World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:2, No:6, 2008

Device Discover: A Component for Network Management System using Simple Network Management Protocol

Garima Gupta, Daya Gupta

***Abstract*—**Virtually all existing networked system management tools use a Manager/Agent paradigm. That is, distributed agents are deployed on managed devices to collect local information and report it back to some management unit. Even those that use standard protocols such as SNMP fall into this model. Using standard protocol has the advantage of interoperability among devices from different vendors. However, it may not be able to provide customized information that is of interest to satisfy specific management needs.

In this dissertation work, different approaches are used to collect information regarding the devices attached to a Local Area Network. An SNMP aware application is being developed that will manage the discovery procedure and will be used as data collector.

***Keywords*—**ICMP Scanner, Network Discovery, Network Management, SNMP Scanner.

I. Introduction

NOW a days, Networks and distributed processing systems are of growing importance and, indeed have become critical in the business world. Within a given organization, the trend is toward larger, more complex networks supporting more application and more users. A large network cannot be put together and managed by human effort alone. The complexity of such systems dictates the use of automated network management tools [1].

The Simple Network Management Protocol (SNMP) is one step in the direction of standardization of Network Management. SNMP has been developed to provide a tool for multi-vendor, interoperable network management [2].

A network management system is an integrated collection of tools for network monitoring and control. A network management system consists of incremental hardware and software additions implemented among existing network components. The software used in accomplishing the network management task resides in the host computers and communication processors (for example, front-end processors, terminal cluster controllers, bridges, and routers) [1]. A network management system is designed to view the entire network as a unified architecture, with addresses and labels assigned to each pint and the specific attribute of each element and link known to the system. The active elements of network provide regular feedback of status information to the network control center.

Each network node contains a collection of software devoted to the network management task, called as network management entity (NME). At least one node in the network is designated as the network control host, or manager. In addition to the NME software, the network control host includes a collection of software called the network management application (NMA). The NMA include an operator interface to allow an authorized user to manage the network [1], [3].

A network management platform deployed in the enterprise manages an infrastructure that consists of multi-vendor network elements. The platform receives and processes events from network elements in the network. Events from servers and other critical resources can also be forwarded to a management platform. The following commonly available functions are included in a standard management platform:

- Network discovery.
- Topology mapping of network elements.
- Event handler.
- Performance data collector and grapher.
- Management data browser.

The strategy that has traditionally been favored by both mainframe vendor and information system executives is of a centralized network management strategy, with a single network control center and perhaps a standby center.

A distributed management system replaces the single network control center with interoperable workstations located on LANs distributed throughout the enterprise. This strategy gives departmental-level managers, who must watch over downsized applications for their local end users

This research work has focused towards the first function of the most of the NMS systems i.e. Network Discovery. The discovery function of most network management platforms is intended to provide a dynamic listing of devices found in the network. A discovery engine provides detailed configuration

Garima Gupta is with Maharaja Agrasen Institute of Technology, Delhi as a lecturer. She is also a final year student of M.E., Delhi College of Engineering, India. (Address: C-325, 1st Floor Vivek Vihar, Delhi – 110095, Phone: +919873186367; e-mail: garima.chadha@gmail.com).

Daya Gupta is Assistant Professor with Delhi College of Engineering (Address: 119, Gagan Vihar-Main, Delhi-110051; email:daya_gupta2005@yahoo.co.in).

information of network devices. Knowledge of the underlying network is becoming one of the key requirements of the new generation of Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). Lacking this knowledge can result in numerous ambiguities when interpreting alerts and making decisions on adequate responses. Acquiring this knowledge can be accomplished by a variety of techniques that can be placed in two general categories:

- Active Discovery Techniques.
- Passive Discovery Techniques.

Passive network discovery and monitoring is a technology that processes captured packets from a monitored network in order to gather information about the network, its active elements, and their properties. It is usually installed at a network chokepoint [3], [4].

The underlying methodology behind active profiling techniques involves actively probing a target device, for which there is a wide variety of techniques, and then analyzing the device's response. Through the use of these techniques, various pieces of information about the network and its devices can be discovered, including the network topology, device availability, the protocols in use, and much more.

While the monitoring methods employed in passive techniques create no additional traffic on the network, all active techniques create varying amounts of additional network traffic. As the number of hosts and ports to be scanned increases, the amount of increased traffic can interfere with normal network and host operation.

The general methodologies behind these two categories, as well as the various techniques within each, have their own distinct advantages and disadvantages, which determine their applicability within real world implementations.

II. System Overview

A. ICMP Echo Scan

This technique involves sending an ICMP Echo datagram to the destination host and waiting to see if there is a response. If the target host is active, an ICMP Echo reply is sent back from the target host. If the source host receives no response to its ICMP Echo request, either the target host is inactive or the ICMP protocol is being filtered. Unfortunately, the ability to differentiate between the two causes of a failure by the target to respond with an ICMP Echo reply is beyond the ability of this scanning technique [5].

It is also possible that an ICMP error message will be returned, such as a: ICMP Host Unreachable or ICMP Destination Unreachable port unreachable error. In the case of an ICMP Host Unreachable error, it indicates that the targeted host is either temporarily down or does not exist. For the port unreachable error, it can be determined that the host is alive and reachable, but the port is closed.

B. Discovery Procedures

Discovery process designed for 'Device Discover' is not limited to 'ICMP Echo Scan', but it also does the SNMP Ping on each device discovered during the ICMP Scan. SNMP Ping is used to extract some more information related to device being discovered [2], [4], [5].

Device Discover has a Ping Scanner (for doing ICMP Echo Scan) and a SNMP Scanner (For doing SNMP Ping to the devices discovered by Ping Scanner).

On start of discovery process, a range of IP-Address is given as an input to the Ping Scanner module. This range can be specified either as start or an end ip-address or can be input as single ip-address and ask the Ping Scanner to look for all the address in the corresponding network. Ping Scanner sends an ICMP Echo (ping) request to each and every ip-address within the range. And it sends all the ICMP packets in one go and then wait for ICMP echo replies. It waits for configurable amount of time before moving on to next step. After waiting 'Ping scanner' passes the list of discovered hosts/devices to SNMP scanner to do the further processing.

SNMP Scanner sends SNMP Get request on all the discovered hosts/devices in one go, and then wait for configurable amount of time to receive SNMP response.

Final collected information is stored in internal data-structures and displayed to user through a third party SNMP Browser, using internally developed MIB [2], [6], [7]. At the end of the discovery process, the new data collected is compared with that of the last time. If a new device is discovered i.e. discovered during the current scan but not present in the last scan, a trap 'deviceUp' is raised. Similarly, if a device goes down i.e. not discovered during the current scan but present in the last scan, a trap 'deviceDown' is raised. Overall functionality can be described using Figure 1.

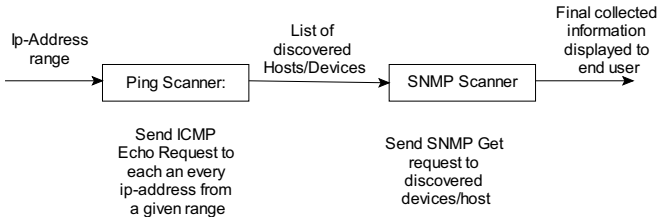

Fig. 1 Discovery process outline

C. Device Discover Architecture

Device Discover provides SNMP Interface for controlling and monitoring device discovery process. To accomplish this, a non-standard MIB is designed and implemented. An SNMP MIB browser (acting as SNMP manager) is used for managing discovery process. MIB is added on to that browser and same MIB is implemented in the CMU-SNMP agent [2], [6], [7].

Some of the configuration for various common parameters of Device Discover is being done through a configuration file. Each line of the configuration file is parsed by a parser. Discovery thread is the core of 'Device Discover' and active discovery method 'ICMP Scan' is extended with 'SNMP Scanner' to accomplish the task of discovery process. Figure 2

explains the architecture of Device Discover. As depicted in the figure, Device Discover has following components:

- SNMP Browser (SNMP Manager): SNMP browser is acting as SNMP manager. It is used to start/stop and manage the discovery process. Device Discover make use of SimpleMibBrowser as a SNMP manager. SimpleMIBBrowser is an easy to use, graphical tool that simplifies the process of interacting with SNMP

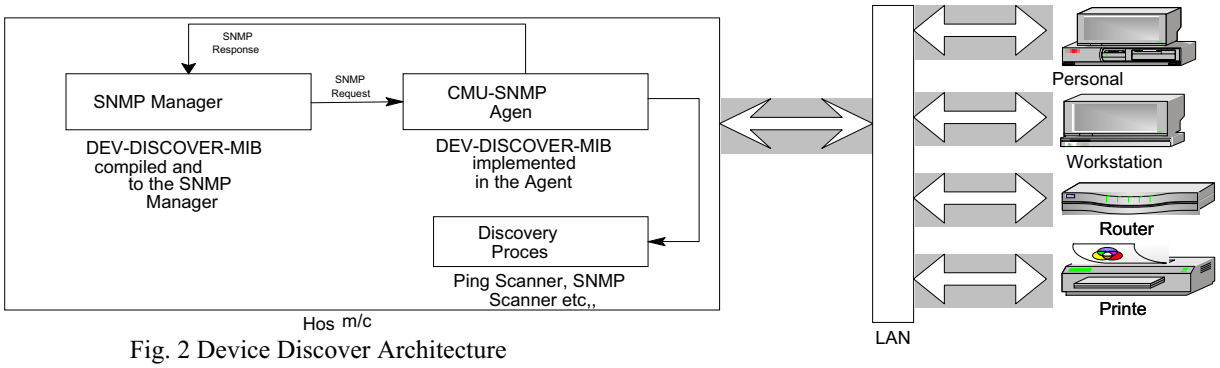

Fig. 2 Device Discover Architecture

enabled devices to retrieve and configure management information. Management Information Base (MIB) ASN.1 documents can be loaded into the SimpleMIBBrowser and their contents displayed in a graphical tree. Queries can then be generated by selecting nodes on the managed object tree and sent to the device/SNMP agent and the responses shown in a tabular fashion.

- SNMP Agent: CMU-SNMP is used as SNMP agent. A non-standard MIB i.e. DEVICE DISCOVER-MIB is added into the CMU-SNMP. This agent is interacting with the manager to control and monitor the discovery process. CMU SNMP agent has supports for various common MIBs like mib-2 (defined in RFC 1213) etc.
- Discovery Process: It is the main component of the Device Discover. This component includes Ping Scanner, SNMP Scanner and all the data-structure used to store the information collected during discovery process. Discovery process runs as a separate thread along with SNMP Manager and SNMP agents. It directly interacts with the agent, and it does all the functional part of Device Discover. On starting the discovery procedure via setting an object of MIB, a SNMP set request is sent to the SNMP agent and then agent triggers the discovery process.

D. Device Discover-MIB

This section describe a non-standard MIB i.e. DEVICE DISCOVER-MIB that is designed and implemented for managing the whole discovery process through SNMP manager. Figure 3 shows the hierarchal structure of this MIB [6], it shows how each and every object of this MIB are linked together in a hierarchal fashion.

This MIB is added under enterprises in mib-2 (defined in RFC 1213). OID of the first object in this MIB is "1.3.6.1.4.1.1024" (or can be represented as iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). devDiscover(1024)). There are three group objects in this MIB:

- devDiscoverCommon(1.3.6.1.4.1.1024.1): This contains many common objects e.g. Ethernet interface to be scanned, discovery state etc…
- devDescoverDeviceTable (1.3.6.1.4.1.1024.2): This object represent a two dimensional object i.e. a table containing information of discovered devices e.g. deviceName, deviceAddress, devIsSnmpEnabled etc.

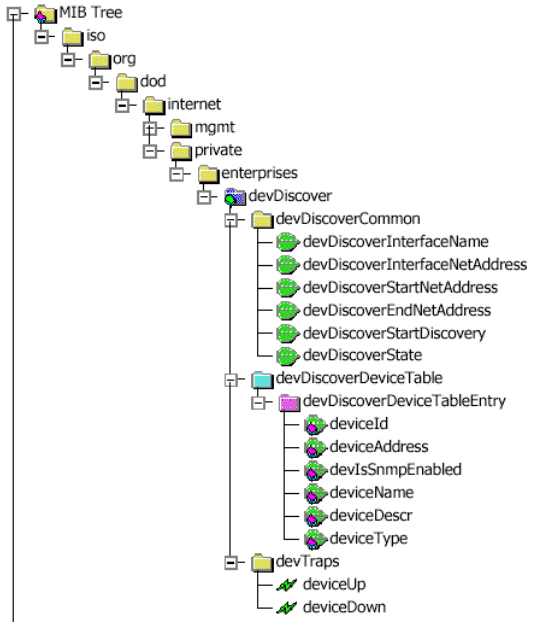

Fig. 3 Device Discover-MIB

- devTraps (1.3.6.1.4.1.1024.3): This object defines two traps that will be sent to SNMP manager when a device comes up or goes down on a network.

E. Threaded Design

As mentioned earlier, discovery process is implemented as

a separate thread running parallel to SNMP agent daemon ('snmpd'). For this a separate thread is created from the 'main' function of SNMP agent.

In SNMP agent after the initializations for 'snmpd' completes, a separate thread for discovery process is created and a communication infrastructure is setup between 'snmpd' and 'discoveryThread'. A FIFO (standard IPC mechanism in POSIX compliant OS) is created for communication between 'snmpd' and 'discoveryThread' and message based communication is provided between these two threads [8]. For example, if 'snmpd' received a request to start/stop the discovery procedure then 'snmpd' will allocate a message of particular type and send it to the 'discoveryThread' over the FIFO created for message exchange.

Since threaded approach has been chosen to implement discovery process, so passing a pointer to message structure is sufficient (as 'discoveryThread' is having the same address space as that of 'snmpd' and so a particular pointer in both the thread will point to same object) and this will save amount of data that has to be transferred and will improve the overall efficiency of the discovery process.

```
Pseudo-code for communication setup:
snmpd's main( )
{
   Start snmp agent as a daemon 'snmpd'.
   Do the initializations.
/*Communication setup*/
   Create a FIFO for communication between 'snmpd' and
'discoveryTherad'.
   Create 'discoveryThread'.
}
```

When 'snmpd' received a set request for an object of DEVICE DISCOVER-MIB, it then creates a message containing that request and passes that message to 'discoveryThread'. 'discoveyThread' continuously waits for messages coming on FIFO and on receiving a message it decide what has to be done. E.g. if a discovery process is running and it receive a message to modify 'devDiscoverStartNetAddress' then that message is simply freed without doing any operation.

Thus in this way the set operations is synchronized with respect to 'discoveryThread'.

```
/*Message structure*/
typedef struct
{
eMsgType   messageType; /*Specify the
object being set*/
union
    {
      BOOL enable;    /*Start/Stop discovery*/
      char ifName[IFNAME_MAX_LAN], /*Interface name
to be scanned*/
      unsigned long startAddr,/*Start address for scanning*/
      unsigned long endAddr, /*End address for scanning*/
      }obj;
}msg;
```

'messageType' field identifies the object for which this message is created. Figure 4 shows the simple scenario of starting the discovery process.

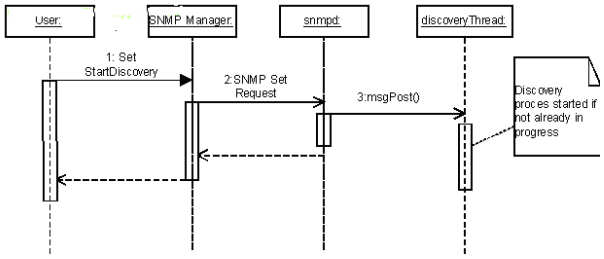

Fig. 4 Sequence diagram for Simple scenario.

F. ICMP Scanner

ICMP Scanner is one of the main components of 'discoveryThread'. It scans the whole network or IP address range to find out the devices/machines that are active in the network. In order to start scanning process, either a network address or start and end IP-address are provided to ICMP scanner [5]. ICMP Scanner then creates ICMP echo message and send it to all the ip-addresses one by one. And if ICMP reply is received then ICMP scanner comes to know that a particular device is active on the network. Thus when this activity completed, ICMP scanner is having the list of devices that are active on the network.

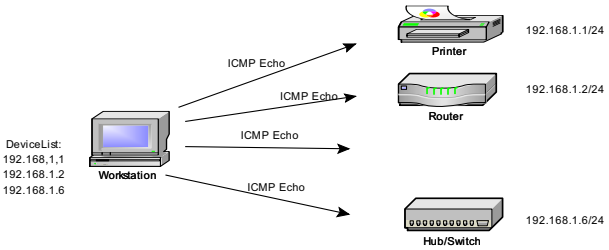

Fig. 5 ICMP Scanning

G. SNMP Scanner

SNMP scanner is another important component of 'discoveryThread'. It does the SNMP scan over the list of devices/machines discovered by the ICMP scanner in the last step.

The SNMP protocol is a stateless, datagram oriented protocol. SNMP scanner sends SNMP requests to multiple IP addresses (discovered during ICMP scan), trying different community strings and waiting for a reply. Unfortunately SNMP agent do not respond to requests with invalid community strings and the underlying UDP protocol does not reliably report closed UDP ports [2]. This means that 'no response' from the probed IP address can mean either of the following:

- machine unreachable

- SNMP agent not running
- invalid community string
- the response datagram has not yet arrived

The approach taken by most SNMP scanners is to send the request, wait for n seconds and assume that the community string is invalid. If only 1 of every hundred scanned IP addresses responds to the SNMP request, the scanner will spend 99*n seconds waiting for replies that will never come. This makes traditional SNMP scanners very inefficient.

But SNMP scanner implemented for 'Device Discover' takes a different approach to SNMP scanning. It takes advantage of the fact that SNMP is a connectionless protocol and sends all SNMP requests as fast as it can. Then the scanner waits for responses to come back and store them in the 'NMSDeviceDataTable'. By default it waits for 10 milliseconds between sending packets, which is adequate for 100Mbs switched networks. The user can adjust this value via the configuration file parameter defining in between packet delay. If set to 0, the scanner will send packets as fast as the kernel would accept them, which may lead to packet drop.

SNMP Scanner sends a request for the following OIDs:

system.sysName.0 (1.3.6.1.2.1.1.5.0)
system.sysDescr.0 (1.3.6.1.2.1.1.1.0)

These OIDs belongs to System group of MIB-II and are present on almost all SNMP enabled devices. Return values gives us more details of the system software running on the device.

III. Conclusion

The work carried out is towards the Network discovery feature of the Network Management Systems. Different network discovery approaches (for example, passive discovery techniques, and active profiling techniques) are studied and analyzed; their advantages and shortcomings are analyzed and studied.

Active discovery techniques, ICMP Echo Scan, under Host link layer Techniques is used for the network discovery in the targeted system. ICMP echo scan (or ping scan) is used to discover the active devices on network. In order to find extra information about the devices discovered ICMP scan is followed by SNMP Scan. SNMP Scan (as currently implemented) is used to obtain 'system.sysName' and 'system.sysDescr' objects of 'system' group. This provides information about the device and system software running on the device.

Whole discovery process is controlled and monitored via SNMP manager. And a non-standard MIB is designed and implemented to allow SNMP entities to control the process. This MIB include objects, and set/get on these objects allows managing the discovery process.

The traditional SNMP scanners are very inefficient as they spent a considerable amount of time waiting for replies. But SNMP scanner implemented for 'Device Discover' takes a different approach to SNMP scanning by taking advantage of the fact that SNMP is a connectionless protocol and sends all SNMP requests as fast as it can.

The technique can be used by Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) in obtaining customized information about the devices on the network and preventing numerous ambiguities when interpreting alerts and making decisions on adequate responses.

References

[1] Marshall T. Rose, The Simple Book: An Introduction to Management of TCP/IP-based internets, Prientice-Hall, Englewood Cliffs, 1991.

[2] Stalling W., SNMP,SNMPv2,SNMPv3 and RMON1 and 2, Pearson-Education,2004.

[3] Annie De Montigny-Leboeuf and Frédéric Massicotte, Passive Network Discovery for Real Time Situation Awareness. Communication Research Centre Canada, Ottawa, Ontario, 2004.

[4] J. Treurniet, An Overview of Passive Information Gathering Techniques for Network Security, TM 2004-073. Defense R&D Canada – Ottawa. Ottawa, Ontario, May 2004.

[5] Ofir Arkin, ICMP Usage in Scanning: The Complete Know-How, Version3.0. Sys-Security Group, June 2001.

[6] K.McCloghrie, M.Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", RFC1213, March 1991

[7] CMU SNMP Library, www.net.cmu.edu/groups/netdev/software.html

[8] C Brain, "Algorithms and Techniques Used for Auto-discovery of Network Topology, Assets and Services", Faculty of Computer Science, University of New Brunswick, Canada, March 16, 2006