ICMP, SNMP: Collaborative Approach to Network Discovery and Monitoring

[1]Aman Mahajan ,[2]Haresh Joshi ,[3]Sahil Khajuria , [4]Anil k Verma

[1,3,4]CSE, Thapar University, Patiala, India
[2]Manager-Technology,CGL Mumbai, India
Email:[1]akverma@thapar.edu,[2]haresh.joshi@cgglobal.com,[3] contact.tosahil@gmail.com, [4]contacttoaman@gmail.com

Abstract: The algorithm and realization technique of network topology discovery is a very important technical indicator for a network management system's quality. Based on the knowledge of the network topology discovery theory, the research on ICMP and SNMP programming for Windows, this paper proposes a method of developing the common network topology discovery software which is a conglomerate of ICMP and SNMP protocols so as to monitor a given network and successfully manage it while taking into full advantage of above mentioned protocols.

Keywords: SNMP, ICMP, MIB, TCP, Topology, IP network.

I. INTRODUCTION

A Computer network forms a crucial component in this computer generation and it is imperative to create a technique to monitor and manage a computer network, taking into full advantage of available resources in a network. Network topology is basically a map of the network in which various network devices are interconnected and communicate with each other. The interconnection among devices can be based on any of the network topologies. Monitoring these devices by manually pinging them by entering the information in console window is a tedious process provided the network devices are many and the polling interval is small. So getting a real time update of device's status using above mentioned process is not feasible. So, a solution is required which automates the process of remote monitoring and management of network.

The knowledge and research on ICMP and SNMP protocols helped in automating the desired task. Following the assumptions that (i) the Internet protocol (IP) is a widely accepted and used protocol and all the system within a network are IP based and (ii) Non-IP compliant equipments, from different types and sizes, adopt some kind of mechanism that allows their monitoring or even their management in a broader sense. A tool in C#.NET has been developed to allow user to either create the desired network topology based on the prevalent network infrastructure and enter network devices configuration details or allow the network monitoring tool to self discover the underlying network topology. Monitoring process is then started which takes the IP list and sends ICMP echo request packets and accepts ICMP echo reply to determine device status.

II. ICMP

Internet Control Message Protocol (ICMP), documented in RFC 792[2], is a required protocol tightly integrated with IP. ICMP messages, delivered in IP packets, are used for reporting information related to network operation and faults. Of course, since ICMP uses IP. Currently there are many applications which

substantially make use of ICMP[1, 2, 3] for probing Devices available in network. Some of ICMP functions are to:

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.

Announce Timeouts. If an IP packet's TTL[1] field drops to zero, the router discarding the packet will often generate an ICMP packet announcing this fact. We map network routes by sending packets with small TTL values and watching the ICMP[2] timeout announcements so as to determine the hosts lying in the path until destination.

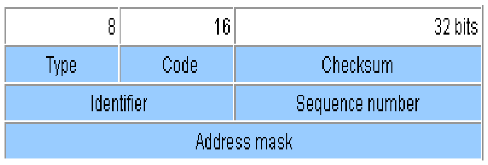

Fig: 1: ICMP header [2]

Figure 1 Shows the ICMP header, here one of important field is type of 8 bits and so can represent 2^8 = 256 types of messages. We extensively make use of type 0 and 8 i.e. ICMP (echo reply/request) messages to keep into track of various hosts alive in a given address space. An improvement is done

by implementing multithreaded ICMP Request capability in application. There is a chance that some hosts firewall may drop/reject ICMP packets. Though there are many suggested approaches [4, 9] to network monitoring relying solely on ICMP but ICMP packets can easily be filtered by firewalls. We use SNMP to query non responding hosts to determine if they are up along with other information.

III. SNMP

The Simple Network Management Protocol (SNMP) is a standard application layer protocol (RFC 1157) that allows a management station (the software that collects SNMP information) to poll agents running on network devices for specific pieces of information; it might report the server's processor utilization and memory usage. What the agents report is dependent on the device.

All SNMP-compliant devices include a specific text file called a Management Information Base (MIB). A MIB is a collection of hierarchically organized information that defines what specific data can be collected from that particular device. SNMP is the protocol used to access the information on the device the MIB describes. MIB compilers convert these text-based MIB modules into a format usable by SNMP management stations. With this information, the SNMP management station queries the device using different commands to obtain device-specific information.

There are five principal operations that an SNMP management station uses to obtain information from an SNMP agent [12]:

• Get-Request Operation: Managing process retrieves information from agent process.

• Get-Next-Request Operation: Request the next value in table, using with Get-Request, it can get each value in table.

• Set-Request Operation: Set one or more parameter value for agent process

• Get-Response Operation: agent process responds Get-Request with this message.

• Trap Operation: Agent process send messages on its own, informing managing process that something happened.

In the manager-agent paradigm, a Network Management System consists of a Network Management Station (NMS), also called monitor or manager that queries a set of agents (acts as a part of Inter-network Operating System (IOS)) for information describing the state of links, devices, protocol entities, and nodes. Agents collect operational data (e.g. performance parameters) and detect exceptional events (e.g. error rates exceeding thresholds). This information 'is kept in the Management Information Base (MIB). Agents may issue alarms to inform the NMS about an exception. The NMS and the agents communicate through a network management protocol. Applications based on the Simple Network Management Protocol (SNMP) [4, 5, 6] are currently widely available.

Consider a Local Area Network (LAN). The traditional approach to monitoring [7, 8] is to have a few managers, usually only one, organized in a tree, each of them responsible for querying a set of agents, and reporting to monitors in higher levels of the tree, as shown in figure 1. In these trees, agents are the leaves, and intermediate nodes are monitors that implement both an agent process (i.e., SNMP server) and a manager process (i.e., SNMP client).

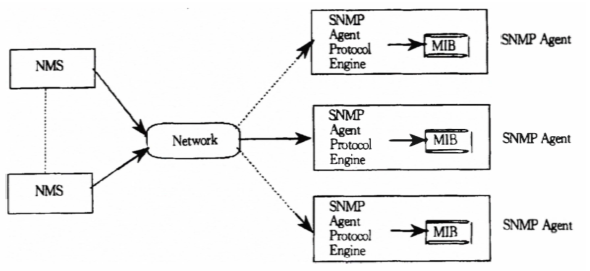
NMS
NMS
Network
SNMP Agent Protocol Engine
MIB
SNMP Agent
SNMP Agent Protocol Engine
MIB
SNMP Agent
SNMP Agent Protocol Engine
MIB
SNMP Agent

Fig 2: Relationship between managing process and agent process in SNMP[13]

IV. ALGORITHM

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

MIBs for Discovery

Our discovery mechanism is based solely on SNMP. Table 1 explains all the SNMP MIB objects required.

TABLE 1 MIB Information For Topology Discovery

MIB-II [10, 11]
sysServices, sysDescr,ifIndex, ifDescr, ifPhyaddress, ipForwarding, ipRouteNextHop, ipRouteType, ipAdEntAddr, ipAdEntNetMask, ipNetToMediaNetAddress, ipNetToMediaPhysAddress,

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

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

B. Send the ICMP packets with TTL [1, 2]field being set to "one" to all the IP addresses discovered previously. If reply address is not same as the address on which request was sent, it is the router address and add it to Router IP list.

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

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

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

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

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

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

Phase 3: Continuous monitoring of discovered devices

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

V. IMPLEMENTATION

We implemented the proposed algorithm in c#.net which provides the user with the capability to manually creating a network topology or run automated network topology discovery mode on a given system. We used C# with .NET framework 4.0 and GDI+ for graphical representation. We developed and tested our system on Windows XP with 2.80-GHz processor, Intel Pentium 4 CPU with 512 MB RAM. When the user selects the automated network topology discovery mode, the above mentioned algorithm gets executed and devices are represented accordingly on screen. Once system has discovered devices, it enters monitoring mode and there after continuously polls the network devices. The user can configure the polling interval as per its requirement. Polling is done to provide real time state of the connected devices, if any of the devices goes down; our system immediately detects it and simultaneously flashes up the corresponding device in red colour with in a fraction of a second.

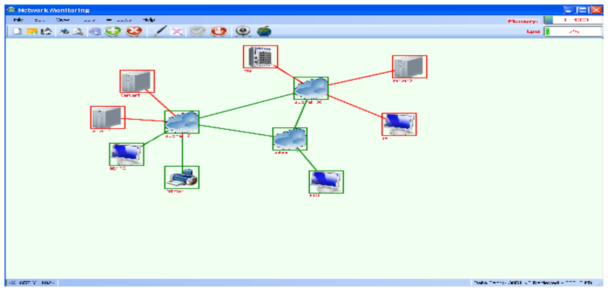
Fig 3: Manually created network topology

The system supports multithreading with ICMP echo requests being sent to devices using multiple threads so as to keep the delay to bare minimum level. It should also be noted that ICMP echo packet are sent with 1 byte of data so as to keep the network traffic generated by this implementation to its lowest possible value. The timeout interval for each ICMP request was kept to 10 ms so as to keep the time taken to Query each of the devices to a minimum. When a manual topology is created and monitoring is started, ICMP requests are sent to all devices presented in CONNECTED_DEVICE list. It is possible that systems like those running windows XP have built in firewall and which can drop/reject ICMP packets. Though if UDP ports of a device are closed *Destination Port unreachable* is received which shows that though remote device is up but its ports are closed. Some of the MIB–II OIDs like (.1.3.1.2.1.1.0, system name), (.1.3.1.2.1.2.0, system object ID) etc are used to query remote device.

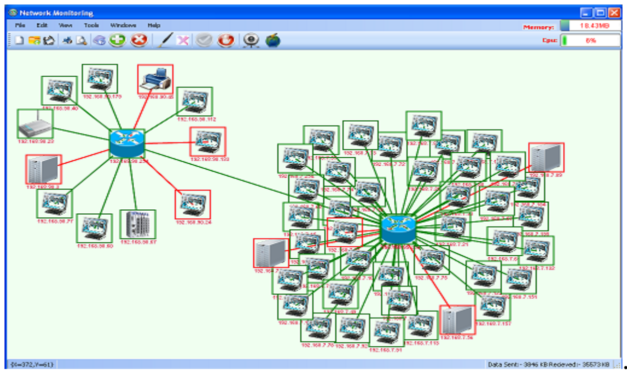

Fig 4: Automatic discovery of network

We scanned the Crompton Greaves Network and it took about 10 seconds to discover 87 devices including Routers. Creating more number of threads lead to degradation of systems performance, so an ideal number of 3 threads were created , one handling ICMP tasks other one dealing with SNMP and the third one Dealing with updating the GUI and placing the devices on the screen providing a clear view of topology to end user.

VI Results

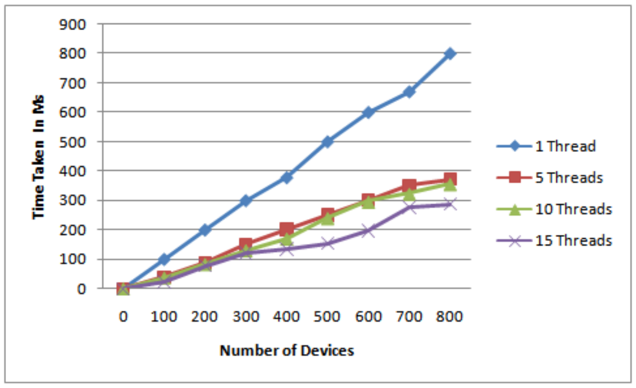

Fig 5: Number of Devices Vs Discovery Time.

Fig 5 shows below shows the variation in time taken to discover number of hosts based on the level of multithreading. As the number of threads are increased the time taken to discover network hosts decreases considerably. However after a certain extent the increase in threads leads to an overhead on system and system performance decreases considerably if agent less network discovery approach is adopted.

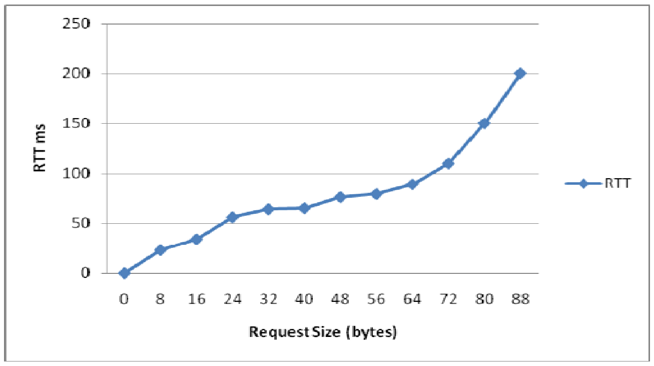

Fig 6: RTT vs Packet Size

Fig 6 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.

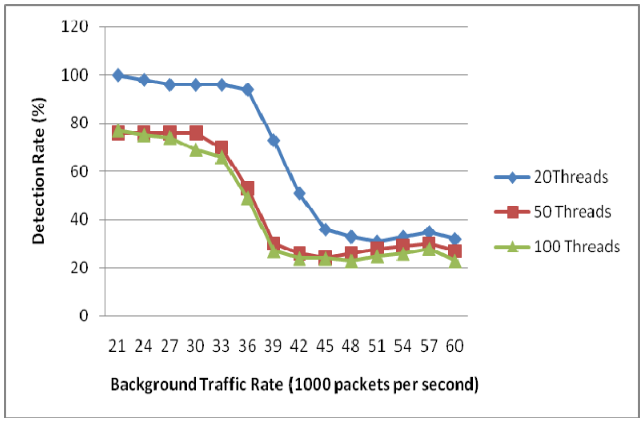

Fig 7: Average Detection rate Vs Background Traffic rate

Fig 7 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.

VII. CONCLUSION

The automatic discovery of the network topology is always an important means for network management. To design and develop an effective and useful tool for the network topology discovery is an important and most difficult part in developing Network Monitoring Solution. It demands deep study and absolute implementation of various network protocols. The discovery mechanism proposed is designed to operate on fast local area networks that won't suffer from the aggressive probing. This paper 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.

REFERENCES

[1] V. Jacobson, "Traceroute Software", Lawrence Berkeley Laboratories, 1989.
[2] J. Postel. "Internet Control Message Protocol", RFC 792, Sep. 1981.
[3] J. Wei-hua, Du jun "The application of ICMP protocol in network scanning", Proceedings of International Conference on PDCAT,pp.904-906 Aug 2003.
[4] J. Schonwalder, H. Langendorfer "How to keep track of your network configuration," Proc. LISA, pp.101–105, Nov. 1993.
[5] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, A. Silberschatz, "Topology Discovery in Heterogeneous IP Networks: The NetInventory System," IEEE/ACM Transactions on Networking, vol. 12, no. 3, June 2004, pp. 401~414.
[6] B. Lowekamp, D. R. O'Hallaron, T. R. Gross, "Topology discovery for large Ethernet networks," ACM SIGCOMM, August 2001, San Diego, CA, USA, pp. 237~248.
[7] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," RFC 1157, IETF, May 1990.
[8] B. Donnet, T. Friedman, "Internet Topology Discovery: A Survey," IEEE Communications Surveys & Tutorials, Vol. 9, No. 4, 4th Quarter 2007, 56~69.
[9] Hwa-Chun Lin, Shou-Chuan Lai, Ping-Wen Chen, "An Algorithm for Automatic Topology Discovery of IP Networks", Proceedings of IEEE, ICC, 1998.
[10] A. Bierman, K. Jones, "Physical Topology MIB", RFC2922, IETF, September 2000.
[11] F. Baker, "IP Forwarding Table MIB", RFC 2096, IETF, January 1997.
[12] Glenn Mansfield, M. Ouchi, K. Jayanthi, Y. Kimura, K. Ohta, Y. Nemoto, "Techniques for automated Network Map Generation using SNMP", infocom, pp.473, Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Vol. 1-3),1996.
[13] Research on the Theory of SNMP and the Technology of SNMP Programming, IEEE 2010, 23-25.
[14] HE Li-juan, WANG Dong-hong, "Topology discovery algorithm of the SNMP-based network--layer protocol", *Journal of Shijiazhuang Vocational Technology Institute*, VOl.21 NO.6, 2009. Li, Yanbing; Ma, Yue; Wang, Wei; Wan, Xiaoqiang, "A link layer topology discovery algorithm based on STP", *Jisuanji Gongcheng/Computer Engineering*, v32, n18, p109-110+113, Sep 20,2006
[15] Yang Qiuxiang; " Algorithm Research of Topology Discovery on SNMP", International Conference on Computer Application and System Modeling (ICCASM 2010),v12, p 496-497.