Design and Development of ICMP 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
SAHIL KHAJURIA
(Roll No. 800932018)

Under the supervision of:

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

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 ICMP 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:

(Sahil Khajuria)

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 Deptt. Thapar University

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

Crompton Greaves Ltd.
Global R & D Centre
Kanjur, Mumbai - 400042
India

Countersigned by

(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 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 members of DARC (CGL) team for their full support.

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.

Sahil Khajuria

(800932018)

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, SCADA.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

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
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

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 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 [1] is very important System for Monitoring and Management of Electric Stations with real time data logging and reporting Functionality.

Fig 1.1: 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.

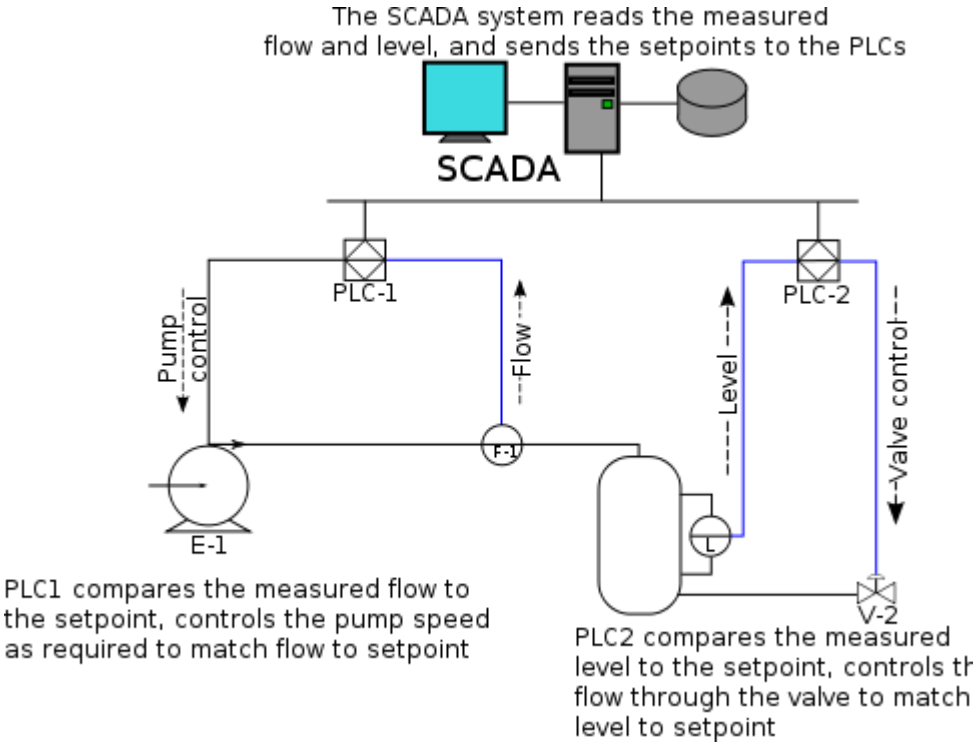


Fig1.2 SCADA Functioning [1]

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

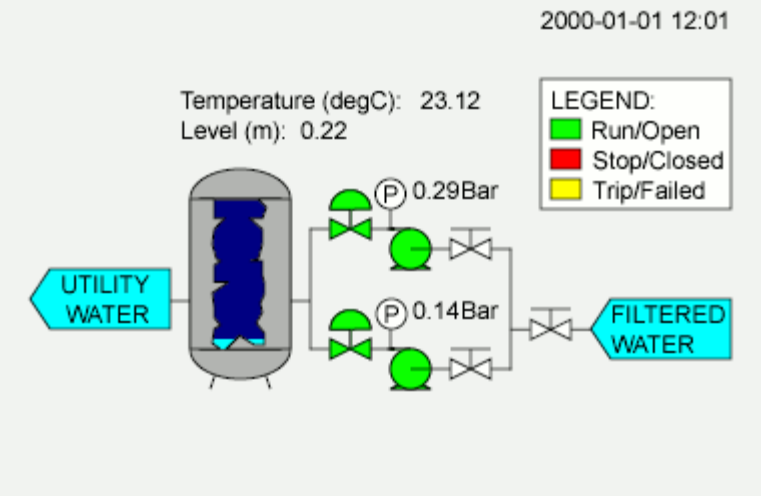


Fig 1.3: Typical Basic SCADA Animations [1]

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 solutions

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.

- **Remote Terminal Unit (RTU)** [2]**:** 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 control the flow of the liquid

- **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 centres. 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 [4] 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".

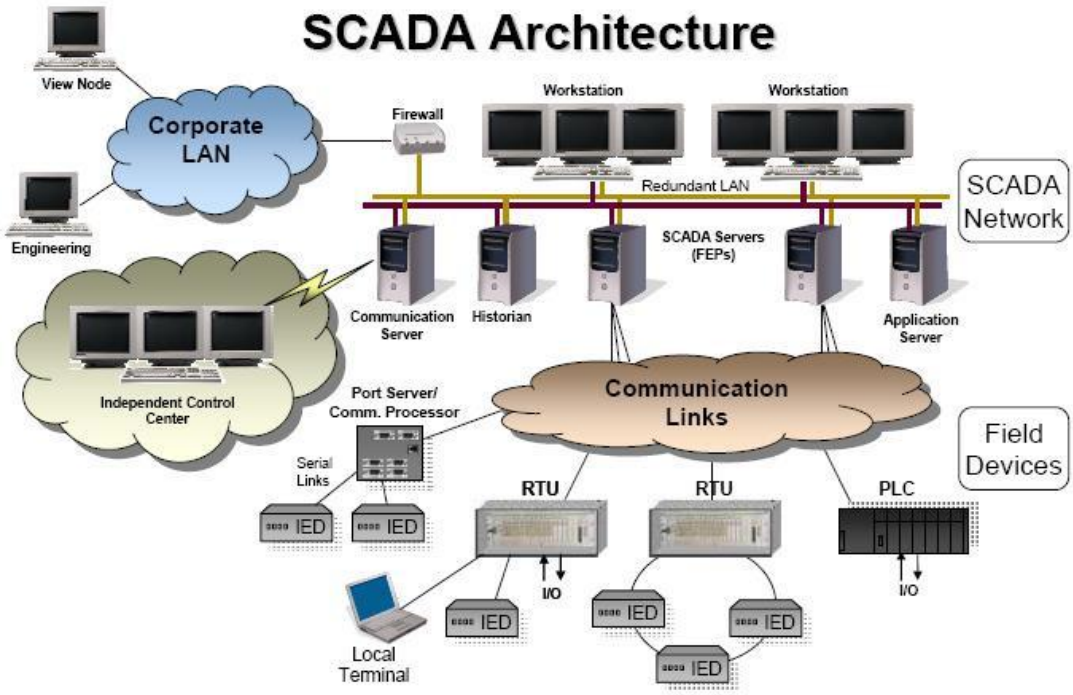


Fig 1.4: SCADA Architecture

SCADA systems have evolved through 3 generations as follows:

- **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 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.

- **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.

- **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 [5, 6]:

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 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
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 op- erate 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] 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.

- **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.

- **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.

- **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.

- **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

Network mapping [9, 15] 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.

- **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. 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" 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 1 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.
- **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 un-discoverable 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 environment 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 ICMP/trace route [17] utilities. ICMP [16] 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 measurement 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 simple 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 an 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. Unfortunately, a finite duration passive measurement campaign will likely result in incomplete measurements, because packets from each host are typically only observed at a subset of the monitors. 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.

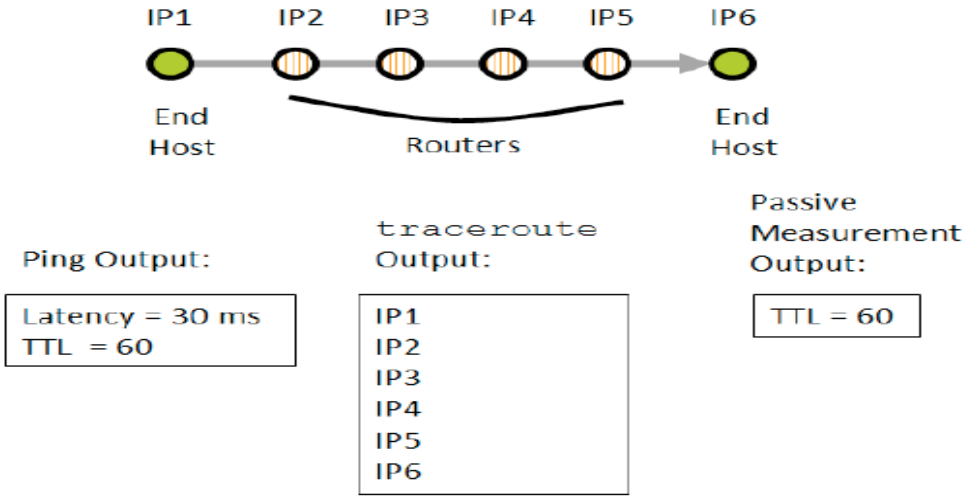


Figure 2.1: Example of the three Internet measurement types (Ping, traceroute, and passive where the monitor is at IP6) between two points in the network, where a time-to-live (TTL) value of 60 indicates that there are 64 − 60 = 4 routers between the two end hosts in the network.

2.3 Internet Control Message Protocol and Application:

The Internet Protocol (IP) [25] is used for host-to-host datagram service in a system of interconnected networks called the Catenet. The network connecting devices are called Gateways. These gateways communicate between themselves for control purposes via a Gateway to Gateway Protocol (GGP) [26]. Occasionally a gateway or destination host will communicate with a source host, for example, to report an error in datagram processing. For such purposes this protocol, the Internet Control Message Protocol (ICMP), is used. The **Internet Control Message Protocol** (**ICMP**) [16] is one of the core protocols of the Internet Protocol Suite. It is chiefly used by the operating systems of networked computers to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. ICMP [16] can also be used to relay query messages. ICMP differs from transport protocols such as TCP and UDP in that it is not typically used to exchange data between systems, nor is it regularly employed by end-user network applications (with the exception of some diagnostic tools like ping and traceroute). ICMP for Internet Protocol version 4 (IPv4) is also known as ICMPv4. IPv6 has a similar protocol, ICMPv6.

ICMP, uses the basic support of IP as if it were a higher level protocol, however, ICMP is actually an integral part of IP, and must be implemented by every IP module. ICMP messages are sent in several situations: for example, when a datagram cannot reach its destination, when the gateway does not have the buffering capacity to forward a datagram, and when the gateway can direct the host to send traffic on a shorter route. The Internet Protocol is not designed to be absolutely reliable. The purpose of these control messages is to provide feedback about problems in the communication environment, not to make IP reliable. There are still no guarantees that a datagram will be delivered or a control message will be returned. Some datagram may still be undelivered without any report of their loss. The higher level protocols that use IP must implement their own reliability procedures if reliable communication is required. The ICMP messages typically report errors in the processing of datagrams. To avoid the infinite regress of messages about messages etc., no ICMP messages are sent about ICMP messages. Also

ICMP messages are only sent about errors in handling fragment zero of fragmented datagrams. (Fragment zero has the fragment offset equal zero).

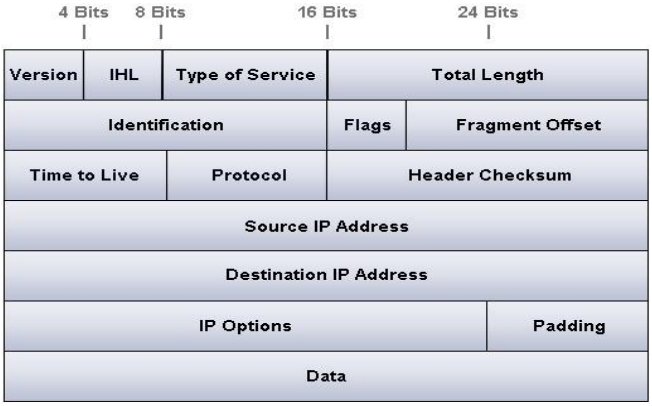


Fig 2.2: IP HEADER [26]

2.3.1 ICMP Functions [27]:-

- **Announce network errors,** such as a host or entire portion of the network being unreachable, due to some type of failure. A TCP or UDP packet directed at a port number with no receiver attached is also reported via ICMP.
- **Announce network congestion.** When a router begins buffering too many packets, due to an inability to transmit them as fast as they are being received, it will generate ICMP *Source Quench* messages. Directed at the sender, these - messages should cause the rate of packet transmission to be slowed. Of course, generating too many Source Quench messages would cause even more network congestion, so they are used sparingly.
- **Assist Troubleshooting.** ICMP supports an *Echo* function, which just sends a packet on a round--trip between two hosts. Ping, a common network management tool, is based on this feature. Ping will transmit a series of packets, measuring average round--trip times and computing loss percentages.
- **Announce Timeouts.** If an IP packet's TTL field drops to zero, the router discarding the packet will often generate an ICMP packet announcing this fact. TraceRoute is a tool which maps network routes by sending packets with small TTL values and watching the ICMP timeout announcements.

2.3.2 ICMP segment structure

2.3.2.1 Header

The ICMP header starts after the IPv4 header. All ICMP packets will have an 8 byte header and variable sized data section. The first 4 bytes of the header will be consistent. The first byte is for the ICMP type. The second byte is for the ICMP code. The third and fourth bytes are a checksum of the entire ICMP message. The contents of the remaining 4 bytes of the header will vary based on the ICMP type and code.

ICMP error messages contain a data section that includes the entire IP header plus the first 8 bytes of the packet that generated the error message. The ICMP datagram is then encapsulated in a new IP datagram [28].

MAC header	IP header	ICMP header	Data

Fig 2.3: MAC Frame [28]

8	16	32bit
Type	Code	Checksum
Message Body		

Fig 2.4: ICMP Header [28]

- **Type** - ICMP type as specified below.
- **Code** - Subtype to the given type.
- **Checksum** - Error checking data. Calculated from the ICMP header +data, with value 0 for this field. The algorithm is the same as the header checksum for IPv4.
- **Message Body/Rest of Header** - Four byte field. Will vary based on the ICMP type and code.

Registry:

Type	Name	Reference
0	Echo Reply	[RFC792]
1	Unassigned	[JBP]
2	Unassigned	[JBP]
3	Destination Unreachable	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
6	Alternate Host Address	[JBP]
7	Unassigned	[JBP]
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Solicitation	[RFC1256]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]
15	Information Request	[RFC792]
16	Information Reply	[RFC792]
17	Address Mask Request	[RFC950]
18	Address Mask Reply	[RFC950]
19	Reserved (for Security)	[Solo]
20-29	Reserved (for Robustness Experiment)	[ZSu]
30	Traceroute	[RFC1393]
31	Datagram Conversion Error	[RFC1475]
32	Mobile Host Redirect	[David Johnson]
33	IPv6 Where-Are-You	[Bill Simpson]
34	IPv6 I-Am-Here	[Bill Simpson]
35	Mobile Registration Request	[Bill Simpson]
36	Mobile Registration Reply	[Bill Simpson]
37	Domain Name Request	[RFC1788]
38	Domain Name Reply	[RFC1788]
39	SKIP	[Markson]
40	Photuris	[RFC2521]
41	ICMP messages utilized by experimental mobility protocols such as Seamoby	[RFC4065]
42-255	Reserved	[JBP]

Fig 2.5: ICMP Message Types [29]

2.3.2.2 Padding data

Padding data follows the ICMP header (in octets):

- Windows "ping.exe" adds, by default, 32 bytes of padding
- The Linux "ping" utility adds, by default, 56 bytes of padding

According to **M.F Schwartz, D.H Goldstein** [30] ICMP provides several discovery mechanisms that can be used in discovering characteristics of a network. The biggest drawback to its normal use is that normal users on most UNIX systems are not permitted to generate ICMP packets, largely because ICMP packets can cause problems with network and host loading if not used carefully.

The ICMP Echo Request and Echo Reply messages can be used in several ways to determine information about a network. For example, the UNIX "ping" utility sends a sequence of ICMP Echo requests to a host. Properly configured hosts will send back ICMP Echo reply packets. Though ICMP Echo Request packets are sent sequentially to number of hosts in a network but a broadcast packet can be sent ("with broadcast address") to discover all packets in LAN. The Main problem with this approach is that it causes significant number of collisions on the Ethernet that some of replies are lost. In addition, on networks that already have problems with "broadcast storms", this may cause even severe storms. This problem can be reduced by setting the Time-To-Live(TTL) field in the packet header to small value, so that broadcast storms last a short time.

Van Jacobsen's "traceroute"[17] utility also uses ICMP Echo Request packets, but sequentially increments the TTL through the values 1,2,3,etc, causing the packet to be returned to the originating host with an ICMP "TTL Expired" message by each router along the chain from source to destination. This allows the source host to discover the route that a normal packet would likely take from originating machine to the ultimate destination (assuming the routes do not change frequently). This is one of the best mechanisms for determining network structure because it is implemented in a majority of routers and because it does not require knowledge of "Private" information on the Router.

The "Pong" [30] utility uses the ICMP Echo Reply packets combined with "loose source routing" to specify that the packet must pass through the "destination" node on the way to the ultimate destination, which is originating node. This tool uses the "Route Record" option of IP to record the addresses of the routers through which the packet travels from source to destination and back to source. This tool suffers form the fact that many systems do not implement the "Route Record" IP option, and that many also do not

handle “loose source routing”, Moreover, the number of slots reserved in the IP packet header for recording route hops is smaller than the current diameter of the Internet.
In addition ICMP also helps in determining the Subnet Mask of a network using Address Mask Request and Address Mask Reply.

J. Schonwalder & H. Langendorfer [31] describes a discovering algorithm which mainly uses the ICMP protocol [16] to discover the structure of a network. This has the advantage that every device on the IP network can be detected since every IP implementation must support the ICMP protocol. Additional information is retrieved from the Domain Name System (DNS) if available. The Algorithm is discussed below:

How Network Discovering Works

The network discovering algorithm is divided into nine steps. The first four steps send probing packets to gather data while the remaining steps are used for data analysis. The algorithm assumes an implementation which will be able to send request packets in a round-robin fashion while waiting for outstanding reply packets to arrive or timeout. This way the total time needed for a given address space remains constant, regardless of the number of responding hosts.

1. Determine IP addresses in use by sending ICMP echo request packets to every address of the given address space. We use a sequential approach since directed broadcast ICMP echo requests tend to cause lots of collisions or even broadcast storms due to errors in some IP implementations. Some IP router even remove incoming broadcast packets reducing the usefulness of directed broadcasts.
2. Trace the routes to all IP addresses discovered in step one using the Van Jacobsen algorithm [17]. The traces are stored for later analysis. Gateways showing up during the traces are added to the IP address list.
3. Determine the network mask for each IP address using the ICMP mask request. The network masks are saved for later analysis.
4. For every IP address, send an UDP packet to an unused port and save the IP address contained in the port unreachable reply packet. Some multi-homed devices respond to an

incoming packet addressed to one of the remote interfaces with a packet containing the IP address of the incoming interface. The returned address is stored for later analysis.

5. Identify networks and subnetworks. Class A, B and C networks are easily recognized by examining the IP addresses. Subnetworks are a bit tricky. First collect all potential subnets based on the netmasks returned in step 3. Afterwards, check if the majority of all members of a potential subnet has reported a suitable netmask. This two level approach is needed to handle incorrect netmasks properly.

6. Identify multi-homed machines based on the traces and the address contained in the port unreachable reply packet of step four. Comparing the Domain Names of the IP addresses gives additional hints to multi-homed machines since the Domain Name Service often contains records with the same name for each interface of a multi-homed host.

7. Connect IP addresses to the networks identified during step six. Gateways are connected based on the traces. We found that gateways often return a different IP address when being traced. Therefore we skip the last hop of traces that end at a gateway machine.

8. Merge the IP addresses of multi-homed machines. This step cleans up duplicate information stored for each interface of a multihomed machine.

9 To the current map in editor add all discovered objects to the map that do not exist yet. Hosts and gateways are mapped to node objects, networks to network objects and IP interfaces to link objects.

The above algorithm can be used to discover routing traces by initializing the list of IP addresses and starting the algorithm at step two. This is a very nice extension to the traceroute program [17]. The resulting map lets us easily identify machines where branches join that are important for our site.

Jiang Wei-Hua, Lei-wei Hua, Du Jun [32] discuss the application of ICMP for network Scanning and monitoring, as suggested in paper there are three levels of network scanning, first to detect whether the host is alive; second to detect the OS of targeted Host; third, to recognize certain application program or version of specified service.

1. **Host Detection by Using ICMP Protocol**:

The purpose of host detection is first to detect whether the host is still alive and then to further find out the possible protocol observed by that host. When using ICMP for host detection, Ping is the most frequently method for a single host, Fping (fast ping) for multiple hosts and broadcast Ping for all hosts on a subnet. If the target responds to these methods, it shows that the host is alive. Otherwise, the host is switched off or a firewall has been set up and access is forbidden. In order to find the possible protocol observed by the host, the protocol field of IP header should be taken full advantage of when designing the report. In version IPV4 [RFC 791] [26], there is a "protocol domain" in the IP header which uses eight digit code to stand for the upper protocol. By using these protocol codes, we can freely fill in the protocol field in IP header of original socket with protocol codes. In doing so we can construct a special data packet and send it to target host. According to returned data, we will be able to tell whether the target host is still alive or not. We may also get a bit further to recognize the protocol observed by target host.

2. OS Fingerprint Recognition [32, 33]**:**

For intruders, if they succeed in acquiring information about the type of OS used by the host, they can try the loopholes in the OS one by one and therefore, they can save a lot of time and improve the scanning efficiency and accuracy. There are many ways to recognize the OS and a popular but complicated one is the fingerprint recognition method based on TCP/IP protocol. This method recognizes the type of the OS by the subtle differences in the definition of the protocol given by different OS. According to different ways of realization, it falls into two types, namely, active detection and passive detection. **Active OS detection** means that the source host sends specified type of date packets to the target host. Certain field of these data packets includes the characteristics of the OS. The returned packets can show the type of the OS or specify the OS by comparing the OS fingerprint database with the corresponding value of certain field in the data packets. While in the **passive OS detection system**, the source host does not need to send detective data packets. It passively hunts reports sent and received by the target host and then finds out the corresponding type of OS by detecting the value of the corresponding field and consulting the fingerprint database. Although these two methods are different in

their realization mechanisms, they are similar in handling and analyzing data packets. In terms of technology, it is categorized into **TCP and ICMP** technology. Comparatively speaking, the **latter** has the following advantages. First, it has a high recognition performance and it is particularly precise in recognizing the Windows OS. Second, it is simple to realize. You just need to send several logically related data packets and as has been experimented, the number of packets sent is less than four. In order to use ICMP for OS detection it is Imperative to set certain IP header Fields.

ID Field: Most OS return with their own ID number and some OS, such as Linux machine based on 2.40--2.4.4 kernel will set the IP identifier to zero in the ICMP query request and reply information. DF Field: Some TCP/IP stacks will set DF digits in **a** wrong ICMP data packet. Others'(such as Linux) will copy all the eight digits and set some of them to zero. The rest will ignore the original DF and set some value related to itself. TTL Field: There are two independent values in the IP TTL field of the ICMP data packet, one is ICMP query Information and the other is query reply setup. The TTL value of the reply packet will reduce by one when passing through each router from the target host to the source host. In accordance with the TTL value in the report and with the reference to the statistics of the TTL value of all types of OS from the following URL h**ttp:/www.switch.chltt-default.hm1,** we can speculate on the original TTL of the target host and then find out the type of its OS. **TOS** Field: Generally speaking, as provided for by RFC 1349, the ICMP reply information should use the same TOS field value in the corresponding ICMP request information. But some OS do not abide by this principle. TOS service field includes three fields, first, a priority field (WC 791) which is three-digit long and has eight priority levels; second, service type field which is four-digit long and describes the type of service operating on the network. It includes the minimal delay, the maximal throughput, the highest reliability and the lowest cost. Thud, the last useful digit field must be set to zero. And the TOS digits and the last field will be replaced in implementing the service type mechanism. If all the four digits are zero, it means ordinary service. We have combined all these fields together and found out some methods for OS recognition by using Ping, Snort and other tools and with the reference to all types of ICMP reports.

1.Comhine the address mask request report with the fragments symbol DF If the target host responds to the address mask request report, then it is Solaris, HP-UXlI.0X, ULTRIX, OpenVMS, Win 95/98/98SE/NT (version lower than SP4). For these OS, we send the address mask request report with fragments symbols to them again. If in the response report, the returned value is address mask and then the systems are ULTRIX, OpenVMS and Win95/98/98SE/NT (version lower than SP4). Then we will differentiate them by the TTL value. When TTL equals to 255, the systems are ULTRIX and OpenVMS. When TTL equals to 128, the systems are Win95/98/98SE/NT (version lower than **SP4).** Generally speaking, systems with no response are Solaris, Hp-UX11.0x.

2. Combine the return request report with the fragment symbol DF If the target host is sent with the ICMP return request report with the setup of DF symbol, there are two types of response reports. One is not to return the DF symbol. Then the common systems are Linux **2.2.x,** ULTRIX, and Novell Netware. All these common systems can be further differentiated by TTL value. When TTL equals to 128, it is Novell Netware. When TTL equals to 255, it is Linux 2.2.x, ULTRIX. For Linux 2.2.x and ULTRIX, we will further send ICMP address mask request report, the one with response is ULTRIX and the one without response is Linux 2.2.x. For those OS that do not return the DF symbol, we will once again send the address mask request report with the DF digit setup. The OS which return the DF digits **is** Solaris, W-UXll.0X, OpenVMS and the OS which does return these digits is Win 98/98SE.

3. Apply TOS digits

(1). Send return report with precedence which is not zero there are two types of return reports: return report with the original precedence value and report with precedence which is zero. For the first type of OS, we will send the time-stamp with the precedence which **is** not zero for requesting report. In the return report, if the precedence is zero, it is usually Win 98/98SE and OpenVMS. Then we will make judgment according to the TTL value. When TTL equals to 225, it is OpenVMS. When TTL equals to 128, it **is** Win98/98SE/ME. Then we will further send address mask request report. If there is response, it is Win98D8SE. If there is no response, it is Win. **ME.** For OS whose precedence is set to zero in the return report, it is usually Win2000, Ultrix and

HPUX11.X. Then further judgment is made according to the TTL value. When 'ITL equals to 128, it is Win20M). When TTL equals to 225, it is Ultrix and HPUX11 .X.
(2) Send report with the type of service field which is not zero. There are two types of return reports: TOS is not zero and TOS is zero. For the OS whose ***TOS*** is not zero, we will further send time stamp with TOS which is not zero for requesting report. If the TOS is zero in the reply report, then it is Win95/98/98SE/ME. We will then make judgment according to the address mask request report. For the OS whose TOS is zero, it is usually Win2000, Ultrix and Novell Netware. As explained earlier, we will then make judgment according to the TTL value.

Xeu Cai, John Heidemann propose a technique based on Active probing using ICMP to classify Internet Address Blocks Used in the Network. Probes results are used to study address allocation strategies and to infer address usage (always be occupied or not in use, use by stable or frequently inaccessible hosts, etc.). Their Goal was to classify Internet address blocks based how addresses respond to frequent probes over the source of one week and to answer following questions:
Do groups of blocks show consistent patterns? How many vary? Groups of consistent patterns will allow clustering and classification of address usage with only external probes. What are the sizes of blocks that show consistent usage? These patterns will suggest how address blocks are managed at fine granularities (smaller than /24s). Can we map consistent, popular patterns to operational network usage? Can we identify groups of servers, cable or DSL customers, dial-up users, etc., by ping responses?

The authors confirm that while **ICMP can be blocked**, studies of a university and random addresses confirm that it is the most accurate method of active probing (more accurate than TCP) and solicits relatively few complaints. Our results here are based on data from June 2007 [10].
Pattern Analysis: They analyzed the usage of each IP address based on the survey replies. Several metrics were defined characterize addresses. First, for each address, consider it to be up when it responds to ICMP Echo request, and define up durations (Ui, for each uptime i) as the time from the first response until the next non-response.

(Precision of up duration's estimates is limited by 11-minutes probe interval.) Define availability as the sum of all up durations, normalized by total survey time. Define volatility as the number of up durations, normalized by the maximum number of possible state changes (half the number of probes). These metrics define an (A, V) plain of address blocks. We also consider the distribution of lengths of up durations, including mean, median and maximum. Clustering: Adjacent IP addresses are clustered with similar usage (based on the above metrics) into blocks. They are currently experimenting with different clustering algorithms to explore all possible blocks with prefix lengths from /29 through /24 (groups of 8 to 256 addresses). Classification: Blocks are classified obtained from clustering into five categories: servers, and blocks that are stable, intermittent, underutilized, or unclassifiable. We trained our classification by manual examination of hostnames and services in several hundred blocks, and then we verify that it works by evaluating it on hundreds of other, randomly chosen blocks.

Server block: highly available and stable (high A, low V) Stable block: stable, usually continuously up for more than 6 hours (low V , and high median U). Statically assigned addresses usually fall into this category.

Intermittent block: short up durations (low median U). For example, many DSL addresses are reassigned every few hours.

Underutilized block: low A values. Most wireless and some dial-up addresses are severely underutilized.

Unclassifiable block: unclassifiable due to too few responders. We frequently have difficulty classifying blocks of size /29 or smaller.

Considering this much analysis of ICMP applications as sufficient we analyze and compare some of already available software which aid into the function of network monitoring and management.

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 varies 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

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

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,

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

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

Chapter 3

PROBLEM STATEMENT

There are various monitoring solutions available in market but most of them demand a particular set of requirements to run and also most of them work on agent based monitoring. Thus there is a need to develop a network monitoring solution for integration with CG SCADA (Crompcada) using monitoring applications of ICMP which would provide the end user with the logical view of the local network and real time information of critical connected devices like RTU (Remote terminal Unit), IED (Intelligent Electronic Devices), PC, printers and simultaneous reporting in the form of alarms in case of some faults with the devices present in the network (i.e. if some device goes down, a visual Remark like popup or some alarm can be raised). This work is very challenging as it involves simultaneous handling of, creation of Graphic presentation of devices, visual remarks, network information processing and multithreading along with following attributes:

1. Agent Less network monitoring and so can be deployed to 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. Integration with Cromp SCADA.
6. Provides other relevant information like measuring the processor utilization of hosts, Packet Loss for links etc.
7. Easy to operate and configure.

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 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 and design 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 allow for future extensions, which was why the 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.

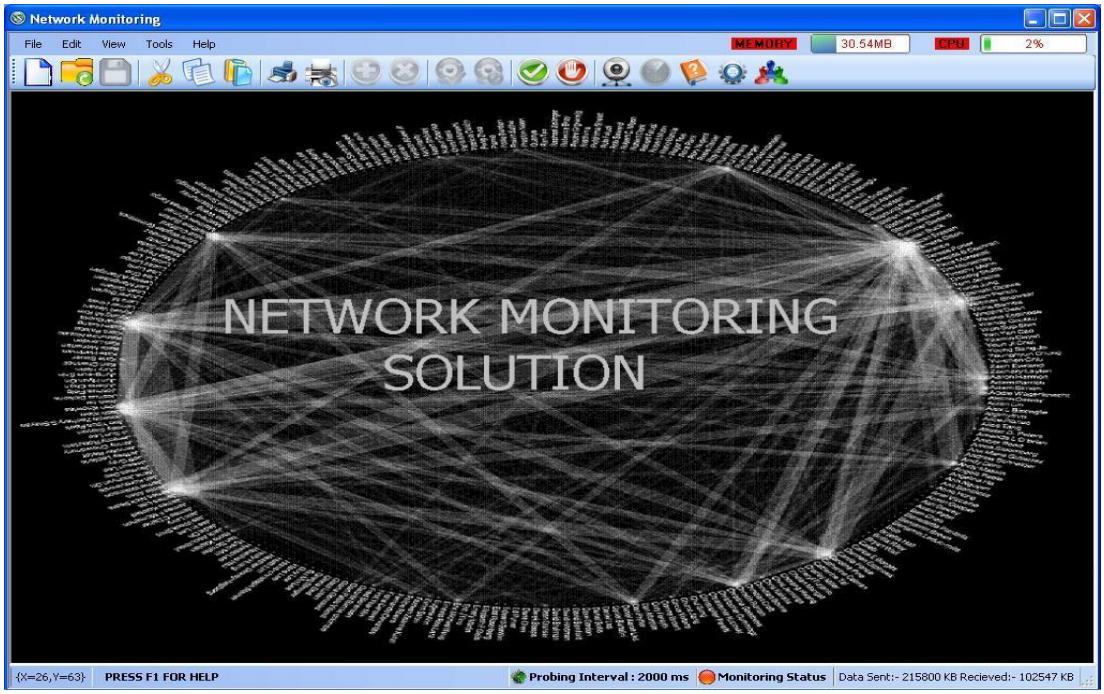


Fig 4.1: Application Main Window.

4.1.1 Programming language

An evaluation of suitable programming languages was performed in order to decide which one to use for the Monitoring Agent implementation. The evaluated languages consisted of Java and C++, C#.net and PHP. At first Java or C++ were favored since I personally had prior experience of these languages. Both these languages have support for object oriented design. Compiled C++ programs are usually very fast and efficient, while Java programs may be used on many different platforms. These benefits were however 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.

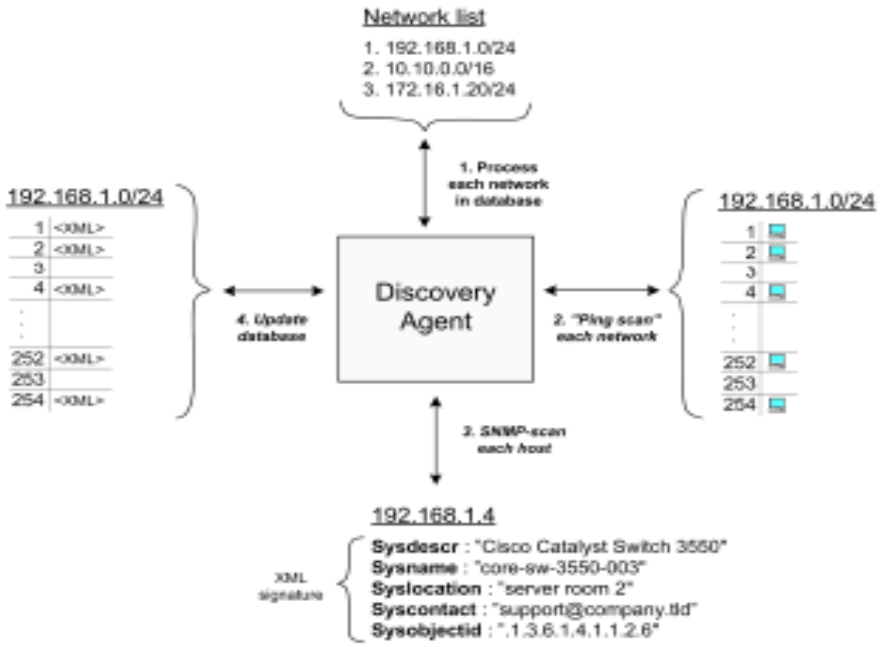


Figure 4.2: The Discovery process

4.1.2 Design tools

For the development of the source code the visual studio 2010 Express Edition environment was selected, with the C#.NET 4 plug-in [13]. This allowed for easy access to help commands, syntax highlighting, and debugging support. For a screenshot of C#.net environment, see Figure. 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.

4.2 Platform

The Network Monitor application running in the Windows 7 environment is developed for .NET 4, a software framework provided by Microsoft.

4.2.1 The .NET 4 Framework

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. For a screenshot refer Fig 4.3.

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 CG selected and aimed at developing full SCADA system in this platform. Although I had prior experience in C# 2.0 framework, developing a system in C# 4.0 was less of a problem.

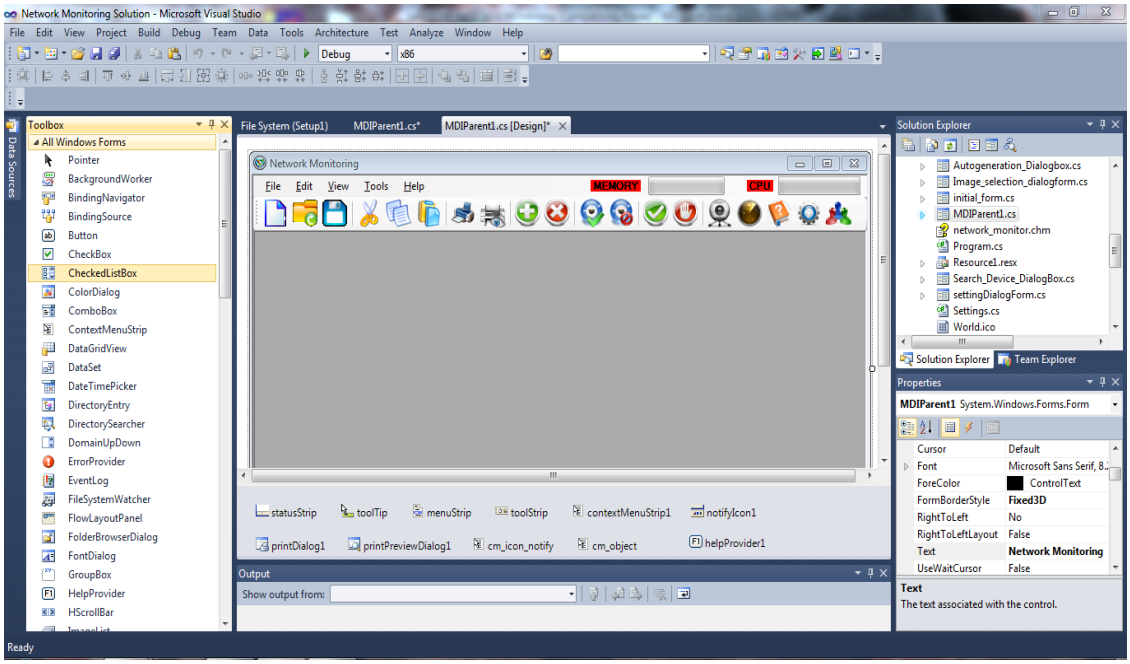


Figure 4.3: Visual Studio 2010 environment

4.3 Implementation

The Discovery/Monitoring Agent is implemented in Network Monitoring System and is 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 net 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 the testing Chapter 7. See also the UML activity diagram in Figure 4.2 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 is 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 net 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 net needs to be rescanned can be made. If it is time for a new scan of the current network, a net scanner thread is spawned.

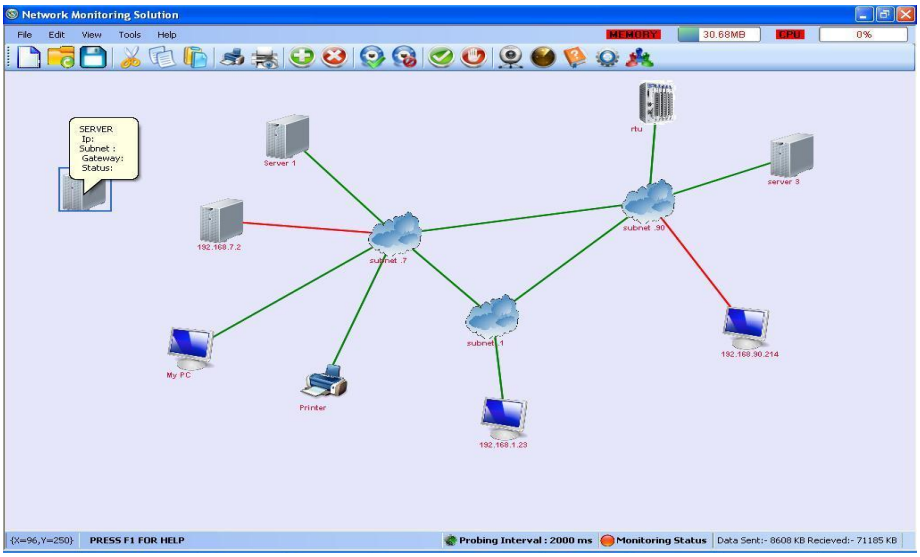


Fig 4.4: Network Scanner processing Device Information.

4.5 Network scanner

The net scanner thread scans a network, designated by a network address and a netmask, to detect which attached hosts that 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 is 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, directed towards specific addresses in a given network, in an attempt to generate a response. However, since the aim of the software is to detect hosts with standard configuration, and since the software must be able to scan a large number of hosts in a limited time, it was decided that a regular ICMP ECHO scan would suffice. 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. For ICMP Class refer to appendix A.1.

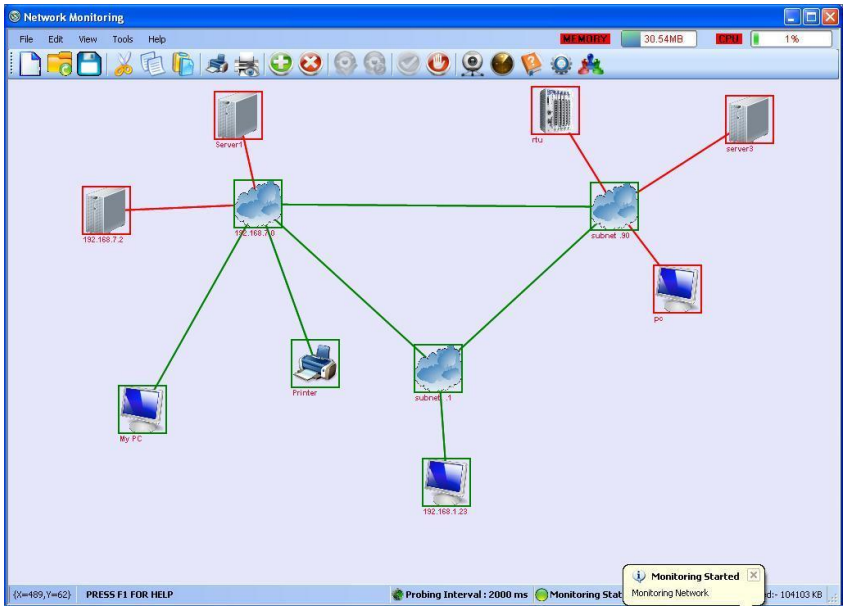


Fig 4.5: Network Scanner sending ICMP Echo requests.

Using this (Appendix A.3) class 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.

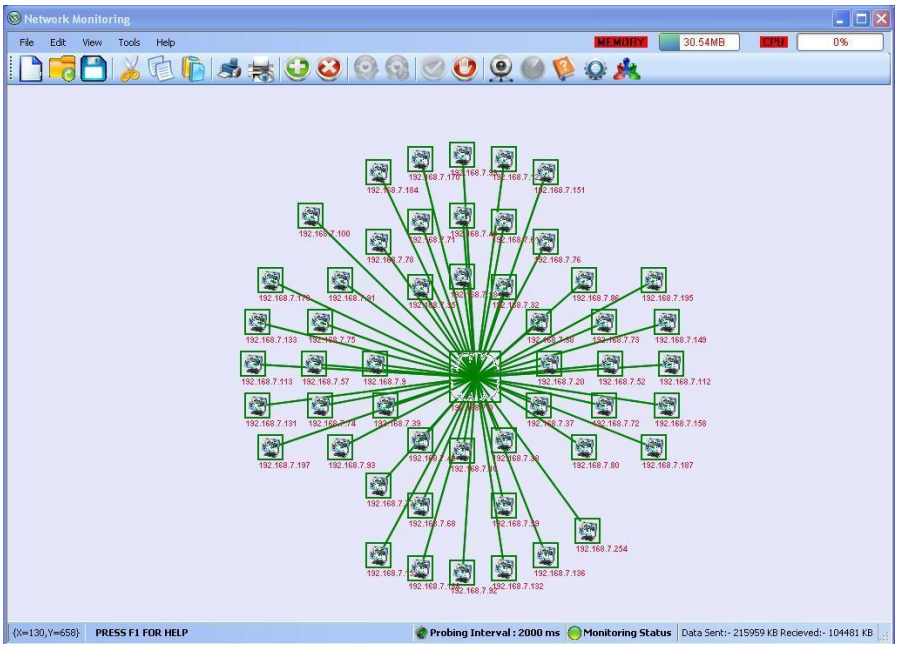


Fig 4.6: Devices Discovered and Monitored in a Subnet

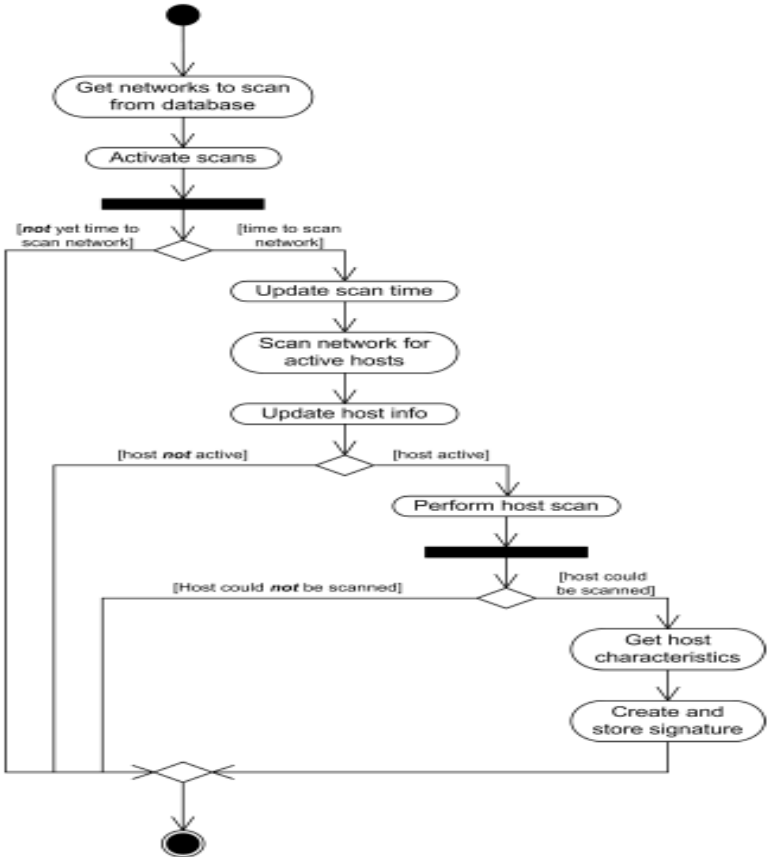


Figure 4.7: Discovery service UML activity diagram

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.

Fig 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 Inhertited properties of Control Class and its own properties like IpAddress,SubnetMask and DefaultGateway etc. GraphicsObject: 

For GraphicsObject Class code refer to Appendix A.2.

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.

```
{
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 is 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 GraphicsOBject Collection, providing basic functions for deletion, updating and addition of objects to Collection. GraphicsObjectCollection class in appendix A.3 provides a view of Implementation.

The created GraphicsObjects are added to GraphicsObjectCollection by Network Processor 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.

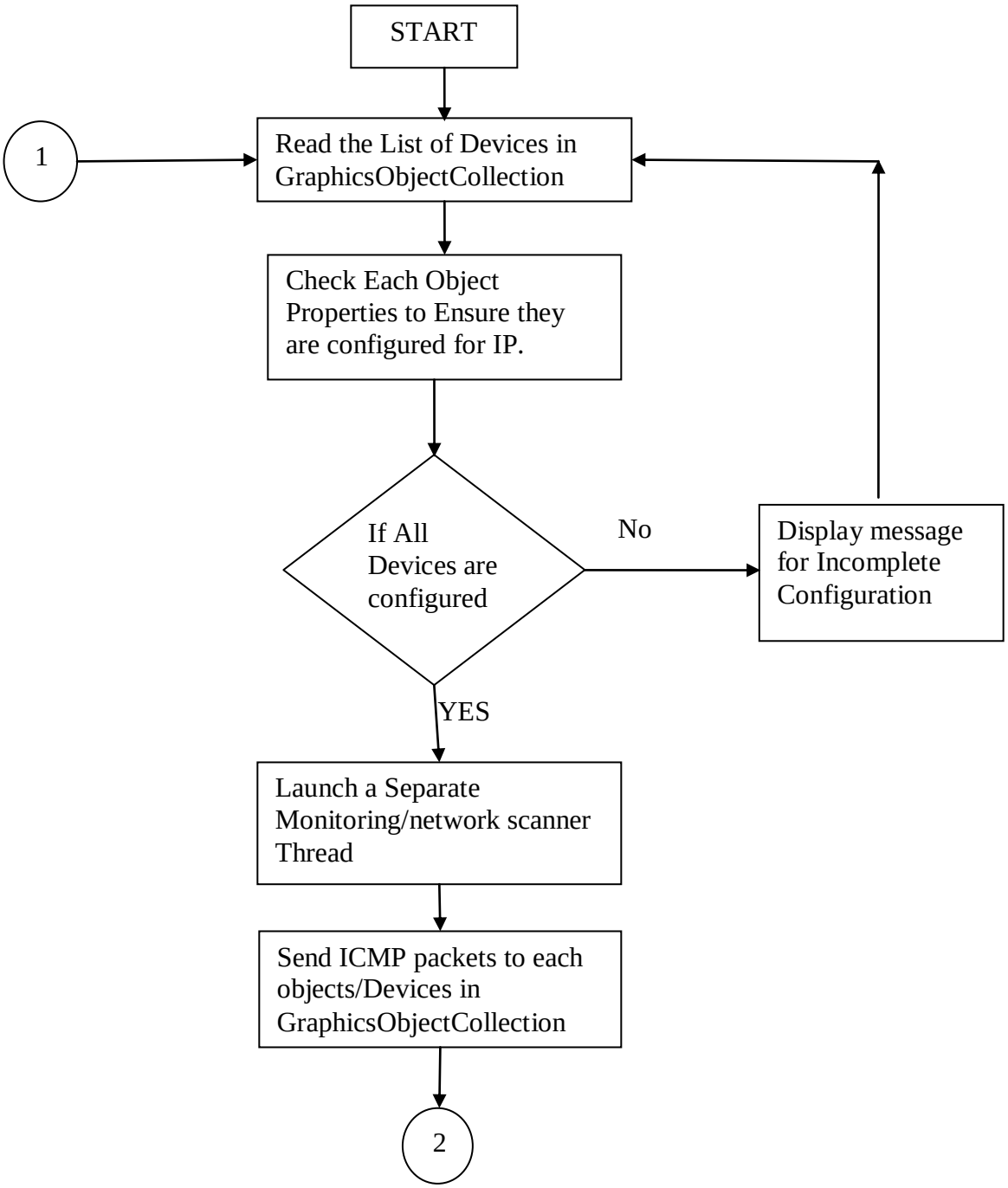
4.9.1 Design

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 runner/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.

In Figure 4.8 the process of the Topology Analysis Module is shown as a component of Complete Monitoring Process.



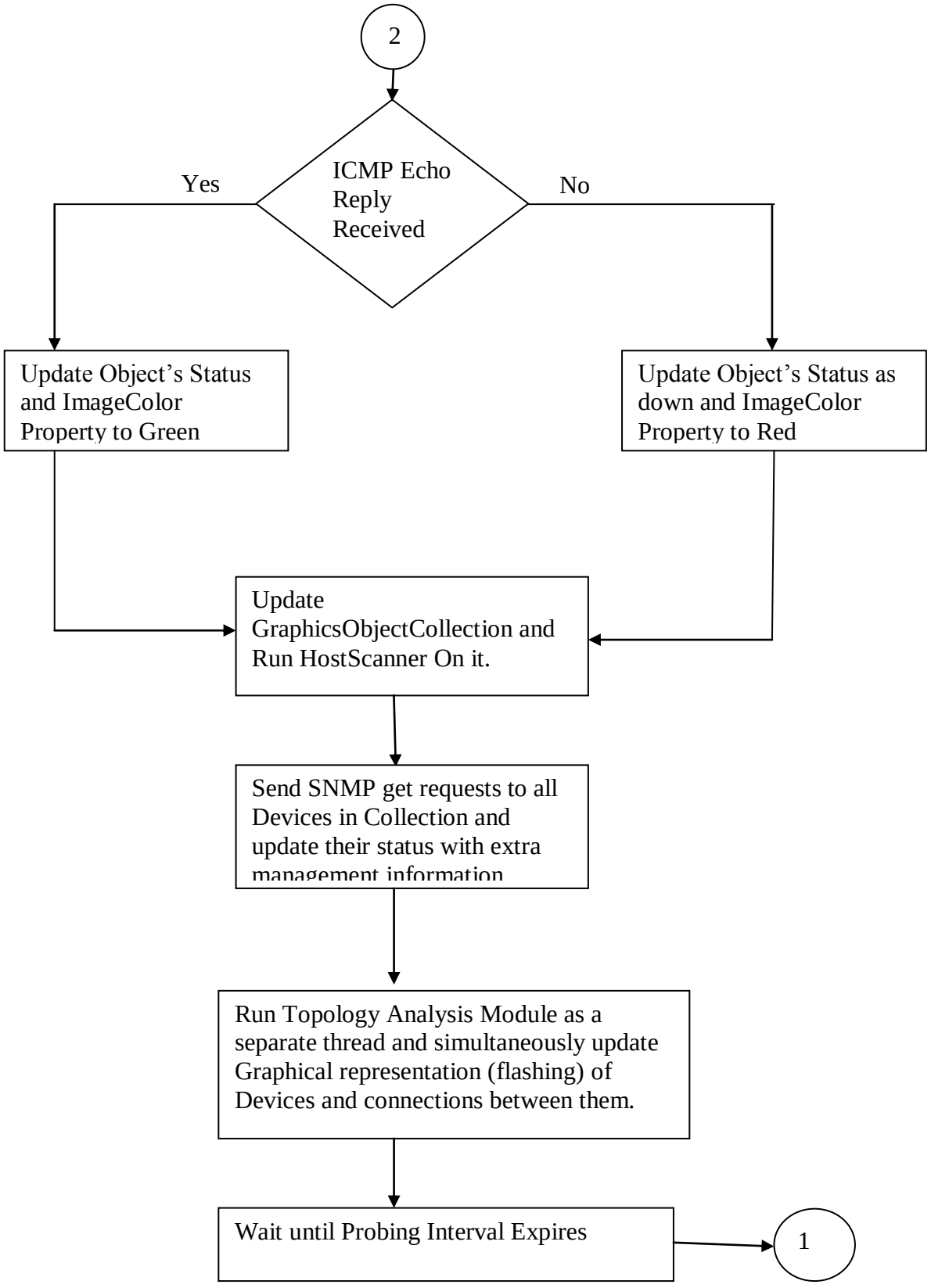


Fig 4.9: Flow Chart Describing Monitoring Process

4.9.2.1 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">iVBORw0KGgoAAAANSUhEUgAAADwAAAA
8CAYAAAA6/ ==</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

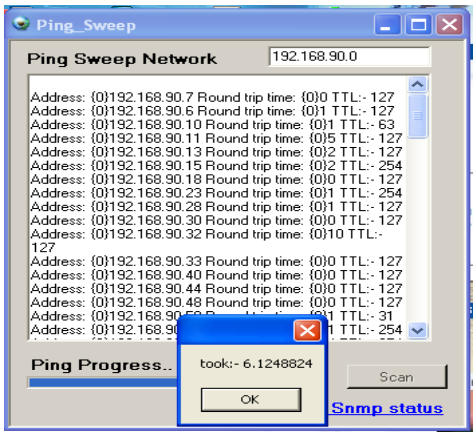
Results and Evaluation

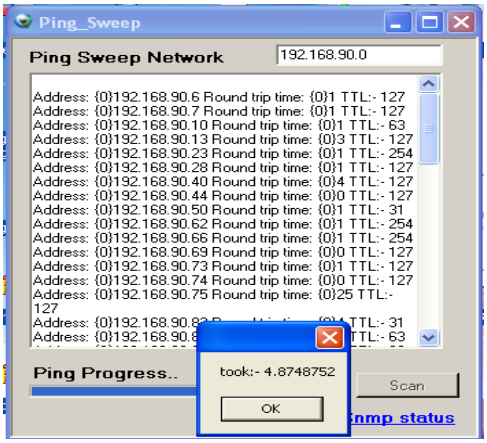
The Network Monitoring Tool was implemented and tested at Crompton greaves R& D department. The system was made to monitor devices located in different subnets and also discovery of subnets was done.

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 devices'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 milisec

data buffer =1 byte

(b) Time taken 4.87 sec

Hosts scanned:- 44

ICMP timeout : 100 milisec

data buffer =1 byte

Fig 5.1 Scanning Using MultiThreading

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

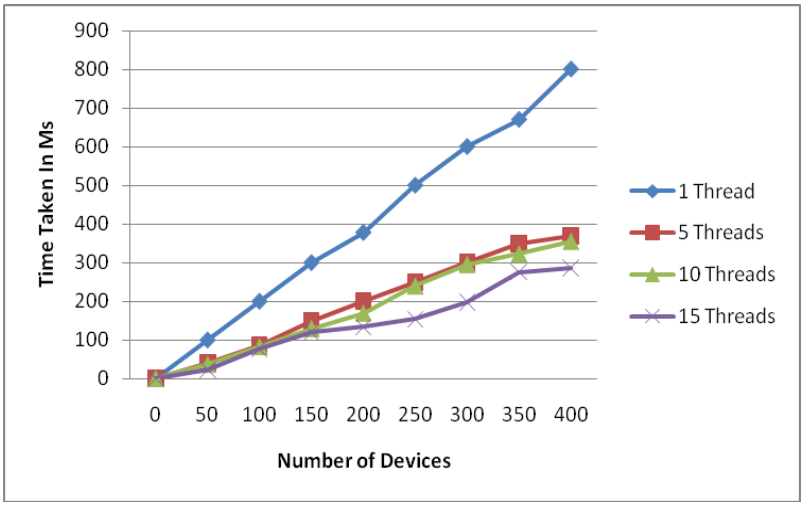


Fig5.2: Number of Devices Vs Discovery Time.

Fig 5.2 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.

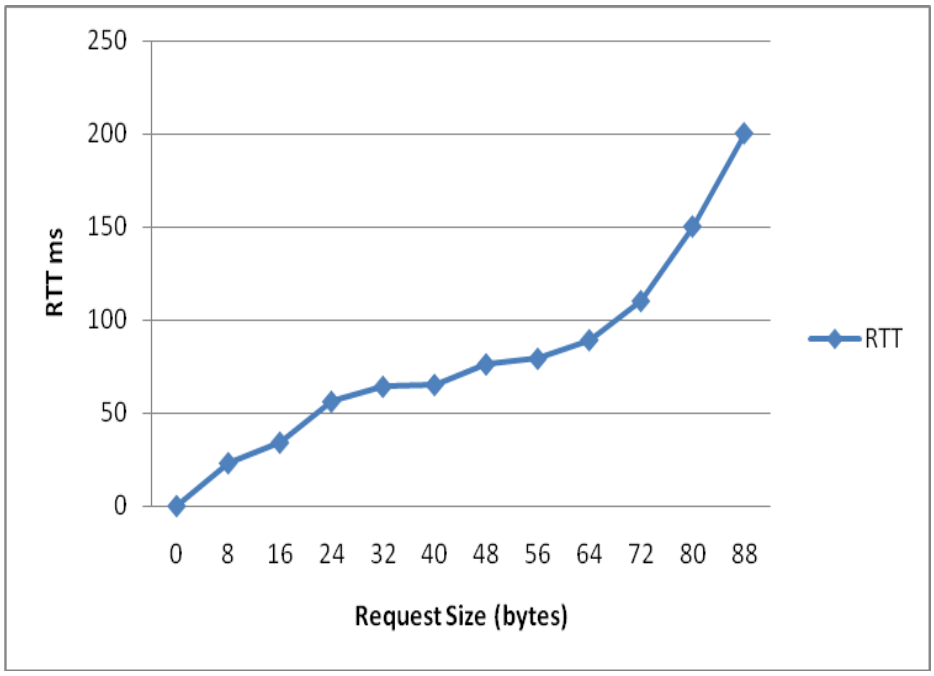


Fig 5.3: RTT vs Packet Size

Fig 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.

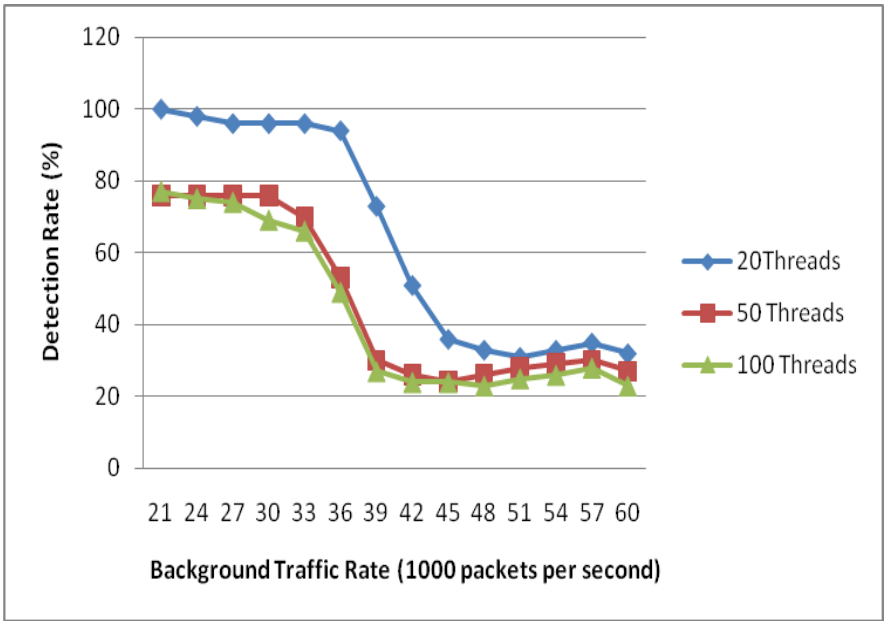


Fig 5.4: Average Detection rate Vs Background Traffic rate

Fig 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 an important means for network management. To design and develop an effective and useful tool for the network topology monitoring/discovery is an important aspect in developing a Network Monitoring Solution. The proposed monitoring system is designed to operate on a fast 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 such as SCADA.

APPENDIX A

CLASS CODE

A.1 ICMP Class:

```
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

      public void SetData(byte[] data, int offset, int length)
      {
         Data_ = new byte[length];
      {
         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; }
      }
```

```
            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);
        }
    }
}
```

A.2 Graphics Object Class:

```
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.GetType());
          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();
        }
```

```
    }
}
```

A.3 GraphicsObjectCollection Class:

```
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 An 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;
        }
    }
}
```

REFERENCES

[1] Higgs, M.A "Electrical SCADA systems from the operators perspective" in HICRCC (IEEE)10 Nov 1998 pages 3/1 – 3/4

[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 2/02/2011.

[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 3/03/2011

[15] “Network Mapping” http://en.wikipedia.org/wiki/Network_mapping visited on 14/04/2011

[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 15/04/2011.

[19] S.Aidarous, T.plevyak, “Principles of Network Management” http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.149.7296 visited on 15/04/2011

[20] N. Duffield and F. L. Presti, “Network Tomography from Measured End-to-End Delay Covariance,” in *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] Strazisar, V., "Gateway Routing: An Implementation Specification", IEN 30, Bolt Beranek and Newman, April 1979.

[27] “ICMP Functions” http://www.freesoft.org/CIE/Topics/81.htm visited on 17/04/2011

[28] “ICMP Header” http://www.networksorcery.com/enp/protocol/icmp.htm.visited on 17/03/2011

[29] “ICMP Message Types and Codes” http://www.iana.org/assignments/icmp-parameters visited on 17/03/2011

[30] Michael F. Schwartz, David H. Goldstein, Richard K. Neves, David C. M. Wood, An Architecture for Discovering and Visualizing Characteristics of Large Internets, Department of Computer Science, University of Colorado, Boulder, Colorado, Technical Report CU-CS-520-91, February 1991

[31]J. Schonwalder, H. Langendorfer “How to keep track of your network configuration,” Proc.LISA, pp.101–105, Nov. 1993.

[32] Jiang Wei-hua; Li Wei-hua; Du Jun, “The application of ICMP protocol in network scanning” in PDCAT 2003, Aug 2003 pages 904-906

[33] “Os Fingerprinting” http://www.phrack.org/issues.html?issue=57&id=7 visited on 11/05/2011

[34] “Active Probing to Classify Internet Address Blocks” www.isi.edu/~xuecai/.../classify_blocks_for_sigcomm_tech_report.pdf visited on 04/03/2011

[35] “Intermapper” http://www.intermapper.com/products/intermapper visited on 23/02/2011

[36] “Pandora Fms” http://pandorafms.org visited on 23/02/2011

[37] “NetXms” http://www.netxms.org visited on 23/02/2011

[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

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

LIST OF PUBLICATIONS

Communicated

- S.Khajuria, H.Joshi, A.K Verma, "ICMP Based Network Discovery and Monitoring for SCADA" International Journal of Computer Science & Information Technologies (IJCSIT), Tamilnadu, July 15, 2011.

- S.Khajuria, A.Mahajan, 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.