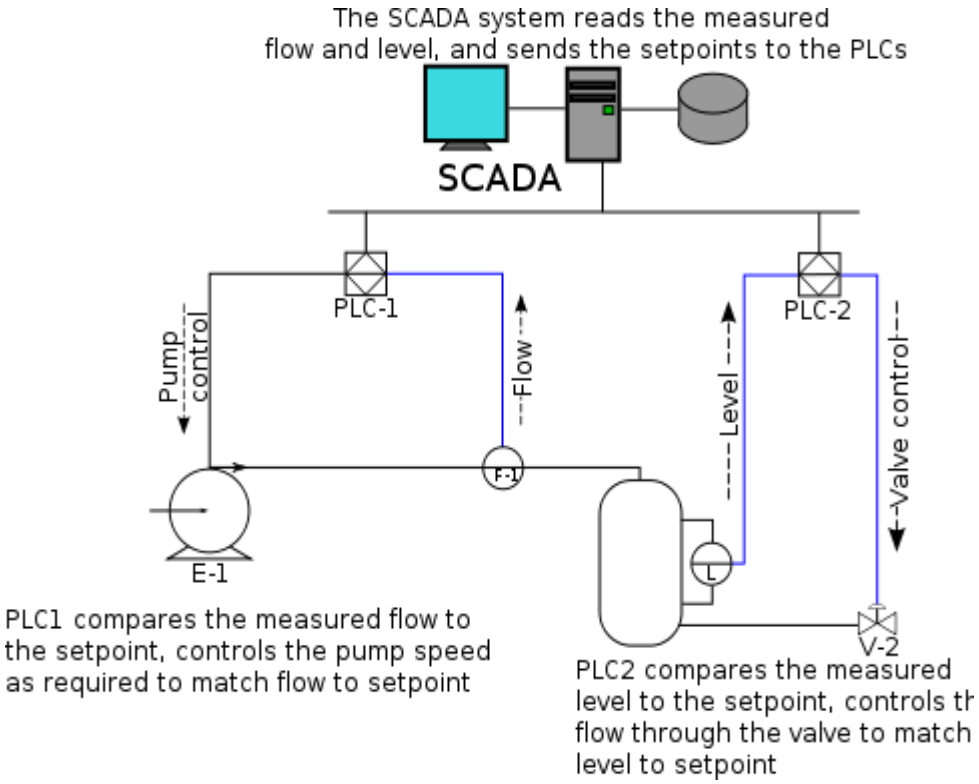


Figure 1.2 SCADA Functioning [1]

Data acquisition begins at the RTU or PLC level and includes meter readings and equipment status reports that are communicated to SCADA as required. Data is then compiled and formatted in such a way that a control room operator using the HMI can make supervisory decisions to adjust or override normal RTU (PLC) controls. Data may also be fed to a Historian, often built on a commodity database management system, to allow trending and other analytical auditing.

SCADA systems typically implement a distributed database, commonly referred to as a *tag database*, which contains data elements called *tags* or *points*. A point represents a single input or output value monitored or controlled by the system. Points can be either "hard" or "soft". A hard point represents an actual input or output within the system, while a soft point results from logic and math operations applied to other

points. (Most implementations conceptually remove the distinction by making every property a "soft" point expression, which may, in the simplest case, equal a single hard point.) Points are normally stored as value-timestamp pairs: a value and the timestamp when it was recorded or calculated. A series of value-timestamp pairs gives the history of that point. It's also common to store additional metadata with tags, such as the path to a field device or PLC register, design time comments, and alarm information.

1.5 Human Machine Interface (HMI)

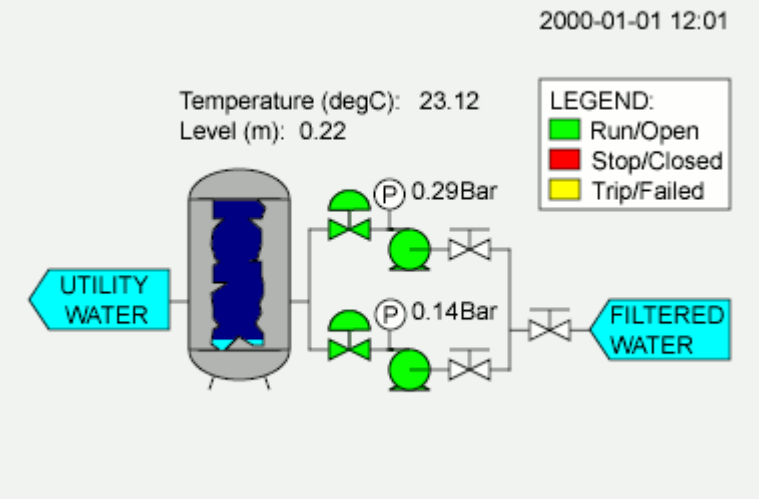


Figure 1.3 Typical basic SCADA animations [1]

A Human-Machine Interface or HMI is the apparatus which presents process data to a human operator, and through which the human operator controls the process.

An HMI is usually linked to the SCADA system's databases and software programs, to provide trending, diagnostic data, and management information such as scheduled maintenance procedures, logistic information, detailed schematics for a particular sensor or machine, and expert-system troubleshooting guides.

The HMI system usually presents the information to the operating personnel graphically, in the form of a mimic diagram. This means that the operator can see a schematic representation of the plant being controlled. For example, a picture of a pump connected to a pipe can show the operator that the pump is running and how much fluid it is pumping through the pipe at the moment. The operator can then switch the pump off. The HMI software will show the flow rate of the fluid in the pipe decrease in real time. Mimic diagrams may consist of line graphics and schematic

symbols to represent process elements, or may consist of digital photographs of the process equipment overlain with animated symbols.

The HMI package for the SCADA system typically includes a drawing program that the operators or system maintenance personnel use to change the way these points are represented in the interface. These representations can be as simple as an on-screen traffic light, which represents the state of an actual traffic light in the field, or as complex as a multi-projector display representing the position of all of the elevators in a skyscraper or all of the trains on a railway.

An important part of most SCADA implementations is alarm handling. The system monitors whether certain alarm conditions are satisfied, to determine when an alarm event has occurred. Once an alarm event has been detected, one or more actions are taken (such as the activation of one or more alarm indicators, and perhaps the generation of email or text messages so that management or remote SCADA operators are informed). In many cases, a SCADA operator may have to acknowledge the alarm event; this may deactivate some alarm indicators, whereas other indicators remain active until the alarm conditions are cleared. Alarm conditions can be explicit - for example, an alarm point is a digital status point that has either the value NORMAL or ALARM that is calculated by a formula based on the values in other analogue and digital points - or implicit: the SCADA system might automatically monitor whether the value in an analogue point lies outside high and low limit values associated with that point. Examples of alarm indicators include a siren, a pop-up box on a screen, or a colored or flashing area on a screen (that might act in a similar way to the "fuel tank empty" light in a car); in each case, the role of the alarm indicator is to draw the operator's attention to the part of the system 'in alarm' so that appropriate action can be taken. In designing SCADA systems, care is needed in coping with a cascade of alarm events occurring in a short time, otherwise the underlying cause (which might not be the earliest event detected) may get lost in the noise. Unfortunately, when used as a noun, the word 'alarm' is used rather loosely in the industry; thus, depending on context it might mean an alarm point, an alarm indicator, or an alarm event.

1.6 Hardware solution

SCADA solutions often have Distributed Control System (DCS) components. Use of "smart" RTUs or PLCs, which are capable of autonomously executing simple logic processes without involving the master computer, is increasing. A standardized

control programming language, IEC 61131-3 (a suite of 5 programming languages including function block, ladder, structured text, sequence function charts and instruction list), is frequently used to create programs which run on these RTUs and PLCs. Unlike a procedural language such as the C programming language or FORTRAN, IEC 61131-3 has minimal training requirements by virtue of resembling historic physical control arrays. This allows SCADA system engineers to perform both the design and implementation of a program to be executed on an RTU or PLC.
A Programmable automation controller (PAC) is a compact controller that combines the features and capabilities of a PC-based control system with that of a typical PLC. PACs are deployed in SCADA systems to provide RTU and PLC functions. In many electrical substation SCADA applications, "distributed RTUs" use information processors or station computers to communicate with digital protective relays, PACs, and other devices for I/O, and communicate with the SCADA master in lieu of a traditional RTU.
Since about 1998, virtually all major PLC manufacturers have offered integrated HMI/SCADA systems, many of them using open and non-proprietary communications protocols. Numerous specialized third-party HMI/SCADA packages, offering built-in compatibility with most major PLCs, have also entered the market, allowing mechanical engineers, electrical engineers and technicians to configure HMIs themselves, without the need for a custom-made program written by a software developer.

1.6.1 Remote Terminal Unit (RTU)

The RTU connects to physical equipment. Typically, an RTU converts the electrical signals from the equipment to digital values such as the open/closed status from a switch or a valve, or measurements such as pressure, flow, voltage or current. By converting and sending these electrical signals out to equipment the RTU can control equipment, such as opening or closing a switch or a valve, or setting the speed of a pump. It also controls the flow of the liquid.

1.6.2 Supervisory Station

The term "Supervisory Station" refers to the servers and software responsible for communicating with the field equipment (RTUs, PLCs, etc), and then to the HMI software running on workstations in the control room, or elsewhere. In smaller

SCADA systems, the master station may be composed of a single PC. In larger SCADA systems, the master station may include multiple servers, distributed software applications, and disaster recovery sites. To increase the integrity of the system the multiple servers will often be configured in a dual-redundant or hot-standby formation providing continuous control and monitoring in the event of a server failure.

1.7 Operational philosophy

For some installations, the costs that would result from the control system failing are extremely high. Possibly even lives could be lost. Hardware for some SCADA systems is ruggedized to withstand temperature, vibration, and voltage extremes, but in most critical installations reliability is enhanced by having redundant hardware and communications channels, up to the point of having multiple fully equipped control centers. A failing part can be quickly identified and its functionality automatically taken over by backup hardware. A failed part can often be replaced without interrupting the process. The reliability of such systems can be calculated statistically and is stated as the mean time to failure, which is a variant of mean time between failures. The calculated mean time to failure of such high reliability systems can be on the order of centuries.

1.8 Communication infrastructure and methods [4]

SCADA systems have traditionally used combinations of radio and direct serial or modem connections to meet communication requirements, although ethernet and IP over SONET / SDH is also frequently used at large sites such as railways and power stations. The remote management or monitoring function of a SCADA system is often referred to as telemetry.

This has also come under threat with some customers wanting SCADA data to travel over their pre-established corporate networks or to share the network with other applications. The legacy of the early low-bandwidth protocols remains, though. SCADA protocols are designed to be very compact and many are designed to send information to the master station only when the master station polls the RTU. Typical legacy SCADA protocols include Modbus RTU, RP-570, Profibus and Conitel. These communication protocols are all SCADA-vendor specific but are widely adopted and used. Standard protocols are IEC 60870-5-101 or 104, IEC 61850 and DNP3. These

communication protocols are standardized and recognized by all major SCADA vendors. Many of these protocols now contain extensions to operate over TCP/IP. Although some believe it is good security engineering practice to avoid connecting SCADA systems to the Internet so the attack surface is reduced, many industries, such as wastewater collection and water distribution, have used existing cellular networks to monitor their infrastructure along with internet portals for end-user data delivery and modification. This practice has been ongoing for many years with no known data breach incidents to date. Cellular network data is fully encrypted, using sophisticated encryption standards, before transmission and internet data transmission, over an "https" site, is highly secure.

RTUs and other automatic controller devices were being developed before the advent of industry wide standards for interoperability. The result is that developers and their management created a multitude of control protocols. Among the larger vendors, there was also the incentive to create their own protocol to "lock in" their customer base. A list of automation protocols is being compiled here.

Recently, OLE for Process Control (OPC) has become a widely accepted solution for intercommunicating different hardware and software, allowing communication even between devices originally not intended to be part of an industrial network.

1.9 SCADA architectures

The United States army's training manual 5-601 covers "SCADA Systems for C4ISR Facilities"

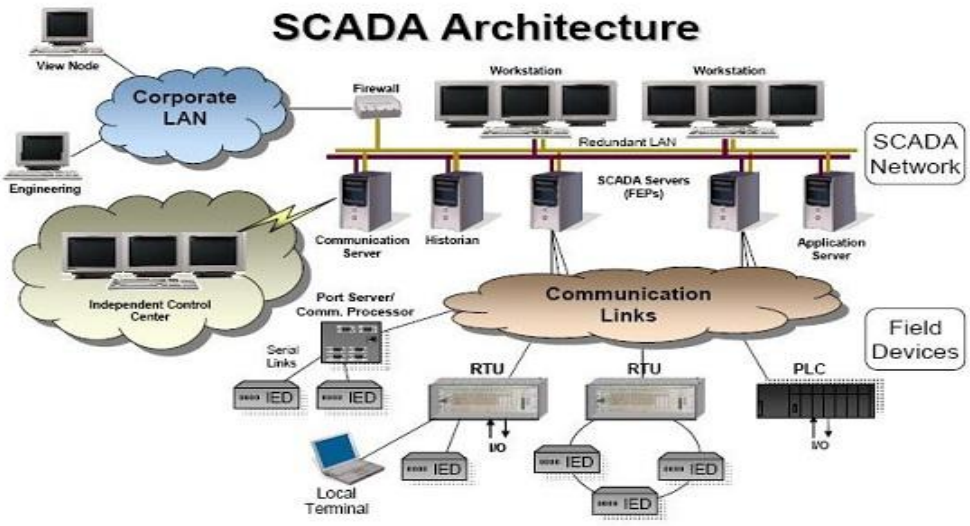


Figure 1.4 SCADA Architecture

SCADA systems have evolved through three generations as follows:

1.9.1 First generation: "Monolithic"

In the first generation, computing was done by mainframe computers. Networks did not exist at the time SCADA was developed. Thus SCADA systems were independent systems with no connectivity to other systems. Wide Area Networks(WAN) were later designed by RTU vendors to communicate with the RTU. The communication protocols used were often proprietary at that time. The first-generation SCADA system was redundant since a back-up mainframe system was connected at the bus level and was used in the event of failure of the primary mainframe system.

1.9.2 Second generation: "Distributed"

The processing was distributed across multiple stations which were connected through a LAN and they shared information in real time. Each station was responsible for a particular task thus making the size and cost of each station less than the one used in First Generation. The network protocols used were still mostly proprietary, which led to significant security problems for any SCADA system that received attention from a hacker. Since the protocols were proprietary, very few people beyond the developers and hackers knew enough to determine how secure a SCADA installation was. Since both parties had invested interests in keeping security issues quiet, the security of a SCADA installation was often badly overestimated, if it was considered at all.

1.9.3 Third generation: "Networked"

These are the current generation SCADA systems which use open system architecture rather than a vendor-controlled proprietary environment. The SCADA system utilizes open standards and protocols, thus distributing functionality across a WAN rather than a LAN. It is easier to connect third party peripheral devices like printers, disk drives, and tape drives due to the use of open architecture. WAN protocols such as Internet Protocol (IP) are used for communication between the master station and communications equipment. Due to the usage of standard protocols and the fact that many networked SCADA systems are accessible from the Internet; the systems are potentially vulnerable to remote cyber-attacks. On the other hand, the usage of

standard protocols and security techniques means that standard security improvements are applicable to the SCADA systems, assuming they receive timely maintenance and updates.

1.10 NETWORK MANAGEMENT [19]

In the rapid pace of growing technology, networks tend to be large and complex, filled with many types of equipment from different vendors. Managing such a network is increasingly more difficult, involving multiple management tools and protocols to support different proprietary devices on the network. Therefore, the network administrator needs a network management tool that can monitor the network's availability, utility and performance with the most industry-acceptable standards as possible to reduce the conflict of the network management system. Network management is the collection of tasks performed to maximize availability, performance, security and control of a network and its resources.

Typically, each managed device in the network includes an agent module responsible for collecting local management information and transmitting it to one or more management stations. Each management station includes network management application software plus software to communicate with agents. Information may be collected actively, by means of polling by the management station, or passively, by means of event reporting by the agents.

The International Organization for Standardization (ISO) network management forum has divided network management into five functional areas:

1. Fault Management

The objective of fault monitoring is to identify faults as quickly as possible after they occur and to identify the cause of the fault so that remedial action may be taken. Comprehensive fault management is the most important task in network management. Fault management tools can help increase the reliability of the network by quickly identifying the fault, isolating the cause of the fault, and then, if possible, correcting the fault.

2. Configuration Management

Configuration management deals with the initialization,

modification, and shutdown of a network. Networks are continually adjusted when devices are added, removed, reconfigured, or updated. The process of configuration management involves identifying the network components and their connections, collecting each device's configuration information, and defining the relationship between network components. In order to perform these tasks, the network manager needs topological information about the network, device configuration information, and control of the network component.

3. Performance Management

Performance management involves measuring the performance of a network and its resources in terms of utilization, throughput, error rates, and response times. With performance management information, a network manager can reduce or prevent network overcrowding and inaccessibility. This helps provide a more consistent level of service to users on the network, without overtaxing the capacity of devices and links.

4. Accounting Management

In the area of accounting management, network monitoring is concerned with gathering usage information to the level of detail required for proper accounting. This type of information helps a network manager allocate the right kind of resources to users, as well as plan for network growth. This type of management also involves monitoring access privileges and usage quotas of the users.

5. Security Management

Security management deals with ensuring overall security of the network, including protecting sensitive information through the control of access points to that information. Sensitive information is any data that an organization wants to secure, so protecting this sensitive data from unauthorized access is a common requirement. Security concerns can be assuaged with a well-designed and implemented security management system[5]. All these types of functional areas are instrumental to network management. However, the research in this thesis focuses on only two types of the functional areas: Fault Management and configuration management which are the basic utility of most network application.

Table 1.1 OSI Requirements Definition [5]

Fault management	The facilities that enable the detection, isolation and connection of abnormal operation of the environment.
Accounting management	The facilities that enable charges to be established for the use of managed objects, as well as costs to be identified for the use of those objects.
Configuration and name management	The facilities that exercise control over, identify, collect data from, and provide data to managed objects for the purpose of assisting the providing for continuous operation of interconnecting services.
Performance management	The facilities needed to evaluate the behavior of managed objects and the effectiveness of communication activities.
Security management	Addresses those aspects of OSI security essentially to operate with OSI network management correctly and to protect managed objects.

1.11 Network Monitoring [7] [8] [9]

It is one of the core components of network management and SCADA system which aids in monitoring and management of various network devices in SCADA system. SCADA system has integrated network diagnostics application. This application includes interactive interface which helps to visualize and control network in real time. Whether it's building-wide, campus-wide, or worldwide, it shows exactly what is going on with network from a single location.

The basic functionality required from Network Monitoring Module is [8]:

- Live Network Mapping
- Real Time Network Monitoring
- Application Monitoring
- Server Monitoring
- Wireless Equipment Monitoring
- Alerts and Notifications

The term **network monitoring** [7] describes the use of a system that constantly monitors a computer network for slow or failing components and that notifies the network administrator (via email, pager or other alarms) in case of outages. It is a subset of the functions involved in network management. While an intrusion detection system monitors a network for threats from the outside, a network monitoring system monitors the network for problems caused by overloaded and/or crashed servers, network connections or other devices. For example, to determine the status of a web server, monitoring software may periodically send an HTTP [9] request to fetch a page. For email servers, a test message might be sent through SMTP [9] and retrieved by IMAP or POP3 [9].

Commonly measured metrics are response time, availability and uptime, although both consistency and reliability metrics are starting to gain popularity. The widespread addition of WAN optimization devices is having an adverse effect on most network monitoring tools especially when it comes to measuring accurate end-to-end response time because they limit round trip visibility.

Status request failures such as when a connection cannot be established, it times-out, or the document or message cannot be retrieved usually produce an action from the monitoring system. These actions vary an alarm may be sent (via SMS, email, etc.) to the resident sysadmin, automatic failover systems may be activated to remove the troubled server from duty until it can be repaired, etc. Monitoring the performance of a network uplink is also known as network traffic measurement, and more software is listed there. Based on the techniques used for monitoring a network they can be classified as:

1.11.1 Passive monitoring [9] [10] [14]

It is a technique used to capture traffic from a network by generating a copy of the traffic, often from a span port or mirror port or via a network tap. Once the data (a stream of frames or packets) has been extracted, it can be used in many ways.

- It can be analyzed in a sniffer such as Wireshark.
- It can be examined for flows of traffic, providing information on "top talkers" in a network as well as TCP round-trip time.
- It can be reassembled according to an application's state machine into end-user activity (for example, into database queries, e-mail messages, and so on.) This

kind of technology is common in Real User Monitoring when applied to the http protocol in web applications.

- In some cases, http reassembly is further analyzed for web analytics

Passive monitoring [10] [13] can be very helpful in troubleshooting performance problems once they have occurred. Passive monitoring differs from synthetic monitoring in that it relies on actual inbound web traffic to take measurements, so problems can only be discovered after they have occurred. While initially viewed as competitive to synthetic monitoring approaches, most networking professionals now recognize that passive and synthetic monitoring are complementary.

1.11.2 Active Monitoring [11] [13] [14]

Active scanning is done through sending multiple probe requests and recording the probe responses. It is a technique relies upon data gathered from probe packets injected into the network. E.g. if ICMP [13] [14] echo packets are sent to a remote host either windows or linux, the reply packets accordingly are different and hence aiding in detection of remote host. A good example is OS fingerprinting.

1.11.2.1 Network tomography

Network tomography is an important area of network measurement, which deals with monitoring the health of various links in a network using end-to-end probes sent by agents located at vantage points in the network/Internet.

1.11.2.2 Route analytics

Route analytics is another important area of network measurement. It includes the methods, systems, algorithms and tools to monitor the routing posture of networks. Incorrect routing or routing issues cause undesirable performance degradation or downtime.

1.11.2.3 Various types of protocols

Website monitoring service can check HTTP pages, HTTPS, SNMP, FTP, SMTP, POP3, IMAP, DNS, SSH, TELNET, SSL, TCP, ICMP, SIP, UDP, media streaming and a range of other ports with a variety of check intervals ranging from every four hours to every one minute. Typically, most network monitoring services test your server anywhere between once-per-hour to once-per-minute.

1.11.2.4 Servers around the globe

Network monitoring services usually have a number of servers around the globe - for example in America, Europe, Asia, Australia and other locations. By having multiple servers in different geographic locations, a monitoring service can determine if a web server is available across different networks worldwide. The more the locations used, the more complete is the picture on network availability.

1.12 Network mapping [9] [15]

Network mapping is the study of the physical connectivity of networks. **Internet mapping** is the study of the physical connectivity of the internet. Network mapping often attempts to determine the servers and operating systems run on networks. It is not to be confused with the remote discovery of which characteristics a computer may possess (operating system, open ports, listening network services, etc), an activity which is called network enumerating and is more akin to penetration testing.

1.12.1 Large-scale mapping project

Images of some of the first attempts at a large scale map of the internet were produced by the internet mapping project and appeared in wired magazine. The maps produced by this project were based on the layer 3 or IP level connectivity of the internet (see OSI model), but there are different aspects of internet structure that have also been mapped.

More recent efforts to map the internet have been improved by more sophisticated methods, allowing them to make faster and more sensible maps. An example of such an effort is the OPTE project, which is attempting to develop a system capable of mapping the internet in a single day. The "map of the internet Project" [1] maps over 4 billion internet locations as cubes in 3D cyberspace. Users can add URLs as cubes and re-arrange objects on the map.

In early 2011 Canadian based ISP PEER hosting created their own map of the internet that depicts a graph of 19,869 autonomous system nodes connected by 44,344 connections. The sizing and layout of the autonomous systems was calculated based on their eigenvector centrality, which is a measure of how central to the network each autonomous system is. Graph theory can be used to better understand maps of the internet and to help choose between the many ways to visualize internet maps. Some projects have attempted to incorporate geographical data into their internet maps (for

example, to draw locations of routers and nodes on a map of the world), but others are only concerned with representing the more abstract structures of the internet, such as the allocation, structure, and purpose of IP space.

1.12.2 Enterprise network mapping

Many organizations create network maps of their network system. These maps can be made manually using simple tools such as Microsoft Visio, or the mapping process can be simplified by using tools that integrate auto network discovery with network mapping. Many of the vendors from the notable network mappers list enable you to customize the maps and include your own labels, add undiscoverable items and background images. Sophisticated mapping is used to help visualize the network and understand relationships between end devices and the transport layers that provide service. Items such as bottlenecks and root cause analysis can be easier to spot using these tools.

There are three main techniques used for network mapping: **SNMP** based approaches, **Active Probing** and **Route analytics.**

The SNMP based approach retrieves data from router and switch MIBs in order to build the network map. The active probing approach relies on a series of traceroute-like probe packets in order to build the network map. The route analytics approach relies on information from the routing protocols to build the network map. Each of the three approaches has advantages and disadvantages in the methods that they use.

Network topology information can be valuable in a variety of situations; it can be used for network administration (including fault-detecting and avoiding [5], network inventory and planning [5, 9], protocol and routing algorithm development [11], performance prediction and monitoring as well as accurate network simulation. From a network security perspective, topology information can find application in threat detection [1], network monitoring [9], network access control and forensic investigations.

Manual network mapping is becoming increasingly difficult [5] [9] due to the size and dynamic behavior of networks. Automatic topology discovery tools and algorithms will therefore play an important role in network security, management and administration.

Research efforts concerned with physical topology discovery have focused mainly on cooperative network environments where it is assumed that network elements are intelligent and can be queried for topology related information.

CHAPTER 2

LITERATURE SURVEY

Efficient and cost-effective measurement of network characteristics is pivotal for distributed systems deployed on the Internet. The network characteristics are utilized by Internet-based distributed systems to provide better service to the user and enhance performance for the application.

Various papers have been proposed in past in the field of Network Management and monitoring relying on the use of SNMP, ICMP/trace route [17] utilities. SNMP along with the host status also provides other relevant information which helps in estimation of bandwidth usage and network congestion.

2.1 Network characteristics relevant for measurements/monitoring [18]

The significance and relevance of network characteristics varies with applications. A content distribution network may be interested in measuring latency from its clients to the application server, whereas a load balancing application may find available bandwidth as a useful criterion to adjust the load on the servers. Other network characteristics such as congestion, queuing delay, network failure, topology discovery, and bottleneck determination also have great significance. However, there are four network characteristics that constitute the basic paradigm of network measurements and could be utilized in the computation of other related network characteristics. They are mentioned below:

- ***Latency*:** Latency refers to the time taken by the message to traverse from the sending host to the destination host. It is usually measured in milliseconds (ms). Latency of a path has a great significance for many applications. For many applications latency is used to determine the availability of the servers and select nearest available server in terms of latency, specifically in the content distribution networks. Further, variation in latency (i.e. the deviation between the successive latency measurements) is used to determine the quality of the path and detect network congestion.

- ***Packet Loss*:** If the rate of packets sent from the source host is not equal to the rate of Packets received at the destination host then the path experiences packet loss. Loss rate of a path is the rate (in percentage) at which packets are being lost while traversing through the network path. Lost packets affect the performance of the application as they are often required to be retransmitted. Even when the packets cannot be retransmitted (for example, such as the live transmission of audio or video streams) a high packet loss could lead to low application performance. Due to these reasons it is always desirable to select paths with low loss rate.

- ***Path Detection*:** When a message is sent across the Internet, it traverses through various intermediate routers to reach the destination. Path detection is the process of determining the actual path taken by the message in reaching from source to destination. Since there are many possibilities for a path from one host to another, path detection enables an application to determine the actual path taken by the message during transmission. Path changes are possible on the Internet, therefore path detection also facilitate in determining a change in the path between two nodes. It can also be used as a sanity check to determine or isolate network failure.

- ***Bandwidth*:** Bandwidth refers to the capacity of the path to transfer data in a unit time. It is measured in bits per second (bps). Bandwidth has great implication for multimedia applications. Such applications generally have high bandwidth requirements. If the available path has limited bandwidth than the multimedia application suffers degraded performance. Bandwidth detection tools can be used to infer the path quality and select appropriate path.

2.2 Internet mapping techniques

There are two prominent techniques used today to create Internet maps. The first works on the data plane of the Internet and is called active probing. It is used to infer internet topology based on router adjacencies. The second works on the control plane and infers autonomous system connectivity based on BGP data.

2.2.1 Active probing

This technique relies on trace route-like probing on the IP address space. These probes report back IP forwarding paths to the destination address. By combining these paths one can infer router level topology for a given POP. Active probing is advantageous in that the paths returned by probes constitute the actual forwarding path that data takes through networks. It is also more likely to find peering links between ISP's. However, active probing requires massive amounts of probes to map the entire Internet. It is more likely to infer false topologies due to load balancing routers and routers with multiple IP address aliases. Decreased global support for enhanced probing mechanisms such as source-route probing, ICMP echo broadcasting, and IP address resolution techniques leaves this type of probing in the realm of network diagnosis.

2.2.2 AS PATH inference

This technique relies on various BGP collectors who collect routing updates and tables and provide this information publicly. Each BGP entry contains a path vector attribute called the AS Path. This path represents an autonomous system forwarding path from a given origin for a given set of prefixes. These paths can be used to infer AS-level connectivity and in turn be used to build AS topology graphs. However, these paths do not necessarily reflect how data is actually forwarded and adjacencies between AS nodes only represent a policy relationship between them. A single AS link can in reality be several router links. It is also much harder to infer peering between two AS nodes as these peering relationships are only propagated to an ISP's customer networks. Nevertheless, support for this type of mapping is increasing as more and more ISP's offer to peer with public route collectors such as route-views and RIPE. New toolsets are emerging such as Cyclops and NetViews that take advantage of a new experimental BGP collector BGPMon. NetViews can not only build topology maps in seconds but visualize topology changes moments after occurring at the actual router. Hence, routing dynamics can be visualized in real time.

2.2.3 Active Measurements

A vast majority of prior Internet characteristic discovery research is dependent on *Active Measurements* [20] [21], which here we will specify as a tomographic network measurement performed by specifying a probe destination target from a set origin point in the network. The active measurement output will consist of some

characteristic of the network between the origin and destination (*e.g.*, delay, router topology, etc.). In this dissertation, we focus on two specific active measurement probes, ping and traceroute.

2.2.3.1 Ping Measurements

The most basic active probe considered in this dissertation is SNMP and ICMP ping probes [16]. Using an ICMP echo request packet, the host origin computer will send a probe to a targeted destination end host, returning both the round trip time latency (RTT) in milliseconds and the time-to-live (TTL) value, indicating the number of routers between the host computer and the targeted end host. Advantages of ping measurements are that the probes are lightweight with very little load on the network path, while the main disadvantage is that no further useful topology characteristics (shared path lengths of two network routes, specific routers traversed by a path, etc.) are returned by individual ICMP ping measurements.

2.2.3.2 Traceroute Measurements

In prior research (*e.g.*, [22][23]), the predominant measurement for acquiring Internet topology characteristics has been based on tools similar to traceroute probes to gather data. Standard traceroute probes further exploit ICMP packets to return both the number of routers (*i.e.*, the hop count between two points in the network), and the set of router interface IP addresses along the path between the two probe points in the network. This probing methodology allows for routing adjacencies to be known along the path between the two probe points (*e.g.*, router A is physically connected to router B). In addition, the router interface IP addresses allow for domain name server (DNS) requests for further information about each router along the observed path. This technique, referred to as unDNS [23], creates location hints that have been used frequently on the problem of estimating an end host's geographic location. Unfortunately, effective use of such hints requires significant and frequently updated databases which still introduce the possibility of errors. Great strides have been made in mitigating the problems associated with active probe Internet measurements, such as interface disambiguation in, which is the problem of resolving multiple IP addresses associated with a single physical router. This has enabled accurate mapping of ISP topologies (*e.g.*, [2]) and of the Internet's core (*e.g.*, [17]) using traceroute probes. However, there are still three important limitations in the use of active probing tools for Internet characteristic discovery. First, the vast size of the Internet means that

a set of measurement hosts M and target hosts N where N >> M must be established in order for the resultant measurements to capture the diverse features of the infrastructure (especially on the edges of the network). Second, active probes sent from monitors to the large set of target hosts result in a significant traffic load on the network. Third, in order to prevent reverse engineering of networks, service providers frequently attempt to thwart structure discovery by blocking ICMP probes to specific routers (and thus blocking both traceroute and ping probes). This results in the acquisition of incomplete active measurements due to *(i)* - the inability to perform exhaustive probing of all objects in the network in a reasonable length of time, and/or *(ii)* - the obfuscation of critical network infrastructure by administrators to avoid reverse engineering.

2.2.4 Passive Measurements

One alternative to the use of active probes is to acquire network information passively, where instead of introducing probe-based traffic into the network we instead measure existing Internet traffic. The methodology we will consider in this dissertation consists of a series of monitors on network links sampling traffic. From passively sampled packets, we can obtain the IP address and Time-To-Live (TTL) count off the packet header. At the origin of each packet, it is assigned an operating system dependent integer value (i.e. 64, 128, or 255). As the data packet traverses the network, the TTL count is decremented by a single count at each router encountered. When the TTL count reaches zero, the packet is discarded, thus preventing packets from forever traversing the network. Using the technique from [24], the TTL count can be translated into the number of routers between the end host and the passive monitor. It is not uncommon to observe packets from a single end host source at several of the passive monitors, resulting in a vector of hop-count distances from each monitor to that source. These vectors provide an indication of the topological location of the source relative to the monitors, with no additional load added to the network by measurement probes. This can be due either to packet sampling restrictions at our monitors (where only a subset of traffic will be observed) or an end host not directing any traffic towards the locations of specific monitors. Due to the inherently incomplete nature of passive measurements, there is relatively little prior work that addresses passive network monitoring.

An example of the three Internet measurement types considered in this dissertation is shown in Figure 2.1.

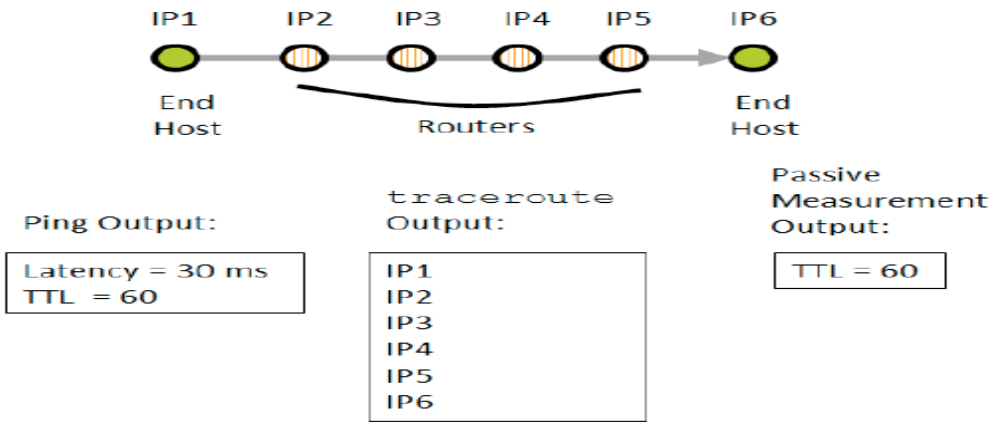


Figure 2.1 Example of the three Internet measurement types (Ping, traceroute, and passive where the monitor is at IP6) between two points in the network, where a time-to-live (TTL) value of 60 indicates that there are 64 − 60 = 4 routers between the two end hosts in the network.

2.3 Simple Network Management Protocol (SNMP)

2.3.1 Foundation

In the early years of small networks (roughly until the mid-eighties), small applications as ping and traceroute were powerful enough to provide basic management functionality. But with the exponential growth of networks since the late eighties, the need arose for a management protocol with much more functionality. Several approaches were evaluated and finally SNMP was selected as a short-term solution, because of its simplicity. In the long-term it was thought to make way for a different, more elaborate management protocol which was to be part of the OSI model [29].

Being a part of the TCP/IP suite, SNMP followed a similar development as TCP/IP. Both were thought to be simple and short-term solutions, as they would in the future be replaced by the OSI standards. However, since they experienced a vast deployment in rapid growing networks, both protocols out-lasted their lifetimes by far. In fact, they are still widely used nowadays and the OSI models remain reference models. To date, almost all vendors of computers, bridges, routers, etc. offer SNMP support for their products.

SNMPv1 was released in 1989 followed by a proposal for version SNMPv2

in1993 and a revision of this version in 1995. Then in 1998 SNMPv3 was issued, which experienced a big focus shift to security. It extends both SNMPv1 and SNMPv2. All of these versions are extensions of the following three foundation specifications [26]:

- Structure and identification of management information for TCP/IP- based networks (RFC 1155 [30]).
- Management information base for network management of TCP/IP-based Internets: MIB-II (RFC 1213 [31]).
- Simple Network Management Protocol (RFC 1157 [32]).

2.3.2 Architecture

The SNMP network management architecture makes a clear distinction between the roles of SNMP-enabled networked devices and systems. A management agent is a device or system that is being managed and a management station is a system from which agents are managed. A management station is sometimes also referred to as a manager.

The architecture also defines a management protocol that provides the link between the managers and agents. Management information itself is standardized and defined in the form of Management Information Bases (MIB). If an agent is said to support a certain MIB, the manager consequently knows how to address the agent in order to access information defined in this MIB. The overall SNMP network management architecture is depicted in figure 2.2.

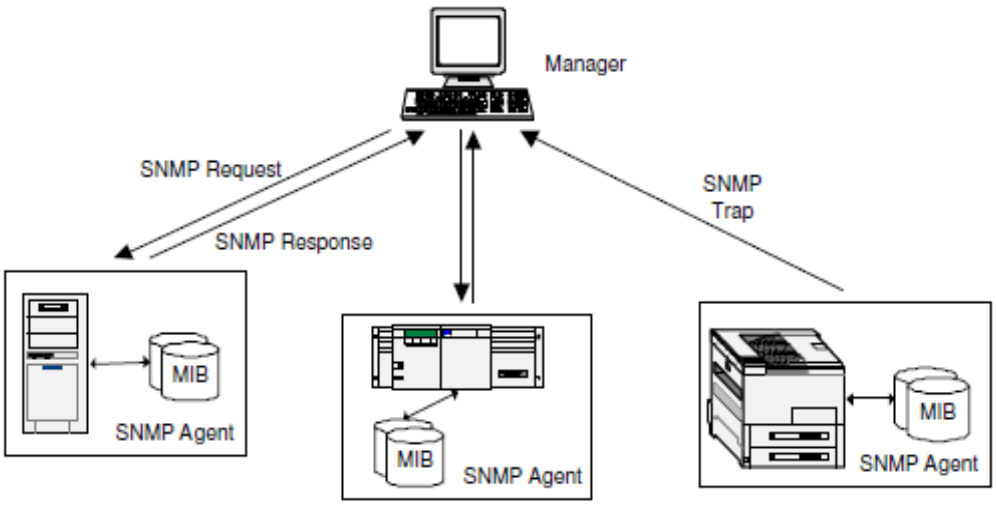


Figure 2.2 SNMP basic operation

Many networked devices, such as PC's, routers, printers, hubs, switches, etc. can contain an SNMP agent. A manager is the interface for a human network operator to monitor and configure these networked devices. Therefore mostly a PC or workstation contains a manager, since these are able to present a (graphical) user interface to the operator. These systems itself can also contain an agent, allowing a management application to also manage the system it is running on. Typical applications on a manager include data analysis and fault recovery. A manager could for instance also be connected to a database system, in order to periodically store management information for statistical purposes.
A management agent in its turn is able to retrieve the actual data from the device it is running on. An agent either waits for requests from a manager, or it can initiate action by sending a so-called trap to the manager. Agents and managers are able to communicate with each other by means of SNMP primitives, as defined by the SNMP protocol. The key capabilities that these primitives offer are to get and set management information from an agent by a manager, and to send trap notifications from an agent to a manager. Traps are used to inform a manager of unusual events that have occurred on the agent-side, such as a reboot after a crash of the system, a link that is down or some pre-defined condition that is fulfilled. For instance, if the agent notices that the percentage of TCP error packets of the total amount of TCP packets is above a certain level, it can notify the manager by means of a trap.

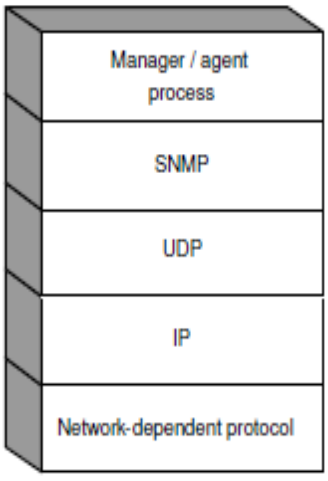


Figure 2.3 SNMP network stack

SNMP is designed to be a part of the TCP/IP protocol suite. Its intended use is on top of the User Datagram Protocol (UDP) [27], because UDP is connectionless. This would allow a management application to be in full control of retransmission strategies, in case connections are lost or congestions have occurred [28]. Also it does not have a lot of protocol overhead. The datagram headers that need to be created are very small, compared to TCP for instance, which limits the size of the datagram . Within the IETF, there has been an attempt to use SNMP on top of TCP [33], but this is still experimental. Figure 2.3 shows examples of the SNMP protocol stack which should be existent on both a manager and an agent. Each manager and each agent must at least implement IP, UDP and SNMP. This excludes any networked device that does not implement the TCP/IP stack from being managed by SNMP. There are provisions however, to create so-called proxy agents in order to translate SNMP requests to a different management protocol and vice versa. The SNMP agent in this case maintains the management information on behalf of one or more non-SNMP devices.

2.3.3 Management Information Base (MIBs)

Resources in a networked device are defined as managed objects. This concept of an object should not be confused with the concept that is commonly known from object-oriented programming. These are not the same. In fact, a managed object is merely a data variable representing one aspect of a resource. For example, suppose a router is a resource, then the system uptime would be one aspect of the router. Thus “system uptime" could be called a managed object in SNMP terminology.

A collection of managed objects that are in some way related to each other are grouped together in a structured format called a Management Information Base, also called a MIB module or a MIB. A formal definition of a MIB module is the following [34]:

"MIBs are specifications containing definitions of management information so that networked systems can be remotely monitored, configured and controlled."

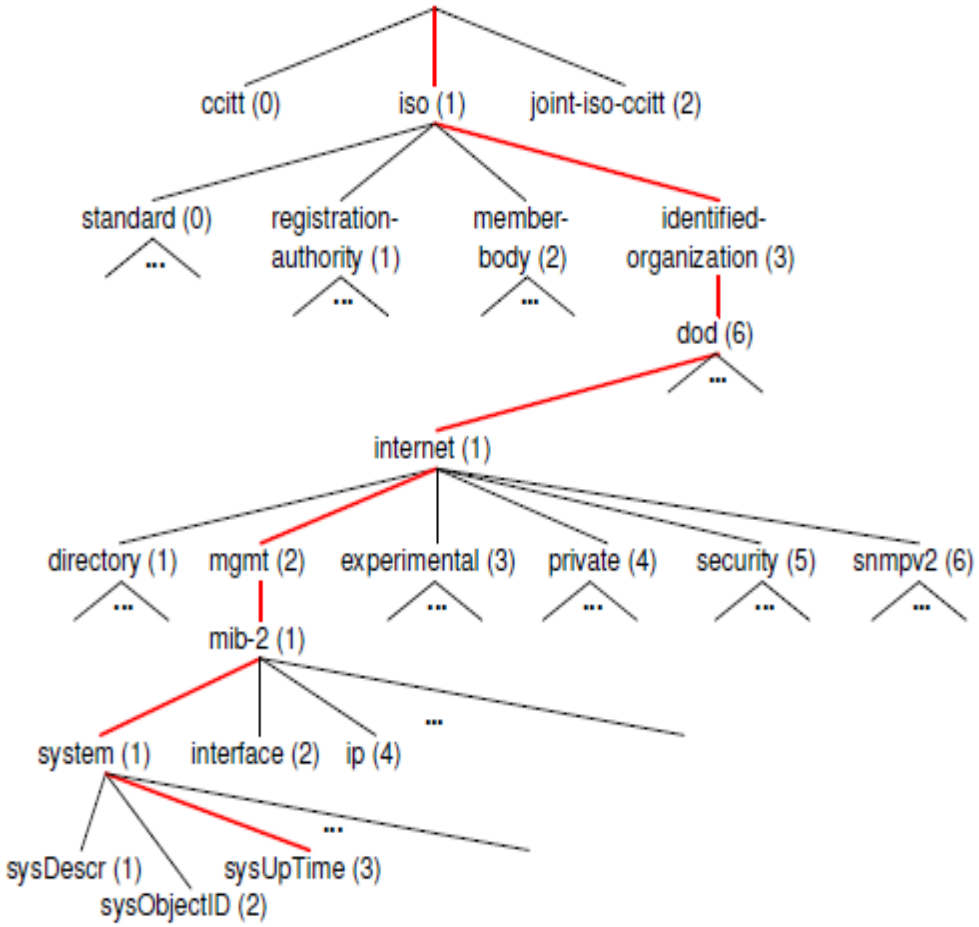


Figure 2.4 SNMP naming tree

The main objective of a MIB is to enhance interoperability across networked devices. One way this is accomplished is to have managed objects representing a particular resource the same at each system. For instance, a managed object representing the system uptime on one system should have the same name and function on another system. It would be a great cause of confusion when, for example, one system would regard it as the time elapsed since the last reboot, and another system regards it as the time elapsed since the operating system was installed regardless of any reboots. Such a situation is not feasible of course.

A second way to enhance interoperability is to have a common definition language for the representation of MIBs: the Structure of Management Information (SMI) [30]. SMI is based on the ASN.1 notation, but uses only a very small subset of it for the sake of simplicity. SMI identifies only several basic data types and specifies how resources are represented and named. New types can

be defined based on the basic types, but they can only be either scalars (based on integer, octet string, null or object identifier) or two-dimensional arrays of scalars (with sequence and sequence of). This rule out all possibilities for more complex data structures.

Managed objects are arranged hierarchically in a tree structure, where each leaf represents a managed object. All nodes and leafs in the tree are given a permanent number, so that each managed object can be uniquely identified on a single networked device by a sequence of these numbers: the object identifier (OID). This OID is defined in a MIB where the corresponding managed object is also defined. An example of (a part of) the SNMP naming tree is given in figure 2.4.

If for instance someone wants to retrieve the system uptime of a certain networked device, it has to provide the corresponding OID to the get primitive. The OID represents the place of the system uptime variable in the naming tree. Starting from the root, the following path should be followed to reach this

ifTable		ifIndex	ifDescr	ifType	ifMtu	ifSpeed	...
	ifEntry						
	ifEntry						
	ifEntry						
	...						

Figure 2.5 Conceptual table: ifTable (Interfaces MIB)

Variable:

iso → org → dod → internet → mgmt → mib-2 → system → sysUpTime

When the names of the nodes are replaced by the node number, the following OID is retrieved:

1.3.6.1.2.1.1.3

If this OID is provided with the SNMP get primitive, the addressed agent retrieves this value from the system it is running on and sends it to the manager in a response message.

Two-dimensional arrays are a very simple way of structuring data. It allows for the creation of conceptual tables: they appear as tables, but they can only be addressed cell by cell. Such a table consists of instances of a certain row object-

type. For example, the Interfaces MIB [41] defines an ifEntry object-type. ifEntry itself is a sequence of scalar values, which are in essence instances of simple object-types. In pseudo-code:

IfEntry: = SEQUENCE {ifIndex, ifDescr, ifType, ifMtu, ifSpeed ...}

The ifTable then is a sequence of instances of ifEntry. In pseudo-code:

IfTable: = SEQUENCE OF IfEntry

This creates a conceptual table, as depicted in figure 2.5. Nesting of tables is not allowed, i.e. an element of a table (or of ifEntry in the example) can itself not be another table. This would make a MIB overly complex and thus is chosen to allow restrictions of this kind in SMI.

2.3.4 Standard MIB variables encoding

The management application is using a special format for encoding queries and for decoding the answers from the agents, which are the *Basic Encoding Rules (BER)*.

Perkins mentions, that although the Basic Encoding Rules are not the very most compact and efficient way to encode values, BER is by CCIT and ISO standardized and a widely used scheme.

2.3.5 MIBv2

MIBv2 is the second version of the management information base, which is defined in this RFC1213 [31]. It is a superset of the first approach with additional objects and groups, and it is always available in every MIB.

Figure 2.6 lists the MIB-2 objects, which are not the most important ones for systems management, but which provide a basic management capability. The below sections explain these groups briefly:

- **System Group:** The system group gives an overview about the managed system. It provides a description address, how long this system is already running, who is responsible for it, and at which network layer (in the OSI layer model) this device provides its services.
- **Interfaces Group: The** interfaces group delivers generic information about the physical interfaces, the configuration and statistics about inbound and outbound traffic.
- **Address Translation Group:** The address translation group is built up

with a single table that provides the IP address and the MAC address of every device that this computer holds in its ARP cache. This is very useful for reports, that operate on the media access control layer, or for any application that needs to find a connection between the *MAC* and the IP address.

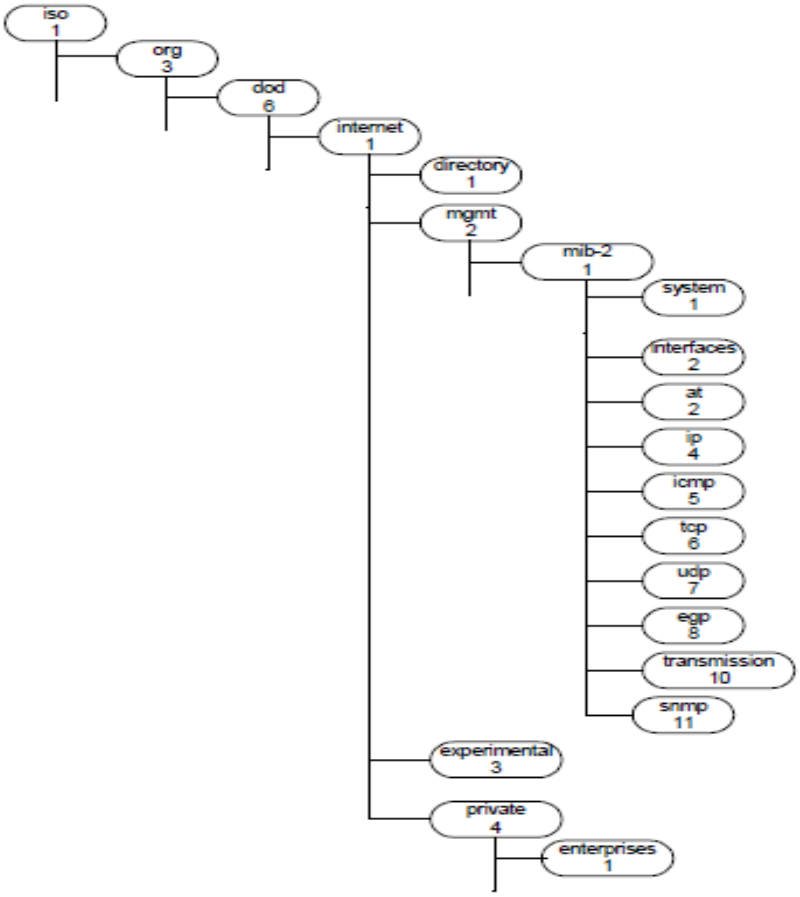


Figure 2.6 Standard MIB groups

- **IP Group:** The IP group provides information and statistics about the IP implementation at a node, about routing tables and failures.
- **ICMP Group:** The ICMP group provides information and feedback about communication problems.
- **TCP Group:** The TCP group provides information about the TCP stack in a device. It derives the open connections at this moment or information about TCP specific indicators.
- **UDP Group:** The UDP group provides information about ports that accept UDP datagram, about encountered failures and errors.
- **EGP group:** The EGP group provides information about the

external gateway protocol.

- **Transmission group:** This group provides information about the transmission medium for each interface, i.e. Token Ring or Ethernet. Actually this is not a group on its own, but a container, that allows access to some interface specific objects.
- **SNMP Group:** The SNMP group provides information about the implementation of the device's specific SNMP stack. It allows the management station to do some basic configuration and provides some basic statistics.

2.3.6 Private MIBs

The structure and the design of the MIB tree is well known, which makes it easy to enhance this tree, as Figure 8 shows. Every vendor can get its own private enterprise number, and snaps in at the enterprise tree at that point, defining its own subtree, with its own values, objects and structure. A list of the currently registered companies can be found at this webpage [43].

In the test environment for the *Watchdog* application we have also extended the standard MIB with private extensions. For *Windows-NT* we used an unsupported tool, which is provided by MICROSOFT. For the *Linux host* we used the CMU specific SNMP extension, which will be described later. *Windows-NT* is very widely used, but it provides no facilities for system management on the SNMP layer. On the other hand it provides a lot of performance monitoring tools, like the standard *Performance Monitor* or *HP Openview ManageX* which accesses the MICROSOFT system objects via the *Distributed Component Object Model (DCOM)*.

A lot of companies tried to get a solution for their network management applications, because DCOM is a proprietary windows mechanism, which does not work in a heterogeneous environment, so MICROSOFT was almost forced to establish a solution for that request. MICROSOFT provided in the *NT Resource Kit* a tool (*perf2mib.exe*) which allows accessing the DCOM system objects via SNMP. This tool allows to generate own MIB definition extension, by just ex- tending the standard *Windows-NT* MIB with the objects administrators are really interested in, and creates a MIB definition file in ASN.1 on its own. Unfortunately this tool is not supported and not many

companies know about the facilities. As explained prior, this utility enhances the standard SNMP service, which is provided with *Windows-NT*, with the system objects and performance indicators most administrators are interested in. The *CMU-SNMP daemon* which is provided with most of the *Linux* distributions does something similar, although it is not possible to define own performance objects that easily. This daemon enhances also the standard MIB, and allows to query system and performance variables like the processor load, processor type, average load and some more.

A lot of vendors extend the standard MIB by defining their own objects in the private subtree. Consider a company like CISCO, which has various devices like routers, switches and hubs. This company has to provide also the interesting values to the management applications, which would not be possible using only view standard objects. Large companies have a lot of different devices and models, with different objects. Therefore they must be able to define their own structure of their part in the tree, which is normally done in the private/enterprises MIB section.

2.3.7 SNMP protocol operation

In section 2.3.2 has already briefly mentioned that agents and managers communicate with each other by means of SNMP primitives. These primitives are also referred to as protocol operations. Each operation has a corresponding message structure to comprise any parameters necessary to fulfill the operation's goal.

This section will discuss the message structure of the operations defined for SNMPv2 (and SNMPv3), which is a superset of SNMPv1. The original specification can be found in RFC 3416 [42] and the corresponding SMI definitions in appendix A. SNMPv2 distinguishes several operations: get, get-next, get-bulk and set are used to retrieve or modify management information on an agent by a manager. This is done by exchanging request and response messages.

Furthermore, there is an operation trap that is used for agent to manager communication, usually in case some serious problem has occurred. And finally there is an operation inform for manager to manager communication, usually to exchange information between higher level applications.

Interaction between managers and agents takes place by the exchange of messages. Each SNMP message has a general structure. It consists of a version field (to denote the SNMP version), a community field (used for access control) and a PDU field. This is shown in figure 2.7(a). A PDU stands for a Protocol Data Unit and its internal structure is dependent on which operation is comprised in the SNMP message. In other words, each operation defines a PDU that holds the operation name and its parameters and/or values.

Version	community	SNMP PDU

(a) SNMPv2 message structure

PDU type	request-id	0	0	variable-bindings

(b) GetRequest-PDU, GetNextRequest-PDU, SetRequest-PDU, SNMPv2-Trap-PDU, InformRequest-PDU

PDU type	request-id	non-repeaters	max-repetition	variable-bindings

(c) GetBulkRequest-PDU

PDU type	request-id	error-status	error-index	variable-bindings

(d) Response-PDU

name_1	value_1	name_2	value_2	...	name_n	value_n

(e) Structure of variable-bindings

Figure 2.7 SNMP message and PDU formats [26]

Figure 2.8 shows the message interaction for each operation and shows the PDUs involved in each exchange. Even though PDUs can have different internal structures, they all share a PDU type field and a request-id field. The PDU type field is used to denote what kind of PDU structure follows, e.g. the

Response-PDU. In case there are multiple outstanding requests, there should be a mechanism to distinguish to which request an incoming response belongs. This is done by putting a uniquely defined request-id in the corresponding field of both a request PDU and its Response-PDU. This value should be the same for each request-response pair.

It is clear from figure 2.8 that operations that require a response message all use the same PDU for it. The structure of a Response-PDU is shown in figure 2.7(d). Besides the already mentioned PDU type and request-id fields, it has an error-status and an error-index field. A non-zero value for error-status means that an error has occurred and the value denotes the specific error, while the error-index contains the index of the variable in the variable-bindings list that caused the error. This will be explained later. Most PDUs involved in a request also share a similar structure.

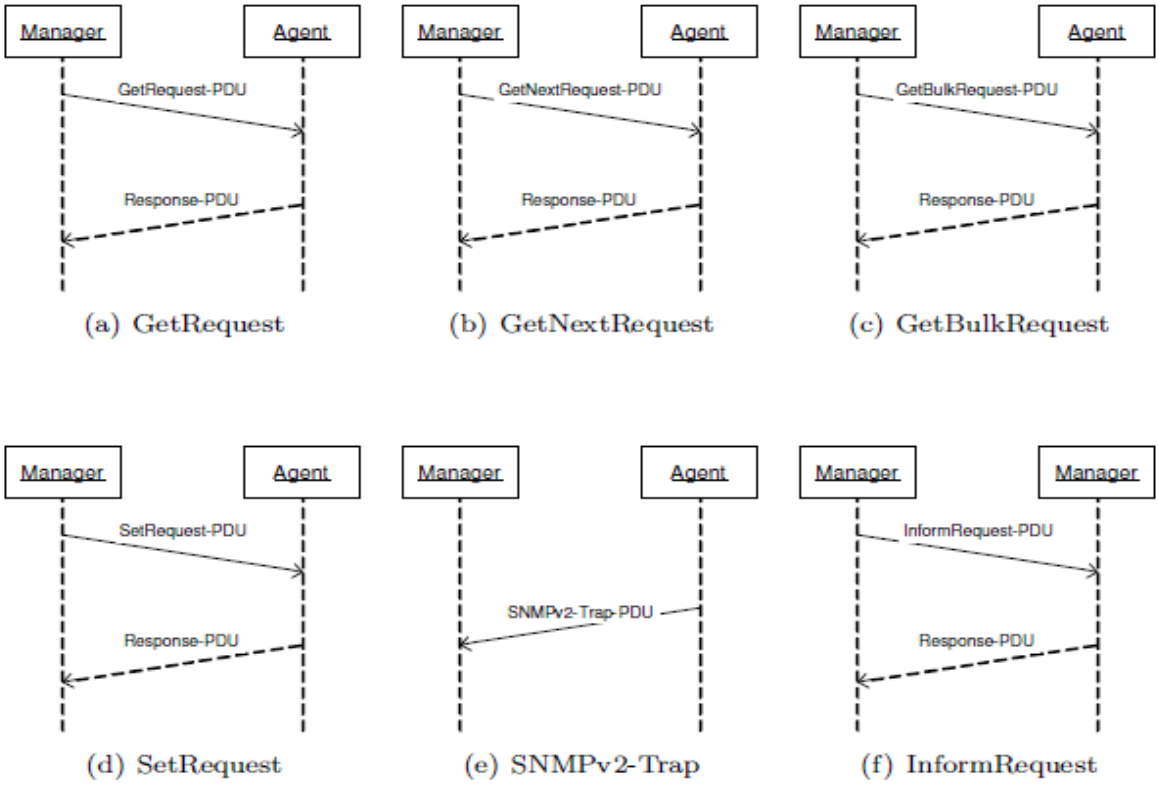


Figure 2.8 SNMPv2 PDU sequences [26]

2.7(b) shows the PDU format of the GetRequest-PDU, GetNextRequest-PDU, SetRequest-PDU, SNMPv2-Trap-PDU and the InformRequest-PDU. The fields that are used in the Response-PDU for error-status and error-index have a

value of 0 in a request PDU. The one exception is the GetBulkRequest-PDU. Instead of the two error fields, it has a non-repeaters and a max-repetitions field. The GetBulkRequest-PDU is added to SNMP to enhance the efficiency of the GetRequest-PDU in the way it retrieves large amounts of variables. More information on this can be found in [26].

The main difference between the PDUs used for a certain operation can be found in the contents of variable-bindings. Its structure is depicted in figure 2.7(e). The variable-bindings field is a list of name-value pairs, where each name is an object identifier. Depending on the type of PDU, the value is an object instance value, unspecified, noSuchObject, noSuchInstance or endOfMibView.

For example, a get request will only hold the names of the variables and each value field should in all cases be set to unspecified. The value fields should contain the values for each requested variable in the response (unless some error has occurred, then it is replaced by noSuchObject, noSuchInstance or endOfMibView). However, the set operation will contain the values already in its request PDU, since they will be used for modifying the values in an agent. The result of this operation is then put in the Response-PDU.

Finally, it is worth mentioning that the protocol operations for SNMPv2 specification (RFC 3416 [42]) defines one more operation: report. The corresponding PDU would be the Report-PDU, however the usage and semantics of this operation are not defined and therefore it will not be discussed in this thesis.

2.3.8 SNMP Security

Since SNMP lacks any authentication capabilities, this results in its vulnerability to a variety of security threats. These include masqueraded occurrences, modification of information, message sequence and timing modifications, and disclosure. Masqueraded occurrences consist of an unauthorized entity attempting to perform management operations by assuming the identity of an authorized management entity.

Modification of information involves an unauthorized entity attempting to alter a message generated by an authorized entity so that the message results in unauthorized accounting management or configuration management operations. Message sequence and timing modifications occur when an unauthorized entity reorders, delays, or copies and later replays a message generated by an authorized entity. Disclosure results when an unauthorized entity extracts values stored in managed object, or learns

of events by monitoring exchanges between managers and agents.

Therefore many vendors do not implement the "SET" operation because SNMP does not have enough authentication security, thereby reducing SNMP to a monitoring facility. However some features added to SNMPv3 including security functionality and access control even though does not widely use yet.

2.4 Analysis and comparison of available Network Monitoring Software

2.4.1 InterMapper [35]

- InterMapper is a cross-platform network monitoring program distributed by Dart ware, LLC.
- The current version of InterMapper is written in java (Agent Less).
- It includes variety of network probes based on ICMP, SNMP, http and other network protocols.
- It discovers network devices by probing a network, building a map of devices found in each network segment starting from a single IP address.
- InterMapper also supports alarms for devices that have disappeared from the network.
- The software can be configured to display graphs of performance data stored at variable intervals.

2.4.2 Pandora FMS [36]

- **Pandora FMS** stands for **Pandora F**lexible **M**onitoring **S**ystem.
- It is released by Artica under the terms of the GNU General Public License (GPL).
- Pandora FMS is free software and uses ICMP, SNMP based probes.
- It allows monitoring in a visual way the status and performance of several parameters.
- These parameters vary from different OS, servers, applications and hardware systems.
- It can obtain information from any OS or device, with specific agents for each platform to collect the data and send it to the server. (Agent Based).
- Trend Prediction

2.4.3 NetXMS [37]

- NetXMS is an open source network monitoring application software
- It is released by Raden Solutions under the terms of the GNU General Public License (GPL)
- It can be used for monitoring entire IT infrastructures, starting with SNMP-capable hardware and ending with applications on servers
- Unified platform for management and monitoring of the entire IT infrastructure
- Native support for many popular platforms and operating systems
- Distributed network monitoring
- Automated network discovery (Agent Based).
- Flexible and easy to use event processing

Some of the Common features which have been seen among these software's are as follows:

- Monitoring of network devices, servers and applications from one management server (SMTP, POP3, HTTP, ICMP, SNMP, FTP)
- Configuration and access to monitoring data take place through the user-friendly and customizable windows-based GUI
- The ability to send e-mails and SMS notifications or alarms
- Layer 3 IP topology auto-discovery
- Remote actions
- Flexible access control configuration

Based on above discussion and analysis of various features of these three softwares a comparison table is presented on next page.

Table 2.1 Comparison of features of Analyzed softwares

Features	Intermapper	NetXMS	Pandora FMS
Vendor	Dartware	Raden Solutions	Artica Solution Technologies
Supporting OS	MacOS X Server10.4, Windows 2000/XP/2003 /Vista/ Win Server 2008/Win 7,Linux, Solaris	Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Linux, Solaris, HP-UX	Windows 2000, XP, 2003, Vista, 2008, 7 (32/64bit), HP-UX, Solaris, Linux
Logical Grouping	Yes	Yes	Yes
Trending	Yes	Yes	Yes
Trend Prediction	No	No	Yes
Auto Discovery	Yes	Yes	Yes
Agent	Yes	Yes	Yes
SNMP support	Yes	Yes	Yes
ICMP support	Yes	Yes	Yes
System Log	Yes	Yes	Yes
Plugins	Yes	No	Yes
Triggers/Alerts	Yes	Yes	Yes
Web App	viewing	Full control	Full Control
Distributed Monitoring	Yes	Yes	Yes
Inventory	Yes	Yes	Yes
Database supported	PostgreSQL	MySQL, Oracle, PostgreSQL, MS SQL	MySQL
License	Limited free, Commercial	General Public License	General Public License
Graphical Maps	Yes	Yes	Yes
Access Control	Granular	Yes	Granular
IPv6 support	Yes	No	Yes
Customers	Google, Apple Computers	ABB SIA, CTXM	Forensics technology,

CHAPTER 3

PROBLEM STATEMENT

There are various network monitoring solutions that are available in market but most of them have a particular set of requirements to run and also many of them were agent based solutions. Thus there was a need to develop a network monitoring solution which was to be integration with Crompton Greaves's SCADA i.e.CrompScada. This solution was thought to be using standard management protocols i.e. SNMP, ICMP etc so that the final end product should not be vendor specific. This application would provide the end user with the logical view of the underline network and real time information of critical network devices such as RTU (Remote terminal Unit), IED (Intelligent Electronic Devices), PC, printers and reporting the faults in the form of alarms to the concerned person (i.e. in case some device goes down, a visual Remark like popup or alarm can be raised). This project was very challenging as it involves simultaneous handling of real time network topology discovery which involves graphics components to display the discovered topology in a user friendly manner, networking aspects i.e. using standard management protocols for network topology discovery and real-time monitoring and use of multithreading for dealing with performance issues. The following are few of the attributes:

1. Agent Less network monitoring and so can be deployed in any environment.
2. Detection of web and email servers and their status.
3. Alarm alerts in case of network criticality.
4. Real time network visualization.
5. Using SNMP & ICMP collaboratively to discover and monitor the discovered topology.
6. Integration with Cromp SCADA.
7. Provides other relevant information such as processor utilization of hosts, packet loss for links, management information like system description, up time, MAC address etc.
8. Easy to operate and fully user configurable.

CHAPTER 4

DESIGN AND IMPLEMENTATION

This section gives insight to the technicalities and the methods involved in the development of the required application. This section gives a description of various modules involved in processing network information and its simultaneous representation on GUI.

4.1 Design

The design of the network monitoring system was decided to be based on object oriented approach rather the traditional procedural programming paradigm, with a top-down approach. The reason for this was that there was a need to perceive every device as an object, each having unique properties and moreover they were required to respond uniquely to user actions. Many of the functions previously implemented on network mappers also used this approach. There was a need to store the network information in a format which was easy to understand. The designing of the discovery information template was done after an evaluation of the information that was desired, by studying the previous mentioned software and the requirement stated. It was clear from the beginning that the design had to flexible that allow future extensions, this was the reason, XML format was chosen for describing capabilities of individual hosts. Serialization technique was adopted so as to save all of the object/device data in XML format thus providing efficiency and scalability.

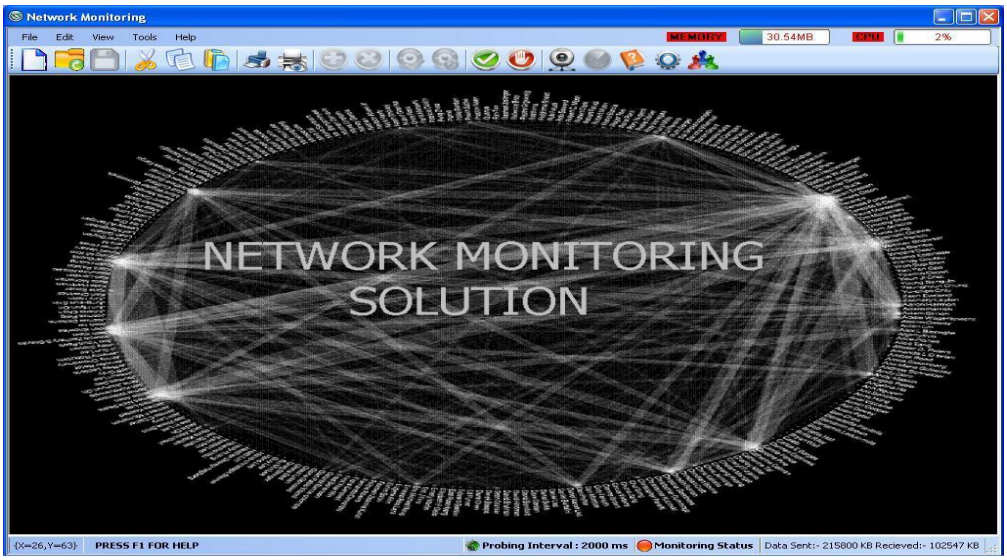


Figure 4.1 Network Monitoring Solution Main Window

4.1.1 Programming language

An evaluation of suitable programming languages was performed in order to decide which programming language best suits the needs for network monitoring application implementation. The evaluated languages consisted of Java, C++, C#.NET and PHP. At first glance Java or C++ was favorable, since I had a prior experience on these languages. Both these languages have support for object oriented design. Advantage of compiled C++ programs was that it usually runs very fast and efficient execution, while Java programs may be used on many different platforms. However, these benefits were not very relevant in the current situation, as current requirements for SCADA were based on windows systems and more over C#.NET provides comprehensive GDI+ library for developing graphics intensive applications and also .NET framework has inbuilt support for ICMP.

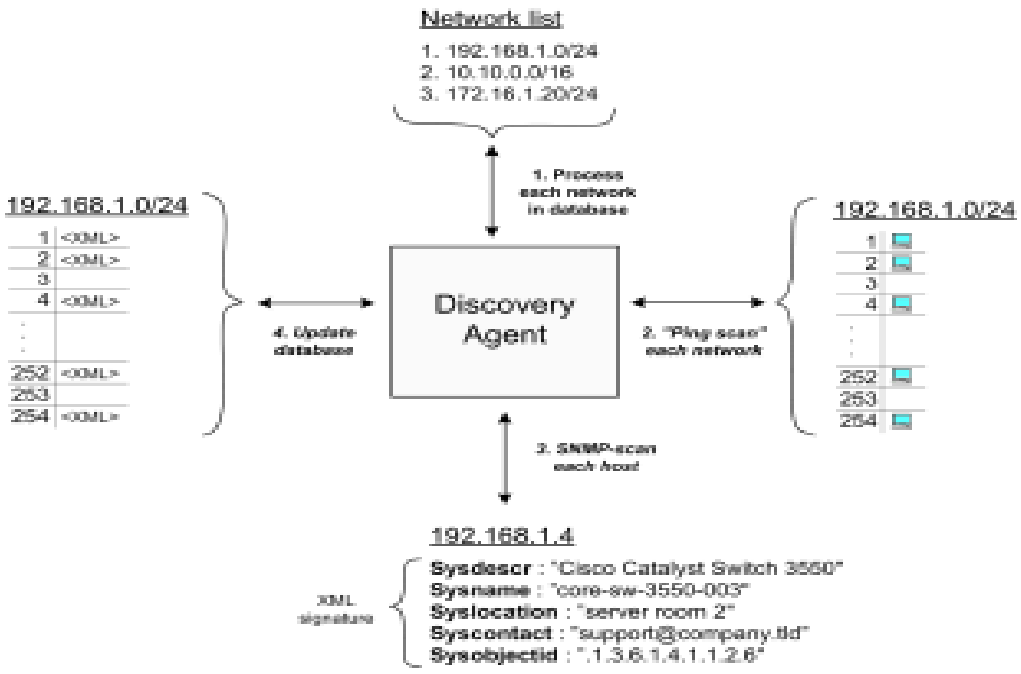


Figure 4.2 The Discovery process

4.1.2 Design tools

For the development of the source code visual studio 2010 express edition environment was selected, with the C#.NET was selected as the programming language for .NET framework 4.0 plug-in [13]. This allowed our team for easy access to help commands, syntax highlighting, and debugging support. For a screenshot of C#.NET environment, see figure 4.3. For design and configuration of the Devices

GDI+ library was used. It is a graphical API which allows for easy drawing and image Handling, and 2D graphics capability.

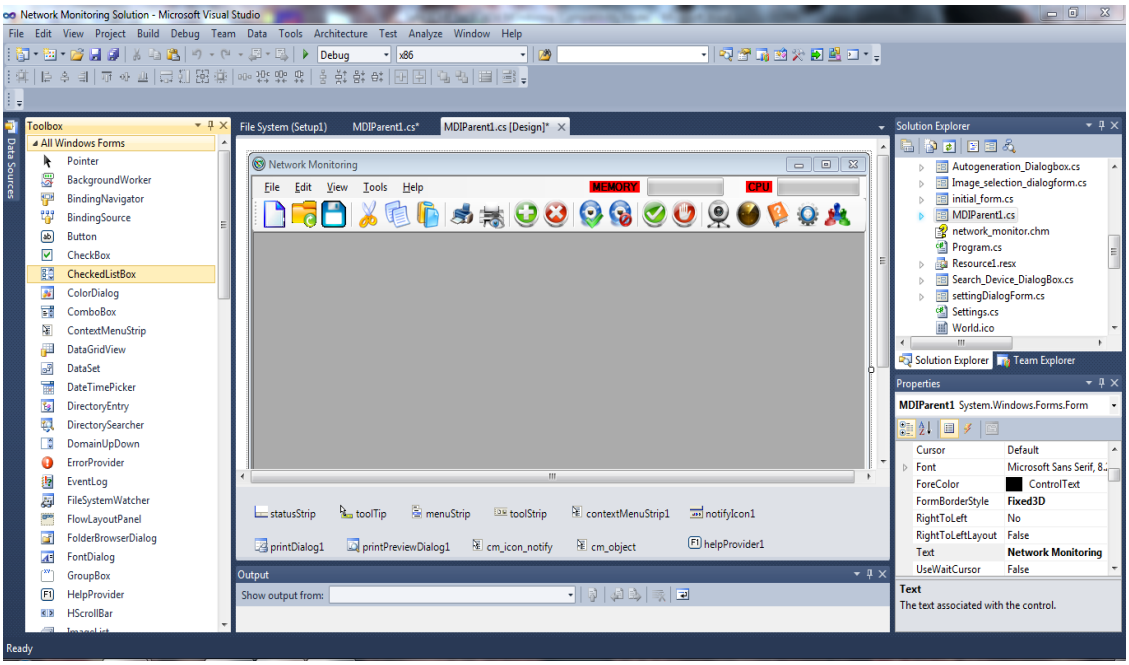


Figure 4.3 Visual Studio 2010 environment

4.2 Platform

The network monitoring application running on the Windows 7 environment is developed for .NET framework 4.0, a software framework provided by Microsoft.

4.2.1 The .NET Framework 4.0

The goal of .NET is to facilitate the development of windows applications and to reduce the security vulnerability of these applications and the computers they are running on. The .NET framework also contains a large body of pre-coded software solutions to common program requirements, such as data management, database connectivity and network communication. The .NET framework allows execution of applications written in many different languages, by use of a Common Language Infrastructure (CLI). The source code is first compiled into platform-neutral language called Common Intermediate Language (CIL) by an intermediary compiler. The CIL file is then compiled by a platform-specific Common Language Runtime (CLR) into bytecode. The CLR provides the appearance of a virtual machine, so that programmers do not need to take the capabilities of the specific CPU type into

consideration. The CLR also provides other important services such as security mechanisms, memory management, and exception handling. The class library and the CLR together compose the .NET Framework.

4.2.2 Visual C# .NET

As mentioned above, programs written in any language that has a .NET compiler can utilize the .NET framework. The most common languages used with .NET are Microsoft's Visual Basic .NET and Visual C#. However, since all .NET compatible languages use the .NET set of classes (library) for their functionality, the difference between them is merely syntactical. The choice of language for the implementation fell on Visual C# .NET mainly because it was the language that SCADA development team at Crompton Greaves (CGL) selected and it was aimed at developing full SCADA system in this platform. Although I had prior experience in C#.NET 2.0 framework, developing a system in C#.NET 4.0 was less of a problem.

4.3 Implementation

The discovery/monitoring agent was implemented in network monitoring system and was the active data gathering component of the monitoring system. The discovery process is illustrated in figure 4.2. The monitoring system consists of three major parts, the **networks processing** part**,** the **net scanner** and **host scanner**. These modules run individually, run in a sequential top-down fashion where the networks processing module spawns instances of the network scanner, which in turn spawns instances of the host scanner. This is done in order to achieve the degree of parallelization needed to complete the tasks in a limited timeframe. The results both of single and multi-process scanning are discussed in Chapter 7. The UML activity diagram in Figure 4.7 can be viewed for an illustration of the workflow behavior of the discovery/monitoring process.

4.4 Networks processor

This module uses the Graphics collection (collection of Graphics Devices) and retrieves the main information about the devices (GraphicsObject) which are to be scanned. The device information was updated by the topology analysis module when a user changes the settings in the user interface. Each network entry in the graphics collection list is processed to see if it is time to perform a new scan of the current

network. This is controlled by the cycle scan time option, which tell how often the network should be scanned (in seconds) and when the last scan was performed. With these variables, in combination with the current date and time, a decision on whether the network needs to be rescanned can be made. If it is time for a new scan of the current network, a network scanner thread is spawned.

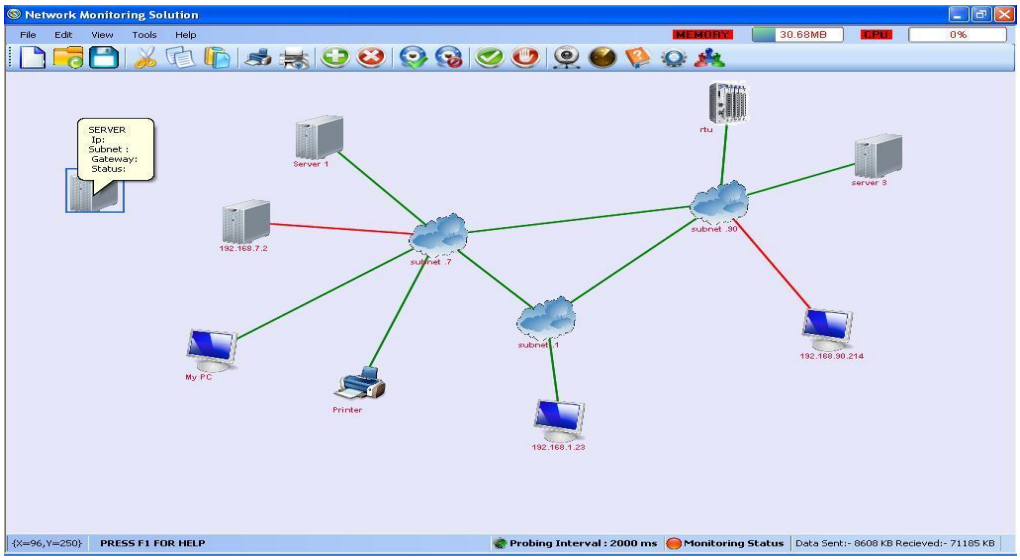


Figure 4.4 Network scanner processing device Information.

4.5 Network scanner

The net scanner thread scans a network, designated by a network address and a network mask, to detect which attached hosts are active. Since there are several possible methods of scanning that can be utilized, a decision had to be made on which to use in this software.

4.5.1 Scanning approaches

As mentioned in section 3.7, two major ways of network scanning are active and passive scanning, where active scanning tries to detect hosts by active probing [14] and passive scanning listens to detect traffic generated by the hosts [15]. In the current instance of the architecture active scanning has been favored since the devices we primarily want to detect (i.e. switches, routers, RTU, IED etc.) very seldom generate any unprovoked traffic. Active scanning is performed by sending out data on the network targeted at the address which you want to detect. What kind of data that you send can vary, but a common and straightforward approach is to send an ICMP (Internet Control Message Protocol) ECHO request. This process is commonly called

"Pinging" a host, from the name of a tool used to send these messages. One problem with using the ICMP protocol was that not all hosts will reply to it, due to firewalls blocking this type of traffic or giving it a low priority. Other possible ways to actively scan for hosts are by using TCP or UDP datagram (SNMP), directed towards specific addresses in a given network, in an attempt to generate a response. Since, in many of the environments ICMP echo request packets are blocked, so we decided to discover the underlying network topology using SNMP (Simple Network Monitoring Protocol), its standard MIBs and use these collected information to form a topology graph that is viewable to the end user. The detailed algorithm used is explained in section 4.9.2. However, since the aim of the software is to detect hosts with standard configuration, and the software must be able to scan a large number of hosts in a limited time, it was decided that SNMP MIBs were to be used for network topology detection and in order to detect the devices which donot support SNMP, regular ICMP ECHO scan would suffice. The abstract details of the SNMP namespace are given in appendix A. The ICMP functionality is implemented into host scanner by implementing the ICMP class and doing all of CRC checks in class only. This allows application the capability to create any type (0-255) of ICMP packets and accordingly obtain reply from the network components. The class definition of ICMP is given in Appendix-B.

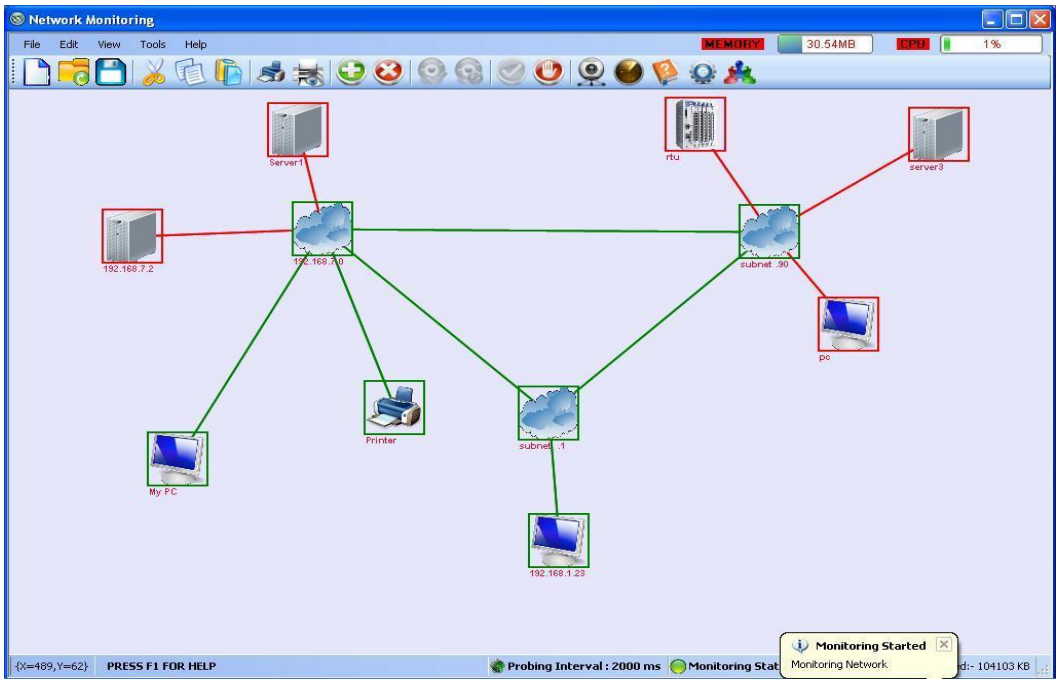


Fig 4.5: Network Scanner sending ICMP Echo requests.

Using the class in appendix-b we construct ICMP Echo Request packets by setting type to (0x08) and code to (0x00) and simultaneously send the requests to multiple hosts using multithreaded approach.

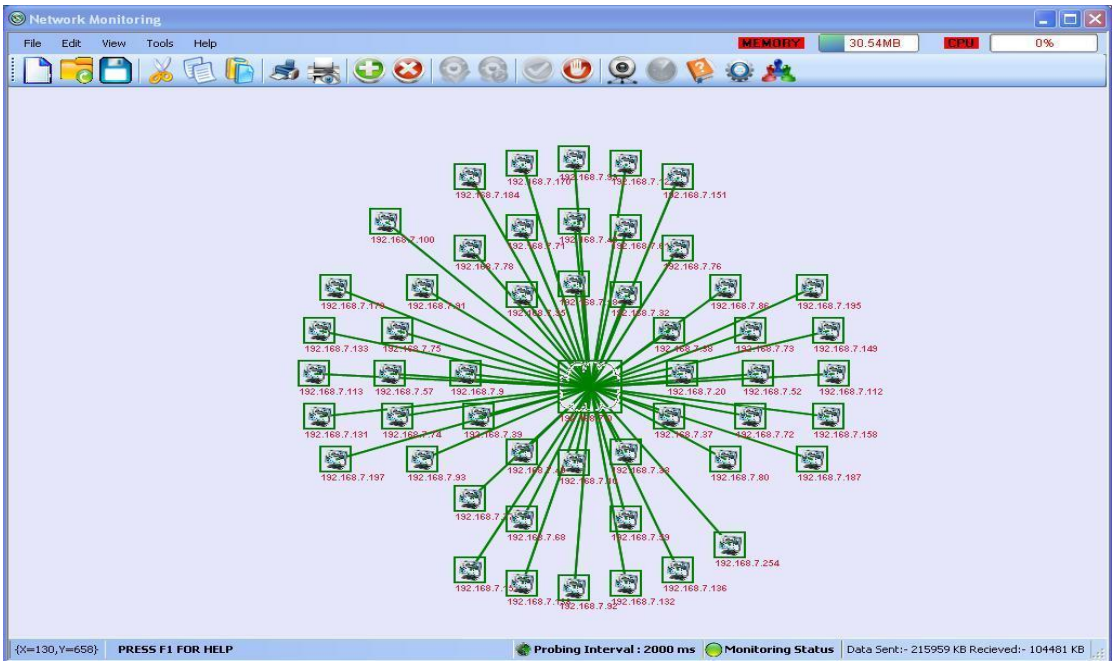


Figure 4.6 Devices Discovered and Monitored in a Subnet

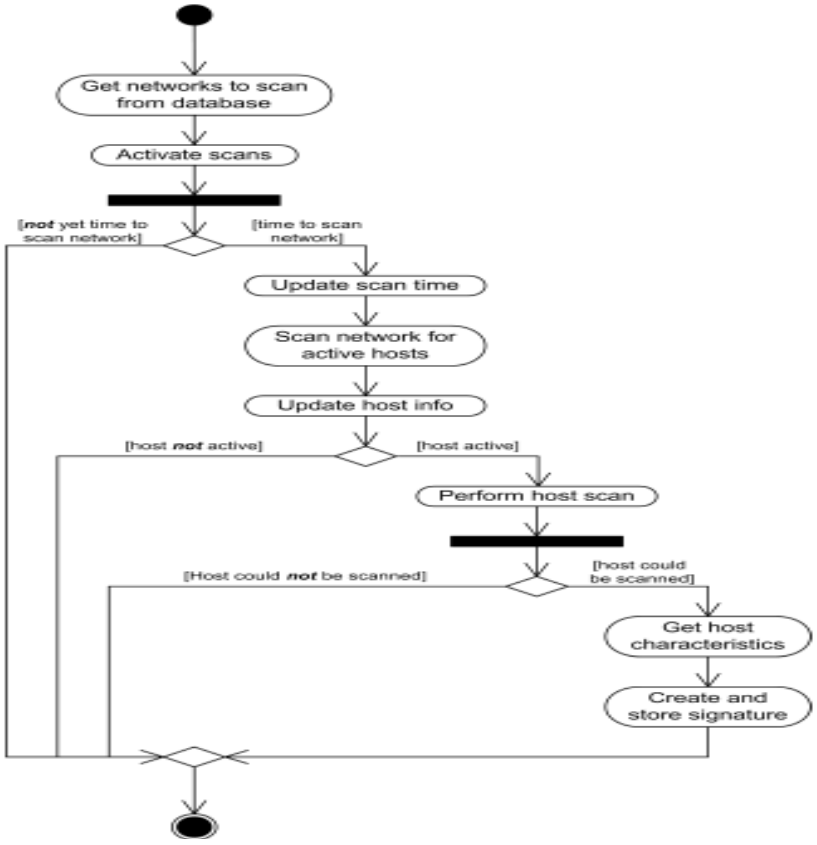


Figure 4.7 Discovery service UML activity diagram

4.6 Host scanner

After a scan of a particular network has been completed, the list of detected active hosts is processed to establish which hosts are new and which have been seen before. This is determined by IP address and imageColor property. Thus, a host which was previously inactive (e.g. turned off for a period of time), is considered new if it becomes active again. All new hosts are then further examined individually by the host scanner. This module performs the information gathering from the individual active hosts discovered by the network scanner. The main protocol used is the Simple Network Management Protocol (SNMP), which is implemented in most manageable network equipment. More information about the hosts (e.g. available services or running operating system) can also be retrieved by using either Nmap, or more specialized tools such as Xprobe [18].

4.6.1 Communities

To be able to read information from the nodes via SNMP, an SNMP community string is required, which serves as a password for access control. The community strings to be tried and used in our application are generally public but for SCADA specific device like Xcell RTU it has to be preconfigured for SNMP Agent. They are ultimately based on information entered by the system administrator when configuring the system. The communities can be stored either in plain text or encrypted (if using SNMPv3). In order to optimize the scan speed, a working SNMP community, if found, is stored in the host entry to be tried first the next time. The SNMP function in host scanner is implemented as a separate class for SNMP V2 which helps in achieving the desired level of operation.

4.6.2 Host signature

In the basic setting, some standard SNMP information is gathered from the host: system description, system name, system location, system contact and system object ID. Which information that is gathered can however be dynamically configured. Based on this information, a compound host signature is generated and stored in the GraphicsObject status Property for use by Topology Analysis Module, so as to present it to the user.

Figure 4.8 Host Scanner retrieving Host Signature via SNMP

4.7 Constructing an Interactive Graphics Device:

Creating an interactive device on screen with specific Image required a significant brainstorming. Initially we adopted a method which involved creating a GraphicsObject and drawing it on panel using panel's OnPaint( ) drawing event and panel's graphics object. But this process was quite tedious and complex as all of controls had to be drawn in paint event and interacting with multiple objects became difficult as they had to be detected based on their location and user clicked point location on panel. Another approach was adopted which involved creating GraphicsObject as a control and then adding this control to network panel. The GraphicsObject class was created with inherited properties of control class and its own properties like ipAddress, subnetMask and defaultGateway etc. The class definition of the GraphicsObject is given in appendix-c.

4.7.1 Interactive Graphics Object:

Making a GraphicsObjectControl as Interactive involves associating various events with it so that appropriate action can be taken in response to user action. Following code snippet shows how various properties and mouse events are associated with the Graphics Object.

```
//Brief code that was used to make object generic

    GraphicsObject g_object_image = new GraphicsObject();
g_object_image.image = new Bitmap(image);
g_object_image.Location = new Point(x, y);
g_object_image.type = type;
g_object_image.ID = graphics_objectcount;
g_object_image.Status = "";
g_object_image.Size = image.Size;

g.ContextMenuStrip = this.add_contextmenu();
g.MouseMove += new MouseEventHandler(g_object_image_MouseMove);
g.MouseUp += new MouseEventHandler(g_object_image_MouseUp);
g.MouseDown += new MouseEventHandler(g_object_image_MouseDown);
```

```
        g.MouseClick += new MouseEventHandler(g_object_image_MouseClick);
        tip.SetToolTip(g, g.type + "\n Ip: " + g.IPAddress + "\nSubnet : " + g.SubnetMask + "\n
Gateway: " + g.DefaultGateway + "\n Status: " + g.Status);
```

Here a context menu was also added to allow user to right click on GraphicsObject and view various options available for that particular GraphicsObject. Further the above created GraphicsObject are arranged and managed by network processor module in a collection named GraphicsObjectCollection, providing basic functions for deletion, updating and addition of objects to collection. Following GraphicsObjectCollection class provides a view of implementation.

```
namespace GUI_Network_Monitoring.graphics
{
   /// <summary>
   /// This class Provides a Collection for Holding the Graphics Objects.
   /// </summary>
   [XmlRootAttribute("GraphicsObjectCollection", Namespace = "GUI_Network_Monitoring",
IsNullable = false)]
   [Serializable()]
   public class GraphicsObjectCollection:System.Collections.CollectionBase
   {
     /// <summary>
     /// This Function adds an Graphics Object To List.
     /// </summary>
     /// <param name="g_object"></param>
      public void Add(GraphicsObject g_object)
      {
         List.Add(g_object);
      }
     /// <summary>
      /// This Function Removes An Object From Specified Index In Collection.
      /// </summary>
      /// <param name="index"></param>
      public void Remove(int index)
      {
         // Check to see if there is a g_object at the supplied index.
         if (index > Count - 1 || index < 0)
         // If no widget g_object, a messagebox is shown and the operation
         // is cancelled.
         {
            System.Windows.Forms.MessageBox.Show("Index not valid!");
         }
         else
         {
            List.RemoveAt(index);
         }
      }
     /// <summary>
     /// This Function Returns An Object From A Specified Index In Collection.
     /// </summary>
     /// <param name="Index"></param>
     /// <returns></returns>
      public GraphicsObject Item(int Index)
      {
         // The appropriate item is retrieved from the List object and
```

```
            // explicitly cast to the Grahicsobject type, then returned to the
            // caller.
           return (GraphicsObject)List [Index];
        }
       /// <summary>
       /// This Function Returns The Index Of A Graphics Object In Collection.
       /// </summary>
       /// <param name="gob"></param>
       /// <returns></returns>
        public int IndexOf(GraphicsObject gob)
        {
           return List.IndexOf(gob);
        }
        /// <summary>
        /// This Method Updates the Location of Graphics Object at the Specified Index.
        /// </summary>
        /// <param name="Index"></param>
        /// <param name="Location"></param>
        public void Update(int Index, Point Location)
        {
           GraphicsObject obj_toupdate = (GraphicsObject)List[Index];
           obj_toupdate.Location = Location;
           Console.WriteLine(Location);
           List[Index] = obj_toupdate;
        }
    }
}
```

The created GraphicsObjects are added to GraphicsObjectCollection by the application and later all operations are performed on the GraphicsObjects present in the collection. The collection is later on passed on to network scanner module which traverses through the collection and decides which hosts are to be active probed and as per the reply received it updates the GraphicsObject properties. Topology analysis module then in parallel reflects the changes made in GraphicsObject by flashing up the status, color of devices on the network panel and also color of the connection in between devices. If both devices are detected as active then the connection color in between them is drawn in green, otherwise color is shown in red.

4.8 Performance issues

In the first implementation of the discovery agent, all scans of the networks assigned to an individual agent were done in a serial fashion without any concerns about scalability. Initially this did not cause any problems, but when the implementation was tested in a simulated large network, performance issues arose. A scan of a network with a thousand hosts could take an hour to complete. This might not seem too bad at first glance, but in even larger networks the scanning times increased linearly. Due to the fact that network topology changes may occur quite frequently,

better performance is needed to capture these changes more quickly. So applying a single running process in application is unfeasible solution and rather multithreaded approach would suffice in scanning multiple hosts at the same time and retrieving the network information.

4.8.1 Multithreading

The above mentioned performance requirements demanded a redesign of the scanning implementation. Instead of using a traditional top-down scanning method which would lead to freezing of GUI, a parallel approach was needed. This was implemented by dividing the network scanning agent into separate modules as threads which could be run simultaneously. By scanning several large networks and their respective hosts in parallel, it was shown that completion times could be significantly reduced. More information about the performance tests can be found in chapter 5.

C#.NET provides System.threading package which allows creating multiple threads with an option to create either parameterized or non parameterized thread. Code snippet below shows how to create a thread and make it to run in background thus avoiding the interference with the GUI.

```
Thread Pinger = new Thread(new ThreadStart(Ping_Host_Synchronous));
 Pinger.IsBackground = true;
 Pinger.Start();
```

4.9 Topology Analysis Module

The responsibility of the topology analysis module (runner) is to retrieve the data collected by the discovery agent, to process it and finally store it as useful information which can be presented to the user. The runner compares the model of the network which exists in the network panel topology with the real world data gathered by the network scanner, and alerts the user of inconsistencies which might be important to attend to.

4.9.1 Design

The topology analysis module is designed as a runner (thread) in the Network Monitor Solution – a program scheduled to run constantly in the background. Since there was already an existing underlying framework for creating a runner, through which the new functionality was to be implemented, many design decisions were dependent on compatibility with the existing system.

4.9.2 Implementation

The discovery/topology analysis module is implemented in Microsoft Visual C# .NET with the main functions in a .cs file, available to the rest of the system through GUI_NETWORK_MONITORING namespace.

4.9.2.1 Network Topology Discovery/Monitoring Algorithm (Proposed algorithm)

In this section, we present our approach to discover network nodes and connectivity among them. The basic inputs to our system are boundary information, i.e., one or multiple range of IP address (es); one or multiple community string(s); SNMP port number; and database credentials. Since our approach is mainly based on ICMP and SNMP, we first analyse the major management information base (MIB) objects required to build our algorithm. These MIBs are used to build a discovery algorithm, which is basically divided into three different modules, namely *device discovery*, *device type discovery*, and *connectivity discovery*. Our proposed algorithm is basically divided into three phases; (I) Determining the ip address of the nearest router using ICMP protocol. (II) Further this ip address of the router will be used to find the other connected devices using SNMP and ICMP protocol. (III) Continuous monitoring of discovered devices using ICMP.

MIBs for Discovery

Our discovery mechanism is based solely on SNMP. Table 4.1 explains all the SNMP MIB objects required.

Table 4.1 MIB Information for topology discovery

MIB-II RFC 1213 [31]
sysServices, sysDescr,ifIndex, ifDescr, ifPhyaddress, ipForwarding, ipRouteNextHop, ipRouteType, ipAdEntAddr, ipAdEntNetMask, ipNetToMediaNetAddress, ipNetToMediaPhysAddress,

Phase 1: Finding the IP address of nearest router using ICMP protocol

A. Algorithm starts with determining all the active IP addresses by sending ICMP echo request packets to every address of the given address space.

B. Send the ICMP packets with TTL field being set to “one” to all the ip addresses discovered previously. If reply address is not same as the address on which request was sent, it is the router address and add it to router IP list.

C. Add the ip address of the first router to the CONNECTED_DEVICE list. This list contains all the ip addresses that have been added on to the topological graph.

Phase 2: Finding other connected devices using SNMP and ICMP protocol

A. Read the routing table of router. A routing table of the device is maintained by the *ipRouteTable* object; it contains entry for each route known to this entry in *ipRouteEntry*. By performing SNMP GETNEXT operation to ipRouteTable object, management station can get the routing table from router. We utilize only *ipRouteNextHop* and *ipRouteType* entries from these tables. *ipRouteNextHop* is the IP address of the next hop in the route. It shows the next router’s IP. *ipRouteType* can be one of four types: *direct, indirect, invalid,* or *other.* We filter the records and take only those entries that are of type *indirect.* Indirect route shows that destination network or destination host still has to pass through other routers. Thus by getting one router, other routers that direct connect with the first one will be found. By searching the next router’s ipRouteTable, more routers are shown. Connect these routers one by one, the main map of topology is shown and add these ip addresses to CONNECTED_DEVICE list.

B. Search other network devices in every network. In addition to ipRouteNextHop, IpRouteTable also contains other information, such as ipRouteType, ipRouteDest and ipRouteMask. ipRouteType indicates the relationship between network and router. IpRouteDest shows the network address while ipRouteMask shows the net mask. For every adjacency network, start IP and end IP are calculated by ipRouteDest and ipRouteMask information. Enumerating every IP in this network, searching program will test the SNMP GET operation. If it responds, we can confirm there is a network device on this IP, and the name of this device is displayed as SysDescr. Connect this device with router indicated by ipRouteNextHop and add the ip address to CONNECTED_DEVICE list, the graph hierarchy has started to build.

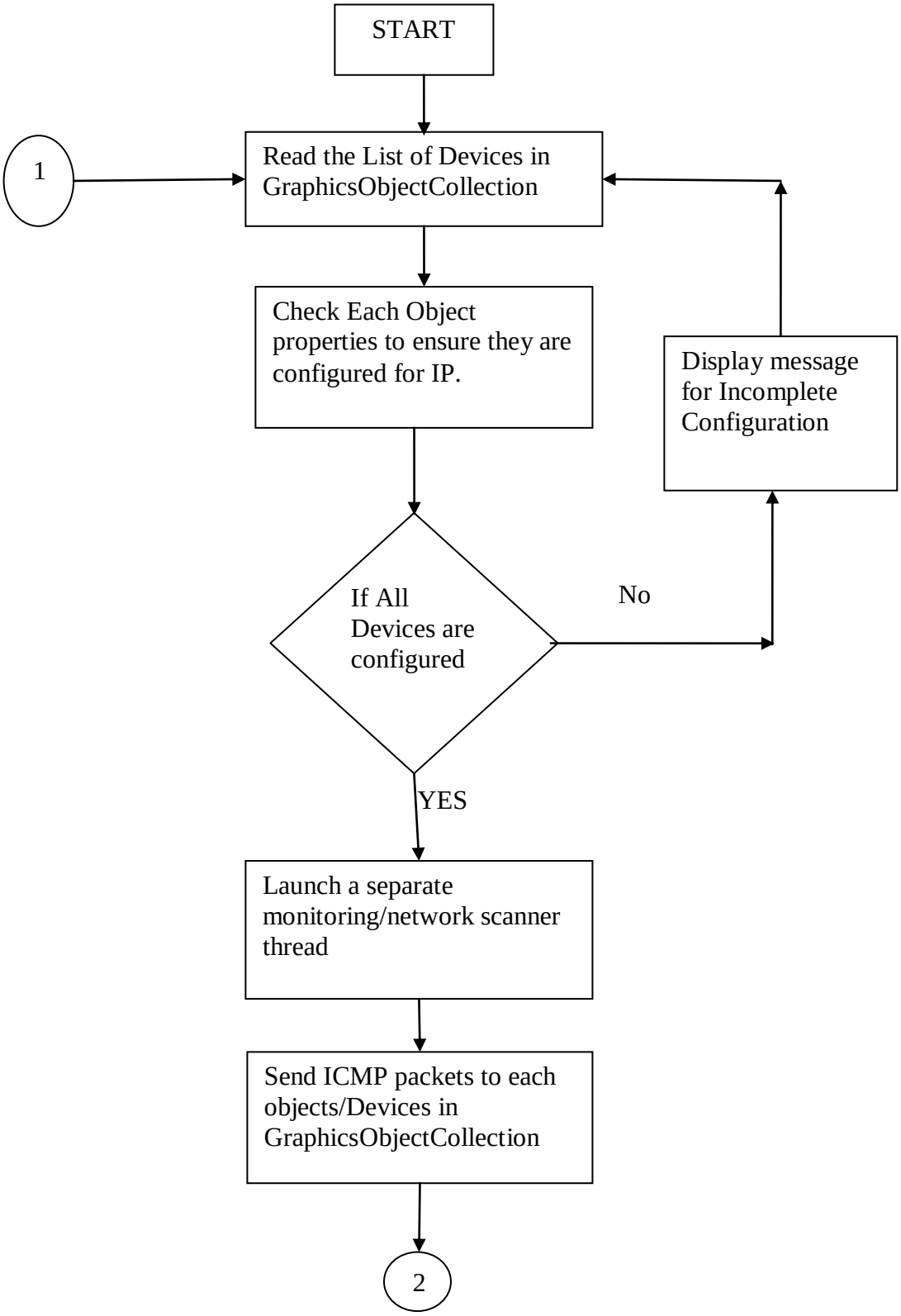
C. Get computers of every network. For getting the PCs' IP and MAC information from switch, ipNetToMediaTable object in switch will be visited. ipNetToMediaTable records the relationship between switch's port, connecting PC's IP and MAC. For every IP in ipNetToMediaTable, if it is not a device that has been found, it should be a PC and will be added to new device list. Inside ipNetToMediaTable object, the ipNetToMediaIfIndex indicates switch's port index, which can be matched with ifIndex of ifTable object, at the same time object ipNetToMediaPhysAddress and object ipNetToMediaNetAddress indicate the MAC and IP of this PC. As soon as we discover a node, we use all unique *ipNetToMediaNetAddress* (i.e. the entries of new device list) entries to discover another set of new nodes. After this step, connect the PC with the corresponding subnet within topologic graph. After completing these three steps, the topology is almost done.

D. Discovering devices that do not support SNMP. For devices that do not support SNMP, ICMP echo requests are used to check whether a device is alive or not. ICMP address-mask requests are used to obtain subnet information about those devices. *ipAddrTable* contains the IP address assigned to the multiple interfaces in the managed node, and there can be one interface for one sub network. To check for this condition, a table of synonyms of the already-discovered devices is maintained, and before confirming that the discovered device is new, it is added on to the ICMP list. ICMP echo requests are sent to all the ip addresses present in ICMP list. After the ICMP echo reply is received, corresponding device is added at an appropriate location within the topology.

Phase 3: Continuous monitoring of discovered devices

Scheduled ICMP algorithm is necessary in order to ensure that the devices found during discovery phase are alive and connected. Asynchronous ICMP echo requests are very important. Packets are constructed in a socket in accordance with the ICMP normative, and the packets are sent to multiple addresses and then receiving replies are started. In this way there will exist small response time for waiting as compared to synchronous ping request and reply. But this is an important step in network monitoring as asynchronous ping request is sent to all the devices listed in the topology after a specified probing interval, so as to continue monitor the health of the connected devices and indicate the user of any change that had been accommodated within the network.

In Figure 4.9 the process of the topology analysis module is shown in the form of a flowchart



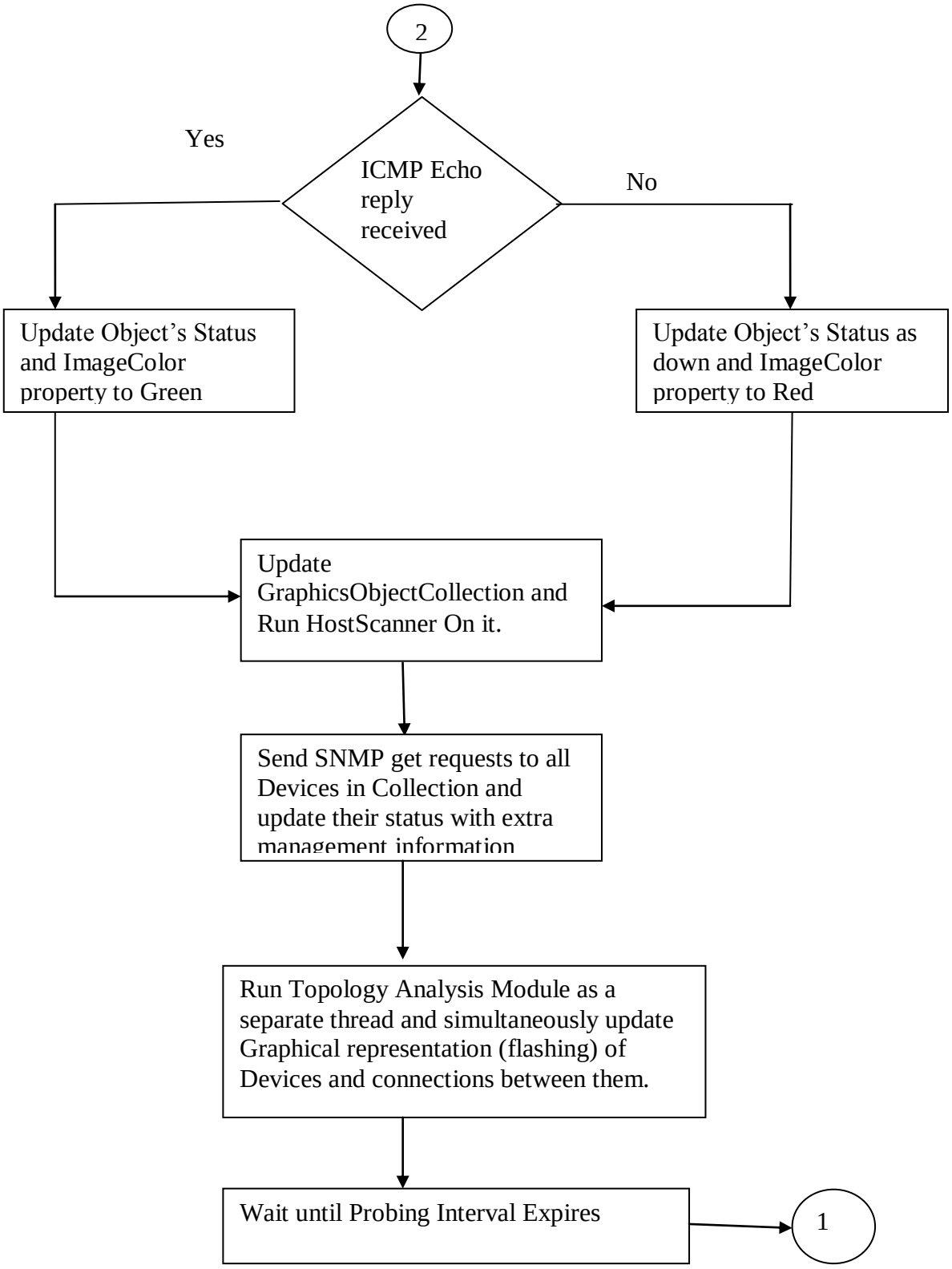


Figure 4.9 Flow chart describing Monitoring Process

4.9.2.2 Saving the Topology data

Topology analysis module is responsible for saving the topology data in XML format as discussed above. The technique used to perform this process is serialization. We use XMLSerializer available System.XML.Serialization namespace. It provides the capability to serialize and then deserialize the Topology data when required. The XML signature below gives a brief Idea of the process.

```
<? xml version="1.0" encoding="utf-8"?>
<ArrayOfGraphicsObjectserializable
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <GraphicsObjectserializable>
    <Location xmlns="GUI_Network_Monitoring">
      <X>100</X>
      <Y>100</Y>
    </Location>
    <Size xmlns="GUI_Network_Monitoring">
      <Width>60</Width>
      <Height>60</Height>
    </Size>
    <Picture
xmlns="GUI_Network_Monitoring">iVBORw0KGgoAAAANSUhEUgAAADwAA
AA8CAYAAAA6/ ==</Picture>
    <imagecolor xmlns="GUI_Network_Monitoring">0</imagecolor>
    <type xmlns="GUI_Network_Monitoring">PC</type>
    <IP Address xmlns="GUI_Network_Monitoring" >192.168.1.34</IP Address>
    <Subnet Mask xmlns="GUI_Network_Monitoring">255.255.255.0</Subnet
Mask>
    <Status xmlns="GUI_Network_Monitoring" >up, Packets Sent: 78
received:56</Status>
    <List xmlns="GUI_Network_Monitoring" />
  </GraphicsObjectserializable>
```

Above XML signature shows a GraphicsObject which has been serialized and its various associated properties and network information are saved (location on graph, color, picture, ip address, subnet mask, status etc).

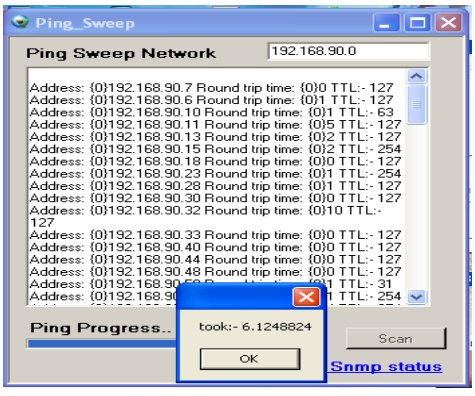
CHAPTER 5

RESULTS AND EVALUATION

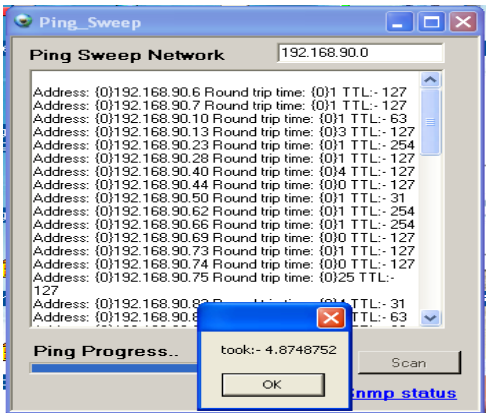
The Network Monitoring Tool was implemented and tested at Crompton Greaves Global Research & Development department. The system was made to monitor devices located in different subnets and also discovery of subnets was done. We successfully monitored a maximum of 400 devices without much compromising with the systems performance.

Initially we developed the system with single process approach and made this process to handle network processing, network scanning and host scanning tasks. The approach worked fine with few computers but with the increase in number of hosts the system performance started degrading, GUI started to freeze because the tool at the same time was probing the network devices and simultaneously handling the updation of device's status and other GUI elements. Scanning a subnet with 255 devices with this sequential approach took ages and the solution was unfeasible.

We then tested the tool with simultaneous spawning of 254 threads, each probing a single host and querying it for host signature information. Though this approach was faster and fetched results but it also lead to CPU overhead and also simultaneous spawning of huge number of threads lead to destabilization of development environment.



(a) Time taken 6.12 sec
Hosts scanned: - 66
ICMP timeout: 100 millisecond
data buffer =1 byte



(b) Time taken 4.87 sec
Hosts scanned: - 44
ICMP timeout: 100 millisecond
data buffer =1 byte

Figure 5.1 Scanning using multithreading

The Impact of Multithreading on scanning time for a given number of devices is clearly indicated in Figure 5.2

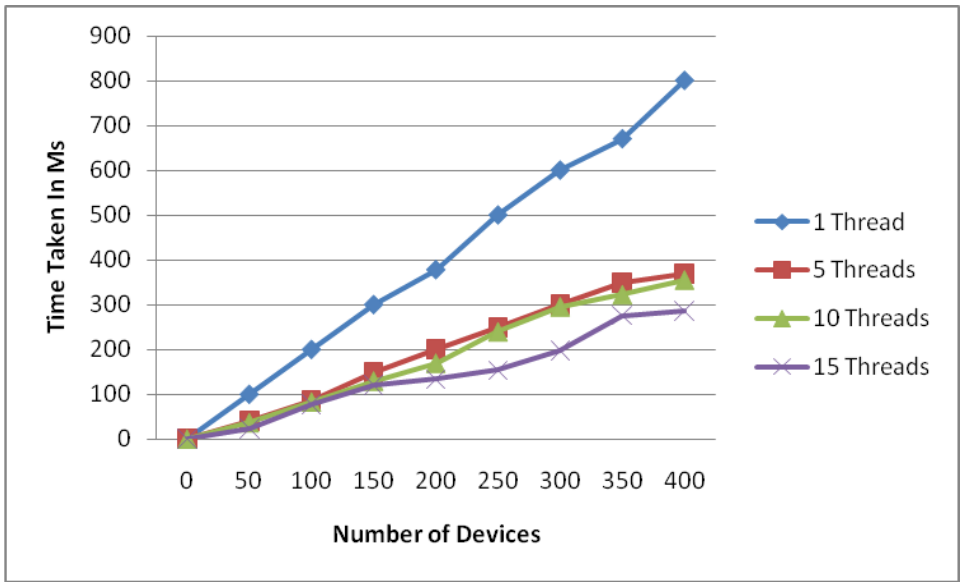


Figure 5.2 Number of Devices Vs Discovery Time.

Figure 5.2 shows below shows the variation in time taken to discover number of hosts based on the level of multithreading.

As the number of threads are increased the time taken to discover network hosts decreases considerably. However after a certain extent the increase in threads leads to an overhead on system and system performance decreases considerably if agent less network discovery approach is adopted.

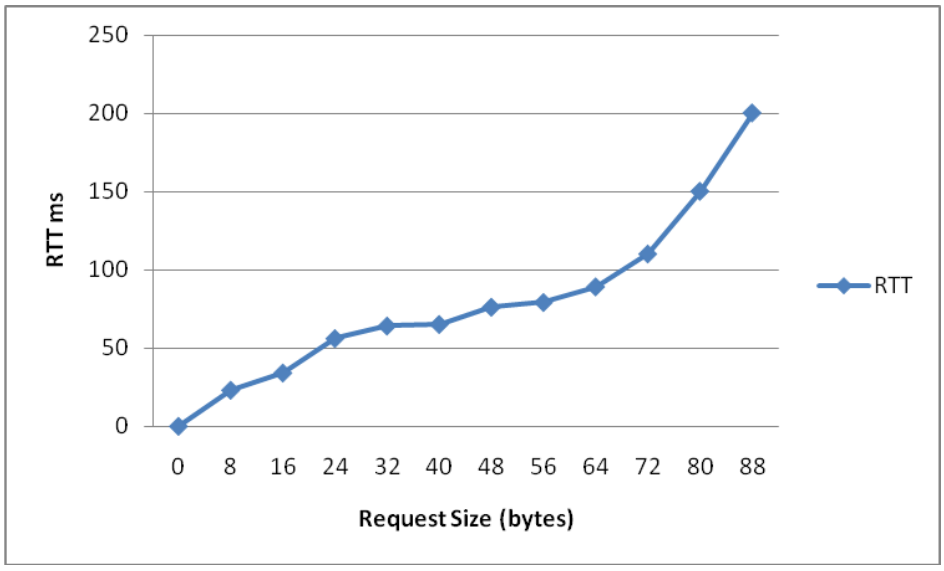


Figure 5.3 RTT vs Packet Size

Figure 5.3 shows the variation in RTT (Round Trip Time) obtained in Echo reply packets if variable sized Echo request packets are sent to network devices. Although sending multiple Echo request packets on network may lead to congestion but results suggest that sending echo request of larger packet size lead to increase in RTT clearly indicating increase in network load.

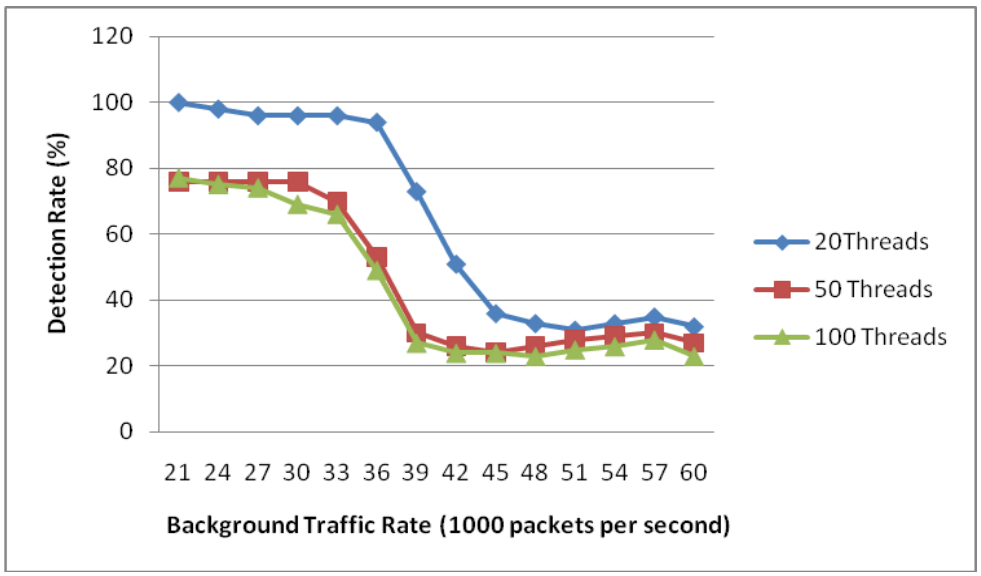


Figure 5.4: Average Detection rate Vs Background Traffic rate

Figure 5.4 illustrates behaviour of SNMP agent under different combination of number of threads and background traffic rate. One can see that from 33000packets/second the detection rate drops quickly. At higher transmission rates the system load increases, reducing the detection rate of the agent manely due to lack of CPU resources. This mainly happens if very large network is scanned.

Our study also confirms that if all devices in a network are reachable then the network scan time will be less in comaprison to if a network has certain devices which are down. This phenomenon results because the network scanner waits for some timeout period for a device to reply before moving on to next device. This can be removed by switching network scanning mode to asynchronous.

Clearly the above implemanted system is a effective, cost efficient solution and is implemented at Crompton Greaves which allows to monitor various SCADA specific devices like Xcell RTU, IED etc.

CHAPTER 6
CONCLUSION & FUTURE SCOPE

The automatic monitoring of the network topology is always an important means for network management. To design and develop an effective and useful tool for the network topology monitoring/discovery is an aspect in developing Network Monitoring Solution. The proposed monitoring system is designed to operate on fast Local Area Network (LAN) that won't suffer from the aggressive probing. This work proposes and provides a mean to implement another solution which involves an effective and efficient use of ICMP and SNMP protocols for effective network monitoring, both complementing each other with network and management information.

The presented architecture, and its incorporated methods, has shown to offer an effective service for flexible network node discovery/monitoring and topology analysis. Implemented in C#.NET on windows platform, it has demonstrated the benefit of synchronization between off-line data and real-world network topology. Using XML and regular expressions to define and evaluate parameters and conditions makes the system highly adaptable and scalable for future improvements. The presented work provides a unique advantage for easy support and management/monitoring of SCADA specific devices like RTU, IED thereby allowing effective integration with SCADA architecture. This work thus lays the ground for further extensions in a security-aware enterprise network management and monitoring system. Adding further data, e.g. about current system versions or patch levels, is a straightforward process and prepares for automatic vulnerability assessment of the nodes in an enterprise network. Future work includes the combination of such a service with a risk analysis and threat assessment infrastructure for business critical systems.

APPENDIX-A : Brief about SNMP Class implementation

```
//SNMP Namespace
namespace SnmpSharpNet
{
   public class AgentParameters : IAgentParameters

   public abstract class AsnType : ICloneable

   public enum AsyncRequestResult

   internal class AsyncRequestState

   public sealed class Authentication

   public enum AuthenticationDigests

   public class AuthenticationMD5 : IAuthenticationDigest

   public class AuthenticationSHA1 : IAuthenticationDigest

   [Serializable]
   public class Counter32 : UInteger32, ICloneable

   [Serializable]
   public class Counter64 : AsnType, IComparable<ulong>,
IComparable<Counter64>, ICloneable

   [Serializable]
   public class EndOfMibView : V2Error, ICloneable

   [Serializable]
   public class EthernetAddress : OctetString, ICloneable

   [Serializable]
   public class Gauge32 : UInteger32, ICloneable

   public interface IAgentParameters

   public interface IAuthenticationDigest

   [Serializable]
   public class Integer32 : AsnType, IComparable<Integer32>, IComparable<int>,
ICloneable

   [Serializable]
   public class IpAddress : OctetString, IComparable, ICloneable

   public interface IPrivacyProtocol
```

public class **MsgFlags** : AsnType, ICloneable

[DefaultMember("Item")]
public class **MutableByte** : ICloneable, IComparable<MutableByte>, IComparable<byte[]>

[Serializable]
public class **NoSuchInstance** : V2Error, ICloneable

[Serializable]
public class **NoSuchObject** : V2Error, ICloneable

[Serializable]
public class **Null** : AsnType, ICloneable

[Serializable, DefaultMember("Item")]
public class **OctetString** : AsnType, ICloneable, IComparable<byte[]>, IComparable<OctetString>, IEnumerable<byte>, IEnumerable

[Serializable, DefaultMember("Item")]
public class **Oid** : AsnType, ICloneable, IComparable, IEnumerable<int>, IEnumerable

[Serializable]
public class **Opaque** : OctetString, ICloneable

[DefaultMember("Item")]
public class **Pdu** : AsnType, ICloneable, IEnumerable<Vb>, IEnumerable

public enum **PduErrorStatus**

public enum **PduType**

public class **Privacy3DES** : IPrivacyProtocol

public class **PrivacyAES** : IPrivacyProtocol

public class **PrivacyAES128** : PrivacyAES

public class **PrivacyAES192** : PrivacyAES

public class **PrivacyAES256** : PrivacyAES

public class **PrivacyDES** : IPrivacyProtocol

public sealed class **PrivacyProtocol**

public enum **PrivacyProtocols**

public class **ScopedPdu** : Pdu

public class **SecureAgentParameters** : IAgentParameters

[Serializable]
public class **Sequence** : AsnType, ICloneable

public class **SimpleSnmp**

internal delegate void **SnmpAsyncCallback**(AsyncRequestResult status, IPEndPoint peer, byte[] buffer, int length)

public delegate void **SnmpAsyncResponse**(AsyncRequestResult result, SnmpPacket packet)

public class **SnmpAuthenticationException** : SnmpException

public sealed class **SnmpConstants**

public class **SnmpDecodingException** : SnmpException

public sealed class **SnmpError**

public class **SnmpErrorStatusException** : Exception

[Serializable]
public class **SnmpException** : Exception

public class **SnmpInvalidPduTypeException** : SnmpException

public class **SnmpInvalidVersionException** : SnmpException

public class **SnmpNetworkException** : SnmpException

public abstract class **SnmpPacket**

public class **SnmpPrivacyException** : SnmpException

public class **SnmpV1Packet** : SnmpPacket

public class **SnmpV1TrapPacket** : SnmpPacket

public class **SnmpV2Packet** : SnmpPacket

public class **SnmpV3Packet** : SnmpPacket

public sealed class **SNMPV3ReportError**

public enum **SnmpVersion**

```
    [Serializable]
    public class TimeTicks : UInteger32, ICloneable

    public class TrapAgent

    public class TrapPdu : AsnType, ICloneable

    public class UdpTarget : UdpTransport, IDisposable

    public class UdpTransport : IDisposable

    [Serializable]
    public class UInteger32 : AsnType, IComparable<UInteger32>,
IComparable<uint>

    public class UserSecurityModel : AsnType, ICloneable

    public class V2Error : AsnType, ICloneable

    public class V2PartyClock : UInteger32, ICloneable

    public class Vb : AsnType, ICloneable

    [DefaultMember("Item")]
    public class VbCollection : AsnType, IEnumerable<Vb>, IEnumerable
}
```

APPENDIX-B: Brief about ICMP Class implementation

```
namespace GUI_Network_Monitoring.Network
{
   /// <summary>
   /// Summary description for IcmpPacket.
   /// </summary>
   public class Icmp
   {
      private IPEndPoint Source_ = null;
      private IPEndPoint Destination_ = null;
      private byte Type_;
      private byte Code_;
      private UInt16 Checksum_;
      private int DataSize_;
      private byte[] Data_ = new byte[1024];

public Icmp()
      {

      }

      public IPEndPoint Source
      {
         get { return Source_; }
         set { Source_ = value; }
      }

      public IPEndPoint Destination
      {
         get { return Destination_; }
         set { Destination_ = value; }
      }

      public String DestinationIP
      {
         get { return
this.Destination_.Address.ToString(); }
      }

      public String SourceIP
      {
         get { return
this.Source_.Address.ToString(); }
      }

      public byte Type
      {
         get { return Type_; }
         set { Type_ = value; }
      }

      public byte Code
      {
         get { return Code_; }
         set { Code_ = value; }
      }

      public UInt16 Checksum
      {
         get { return Checksum_; }
         set { Checksum_ = value;
}
      }

      public int DataSize
      {
         get { return DataSize_; }
         set { DataSize_ = value; }
      }

      public Byte[] Data
      {
         get { return Data_; }
         set { Data_ = value; }
      }

      public void SetData(byte[] data, int offset, int length)
      {
```

```
            Data_ = new byte[length];
            Array.Copy(data, offset, Data_, 0, length);
        }

        public String GetHashString()
        {
            String format = "Icmp:{0}:{1}";
            return String.Format(format, this.SourceIP, this.DestinationIP);
        }


        public Icmp(byte[] data, int size)
        {
            Type_ = data[20];
            Code_ = data[21];
            Checksum_ = BitConverter.ToUInt16(data, 22);
            DataSize_ = size - 24;
            Buffer.BlockCopy(data, 24, Data_, 0, DataSize_);
        }
        public byte[] getBytes()
        {
            byte[] data = new byte[DataSize_ + 9];
            Buffer.BlockCopy(BitConverter.GetBytes(Type_), 0, data, 0, 1);
            Buffer.BlockCopy(BitConverter.GetBytes(Code_), 0, data, 1, 1);
            Buffer.BlockCopy(BitConverter.GetBytes(Checksum_), 0, data, 2, 2);
            Buffer.BlockCopy(Data_, 0, data, 4, DataSize_);
            return data;
        }
        public UInt16 getChecksum()
        {
            UInt32 chcksm = 0;
            byte[] data = getBytes();
            int packetsize = DataSize_ + 8;
            int index = 0;
            while (index < packetsize)
            {
                chcksm += Convert.ToUInt32(BitConverter.ToUInt16(data, index));
                index += 2;
            }
            chcksm = (chcksm >> 16) + (chcksm & 0xffff);
            chcksm += (chcksm >> 16);
            return (UInt16)(~chcksm);
        }
    }
}
```

APPENDIX-C : Brief about Graphics object Class

GraphicsObject: 


```
namespace GUI_Network_Monitoring.graphics
{

    /// <summary>
    /// This Class Inherits Control Class To Draw an Image as a Control On Drawing Panel.
    /// Cannot be serialized straightaway, so we have to rip off Control class properties inherited.
    /// we created GraphicsObjectserializable class in XML folder to assign "GraphicsObject"  properties to
    /// GraphicsObjectserializable but leaving aside "Control" (base class properties).
    /// </summary>

    [XmlRootAttribute("GraphicsObject", Namespace = "GUI_Network_Monitoring", IsNullable = false)]
    [Serializable()]
    public class GraphicsObject : Control
    {
        #region Graphics Object Properties
        Bitmap _img;
        [XmlAttributeAttribute(DataType = "String")]
        String _ipaddress;
        [XmlAttributeAttribute(DataType = "String")]
        String _subnetmask;
        [XmlAttributeAttribute(DataType = "String")]
        String _defaultgateway;
        [XmlAttributeAttribute(DataType = "String")]
        Color _imgcolor;
        [XmlAttributeAttribute(DataType = "String")]
        String _type;
        [XmlAttributeAttribute(DataType = "String")]
        String _status;
        [XmlAttributeAttribute(DataType = "int")]
        int _objectid;
        [XmlAttributeAttribute(DataType = "bool")]
        bool _obj_type;

        [XmlArray("GraphicsObjects"), XmlArrayItem("object", typeof(GraphicsObject))]
        List<GraphicsObject> connection_List = new List<GraphicsObject>();
        #endregion

        public GraphicsObject()
```

```
        {
        }

        #region Getter/Setter Methods For
Graphics Object Properties

        public bool object_type
        {
            get
            {
                return _obj_type;
            }
            set
            {
                _obj_type = value;
            }
        }
        [XmlIgnoreAttribute()]
        public Bitmap image
        {
            get
            {
                return _img;
            }
            set
            {
                _img = value;
            }
        }
        [XmlElementAttribute("Picture")]
        public byte[] PictureByteArray
        {
            get
            {
                if (_img != null)
                {
                    TypeConverter
BitmapConverter =

TypeDescriptor.GetConverter(_img.Ge
tType());
                    return (byte[])

BitmapConverter.ConvertTo(_img,
typeof(byte[]));
                }
                else
                    return null;
            }
            set
            {
                if (value != null)
                    _img = new Bitmap(new
MemoryStream(value));
                else
                    _img = null;
            }
        }
        public Color imagecolor
        {
            get
            {
                return _imgcolor;
            }
            set
            {
                _imgcolor = value;
            }
        }
        public String type
        {
            get
            {
                return _type;
            }
            set
            {
                _type = value;
            }
        }
        public int ID
        {
            get
            {
                return _objectid;
            }
            set
            {
                _objectid = value;
            }
        }
        public String IPAddress
        {
            get
            {
                return _ipaddress;
```

```
            }
            set
            {
                _ipaddress = value;
            }
        }
        public String SubnetMask
        {
            get{
                return _subnetmask;
            }
            set
            {
                _subnetmask = value;
            }
        }
        public String DefaultGateway
        {

            get
            {
                return _defaultgateway;
            }
            set
            {
                _defaultgateway = value;
            }
        }
        public String Status
        {
            get
            {
                return _status;
            }
            set
            {
                _status = value;
            }
        }
        public List<GraphicsObject> List
        {
            get
            {
                return connection_List;
            }
        }
        #endregion

        /// <summary>
        /// Overriden Control's Paint Method Used For Drawing Image on Control Client
Area.
        /// </summary>
        /// <param name="e"></param>
        protected override void OnPaint(PaintEventArgs e)
        {
            Graphics g = e.Graphics;
            g.SmoothingMode = SmoothingMode.HighSpeed;
            _img.MakeTransparent(_img.GetPixel(1, 1));
            g.DrawImage(_img, this.ClientRectangle);
            this.BackColor = Color.Transparent;
            g.DrawRectangle(new Pen(new SolidBrush(this.ForeColor), 3),
this.DisplayRectangle.X, this.DisplayRectangle.Y, this.DisplayRectangle.Width - 1,
this.DisplayRectangle.Height - 1);
            g.Dispose();
        }
    }
}
```

REFERENCES

[1] Higgs, M.A "Electrical SCADA systems from the operators perspective" in HICRCC (IEEE) Nov 1998, pages 31 – 34

[2] Chandna, V.K.; Kumar, P.; Thomas, M.S., "Innovation in the Design of RTU and Migration to IED" in POWERCON 15 Oct. 2008, pages 1 – 6

[3] Boettger, Klaus; Reuschel, Klaus, "Telecommunication Power Supply with Programmable Logic Control" in INTELEC june 1987, page 597

[4] Zhu Yongli; Wang Dewen; Wang Yan; Zhao Wenqing, "Study on interoperable exchange of IEC 61850 data model" in ICIEA May 2009, page 2724-2728

[5] Raman, L "OSI systems and network management" in Communications magazine IEEE March 1998, pages 46-53.

[6] Chenyue Duan; Qin Zhao; Yan Ma "Object-oriented IPV4/IPV6 distributed network management model" in (IC-BNMT) IEEE Oct 2010, pages 238-242

[7] Zhang Shiyong; Wu Chengrong; Guo Wei "Network monitoring in broadband network" ICWISE Dec 2001, pages 171-177

[8] "Network Monitoring White Paper" www.telogic.com.sg/PDF/Monitoring_White_Paper.pdf visited on 2/02/2011

[9] "Network Monitoring Basics"

www./en.wikipedia.org/wiki/Network_monitoring visited on 5/03/2011

[10] Zangrilli, M.; Lowekamp, B.B "Using passive traces of application traffic in a network monitoring system" ICHPDC June 2004, pages 77-86

[11] "Internet Monitoring Techniques" www.slac.stanford.edu/comp/net/wan.../passive-vs-active.html visited on 12/03/2011.

[12] Chen, T.M, Hu, L. "Internet performance monitoring" Proceedings of IEEE sept 2002, pages 1592-1603

[13] Matthews, W.; Cottrell, L The PingER project: active Internet performance monitoring for the HENP community, communications magazine vol 38, pages 130-136

[14] A Summary of Network Traffic Monitoring and Analysis Techniques http://www1.cse.wustl.edu/~jain/cse567-06/ftp/net_monitoring/index.html visited on 18/03/2011

[15] "Network Mapping"

http://en.wikipedia.org/wiki/Network_mapping visited on 5/04/2011

[16] "Internet Control Message Protocol" RFC-792 .

http://www.faqs.org/rfcs/rfc792.html

[17] V. Jacobson, "Traceroute Software", Lawrence Berkeley Laboratories, 1989.

[18] J. Shamsi, M. Brocmeyer, "Principles of Network Monitoring" http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.7296&rep=rep1&type=pdf visited on 22/04/2011.

[19] S.Aidarous, T.plevyak, "Principles of Network Management" http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.149.7296 visited on 28/04/2011

[20] N. Duffield and F. L. Presti, "Network Tomography from Measured End-to-End Delay Covariance", IEEE/ACM Transactions on Networking, vol. 12, no. 6, 2004, pp. 978–992.

[21] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, "Network Loss Tomography using Striped Unicast Probes," in IEEE/ACM Transactions on Networking, vol. 14, no. 4, 2006, pp. 697–710.

[22] CAIDA, "The Skitter Project," http://www.caida.org/tools/measurement/skitter/, 2007.

[23] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP Topologies with Rocketfuel,” in Proceedings of ACM SIGCOMM Conference, Pittsburgh, PA, August 2002.

[24] C. Jin, H. Wang, and K. Shin, “Hop-Count Filtering: An Effective Defense Against Spoofed Traffic,” in Proceedings of IEEE INFOCOM Conference, San Francisco, CA, April 2003.

[25] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, USC/Information Sciences Institute, September 1981.

[26] William Stallings. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. Ad- dison-Wesley, Reading, MA, USA, third edition, 1999.

[27] J. B. Postel. User datagram protocol. RFC 768, Internet Engineering Task Force, August 1980.

[28] J. Schönwälder, A. Pras, and J.P. Martin-Flatin. On the future of internet management technologies. IEEE Communications Magazine, 41(10):90–97, October 2003.

[29] Information technology – Open Systems Interconnection – Basic Reference Model. ISO/IEC 7498.

[30] Marshall T. Rose and K. McCloghrie. Structure and identification of management information for TCP/IP-based internets. RFC 1155, Internet Engineering Task Force, May 1990.

[31] K. McCloghrie and Marshall T. Rose. Management information base for network management of TCP/IP-based internets:MIB-II. RFC 1213, In- ternet Engineering Task Force, March 1991.

[32] J. D. Case, M. S. Fedor, M. L. Schoffstall, and C. Davin. Simple network management protocol (SNMP). RFC 1157, Internet Engineering Task Force, May 1990.

[33] J. Schönwälder. Simple network management protocol over transmission control protocol transport mapping. RFC 3430, Internet Engineering Task Force, December 2002.

[34]] D. Perkins and E. McGinnis. Understanding SNMP MIBs. Prentice-Hall, Upper Saddle River, NJ, USA, 1997.

[35] "Intermapper" http://www.intermapper.com/products/intermapper visited on 17/02/2011

[36] "Pandora Fms" http://pandorafms.org visited on 19/02/2011

[37] "NetXms" http://www.netxms.org visited on 25/02/2011

[38] B. Donnet, T. Friedman, "Internet Topology Discovery: A Survey," IEEE Communications Surveys & Tutorials, Vol. 9, No. 4, 4th Quarter 2007, 56~69.

[39] Hwa-Chun Lin, Shou-Chuan Lai, Ping-Wen Chen, "An Algorithm for Automatic Topology Discovery of IP Networks", Proceedings of IEEE, ICC, 1998, pp 97-103.

[40] Research on the Theory of SNMP and the Technology of SNMP Programming, IEEE 2010, pp 23-25.

[41] K. McCloghrie and F. Kastenholz. The interfaces group MIB. RFC 2863, Internet Engineering Task Force, June 2000.

[42] Version 2 of the protocol operations for the simple network management protocol (SNMP). RFC 3416, Internet Engineering Task Force, December 2002.

[43] Private Enterprise Numbers, ftp://ftp.isi.edu/in-notes/iana/assignments/enterprise-numbers, visited on 15/3/ 2011.

LIST OF PUBLICATION

Communicated

- A.Mahajan, S.Khajuria, H.Joshi, A.K Verma, "SNMP, ICMP: A Collaborative Approach to Network Discovery and Monitoring" International Conference on computer Science and Information Technology (CSIT -2011), Bangalore, July 24-25, 2011.
- A. Mahajan, H.Joshi, A.K. Verma "SNMP based Network Discovery and Monitoring for SCADA", National Conference on "Emerging Trends in Computing and Information Technology (NCETCIT 2011), New Delhi, September 9,2011.