Physical Topology Discovery for Metro Ethernet Networks

Myung-Hee Son, Bheom-Soon Joo, Byung-Chul Kim, and Jae-Yong Lee

Automatic discovery of physical topology plays a crucial role in enhancing the manageability of modern metro Ethernet networks. Despite the importance of the problem, earlier research and commercial network management tools have typically concentrated on either discovering logical topology, or proprietary solutions targeting specific product families. Recent works have demonstrated that network topology can be determined using the standard simple network management protocol (SNMP) management information base (MIB), but these algorithms depend on address forwarding table (AFT) entries and can find only spanning tree paths in an Ethernet mesh network. A previous work by Breibart et al. requires that AFT entries be complete; however, that can be a risky assumption in a realistic Ethernet mesh network. In this paper, we have proposed a new physical topology discovery algorithm which works without complete knowledge of AFT entries. Our algorithm can discover a complete physical topology including inactive interfaces eliminated by the spanning tree protocol in metro Ethernet networks. The effectiveness of the algorithm is demonstrated by implementation.

Keywords: Physical topology discovery, metro Ethernet network, SNMP, spanning tree protocol, bridge MIB.

Manuscript received Dec. 09, 2004; revised Mar. 31, 2005.

Myung-Hee Son (phone: +82 42 869 1747, email: mhson@etri.re.kr) and Bheom-Soon Joo (email: bsjoo@etri.re.kr) are with Broadband Convergence Network Research Division, ETRI, Daejeon, Korea.

Byung-Chul Kim (email: byckim@cnu.ac.kr) and Jae-Yong Lee (email: jyl@cnu.ac.kr) are with Information and Communications Engineering Department, Chungnam National University, Daejeon, Korea.

I. Introduction

Metro Ethernet services [1] are now offered by a wide range of service providers. Some providers have extended Ethernet services beyond a metropolitan area network (MAN) and wide area network (WAN). Thousands of subscribers already use Ethernet services and their numbers are growing rapidly. These subscribers have been attracted by the benefits of Ethernet services, including ease of use, cost effectiveness, and flexibility. Now the usage of Ethernets has widened from the LAN environment to MAN and WAN environments.

For a metro Ethernet network, existing technologies for data transport have scalability limitations relating to bandwidth and network management. Actually, most system operators do not manage Ethernet devices because it is not critical in the LAN environment. However, in the metro Ethernet network environment, that approach will not work and a new management capability will be required. In spite of the importance of the problem, earlier research and commercial network management tools have typically concentrated on discovering a logical (that is, Layer-3) topology, which implies that the connectivity of all Layer-2 devices is ignored, or have provided limited proprietary solutions targeting specific product families. The Internet Engineering Task Force (IETF) has acknowledged the importance of this issue by designating a "physical topology" simple network management protocol (SNMP) management information base (MIB) [2], but the proposal merely reserves a portion of the MIB space without defining any protocol or algorithm for obtaining the topology information. Moreover, recent works relating to physical topology discovery in an Ethernet network [3]-[5] have a serious common problem which prevents discovery of the complete physical topology. Their physical topology discovery

was limited to spanning tree paths, which is described in section II.

We explain where a physical topology discovery can be used, for example in capacity provisioning, planning, and management of the growth of a network; in support of configuration management for service provisioning; and finally in support of assurance, for example fault localization and performance management.

For discovering the physical topology including multiple redundant paths in a metro Ethernet network, we suggested a novel and practical algorithm which can perform automatic discovery while minimizing the overhead of processing an enormous number of Layer-2 address forwarding table (AFT) entries in core bridges. Our algorithm uses the standard interface MIB [6] and bridge MIB [7], and we can discover the physical topology whenever it is needed.

We begin by describing related work and our contribution. Section III reviews necessary background information. Sections IV and V describe our physical topology discovery algorithm, which derives locating edge bridges,[1] neighbor bridges, and hosts using the spanning tree protocol (STP) information and the Layer-2 AFT entries. In section VI, we present our experiments including implementation and test results and discuss how our solution can be extended to deal with rapid STP (RSTP) [8] and multiple STP (MSTP) [9]. Finally, section VII concludes the paper.

II. Related Works and Our Contributions

As the trend of moving from Layer-3 communications to Layer-2 communications has developed, many vendors have developed proprietary tools and protocols for discovering physical network connectivity in an Ethernet network. Hitherto, SNMP-based algorithms for automatically discovering Layer-3 topology are featured in many common network management tools, such as HP's OpenView (www.openview.hp.com) and IBM's Tivoli (www.tivoli.com). XML-based algorithms for IP networks [10] are also available; however, these cannot discover Layer-2 topology. In a recent work related to topology discovery for metro Ethernet networks, Breibart and others [3] proposed an algorithm that relies solely on standard AFT information collected in SNMP MIBs to discover the physical topology of heterogeneous networks comprising bridges organized in multiple subnets. Unfortunately, the algorithm assumes that complete AFT information is available from all nodes in the underlying network and thus, cannot cope with hubs or uncooperative switches.[2] In a follow-up paper, Lowekamp and others [4] suggested techniques for inferring network-element connectivity using incomplete AFT information and also discussed how to handle dumb, uncooperative elements. Bejerano and others [5] proposed a more advanced algorithm using a skeleton path for discovering physical topology in large multi-subnet networks. A common feature of those previous works [3]-[5] is that they only use AFT entries. So, they can only find spanning tree paths and exclude multiple redundant paths. Even though their techniques are said to discover the physical topology of Ethernet networks, they have the limitations of discovering only Layer-2 spanning tree paths.

The practicality of our algorithm stems from the fact that it depends solely on standard information routinely collected in the SNMP MIB [11] of bridges, and it requires no modifications to the operating system software running on bridges or hosts. The three previous works [3]-[5] also have the same advantage as above. However these previous works have lots of problems: First, AFT entries typically employ an aging mechanism to evict infrequent source media access control (MAC) addresses from the AFT. In the case of aging-out, topology discovery is not feasible. Second, in the metro Ethernet networks, the size of the AFT ranges from tens of Kbytes to scores of Mbytes. Moreover, most core bridges might have hundreds of Mbytes. It will take up to a very long time to obtain all AFT entries through SNMP and will be difficult to obtain complete information. Third, all data traffic must be bi-directional, but real data traffic doesn't meet that requirement. However, Lemma III.1 presented by Breitbart and others [3] always needs bi-directional complete AFT entries. In their implementation, they addressed this issue by modifying a standard *ping* program in a way that allows their tool to submit Internet control message protocol messages from a given source address to a given destination address. This is overhead, causing unnecessary heavy traffic in the network. Fourth, after a topology change happens, the bridge must flush all AFT entries. In this case, it is impossible to discover the topology in Ethernet networks. Fifth, they cannot find the edges between interfaces that are not active (that is, those eliminated by the STP). Therefore, the topology obtained by the previous method is not a complete physical graph but Layer-2 spanning tree paths excluding multiple redundant paths.

To cope with these problems, in this paper we propose a novel and practical algorithmic solution that can discover an accurate physical topology in a metro Ethernet network. The key idea of our algorithm is as follows: First, to divide metro Ethernet networks into bridged networks and host networks to eliminate the aging-out influence and to avoid the enormous size of AFT entries in core bridges. The second idea is to use STP bridge protocol data units (BPDUs) that are received on

1) To be connected to some hosts or routers.

2) The terms "switch" and "bridge" can be used interchangeably; we will primarily use "bridge" in the remainder of this paper.

inactive port states, such as for blocking and listening to discover multiple redundant paths.

The major contribution of this paper is that we find a true physical topology, not a tree, of metro Ethernet networks, where loop structures are present.

III. Background

This section describes an algorithm that creates AFT entries in Ethernet networks to aid in the understanding of our algorithm. We also describe, at a high level, the operation of STP to motivate our algorithm for discovering neighbor bridges.

1. Creating AFT Entries

In Fig. 1, we show that an AFT entry is composed of a MAC address, a receiving port number, and the aging time indication with a default value of 300 seconds [12], [13]. If the aging time is expired, the entry in the AFT is removed.

All data frames are forwarded only along active connections. This is enforced by dropping those frames received via a blocking or listening port state. If a host moves to another place, then a bridge knows this fact through address learning.

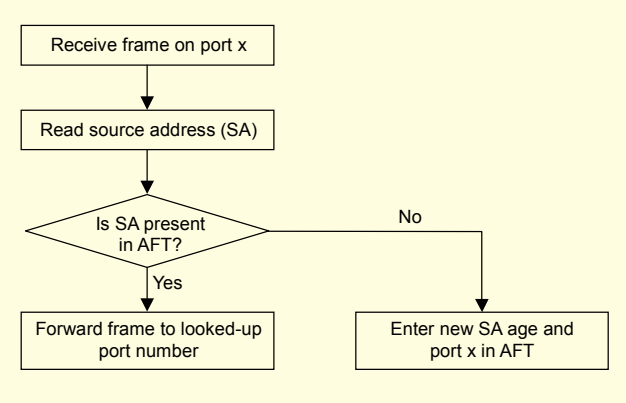


Fig. 1. Flowchart for creating an AFT in a bridge.

2. Spanning Tree Protocol

The STP [13] is a Layer-2 link management protocol that provides path redundancy while preventing undesirable loops in a network. A spanning tree builds an ordered network from a root bridge to designated bridges. The spanning tree algorithm (STA) [13] can be viewed as constructing a spanning tree over a graph of the network topology. Basically, the STA 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 (minimum path length in the graph) to the root bridge and selects one of its ports, called the root port, which is closest to the root bridge. Then, the bridges elect one port on each segment, called the designated port, which is closest to the LAN segment. Each Layer-2 interface on a bridge using STP exists in one of these states: blocking, listening, learning, forwarding, or disabled. Here, a disabled port state means that the physical link connection is not present. Any bridge can receive BPDUs in any port state excluding a disabled state. Also, all port states can send BPDUs, excluding the blocking and disabled state.

The STA is needed to place some of the physical links into a blocked state until such time as a link fails and the backup blocked link is placed in service (that is, frames can be forwarded over it). The two key functions of the STA are 1) to find an active topology without any loops, that is, a spanning tree, and 2) to have a mechanism to discover failures around the network.

IV. Overview of Physical Topology Discovery Algorithm

Most Ethernet Networks have a mesh-type topology to offer redundant networks because of cost effectiveness. For clarity, we define the logical topology and physical topology concepts in metro Ethernet networks. Physical network topology refers to the characterization of the physical connectivity relationships that exist among entities in a communication network. Logical network topology represents the unique forwarding paths to be decided by the STP. From the port state of view for each bridge port in the network, the port state by the former means an electrical active or inactive state, but the port state by the latter has a metaphysical blocking, listening, learning or forwarding state. The goal of our proposed algorithm was to find the redundant physical paths in a metro Ethernet network. To accomplish this goal, we chose to derive the physical network topology using interface MIB [6] and bridge MIB [7] information obtained from all bridges in the metro Ethernet network by the SNMP manager. In this section, we describe our physical topology discovery algorithm, which derives locating edge bridges, designated bridges,[3)] and hosts. A key concept in our topology discovery algorithm is to select the edge bridges defined formally below.

Definition 1. Edge Bridge

An edge bridge is a bridge that contains at least an interface connected to any hosts or any routers. In other words, it is an ingress or egress bridge to or from bridged networks.

A metro Ethernet network containing the nodes N can be divided into the set of all bridges B, the set of all edge bridges

3) These are neighbor bridges connected to a bridge.

Q, and the set of all hosts H. Intuitively, the set of all edge bridges form the boundary between the bridged network and the host network. With the metro Ethernet network split into two sets, it is easier to find the physical topology as compared with the previous approaches [3]-[5].

On the basis of definition 1, we can induce the following property.

Property 1. Edge Bridge

It is impossible for the number of active ports on a bridge to be less than the number of STP enabled ports.

Remark: If we subtract the number of STP enabled ports from the number of active ports, the number of interfaces connected to the hosts remains. We don't need to run STP on those interfaces because the interfaces are not connected to bridged networks forming loops.

The first step in discovering a physical network's topology is to collect some SNMP MIB information, and the second step is to apply our algorithm to them.

Figure 2 illustrates on example of a metro Ethernet network and its active network composed of bridges B1, B2, and B3 and hosts X, Y, and Z. In the metro Ethernet network, as shown in Fig. 2(a), three bridges create a loop. Three hosts transmit a data frame, and then the AFT entries are created as in column 5 or column 7 in Table 1. There are multiple paths with source Y and destination X in accordance with AFT entries in bridge B3. Specifically, there are two paths from Y to X, Y-B3-B1-B2-X and Y-B3-B2-X. Therefore the STP is needed in the metro Ethernet network to decide on a single forwarding path. Figure 2(b) shows an Ethernet active network without any loops, that is, a spanning tree (from the mathematical discipline of graph theory), and the dotted line indicates a link disabled by the STA. Thus, Y-B3-B1-B2-X is the only forwarding path. The reason why this forwarding path is chosen is for STA. The bridge number shown in Fig. 2 represents the bridge priority, and the small number has high priority. That is, bridge B1 has higher priority than bridge B2. Port 2 on bridge B3 has a blocking port state due to a lower priority, so data traffic isn't sent or received in this port state but the STP BPDU can be received by it. The previous authors [3]-[5] can find the spanning tree shown in Fig. 2(b) because they depend only on AFT entries. However, with the algorithm described here we can discover both mesh networks, as shown in Fig. 2(a), and active networks, as shown in Fig. 2(b), as we use information obtained from both the STP and the AFT.

Table 1 summarizes SNMP MIB information to apply our physical topology discovery algorithm to Fig. 2 as an example. The first column lists the devices equipped with an SNMP agent and is mapped to dot1dBaseBridgeAddress in the bridge

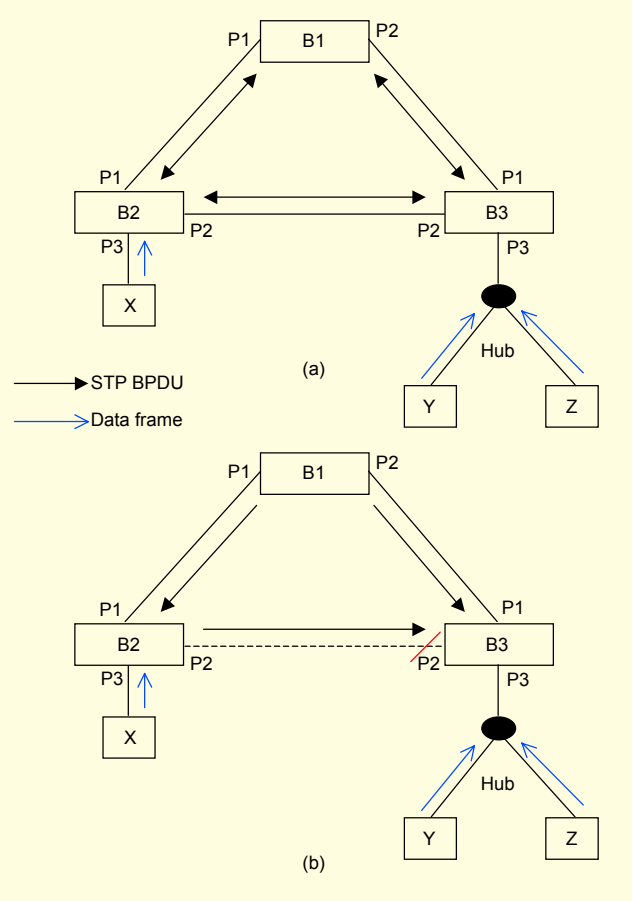


Fig. 2. Example of an Ethernet mesh network vs. Ethernet active network.

MIB [7]. The second column lists active ports in each bridge and can be obtained from ifOperStatus with value 1 in the interface MIB [6]. The third column lists STP enabled ports and can be obtained from dot1dStpPortEnable in the bridge MIB [7]. The fourth column lists a peer bridge and can be obtained from dot1dStpPortDesignatedBridge in the bridge MIB [7]. And finally, the fifth column lists AFT entries and can be obtained from dot1dTpFdbAddress or dot1dStaticAddress in the bridge MIB [7]. The AFT information is only used to find host networks differently from previous works. The value of the designated bridge between mesh networks and active networks is different because of the STP mechanism addressed in section III.2 [13]. However, a metro Ethernet network has an undirected graph so the results of the two types of networks are identical. Thus, we can discover the physical topology regardless of the STP calculation. The SNMP MIB information collected for physical topology discovery shown in Table 1 is inserted into the database (DB) in the SNMP manager. If some topology changes occur, the trap event is noted to the SNMP manager. Therefore, the SNMP manager is able to maintain the latest topology data in the DB.

Table 1. SNMP MIB for physical topology discovery.

Bridge (agent)	Port		Mesh network		Active network	
	OperUP	EnableSTP	DesignatedBridge	AFT	DesignatedBridge	AFT
B1	P1	P1	B2	X	B1	X
	P2	P2	B3	YZ	B1	YZ
B2	P1	P1	B1	YZ	B1	YZ
	P2	P2	B3	YZ	B2	null
	P3	null	null	X	null	X
B3	P1	P1	B1	X	B1	X
	P2	P2	B2	X	B2	null
	P3	null	null	YZ	null	YZ

In the next step we will explain our algorithm with Fig. 2 and Table 1 to gain a better understanding of it. We can model the metro Ethernet network as an undirected graph G=(V, E), where each node in V represents a network element and each edge in E represents a physical connection between two bridge interfaces. Table 2 summarizes the key notations used throughout the paper with a brief description of their semantics. Additional notation will be introduced when necessary.

The first task faced by our algorithm is to find edge bridges in metro Ethernet networks. It is easy to find edge bridges because they have interfaces that are not running STP. Therefore, if a bridge complies with lemma 1, it becomes an edge bridge. The main reason for this procedure is to eliminate the aging-out influence and avoid an enormous number of AFT entries in the core bridges, such as in property 2, because only edge bridges can request AFT entries.

Table 2. Notation.

Symbol	Semantics
$G = (V, E)$	The metro Ethernet network graph
b_i	Bridge $b_i \in B=\{b_1,b_2,\ldots,b_N\}$
D_i	Set of designated bridges connected to b_i. $D_i \subset D$
P_i	Set of active ports on bridge b_i. $P_i \subset P$
T_i	Set of STP enabled ports of bridge b_i. $T_i \subset T$
H_i	Set of AFT entries in edge bridge b_i. $H_i \subset H$
Q	Set of edge bridges
$H_{i,j}$	Set of hosts connected to j-th interface of edge bridge b_i
H_i^j	j-th element of H_i
D_i^j	j-th element of D_i
$e=(v_i,v_j)$	An edge is an element in the set of E
$n(S)$	The number of elements in a set of S

Lemma 1. Edge Bridge Selection

Let b_i be a bridge in metro Ethernet networks and Q be a set of edge bridges. If $n(P_i)>n(T_i)$, then $b_i \in Q$.

Proof: Suppose, to the contrary, that b_i is not an element of Q. Then, the number of active ports on bridge b_i should be equal to the number of STP enabled ports on bridge b_i. The case where the number of active ports on a bridge is less than the number of STP enabled ports violates property 1. □

Property 2. Calculation of Core Bridge's AFT Size

The size of an AFT in a core bridge increases in proportion to the number of active interfaces of the core bridge, and the result is calculated from the following equation:

$$n(AFT_{CoreBridge}) = \sum_{active\ port} n(AFT_{active\ port's\ DesignatedBridge}).$$

Remark: The number of active ports is equal to the number of total active interfaces in a core bridge.

We propose a physical topology discovery algorithm which consists of four major stages. In stage I, we initialize four sets: a set of edge bridges Q, a set of designated bridges $D_{b\in B}$, a set of operational interfaces $P_{b\in B}$, and a set of STP enabled interfaces $T_{b\in B}$. In stage II, we collect physical topology data such as P_i, T_i, and D_i from all agents. The main goal here is to minimize the quantity of data collection and unnecessary overhead instead of collecting all entities of interface MIB [6] and bridge MIB [7]. Here, both *getIFMIB* and *getSTPMIB* are metaphysical terminologies, and the return value of them is

part of interface MIB [6] and bridge MIB [7]. In stage III, we select edge bridges based on definition 1, property 1, and lemma 1. In stage IV, two sub-functions of *PhysicalTopologyDiscovery*, *FindBridgeGraph* for the bridged network and *FindHostGraph* for the host network, are called up. The detailed steps of the physical topology discovery algorithms are described as follows.

```
procedure PhysicalTopologyDiscovery(bridgeSet)
/* bridgeSet B={b1,b2,...,bN} */
  begin
  /* Stage I. Data initialization */
  edgebridgeSet Q←φ
  designatedbridgeSet Di ←φ
  enableportSet Pi←φ
  stpportSet Ti←φ
  /* Stage II. Data collection */
   for each bridge bi∈B   do {
       Pi←getIFMIB(bi)
       Ti←getSTPMIB(bi)
       Di←getSTPMIB(bi)
  }
  /* Stage III. Edge bridge selection */
  for each bridge bi∈B do {
       if(n(Pi) > n(Ti))
          Q←Q∪ {bi}
  }
  /* Stage IV. Sub-procedures call */
  FindBridgeGraph(B, D)
  FindHostGraph(Q)
  end
```

Fig. 3. Our physical topology discovery algorithm.

To help understand our algorithm, we explain it with Fig. 2 as an example. The initial input data for our algorithm is the collection of bridges B in the metro Ethernet network. We can obtain the set of bridges B such as $B=\{b_1, b_2, b_3\}$. As a result of stage II, we can obtain P_i and T_i as follows:

$$P_1=\{p_1, p_2\},\ P_2=\{p_1, p_2, p_3\},\ \text{and}\ P_3=\{p_1, p_2, p_3\}.$$

$$T_1=\{p_1, p_2\},\ T_2=\{p_1, p_2\},\ \text{and}\ T_3=\{p_1, p_2\}.$$

We can also obtain a set of designated bridges, D_i, from stage II with the following two conditions:

First, before spanning tree calculation (Mesh Physical Network),

$$D_1=\{(b_1, b_2), (b_1, b_3)\},\ D_2=\{(b_2, b_1), (b_2, b_3)\},\ \text{and}$$

$$D_3=\{(b_3, b_1), (b_3, b_2)\}.$$

Second, after spanning tree calculation (Tree Active Network),

$$D_1=\{(b_1, b_1), (b_1, b_1)\},\ D_2=\{(b_2, b_1), (b_2, b_2)\},\ \text{and}$$

$$D_3=\{(b_3, b_1), (b_3, b_2)\}.$$

Stage III selects the set of edge bridges $Q=\{b_2, b_3\}$. Therefore, the results of Fig. 3 are the set of agents B, the set of designated bridges D_i, and the set of edge bridges Q, and they are used as input parameters for the *FindBridgeGraph* and *FindHostGraph* procedures.

More specifically, these procedures are performed for discovering some network graphs in the bridged and host networks, respectively. We describe the network graph discovery procedures in the following section.

V. Sub-procedures of Our Physical Topology Discovery Algorithm

We present the *FindBridgeGraph* procedure to find a physical topology in the bridged network and the *FindHostGraph* procedure to discover a physical topology in the host network.

1. The *FindBridgeGraph* Procedure

The main goal of this procedure is to discover a graph in the bridged network. We had sufficient information to perform this procedure through the *PhysicalTopologyDiscovery* procedure shown in Fig. 3.

We first briefly describe how to find the set of bridges in the metro Ethernet network. And we discover edges between bridges with designated bridge information. Figure 4 gives the pseudo-code based on property 3 to discover undirected graph $G=(V, E)$ in the bridged network regardless of calculating a spanning tree. A bridged network is an undirected graph for offering multiple redundant paths. We only need unidirectional information by property 3. The number of sending neighbors has been changed according to whether or not a spanning tree was calculated. In the case where a spanning tree has been calculated, a root bridge can start sending a BPDU; some designated bridges which receive it can also send the BPDU to other designated bridges. However, if before calculating a spanning tree, for example, a topology change happens in the network, all bridges have to start sending a BPDU. If a bridge has a lot of interfaces for running the STP, it has the same number of designated bridges as the number of STP-enabled interfaces.

The main goal of this algorithm is to find the set of edges E and the set of vertices V in bridged networks. This procedure can find the set of edges by means of filtering duplication and dummy information.

Property 3. Connection between Bridges

Let b_i and b_j be bridges in *metro Ethernet networks*, b_j^k be

the k-th interface of a bridge b_i and G=(V, E) be an undirected graph. If b_i^k has a designated bridge b_j then b_i *is* connected to b_j. If the designated bridge of b_i^k is b_i itself, then b_i is not connected to one of any other bridges including b_j.

Remark: In an undirected graph, (b_i, b_j) is the same as (b_j, b_i). And the k-th interface of bridge b_i is an STP-enabled port.

It is simple to find a physical topology in bridged networks because the designated bridge information is offered by a standard SNMP MIB entity. Moreover, we can resolve the problem with property 2 by using AFT entries.

In stage I, we initialize a vertex set V and an edge set E. In stage II, we insert all elements in the bridge set B into the vertex set V. In stage III, we remove the designated bridge, which is a bridge from the designated bridge set D_i, because of applying the graph theory with the following condition: $(b_i, b_i) \notin E$. In stage IV, we can obtain edge set E excluding the following undirected graph condition: $(b_i, b_j) \equiv (b_j, b_i)$. The detailed steps of our algorithm are described as follows.

```
procedure FindBridgeGraph(B, D)
/* G=(V, E) */
    begin
    /* Stage I. Data initialization */
    vertexSet V←Q
    edgeSet E←Q
    /* Stage II. Vertex creation */
    V←V∪B
    /* Stage III. Apply graph theory */
    for each designatedbridgeSet Di⊂D do {
        for each designatedbridge Di^j∈Di do {
            if (bi= Di^j)
                Di←Di−{Di^j}
        }
    }
    /* Stage IV. Edge creation */
    for each designatedbridgeSet Di⊂D do {
        if(Di=φ)
            continue
        for each designatedbridge Di^j ∈Di  do {
            /* (bi, Di^j)≡(Di^j, bi)  ∵ undirected graph */
            E←E∪{(bi, Di^j)}
        }
    }
    end
```

Fig. 4. The FindBridgeGraph procedure.

We explain our algorithm with Fig. 2 as an example. Input arguments of the *FindBridgeGraph* procedure are the set of bridges B and the set of sets of designated bridges D. We can obtain the input arguments as follows:

$$B=\{b_1, b_2, b_3\} \text{ and } D=\{D_1, D_2, D_3\}.$$

The vertex set V can be obtained from stage II as follows: $V=\{b_1, b_2, b_3\}$. We can find some sets of designated bridges with the following two conditions:

First, before spanning tree calculation,

$D_1=\{(b_1, b_2), (b_1, b_3)\}$, $D_2=\{(b_2, b_1), (b_2, b_3)\}$, and $D_3=\{(b_3, b_1), (b_3, b_2)\}$.

Second, after spanning tree calculation,

$D_1=\{\}$, $D_2=\{(b_2, b_1)\}$, and $D_3=\{(b_3, b_1), (b_3, b_2)\}$.

In stage IV, the set of edges E is as follows:

$E=\{e_1, e_2, e_3\}=\{(b_1, b_2), (b_3, b_1), (b_2, b_3)\}$.

2. The *FindHostGraph* Procedure

The last step in physical topology discovery is to find hosts connected to edge bridges. We use AFT entries in the edge bridges for discovering some connectivity with hosts.

We put the following property 4 to decide on the link connection type in the host networks. And we denote the j-th interface of an edge bridge b_i by b_i^j and the MAC address of host B's interface by b, which is expressed in lowercase letters.

Property 4. Connection between an Edge Bridge and Hosts

Let A, B, and C be hosts and b_i be an edge bridge. If b_i has A, B, and C for the same interface j as the AFT entries, then b_i^j is connected to interfaces of A, B, and C through a hub. Thus, we can find the following properties:

First, *if* $(n(H_{i,j})>1)$, *then* the edge bridge b_i is connected to hosts with the number of $n(H_{i,j})$ in the shared segments.

Second, *if* $(n(H_{i,j})=1)$, *then* the edge bridge b_i is connected to only one host. That is, the connection between the edge bridge and the host is a point-to-point link.

Third, *if* $(n(H_{i,j})=0)$, *then* the edge bridge b_i is not connected to any hosts

Remark: Here, $H_{i,j}=\{a, b, c\}$ and b_i^j is an active interface not running STP.

The *FindHostGraph* procedure, as shown in Fig. 5, has the set of edge bridges as an input parameter. The *getAFTMIB* is a metaphysical terminology and carries AFT entries from all edge bridges. To minimize the influence of the size of AFT as property 2, an SNMP manager gets AFT entries from only edge bridges and the information is available for finding host connectivity. Hosts are connected to edge bridges through a hub or direct link as in property 4. We can obtain edge set E after running the following procedure.

In stage I, we initialize vertex set V, edge set E, and host set H_i^j. In stage II, to obtain vertex set V, we need to know AFT entries from BridgeMIB [7] and we can create the vertex set V as $V=Q\cup H_{i,j}$. In stage III, we can decide whether an edge bridge is connected to a host through a point to point link

or whether it is connected to more than two hosts through a hub. That is, edge set E becomes the union of the shared edge set E_s and the point to point edge set E_P. The detailed steps of our algorithm are described as follows.

```
procedure FindHostGraph (Q)
/* G=(V, E) */
begin
    /* Stage I. Data initialization */
    vertexSet V←φ
    edgeSet E←φ
    hostSet H_i^j←φ
    /* Stage II. Vertex creation */
    V←V∪Q
    for each edgebridge b_i∈Q do {
        getAFTMIB(b_i)
        H_ij←findAFTMIB(b_i^j)
        V←V∪H_ij
    }
    /* Stage III. Edge creation */
    for each edgebridge b_i∈Q do {
        if (n(H_ij)≥2) {
            for each host H_i^j∈H_ij do {
                E_s←E_s∪{(b_i, H_i^j)}
            }
        }
        else{
            for each host H_i^j∈H_ij do {
                E_P←E_P∪{(b_i, H_i^j)}
            }
        }
        E=E_s∪E_P
    }
end
```

Fig. 5. The FindHostGraph procedure.

We explain our algorithm with Fig. 2 as an example. An input argument of the *FindHostGraph* procedure is the set of edge bridges Q. We can obtain the input argument as $Q=\{b_2, b_3\}$. We can obtain the vertex set V from stage II as $V=Q\cup H_{2,3}\cup H_{3,3}=\{b_2, b_3, X, Y, Z\}$. Here, $H_{2,3}=\{X\}$ and $H_{3,3}=\{Y, Z\}$. Stage III finds the subset of edge set E by applying property 4 as $E_S=\{(b_3, Y), (b_3, Z)\}$ and $E_P=\{(b_2, X)\}$. Therefore, we can obtain the set of edges as $E=\{e_1, e_2, e_3\}=\{(b_3, Y), (b_3, Z), (b_2, X)\}$.

VI. Experiments

1. Implementation

The basic component processing management functionalities [14] are management application, remote method invocation (RMI) server, and SNMP manager. The algorithm has been implemented in the management application, that is, a graphical user interface (GUI), and was tested on a variety of networks. Figure 6 depicts a high-level view of our implementation architecture.

The management application requests a part of the interface MIB [6] and the bridge MIB [7] to the SNMP manager through the RMI Server. The SNMP manager can submit a query message to all SNMP agents, that is, all bridges. After receiving the interface MIB [6] and the bridge MIB [7] from all SNMP agents, the SNMP manager inserts them into the DB with the following schema: dot1dPTDIfBasicEntry, dot1dPTDBaseInfo, dot1dPTDStp, dot1dPTDStpPortEntry, dot1dPTDTpFdbEntry, and dot1dPTDStaticEntry expressed in Tables 3 through 8. Then, the SNMP manager notifies the fact to the management application. The management application can fetch the physical topology information from the DB and carry out our topology discovery algorithm; it then shows the results via a GUI.

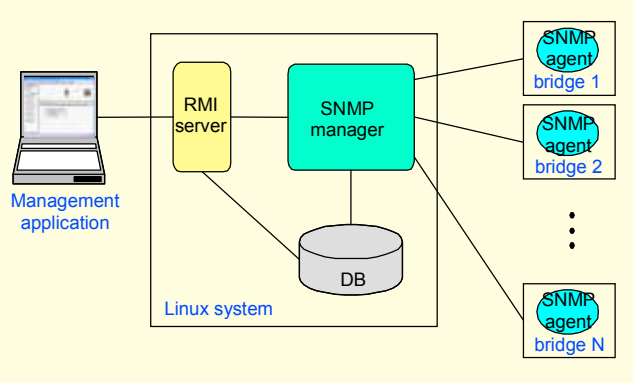


Fig. 6. Implementation architecture.

We implemented the management application on Windows XP using Java language and the SNMP manager on Linux OS using C plus language. The RMI Server which interfaces the management application to the SNMP manager is implemented on Linux OS using Java language.

Tables 3 through 6 show the data used for deciding on the set of edge bridges Q and for discovering the bridges' connections in the bridged network. Tables 7 and 8 show the data used for discovering connections between edge bridges and hosts in the host network. Table 7 is taken from the address learning function and Table 8 is taken by management.

Figure 7 depicts the representative test network for discovering physical topology in the metro Ethernet network. It

Table 3. Schema of dot1dPTDIfBasicEntry.

Name	Syntax	OID
agent_ip	String	
ifOperStatus	Integer	1.3.6.1.2.1.2.2.1.8

Table 4. Schema of dot1dPTDBaseInfo.

Name	Syntax	OID
agent_ip	String	
dot1BaseBridgeAddress	String	1.3.6.1.2.1.17.1.1

Table 5. Schema of dot1dPTDStp.

Name	Syntax	OID
agent_ip	String	
dot1dStpPriority	Integer	1.3.6.1.2.1.17.2.2
dot1dStpRootPort	Integer	1.3.6.1.2.1.17.2.7

Table 6. Schema of dot1dPTDStpPortEntry.

Name	Syntax	OID
agent_ip	String	
dot1dStpPort	Integer	1.3.6.1.2.1.17.2.15.1.1
dot1dStpPortState	Enumerated	1.3.6.1.2.1.17.2.15.1.3
dot1dStpPortEnable	Enumerated	1.3.6.1.2.1.17.2.15.1.4
dot1dStpPortDesignated Bridge	String	1.3.6.1.2.1.17.2.15.1.8

Table 7. Schema of dot1dPTDTpFdbEntry.

Name	Syntax	OID
agent_ip	String	
dot1dTpFdbAddress	String	1.3.6.1.2.1.17.4.3.1.1
dot1dTpFdbPort	Integer	1.3.6.1.2.1.17.4.3.1.2
dot1dTpFdbStatus	Enumerated	1.3.6.1.2.1.17.4.3.1.3

Table 8. Schema of dot1dPTDStaticEntry.

Name	Syntax	OID
agent_ip	String	
dot1dStaticAddress	String	1.3.6.1.2.1.17.5.1.1.1
dot1dStaticReceivePort	Integer	1.3.6.1.2.1.17.5.1.1.2

is composed of twelve bridges and one hundred ninety three hosts. The bridge number shown in Fig. 7 represents the bridge priority; a small number has high priority. We used various products such as Cisco's Catalyst3550 [15], Riverstone's RS1000 and RS3000 [16], and Paxcomm's NDX2104 and NGX2104 [17] to demonstrate interoperability.

2. Results

A primary goal of the experimental study with our physical topology discovery tool is to correctly determine a physical topology including many links eliminated by STP. We found a graph with multiple inactive paths for the metro Ethernet network as well as a spanning tree, which related works also had found. A second goal of our experimental study is to verify the practicality of our physical topology discovery algorithm by measuring its calculation time requirements for various network sizes. We tested the relationship between the number of bridges and physical topology discovery calculation time and the relationship between the number of hosts and physical topology discovery calculation time.

Although the accuracy of the physical topology is the most important criterion when judging the performance of this algorithm, its time performance may also be important in some applications. The majority of the execution time is spent obtaining the interface MIB [16] and bridge MIB [7] from the bridges. The execution time depends principally on the communication delay due to the slow speed of SNMP. To minimize this influence, our algorithm separated the data calculation function from the data collection procedure. Due to backup physical topology data in our DB, we can discover a physical topology at any time without decreasing performance. The performances of previous approaches [3]-[5] decreased significantly in proportion to the number of hosts due to the size of the AFT, as we addressed in section IV. However, our physical topology discovery algorithm is not seriously affected by the number of hosts because we divided the metro Ethernet network into a bridged network and host networks, and for the bridged networks our algorithm does not need AFT entries. It took below one second to calculate our physical topology discovery under the environment of less than 5000 hosts and 100 bridges. Therefore, our algorithm is a very precise solution to discover a physical topology for metro Ethernet networks.

For rapid convergence, RSTP became the IEEE 802.1w [8] standard in 2001. And for supporting multiple trees in VLAN environments, MSTP became the IEEE P802.1s/D15 [9] standard at the end of 2002. However, further work is required to standardize MIBs for RSTP and MSTP. The standard MIB for RSTP is now being processed in an IEEE draft [12] and the MSTP MIB is ready to be proposed in an IEEE draft. When the RSTP MIB is implemented, we don't need to get the

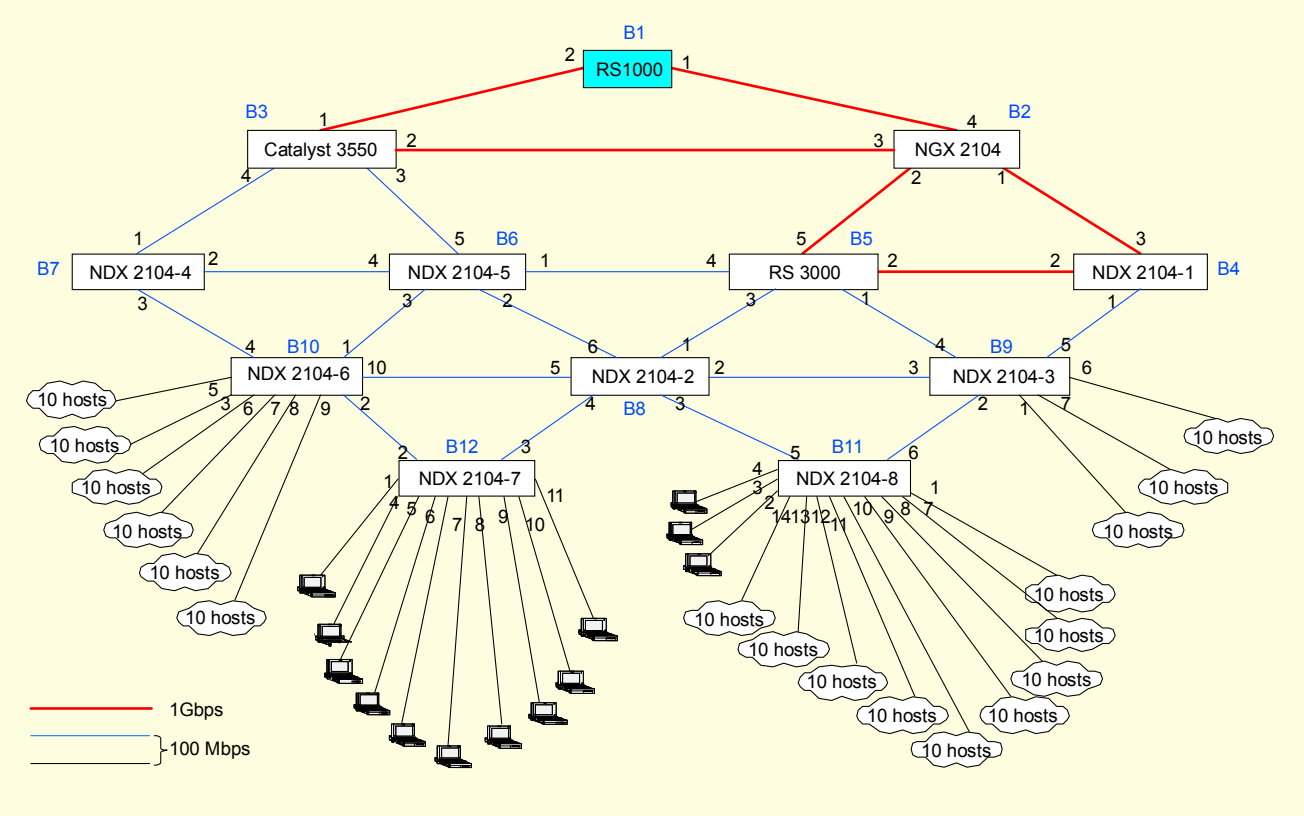


Fig. 7. Test network for discovering a physical topology.

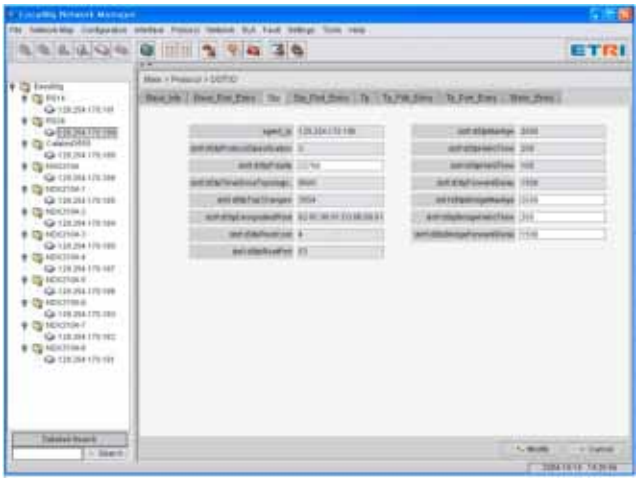


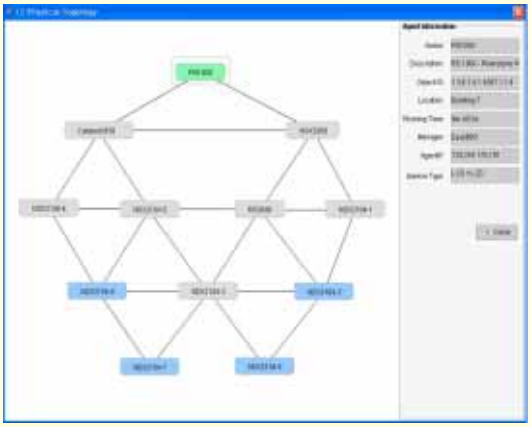
Fig. 8. Network manager's user interface.

interface MIB to decide on the edge bridges because the edge port in the RSTP MIB may be used instead of the number of active ports. Moreover, if we could get an MSTP MIB in a bridged network, we would be able to discover multiple logical VLAN paths. We hope to be able to implement this research in the near future.

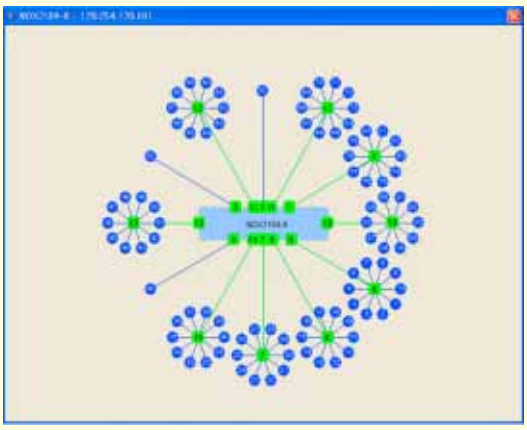
Our algorithm found an exactly complete physical topology including inactive interfaces eliminated by the STP in metro Ethernet networks. Figure 8 shows an example of our network manager's user interface when connected to the twelve bridges depicted in Fig. 7. Figure 9(a) shows the results of a physical topology for our test network in our ETRI laboratory. In Fig. 9(a), the RS1000 is a root bridge and the blue colored four bridges are edge bridges. When we click these edge bridges we can know the physical topology in the host network depicted in Fig. 9(b). Figure 9(b) shows the results of a host network connected to an edge bridge, NDX2104-8. Ports 1, 7, 8, 9, 10, 11, 12, 13, and 14 are connected to the edge bridge NDX2104-8 through a hub, and ports 2, 3, and 4 are directly connected to the edge bridge NDX2104-8.

Although the correctness of the topology is the most important criterion when judging the performance of this algorithm, its execution time performance may also be important in some applications. The majority of the execution time is spent fetching the interface MIB and bridge MIB information from the bridges. This is partially due to the slow speed of SNMP, but most significantly due to a desire not to swamp the bridges with queries. However, the topology of metro Ethernet networks changes rather slowly. To minimize this overhead, we isolated the topology calculation procedure from the MIB data fetch procedure.

Figures 10 and 11 show the performance of the physical



(a) Bridged network topology



(b) Host network topology

Fig. 9. Result of our physical topology discovery.

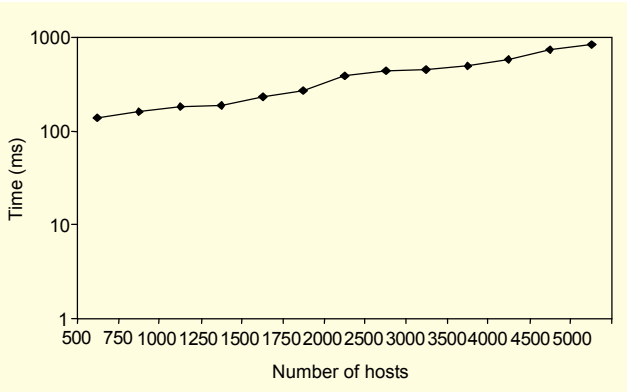


Fig. 10. The performance of the time to calculate a topology with varying numbers of hosts and fewer than 12 bridges.

topology discovery algorithm running on a 2.20 GHz Pentium IV. The number of hosts and bridges were varied by adding the entries for hosts or bridges from a previously collected DB.

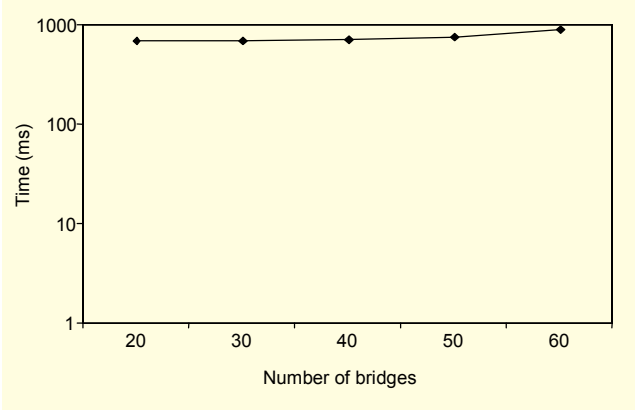


Fig. 11. The performance of the time to calculate a topology with varying numbers of bridges and fewer than 5000 hosts.

VII. Conclusions

Automatic discovery of physical topology plays a crucial role in enhancing the manageability of modern metro Ethernet networks. Despite the importance of the problem, earlier research and commercial network management tools have typically concentrated on either i) discovering logical (that is, Layer-3) topology, which implies that the connectivity of all Layer-2 elements is ignored, or ii) proprietary solutions targeting specific product families. In this paper, we have proposed a novel and practical algorithm for discovering the physical topology in a metro Ethernet network. The common problem of the recent work by other researchers is that their algorithms can only discover spanning tree paths and not the complete physical topology of a metro Ethernet network. They also had a serious performance reducing problem because their algorithm depended on AFT entries only. To our knowledge, our algorithm is the only solution that can find the exact physical topology in a metro Ethernet network. It took below one second to calculate our physical topology discovery in conditions of fewer than 5000 hosts and fewer than 100 bridges. As metro Ethernet network technology develops, we will continue to study topology discovery exploiting RSTP and MSTP as they mature to overcome the limitations of STP.

References

[1] Ralph Santitoro, "Metro Ethernet Services-A Technical Overview," In the *Metro Ethernet Forum 2003*, http://www.metroethernetforum.org.

[2] A. Bierman and K. Jones, *Physical Topology MIB*, Sept. 2000, Internet RFC-2922, http://www.ietf.org/rfc/.

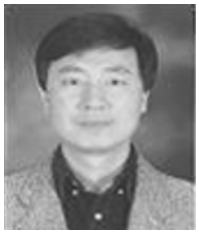
[3] Yuri Breibart, Minos Garofalakis, Cliff Martin, Rajeev Rastogi, S. Seshadri, and Avi Silberschatz, "Topology Discovery in Heterogeneous IP Networks," *IEEE INFOCOM* 2000, Tel Aviv,

Israel, Mar. 2000, pp. 265-274.
[4] Bruce Lowekamp, David R. O'Hallaron, and Thomas R. Gross, "Topology Discovery for Large Ethernet Networks," *ACM SIGCOMM* 2001, San Diego, California, USA, Aug. 2001, pp. 237-248.
[5] Yigal Bejerano, Yuri Breitbart, Minos Garofalakis, and Rajeev Rastogi, "Physical Topology Discovery for Large Multi-Subnet Networks," *IEEE INFOCOM* 2003, San Francisco, USA, Apr. 2003, pp. 342-352.
[6] K. McCloghrie and F. Kastenholz, *The Interface Group MIB*, June 2000, Internet RFC-2863, http://www.ietf.org/rfc/.
[7] RFC-1493, *Definitions of Managed Objects for Bridges*, IETF, E. Decker and P. Langille, July 1993, http://www.ietf.org/rfc/.
[8] IEEE Std 802.1w, *Amendment to IEEE Std 802.1D, 1998 Edition (ISO/IEC 15802-3:1998) and IEEE Std 802.1t-2001, Part 3: Media Access Control (MAC) Bridges-Amendment2: Rapid Reconfiguration*, IEEE, 2001.
[9] IEEE P802.1s/D15, *Draft Standard for Local and Metropolitan Area Networks-Amendment 3 to 802.1Q Virtual Bridged Local Area Networks: Multiple Spanning Trees*, *Amendment to IEEE Std 802.1Q-1999,* IEEE, 2002.
[10] M. J. Choi, "XML-Based Network Management for IP Networks," *ETRI J.*, vol. 25, no. 6, Dec. 2003, pp.445-463.
[11] William Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2,* Third Edition, Addison-Wesley.
[12] Robert Breyer and Sean Riley, *Switched, Fast, and Gigabit Ethernet*, Third Edition, Macmillan Technical Publishing.
[13] ANSI/IEEE Std 802.1D, *Part 3: Media Access Control (MAC) Bridges*, IEEE, ISO/IEC 15802-3:1998, 1998.
[14] W. K. Hong, "An ATM Network Management System for Point-to-Multipoint Reservation Service," *ETRI J.*, vol. 24, no. 4, Aug. 2002, pp.299-310.
[15] Corporate Headquarters, "Catalyst 3550 Multilayer Switch Software Configuration Guide," March 2003, Cisco IOS Release 12.1(13)EA1, http://www.cisco.com.
[16] Riverstone Networks, "RS Switch Router User Guide-Release 9.1," http://www.riverstonenet.com.
[17] PaxComm, *NetSteer NDX-2124 User Guide - Real Ethernet Switching Layer 2*, Ver. 2.62, http://www.paxcomm.com.
[18] V. Ngai and E. Bell, *Definitions of Managed Objects for Bridges with Rapid Spanning Tree Protocol*, March 2004, draft-ietf-bridge-rstpmib-04.txt, http://www.ietf.org/internet-drafts/.

Myung-Hee Son has been an Engineer Staff Member of Electronics and Telecommunications Research Institute (ETRI) since March 2000. She received the BS degree in aerospace engineering from Chungnam National University in 1998 and the MS degree in computer engineering from Chungnam National University in 2000. And she received the PhD degree at the Department of Information and Communications Engineering of Chungnam National University, Korea. Her current research interests include network protocols, mobile RFID technology and analysis, and the study of mobile communications in sensor networks.

Bheom-Soon Joo received the BS degree in electronic engineering from Seoul National University, Korea, in 1983 and the MS degree in electronic engineering from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1999. He has worked with ETRI from 1984. Currently, he is a Principal Engineer and Team Leader at the Ethernet Technology team in ETRI. His research interests span the fields of hardware development for Ethernet switch systems, hardware development of ATM switch systems, network synchronization, and high speed interconnection.

Byung-Chul Kim received the BS degree in electronic engineering from Seoul National University and the MS and PhD degrees in electronic engineering from KAIST, Korea, in 1988, 1990, and 1996. Since 1999, he has been a Professor at the Department of Information and Communications Engineering of Chungnam National University, Korea. Also, from 1993 to 1999, he worked as a Research Engineer at Samsung Electronics. His research interests include computer networks, wireless internet, sensor networks, and mobile communications.



Jae-Yong Lee received the BS degree in electronic engineering from Seoul National University and the MS and PhD degrees in electronic engineering from KAIST, Korea, in 1988, 1990, and 1995. Since 1995, he has been a Professor at the Department of Information and Communication Engineering of Chungnam National University, Korea. Also, from 1990 to 1995, he worked as a Research Engineer at Digicom Institute of Information and Communications. His research interests include computer networks, Internet protocols, traffic control, performance analysis, and wireless Internet.