IP Network Topology Discovery Using SNMP

Suman Pandey[#1], Mi-Jung Choi[#2], Sung-Joo Lee[#3], James W. Hong[#4]

[#] *Dept. of Computer Science and Engineering, POSTECH, Korea*

[1]suman@postech.ac.kr,[2]mjchoi@postech.ac.kr,[3]forstar@postech.ac.kr,[4]jwkhong@postech.ac.kr

***Abstract*— Network topology information helps one analyze faults in IP networks and their locations. Rectifying such faults is a key role of an enterprise IP network management system. Thus, the automatic discovery of enterprise network topology has been the subject of rigorous study for many years. This paper proposes a Simple Network Management Protocol (SNMP)-based solution that handles various types of network devices, including L2/L3/L4/L7 switches, routers, printers, and hosts; and discovers connectivity among these devices. Key contribution of this paper is a simple algorithm to discover end host connectivity with the switches and routers. We show the network topology in combination of graph and tree layouts.**

I. INTRODUCTION

Network topology is the study of the arrangement of links and nodes in a network and the interconnections among the nodes. We can categorize it as a physical network topology, where peers are connected to ports on devices via a transmission link or a logical network topology, in which a network is divided into logical segments through subnets. Network discovery can also be categorized as Internet or backbone discovery and the local area network (LAN) or an organizational-level network such as autonomous system (AS) discovery. An inexperienced network administrator joining an organization faces many difficulties due to the unavailability of a discovery tool, which otherwise would show the topology classification (based on types of devices and subnets) and layout of the networks. Even for the experienced administrator, keeping track of devices and their connectivity details, without having a proper method of visually presenting them becomes a difficult task. Our work concentrates on organizational-level topological discovery.

Many studies [1–4] conducted in the area of automatic discovery of network topology using ping, traceroute, Simple Network Management Protocol (SNMP) [15], and other methods have shown remarkable results; however, they fail to address the following issues:

Discovering various types of devices, Yuri et al. [2] details some information about discovery of device type, but their algorithm is not sufficient in a modern network; we need a better algorithm to identify the types of devices. Our algorithm can discover all kinds of devices, including layer 2 (L2), layer 3 (L3), layer 4 (L4), and layer 7 (L7) switches, routers, end hosts, and printers.

Network topology visualization, is another restriction inherent in most of the latest tools supporting topology discovery.

Discovering complete topology, although many studies have been conducted on discovering L2 and L3-level topologies separately [1, 3 and 4], there has also been little attention paid to interconnecting L2 and L3 topologies [2]. We propose a solution to discover the complete topology of a network. Our main strength is using a simple algorithm to connect the end host with the network. It is simple because even if the host does not support SNMP we can still find the connectivity. This algorithm is based on heuristics.

Network connectivity discovery is a well studied area, and there are many interesting mechanisms—such as ping, traceroute, DNS, address resolution protocol (ARP), and SNMP—available to discover network elements and the connectivity among them. There have been some efforts made by Cisco to standardize the Physical Network Topology Identification and Discovery (PTOPO-MIB) in RFC 2922 [7], but to the best of our knowledge, Cisco is the only vendor supporting this. There are various commercial tools—such as HP OpenView, IBM's Tivoli, and AdventNet OpManager—but these show only L3-level topology. Apart from the commercial efforts, there has been much effort made by the research community in this direction.

R. Siamwalla et al. [1] did a good survey and proposed mechanisms to discover topology by combining ping, traceroot, SNMP, DNS, and ARP. However, these methods could discover only L3-level topology, and the report did not propose any mechanisms to discover L2- or host-level topology, though they proved that SNMP performs better than all other mechanisms. Yuri et al. [2] proposed a mechanism that is heterogeneous, irrespective of any kind of network, but this mechanism requires ICMP spoofing in order to get complete forwarding table, which is not allowed in most of today's networks. These algorithms are also time-consuming and resource-intensive. Though they did a good job in explaining the connectivity algorithm they failed to provide details on SNMP MIBs required for gathering topology information. Lowekamp et al. [3] proposed a mechanism by which we would not require complete forwarding information of bridges; their approach contradicted of Yuri et al. [2]. We extend the work of Lowekamp et al. [3] and propose a complete topological discovery mechanism to discover L2-L2, L3-L3, L2-L3, and L2 and L3 to end host connectivity.

The organization of this paper is as follows. The topology discovery algorithms are explained in Section 2, and our implementation and experiments are discussed in Section 3. Section 4 concludes our work and discusses future research directions.

II. DISCOVERY ALGORITHM

In this section, we present our approach to discovering network nodes and connectivity among them. Since our approach is mainly based on SNMP, we first analyse the major management information base (MIB) objects required to build our algorithm. We then utilize MIBs to build a discovery algorithm, which is basically divided into three different modules, namely *device discovery*, *device type discovery*, and *connectivity discovery*.

A. MIBs for Discovery

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

TABLE I
MIB INFORMATION FOR TOPOLOGY DISCOVERY

MIB-II RFC 1213 [12]
sysServices, sysDescr, ifIndex, ifDescr, ifPhyaddress, ipForwarding, ipRouteNextHop, ipRouteType, ipAdEntAddr, ipAdEntNetMask, ipNetToMediaNetAddress, ipNetToMediaPhysAddress,
BRIDGE-MIB for connectivity discovery [13]
dot1dBasePort, dot1dBasePortIfIndex, dot1dTpFdbAddress, dot1dTpFdbPort, dot1dTpFdbStatus
BRIDGE-MIB for Spanning Tree discovery [13]
dot1dStpPort, dot1dStpPortState, dot1dStpPortDesignatedRoot, dot1dStpPortDesignatedBridge, dot1dStpPortDesignatedPort

B. Overall Discovery Algorithm

Algorithm 1 shows the overall network connectivity discovery. The basic inputs to our system are IP address of at least one gateway router in the enterprise; boundary information, i.e., one or multiple range of IP address(es); one or multiple community string(s); SNMP port number; and database credentials. The device discovery step uses a routing table, an ARP cache table, and ICMP utilities to discover devices. For each discovered device, it verifies SNMP support and then discovers the device type, such as router, L2/L3/L4/L7 switches, printers, or network terminal nodes.

Depending on the type of device, the relevant MIB information is retrieved from SNMP agents and loaded into the database. This MIB information is used to find the connectivity among the devices. In this way, we can discover connectivity between L2 and L2 devices, L2 and L3 devices, L3 and L3 devices, and L2/L3 and end hosts.

ALGORITHM 1: OVERALL ALGORITHM

1. Take network input
2. Device discovery
   a) Device discovery using recursive next hop mechanism
   b) Device discovery using recursive ARP cache mechanism
3. Device type discovery
4. Device grouping based on IP address
5. Connectivity discovery
   a) L2 to L2 connectivity
   b) L2 to L3 connectivity
   c) L3 to L3 connectivity
   d) L2 and L3 to end host connectivity

C. Recursive Device Discovery Algorithm

RFC 1213 defines a simple, workable architecture of managed objects for managing TCP/IP-based networks [12]. The managed objects mentioned in this RFC are standard and implemented by all vendors; we have utilized their minimum and workable architecture to discover topology, and we have found that this information is sufficient for discovering almost all the devices in the network. A routing table of the device is maintained by the *ipRouteTable* object; the *ipRouteTable* object contains an entry for each route presently known to this entity in *ipRouteEntry*. We utilize only *ipRouteNextHop* and *ipRouteType* entries for these tables. *ipRouteNextHop* is the IP address of the next hop in the route. *ipRouteType* can be one of four types: *direct*, *indirect*, *invalid*, or *other*. The type *direct* refers to the same device, having multiple IP addresses; we thus discard the entries of types *direct*, *invalid* or *other*. We filter the records and take only those entries that are of type *indirect*. This routing table is updated by any routing protocol such as OSPF and IS-IS, and this provides topology of the network around one L3 device.

To discover end hosts and L2 devices, we rely on *ipNetToMdeiaTable*, an IP address translation table. For resolving IP address to MAC address mapping, ARP protocol is used; to make this resolution work faster, the router maintains an ARP cache that contains the MAC to IP mapping of the active devices in the network. As soon as we discover a node, we use all unique *ipNetToMediaNetAddress* entries to discover another set of new nodes. One device can help in discovering more devices, and these algorithms comprise a recursive process.

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. One of the important steps to discovering devices is to take care of the synonyms of a device. A device can have multiple IP addresses, depending on the number of subnetworks it is connected to. *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, a verification is performed by checking these synonyms. Algorithm 2 explains the device discovery mechanism.

ALGORITHM 2: RECURSIVE ALGORITHM FOR DEVICE DISCOVERY

1. Visited device set = Set of routers already visited, initially empty
2. Next hop discovery (Router IP address)
   a) If router is not in visited device set
   b) Get all unique next hops of router through *ipRouteNextHop*, where *ipRouteType* is *indirect*
   c) If there is no *ipRouteNextHop*, then return;
   d) Call next hop discovery (*ipRouterNextHop*) recursively
3. ARP cache discovery (IP address)
   a) If IP address is not in the visited device set
   b) Get all the unique *ipNetToMediaNetAddress*
   c) If there is no *ipNetToMediaNetAddress*, then return;
   d) Call ARP cache discovery (*ipNetToMediaNetAddress*) recursively

D. Device Type Discovery Algorithm

To discover the types of devices, we use the *sysServices* MIB object and convert it into a seven-bit string. Each bit corresponds to the 7 layer of the OSI network model. If a device has *sysServices* 78 (1001110)—its second, third, fourth, and seventh bits are set—then the device is an L7 switch that provides services for all these four layers. It uses Bridge MIB [13] information to check whether the device can support interface-to-interface connectivity at L2. We categorize such a device as an L2 and L3 switch. Though a similar approach is taken by Yuri et al. [2] to identify device type still they have not considered some extreme conditions such as an application switch configured to have similar MAC address for multiple interfaces. The *ifTable* MIB helps us decide whether the L3 devices are configured to have the same MAC address for multiple interfaces; this helps us in filtering the L3 devices for those where we cannot show interface-to-interface connectivity with other devices, since they have the same MAC addresses for multiple ports and it is not possible to distinguish ports to which the other devices are connected. Also we used Printer MIB to check whether the device is a printer or not. Figure 1 explains the algorithm in the form of a flow chart; the different output boxes in the flow chart show the different types of discovered devices.

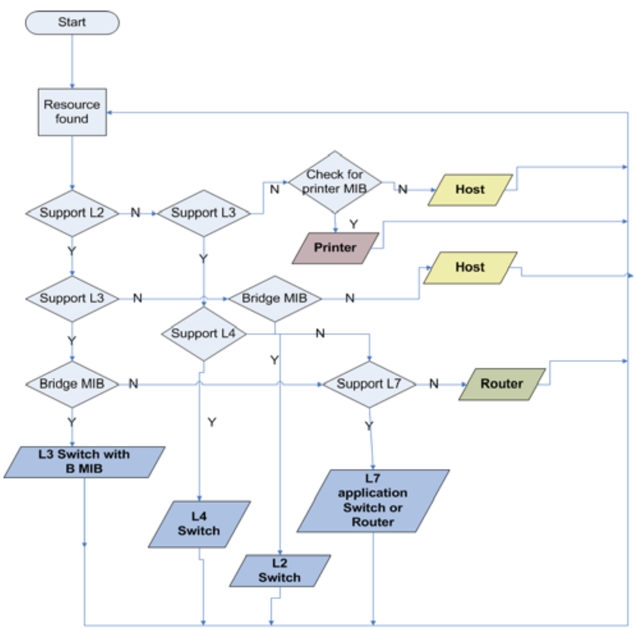

Fig. 1 Device Type Discovery Algorithm

E. Connectivity Discovery Algorithm

An enterprise network is composed of various types of devices. Finding connectivity among the different types of devices is challenging; in this section, we explain connectivity discovery algorithms.

As soon as we discover the device type, we determine what kind of MIB objects need to be stored in the database, this is required because we need not store all the MIBs explained in Table 1 for every type of device. For example, if the device is an L2 or L3, L4 or L7 switch, and if these switches support Bridge MIB, then we retrieve the Bridge MIB; if the device is a Cisco switch, then we apply community string indexing [6] and load the Bridge MIB for each VLAN by appending *@vlanid* on the community string in the SNMP query. If the device is an L3, L4, L7 switch or router, then we load the routing table MIB, and so forth. Furthermore, this information is used to find connectivity among the devices.

The switch (L2)-to-switch (L2) connectivity is discovered using Bridge MIB. Ethernet uses transparent bridging; the presence and operations of transparent bridges are, as the name implies, transparent to the network host. These bridges can learn workstation locations by analysing the source addresses of incoming frames from all network elements.

Using this process, a table is built, which is referred to as the address forwarding table (AFT). When a frame arrives on the bridge interface, the bridge looks up the frame's destination address in the table. If the table contains the association of destination address and the bridge's port, then the frame is forwarded to that port. If there is no association, then the frame is flooded to all the other ports, except the inbound port. To avoid looping, a transparent bridge implements a spanning tree algorithm.

The *dot1dTp* group of Bridge MIB contains the objects that describe the device state with respect to both transparent bridging and Source-route transparent (SRT) bridging. Bridge MIB also has a relationship with the standard MIB-II, so it is assumed that the bridge implementing Bridge MIB also implements at least the system group and the interface group, as defined in MIB-II. The interface group is mandatory, and it contains information about the device's interface, where each interface is attached to a subnetwork, and the port of the bridge that is associated with each interface. Each port is uniquely identified by the port number *dot1dBasePort* in Bridge MIB. *dotdBasePort* is mapped with interface *ifIndex* by *dot1dBasePortEntry*. The Bridge MIB maintains the AFT for each port *dot1dBasePortEntry* and it can be mapped onto *ifIndex* to obtain the actual interface ID. Algorithm 3 describes how to find the connectivity among interfaces of the switches using Bridge MIB.

ALGORITHM 3: SWITCH-TO-SWITCH CONNECTIVITY DISCOVERY

1. Switch set = Filter the L2, L3, L4, and L7 switches that support Bridge MIB
2. Switch pair set = Make pairs of switches; if there are *n* switches there will be n^2 sets
3. For each switch pair set, for ex: { S_i, S_j}
   a) Get the set of MAC for S_i i.e., {$M_{ia}..M_{il}$} from *ifPhyaddress*
   b) Get the set of MAC for S_j i.e., {$M_{ja}..M_{jk}$} from *ifPhyaddress*
   c) If AFT of one of the *dot1dBasePortEntry* (P_i and P_j) of S_i and S_j has at least one MAC address M_{il} and M_{jk} of each other then
      i) Get the mapping of P_i and P_j with the *ifIndex* If_i and If_j
      ii) Set the connectivity of S_i and S_j for interface If_i and If_j
      iii) Store the connectivity information in database.

The switch(L2)-to-router(L3) connectivity is also discovered using Bridge MIB. If a router does not support

Bridge MIB, then we cannot find the interface of the router through which the switch is connected, but we can find the interface of the switch through which the router is connected. Yuri et al[2] explains the details of the switch-to-router connectivity.

The router(L3)-to-router(L3) connectivity is discovered using the routing table. To find the connectivity, we first confirm that connectivity has not already been discovered, using L2-L2 and L2-L3 connectivity. Then we use *ipRouteNextHop* for each pair of routers, to determine whether they are next hops to each other. If the mapping is found, then we establish the connectivity between those L3 devices.

The switches (L2) and routers (L3) to end host connectivity is discovered using the subnet information and spanning tree information of the L2 devices. Each device in an Ethernet belongs to a subnet. We can find the subnet information for each device using the SNMP *ipAdEntNetMask* object; if a device does not support SNMP, the subnet information is obtained through *ICMP address-mask* reply messages. Also various subnet guessing algorithms [1] are utilized to obtain the subnet information. Once we obtain the subnet information about the devices, we group the devices based on the subnet, and then attach the group of the device to the edge node of the already-discovered L2 and L3 network. The edge node can be discovered by utilizing the spanning tree in the bridged network. The *dot1dStp* group contains the objects that denote the bridge's state with respect to the spanning tree algorithm. The spanning tree mechanism works as follows. First, the bridges in the network elect one of their members as a root bridge. Then, each bridge—other than the root bridge—determines its distance to the root bridge and selects one of its ports, called the root port, closest to the root bridge. Then, the bridge elects one port on each subnetwork, called the designated port, which is connected to the designated bridge. The designated bridge will be closest to the LAN subnetwork, and we attach the group of end hosts in the subnetwork to the designated bridge. There is no interface-to-interface connectivity discovered. This method is a heuristic but by verifying manually, we found that this method generates correct results in a POSTECH network. Algorithm 4 describes the mechanism for the host connectivity in the network.

ALGORITHM 4: SWITCH AND ROUTER TO END HOST CONNECTIVITY DISCOVERY

1. Visited device = Set of devices already visited using the switch-to-switch, switch-to-router, and router-to-router connectivity algorithm
2. Not visited device = Set of devices for which connectivity has not already been discovered
3. Retrieve the subnet information of all the devices by applying a bit-wise AND operation to the IP address and the subnet mask retrieved via *ipAdEntNetMask*
4. Group the devices based on subnet
5. For each subnet,
   a) Get the higher-level spanning tree using the *dot1dStp* MIB
   b) Attach all the devices in that subnet to the edge node of the tree
   c) Store the connectivity information in the database

III. IMPLEMENTATION AND EXPERIMENT

We used Java 1.5 and Tomcat 5.5, Oracle 10g, AdventNet SNMP API [11], and JGraphT [8] for graphical representation. We developed and tested our system on Windows XP with a 2.80-GHz, Intel Pentium 4 CPU with 512 MB RAM.

Fig. 2. Graph View of L3 and Tree View of L2 Devices in POSTECH

For better viewing we have shown the L3 level mesh type topology in the graph form, and if we click on one of the L3 device we can get the lower level topology of L2 device and the end hosts in the tree form as shown in fig 2. While showing the tree topology of L2 device, we utilized the spanning tree information of the network, because the spanning tree formed in the network is the active connectivity. This kind of abstraction also helps in viewing the topology and expanding and contracting the nodes to have better high level view.

We applied our discovery system to two different enterprise networks: POSTECH and Korea University. We found in Korea University a total of 2,019 devices, including 29 L2 switches, 14 routers and L3 switches, five L4 switches, and 42 subnets. We discovered in POSTECH a total of 7,495 devices, including 522 L2 switches, eight routers, 76 L3 switches, five L4 switches, and 208 subnets. We noticed that the numbers of L2/L3/L4/L7 switches, routers, subnets discovered in multiple tests were the same, but that the number of discovered end hosts varied, with an error range of 1–2%. The cause of this error is the ARP cache table, we used the ARP cache of all the routers and switches to discover the devices but the entries in the ARP cache are not aged out, so in such situations, we discover some extra devices. We minimized this error by applying an icmp echo request to the devices. In the case of a connectivity discovery among the devices, there are many L2 switches—such as BlackDiamond switches [19]—that do not support SNMP. For such devices, we could not discover the connectivity information accurately and we will do this as a future work.

To improve the efficiency of the discovery method, we implemented a multi-threaded, parallel discovery mechanism.

We compared the time taken for device discovery for various numbers of threads. Figures 5 and 6 show the time taken to discover the number of devices, using different numbers of threads for POSTECH and Korea University, respectively. With an increase in the number of threads, we observed a decrease in processing time; however, we found that there is some packet loss if we increase the number of threads, as it primarily depends on the processing power of the machine, bandwidth, and the underlying SNMP manager implementation. Our system works well with 10 simultaneous threads. Using 10 threads, it took 180 minutes to discover and load the MIB information, 10 minutes to discover connectivity, 34 seconds to discover device types, and 30 seconds to calculate the subnet information of the 7,495 devices in case of POSTECH. The timeout value of SNMP requests in these tests is 5 sec. By reducing the timeout to 3 and 1 sec the time taken was reduced by 34% and 55% respectively.

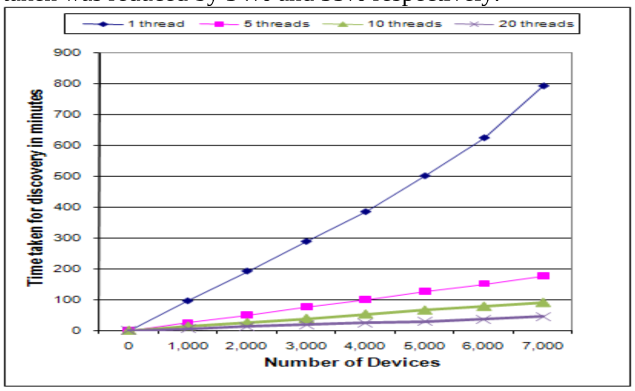

Fig. 5. POSTECH Test Results

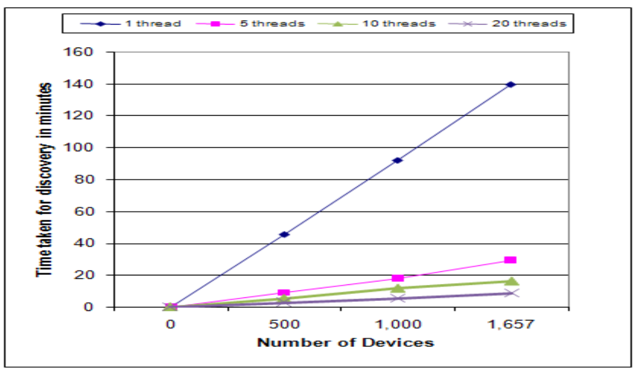

Fig. 6. Korea University Test Results

IV. CONCLUSION AND FUTURE WORK

In this paper, we not only focused on discovery of devices but also on their connectivity. We extended the work of others by introducing an algorithm to connect end host with network. We discovered different types of devices, including routers, L2/L3/L4/L7 switches, printers, and end hosts and enhanced the already existing technique of device type discovery. We utilized the SNMP mechanism, which is the most efficient and generates the least amount of traffic, in comparison to mechanisms in other research [1–4]. Methods based on spoofed ICMP packets and ping cannot be implemented in many of today's networks, because they are not allowed on modern networks for security reasons. Our work can be a guideline in implementing an SNMP-based topology discovery system. Our extensive tests are significant in terms of efficiency and the number of devices discovered: We discovered more than 7,000 devices in a reasonably short time (just over three hours).

This work can be extended by integrating it with weather maps or monitoring tools to provide greater management functionality. Our future goals include integrating more link characteristics—such as link capacity and mean delay—to the links, and discovering connectivity for the L2 switches, such as the BlackDiamond switch, which does not support SNMP. For greater accuracy, our end host connectivity algorithm needs more refinement. We aim to acquire a mechanism to do fast topology update functionality in the future. Various analyses of changes to topology will also be done in future which can help us discover the growth patterns of networks.

REFERENCES

[1] R.Siamwalla, R. Sharma, and S. Keshav, "Discovering internet topology, " Cornell Univ., Ithaca, NY, Techical Report, May 1999.

[2] 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.

[3] 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.

[4] F. Nazir, T.H. Tarar, F. Javed, H. Suguri, H.F. Ahmad, A. Ali, "Constella: A Complete IP Network Topology Discovery Solution," APNOMS 2007, October 2007, Sapporo, Japan, pp. 425~436.

[5] Cisco, How to Get Dynamic CAM Entries (CAM Table) for Catalyst Switches Using SNMP, http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a0080094a9b.shtml.

[6] Cisco, SNMP Community String Indexing, http://www.cisco.com/warp/public/477/SNMP/camsnmp40367.html.

[7] CISCO-VTP-MIB, http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&mibName=CISCO-VTP-MIB.

[8] JGraphT Implementation and Source Code, http://jgrapht.sourceforge.net/.

[9] Bierman, K. Jones, "Physical Topology MIB," RFC 2922, IETF, September 2000.

[10] N. Dufield, F. L. Presti, V. Paxson, D. Towsley, "Network Loss Tomography Using Striped Unicast Probes," IEEE/ACM Transactions on Networking, August 2006, vol. 14, no. 4, 697~710.

[11] AdventNet, AdventNet SNMP API, http://snmp.adventnet.com/.

[12] K. McCloghrie, M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets, MIB-II," RFC 1213, IETF, March 1991.

[13] E. Decker, P. Langille, A. Rijsinghani, K. McCloghrie, "Bridge MIB," RFC 1493, July 1993

[14] E. Bell, A. Smith, P. Langille, A. Rijsinghani,K. McCloghrie, "Q-BRIDGE-MIB," RFC 2674, IETF, August 1999

[15] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," RFC 1157, IETF, May 1990.

[16] D. Passmore, J. Freeman, "The Virtual LAN Technology Report," http://www.3com.com/nsc/200374.html, March 1997.

[17] IEEE: 802.1Q, IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridge Local Area Networks 1998.

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

[19] Extreme Networks: BlackDiamond Core Switching Products, http://www.extremenetworks.com/products/switching-core-family.aspx