INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2003; **13**: 000–000 (DOI: 10.1002/nem.478)

Development of SNMP-XML translator and gateway for XML-based integrated network management

By Jeong-Hyuk Yoon, Hong-Taek Ju*† and James W. Hong

The research objective of our work is to develop a SNMP MIB to XML translation algorithm and to implement an SNMP-XML gateway using this algorithm. The gateway is used to transfer management information between an XML-based manager and SNMP-based agents. SNMP is widely used for Internet management, but SNMP is insufficient to manage continuously expanding networks because of constraints in scalability and efficiency. XML–based network management architectures are newly proposed as alternatives to SNMP-based network management, but the XML-based Network Management System (XML-based NMS) cannot directly manage legacy SNMP agents. We also implemented an automatic specification translator (SNMP MIB to XML Translator) and an SNMP-XML gateway. Copyright © 2003 John Wiley & Sons, Ltd.

Introduction

Since the late 1980s, the Internet has increased explosively with the growth of the World Wide Web.[1] It is a complex network composed of diverse network equipment from various venders. This situation necessitates the need for network management. The Simple Network Management Protocol (SNMP)[2] was devised as a temporary means for Internet management, and is still used as the most popular mode because of its merits, such as ease of implementation and great interoperability.

Despite these strong points, SNMP has drawbacks, in both scalability and efficiency.[3] The SNMP-based Network Management System

Jeong-Hyuk Yoon received his BS degree in electronic engineering from Kumoh National University of Technology in 1995. He has worked for SK Telecom since 1995. He received his MS degree in the Department of Computer and Communications Engineering, Graduate School for Information Technology, Pohang University of Science and Technology (POSTECH) in 2002. His research interests include network management and Web technology.

Hong-Taek Ju is a senior researcher in the PIRL (POSTECH Information Research Laboratory), POSTECH, Pohang, Korea. He received his BSc degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1989, MSc and PhD degrees in Computer Science and Engineering from Pohang University of Science and Technology (POSTECH) in 1991 and 2002, respectively. His research interests include distributed processing and network management.

James W. Hong is an associate professor in the Department. of Computer Science and Engineering, POSTECH, Pohang, Korea. He has been with POSTECH since May 1995. He has been very active as a participant, program committee member and organizing committee member for IEEE CNOM-sponsored symposiums such as NOMS, IM, DSOM and APNOMS. For the last several years, he has been working on various research projects on network and systems management, which utilize Web, Java and CORBA technologies. His research interests include network and systems management, distributed computing and traffic engineering and planning.

*Correspondence to: Hong-Taek Ju, Department of Computer Science and Engineering, Pohang University of Science and Technology, San 31, Hyojadong Namgu, Pohang, South Korea 790-784.
†E-mail: juht@postech.ac.kr*

(NMS) has reached its limit to manage the rapidly growing Internet since the 1990s. Research on NMS using XML[6] has appeared recently to address the limits of SNMP-based NMS and to efficiently manage the copious amount of management data produced in large networks. This research uses XML to transfer, process, and store management data in a Web-based NMS environment. This NMS uses HTTP to transfer management data, XML documents[3,4] and standard XML processing methods to process the documents in a manager system.[5] This XML-based NMS has many advantages, such as high reliability of data transfer, high interoperability, low network overhead, and low latency. However, the newly evolved XML-based NMS does not have the means to manage legacy SNMP agents directly. Most network devices used worldwide are equipped with an SNMP agent. If the XML-based NMS cannot manage legacy SNMP agents directly, the usefulness of the NMS is not sufficient as an NMS for the Internet.

Therefore, research is required to learn the best approach to use the XML-based manager's merits to manage the Internet composed of devices equipped with legacy SNMP agents. We propose an SNMP-XML gateway to solve the problem. The gateway delivers management data between an XML-based manager using HTTP and SNMP-based agents using SNMP. For specification translation of the gateway, we need an algorithm to convert SNMP MIB (Management Information Base)[13] and management information of the SNMP agent to XML. For the interaction translation of the gateway, we need some mapping methods.

In this paper, we first design a translation algorithm that converts SNMP MIB to XML and implements an automatic translator using the algorithm. Next, we develop an SNMP-XML gateway using the translator. The gateway will be adapted to XML-based NMS to manage legacy SNMP agents.

After this introductory section, we discuss related work and its limits, and propose our solution in the second section. In the third section, we develop an SNMP MIB to XML translation algorithm and develop an SNMP MIB to XML translator to realize the algorithm. In the fourth section, we explain the development of our SNMP-XML gateway and practice it. For the gateway, we define interaction translation methods and realize them. We present conclusions and future work in the final section.

Related Work and Proposed Approach

In this section, we first overview SNMP-based and XML-based network management, and discuss some related work on gateway. Then, we analyze drawbacks with previous solutions. Finally, we propose our approach to solve the problems.

—Limits of SNMP-based NMS—

In the late 1980s, the Internet began to increase explosively, empowered by the WWW. However, Internet management was not established beyond finding the end-to-end connection state using an Internet Control Message Protocol (ICMP).[9] Network administrators needed more information than merely the connection status. They also needed a standardized management method to manage various network devices made by multi-vendors in common ways.

SNMP was first standardized by the Internet Engineering Task Force (IETF)[21] in 1990. SNMP version 2 (SNMPv2)[15] that extends SNMPv1 was proposed in 1995. Also, SNMP version 3 (SNMPv3)[16] that adopted string security into SNMP was proposed in 1999. The communication model of SNMP is the Manager/Agent paradigm. An SNMP agent is equipped to manage devices, and collects management information to send to the SNMP manager. The SNMP manager processes the received data and stores or presents the data to the user.

The data transfer methods are polling and trap. Polling is a request/response mechanism, and the trap is event-driven notification. A comparison table of each version of SNMP is given in Table 1. SNMPv1 uses SMIv1 for modeling management information, and defines the four operation methods shown in Table 1. SNMPv2 uses SMIv2 for modeling management information that is a more extended modeling rule, and defines the 'Inform' operation as a communication method between managers in addition and supplements 'trap' action by 'SNMPv2-trap'. SNMPv3 uses a modeling method and operation methods such as SNMPv2, but achieves greater security.

	SNMPv1	SNMPv2	SNMPv3
MO modelling	SMIv1	SMIv2	SMIv2
Protocol operation	Get, GetNext, Set	Get, GetNext, GetBulk Set	Get, GetNext, GetBulk Set
	Trap	SNMPv2-trap, Inform	SNMPv2-trap, Inform
Feature	First standard	Extended SMI, Communication between managers, bulk transfer	Strong security
Standard	Full (1991)	Full (1999)	Draft (1999)
RFC No.	1155, 1212, 1213, 1215	1901~1908, 2578~2580	2570~2576

Table 1. Comparison of SNMP versions

A major problem with SNMP is scaleability and efficiency.

A major problem with SNMP is scalability and efficiency. Scalability refers to the number of agents which can be managed in a single manager system, and efficiency means how quickly and effectively a system operates in such actions as the delivery and processing of data. The problems of scalability and efficiency become an issue in the increase of management data.

When data to manage was less in the past, the 'simplicity' of SNMP was a great advantage to design and implement an NMS. However, the network has evolved enormously, and management data which should be passed through the network and processed in managers and agents also increased in tandem. Thus, there is a problem to manage the huge network with SNMP-based NMS.

Management data increases because the entire managed system increases with the growth of the Internet and the quantity of management data of each agent. As the MIB that the agent supports is also increasing and the network scale increases, the MIB table (IP routing table, TCP connection table, accounting table etc.) is becoming larger. Also, not only simple monitoring but also the network plan and quantity of data that inquire beforehand is increasing. An SNMP-based management system cannot perform the delivery and processing of management with limited efficiency for the following four reasons.[4]

First, increase of network overhead. Because traffic connected with NMS is not generated for user services, in fact the whole traffic for management is only overhead. Therefore, we must keep traffic for network management at its minimum. In the SNMP mechanism, however, overhead increases continuously because it is proportional to the increasing rate of management data. Increasing overhead influences network traffic for actual services.

Second, increase of latency. Latency that takes delivery and processing of data should be limited to the minimum. If latency increases, the time to discover a problem may take too long for normal management, and when a manager cannot properly grasp a problem proportional to the number of agents to manage, the total polling time increases.

Third, the processing capacity of the manager. The processing capacity depends on the performance of hardware resources and decentralized processing. Hardware resources, such as CPU, memory etc. cannot increase without limits because of cost and performance limitations of the hardware itself. Also, SNMP is limited to diminishing the processing subordinate of the manager because SNMP does not have a distributed processing structure that utilizes communication between managers or a management model of a hierarchical structure.

Fourth, limit of the manager's local segment. The extent to which one manager can manage is limited in the SNMP management framework, because SNMP uses a centralized mechanism. Data from all agents is gathered in a segment point in which the manager is connected, and this causes

a bottleneck. Thus, network overhead should be reduced as long as possible. Specifically, it is necessary to improve the efficiency of the communication protocol.

As an attempt to solve these problems of SNMP, network management using XML is studied.

—Advantages of XML for Network Management—

XML[6] is a meta-markup language which was standardized by the World Wide Web Consortium (W3C) in 1998 for document exchange on the World Wide Web. It is reduced SGML (Standard Generalized Markup Language).[8] Therefore, XML has the advantage of easy transmission using HTTP like HTML[7] and has the advantage of extensiveness like SGML. At the same time, it can easily transfer structured documents on the Internet. With a wide expression power of documents and data, XML has a major potential to become the standard interface for management information. The advantages of XML compared to HTML as the standard for exchanging documents on the WWW are as follows.

First, XML, unlike HTML, separates the contents of document and the expression method. The expression method is standardized to XSL (eXtensible Stylesheet Language).[17] A document can be expressed in various ways by changing this style seat. Second, the support structures document definition. It is possible by DTD and XML schema that defines data structure in an XML document. Third, there is an extended link function. XML standard set offers Xpointer[23] and Xlink[22] to link and to indicate an entire document or one part of a document. This supplies extended link methods between documents, such as a mutual link or a multi-way link. Fourth, it offers standard API for document processing. XML supports a standard interface about DTD that define the XML document itself and document model unlike SGML. All programs that support Document Object Model (DOM)[18] can treat XML documents easily for access, abstraction, and storing. Simple API for XML (SAX)[19] achieves work that produces and specifies the beginning and end of each element and document.

XML has many advantages for information transmission and especially for processing network management. For information modeling, all information can easily be modeled by using the powerful modeling function of XML. Specifically, conversion of SNMP MIB structure information can be achieved easily by XML. For information transmission, XML documents can be easily transferred by using an HTTP stack which is already equipped worldwide. For compatibility, XML supports exchange of information between all hardware and software platforms which support HTTP. The XML can used to function as middleware through these advantages. For development cost, XML needs a low cost for system development because it has various standard APIs and free development kits. Also, XML supports delivering data in great quantity at once without the limitation of document size for network management. These advantages of XML are studied by technology that solves the problems of scalability and efficiency of existing SNMP-based NMS.

—XML-based Network Management—

XML-based network management refers to a network management method which defines management information by XML, and exchange of data for management in the form of an XML document, and uses an XML document processing standard method for processing data.

XML-based network management using an embedded web server—Many devices use an embedded web server (EWS) to manage a network device. Research seeks to extend the EWS to network management from element management.[3,25] The research studied network management using XML to pass management data. The managed device has a WBM agent (Web-based Management agent) which has an EWS, and a manager system to interact with the agent to manage the device. The communication model between the manager and the agent is defined in Figure 1.

Figure 1(a) depicts a method where the manager requests the agent for information. The agent responds with an HTTP GET request to the manager. (b) is a method where the manager controls the agent, and the manager sends the

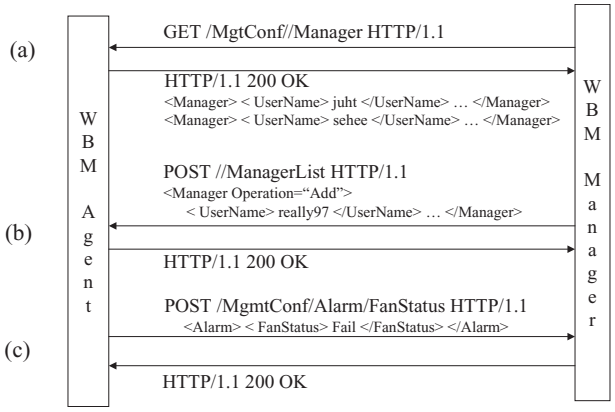

Figure 1. Communication model of XML-based network management

necessary information by an HTTP POST message to the agent, and the agent responds with corresponding actions. (c) displays the communication method where the agent reports to the manager when there is an event that happens in an agent. Each action performs a function similar to the action method of get, set, trap in SNMP. This communication model presents ways to deliver management information by using an XML document over HTTP. This XML-based NMS becomes an alternative to solve the drawbacks of SNMP-based NMS described previously. XML-based NMS has many advantages. However, this system is not supplying the ways to manage network devices equipped only with a legacy SNMP agent. The SNMP agent is used in networks worldwide. It is actually impossible to modify all SNMP agents to support HTTP as the WBM agent.

Therefore, a method to manage the legacy SNMP agent directly without change by using XML-based manager should be studied in order to use the advantages of XML-based NMS and to manage the SNMP agent at the same time.

Concept using XML to network management—J.P. Martin-Flatin presented an idea to use XML for integrated management in his research on Web-based integrated network management architecture.[4] Moreover, he proposed SNMP MIB to XML translation models, namely Model-level mapping and Metamodel-level mapping. Model-level mapping is a method where DTD is made out specifically to SNMP MIB, which make elements and attributes in XML DTD by using SNMP variable identifiers just as they are. An example of model-level mapping follows.

```
<interface>
<bandwidth type="string">100
  Mbit/s</bandwidth>
</interface>
```

The advantage of using XML documents based on this model is that readability is superior and it is easy to understand intuitively. The disadvantage is that this model needs many DTDs, because each MIB needs its own DTD.

The Metamodel-level mapping method defines one generalized DTD, and applies to all MIBs. This model does not use variables defined in MIB but uses general keywords for defining DTD. If we convert the simple example shown in Model-level mapping to Metamodel-level mapping again, it can be expressed as follows.

```
<class name="interface">
<property name="bandwidth"
 type="string">
<value>100 Mbit/s</value>
</property>
</class>
```

Martin-Flatin's research, on Web-based integrated management of an IP network and system, describes the advantage of HTTP/XML-based communication, and presents the basic idea about SNMP MIB to XML conversion. However, it does not provide a specific algorithm about the conversion of SMI to XML, but only shows an example of MIB to XML conversion about a specific MIB node that is an interface group of MIB II.

XML-based manager for SNMP polling—Some research concerned the development of a management data collecting module using XML.[5] The objective of the research was to simplify the development of NMS and apply for Web-based network management. In the research, the manager was composed of an XML-based polling and a data processing module that uses standard APIs for XML handling. This method reduces development cost and time of NMS compared to

prior methods. In the paper, the author developed an SNMP SMI to XML converter that converts various forms of SNMP MIB to XML DTD files. The research uses XML DOM as a mediator for collecting, processing, and storing management data from various SNMP agents.

This research uses DTD for defining XML documents. However, this approach has limits for presenting all information of MIB into XML. Also, the paper did not describe a concrete translation algorithm for SNMP MIB to XML conversion, and appropriate operations for the SNMP trap.

Research on SNMP MIB to XML conversion—Previous research on the methods to translate SNMP MIB to XML is as follows. First, J.P. Martin-Flatin presented an example of a mapping method for MIB to XML translation.[4] In his paper, he describes conversion results about a 'system' group and an 'interface' group of MIB II by example without presenting a method on the whole MIB conversion algorithm. Second, there is research on the XML expression of SNMP MIB in a CORBA/SNMP gateway development project[10] at the Bell Research Institute. However, this research did not arrange a specific translation algorithm either, and merely showed examples of an XML document using XML DTD for definition and an actual data instance document for MIB II nodes. Third, there is Frank Strauss' MIB processing library, 'libsmi',[11] which translates SNMP MIB to other languages, such as Java, Corba, C, XML etc. However, this library program causes information loss, and the converted XML does not contain all the information of the source MIB but merely the structure and node name, with many features of each MIB node ignored. Thus, this program does not have a concrete translation algorithm. Finally, the IBM Research Institute announced a research paper on the translation of ASN.1 to XML.[12] This is about the conversion of ASN.1, which is a superset of SMI. However, it does not support the conversion of macro functions defined in SNMP SMI, and its conversion method uses DTD.

Previous research does not supply specific conversion algorithms, and there is information loss in the conversion process. The largest loss is in various data types of MIB nodes, so these have limits as standard methods.

—Our Proposed Approach—

We examined problems with the current SNMP-based network management system, low scalability, and low efficiency, although SNMP is widely used. As an alternative to this problem, XML-based NMS is being developed. We call the manager system of XML-based NMS an 'XML-based manager.' The XML-based manager presents a method to manage a huge network, but it does not provide a method to manage the widely utilized accommodating SNMP-based agent.

In this paper, we suggest how the XML-based manager directly manages legacy SNMP agents. We design and implement a gateway that relays messages between the SNMP-based agent and the XML-based manager. The XML-based manager uses a method to pass data by XML document over HTTP, and the SNMP agent passes data to SNMP message over SNMP. The gateway converts and relays data between the two protocols.

For the implementation of a gateway, we need both a specification translation and an interaction translation. For a specification translation of the gateway, we translate SNMP MIB into XML. Previous research for this translation does not have a concrete translation algorithm. We defined an SNMP MIB to XML translation algorithm as a standard method. We also implemented an automatic translator with this algorithm. For preventing information loss in the translation process, we used an XML Schema which supports various data types defined in SNMP SMI and powerful modeling.

For the implementation of a gateway, we need both a specification translation and an interaction translation.

For interaction translation of the gateway, we define a mapping model for each operation of SNMP into HTTP to translate mutually between SNMP message and XML document. We also implement the translator as a module of the gateway.

SNMP MIB to XML Translation

In this section we explain an SNMP MIB to XML Schema translation algorithm for a specification translation of our gateway, and show how to implement a translator using the algorithm.

—Translation Algorithm—

First, consideration for conversion of information model for network management keeps all information of a before-conversion model in an after-conversion model. For this purpose, it is important to keep the structure of an information model equally. Table 2 arranges a mapping method of the document structure.

SNMP SMI	XML Schema
Node (macro definition)	Element
Node name	Element name
Clauses in node	Attributes of the element

Table 2. Document structure conversion

Each node of SNMP MIB is converted into an element of the XML Schema, and the name of the node is converted into the name of the element. Interior clauses of the MIB node are converted into attributes of the XML element.

Data type definition—In order to define data types of SNMP MIB into XML, we used an XML Schema which is established as a standard by W3C, because it is impossible to express various data types with XML DTD.

Table 3 shows XML Schema definitions translated from SMIv1 data types. The 'IpAddress' type is specially defined with the range of IP address, [0-255].[0-255].[0-255].[0-255]. Besides, various types, such as number type, string type can be changed without loss of information. Data types defined in SMIv2 converted into XML Schema definition in Table 4.

These types conform with primary data type that define XML Schema. Data types defined by user are converted into XML Schema definition in Table 5.

Data types which is defined in SNMP MIB by user can defines using string replacement such as 'DisplayString : : = OCTET STRING'. Designer can define new data type if use XML Schema although it is impossible with XML DTD.

SMI expression	XML Schema definition
IpAddress : : = OCTET STRING (SIZE (4)) (4))	<xsd:simpleType name = "IpAddress"> <xsd:restriction base = "xsd:string"> < xsd:pattern value = " (([1 - 9] ?[0- 9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}([1-9]?[0 - 9] | 1 [0 - 9] [0 - 9]| 2 [0 - 4] [0 - 9] | 25 [0 - 5]) " / > </ xsd:restriction> </ xsd:simpleType>
Counter : : = INTEGER (0 .. 4294967295)	<xsd:simpleType name = "Counter"> <xsd:restriction base = "xsd:unsignedInt"> </ xsd:restriction> </ xsd:simpleType>
TimeTicks : : = INTEGER (0 .. 4294967295)	<xsd:simpleType name = "TimeTicks"> <xsd:restriction base = "xsd:unsignedInt"> </ xsd:restriction> </ xsd:simpleType>
Opaque : : = OCTET STRING	<xsd:simpleType name = "Opaque"> <xsd:restriction base = "xsd:string"> </ xsd:restriction> </ xsd:simpleType>

Table 3. XML Schema definition of SMIv1 data types

SMI expression	XML Schema definition
Counter64 ::= INTEGER (0 .. 18446744073709551615)	<xsd:simpleType name = "Counter64"> <xsd:restriction base = "xsd:unsignedLong"> </ xsd:restriction> </ xsd:simpleType>
Gauge32 :: = INTEGER (0 .. 4294967295)	<xsd:simpleType name = "Gauge32"> <xsd:restriction base = "xsd:unsignedInt"> </ xsd:restriction> </ xsd:simpleType>
Unsigned32 ::= INTEGER (0 .. 4294967295)	<xsd:simpleType name = "Unsigned32"> <xsd:restriction base = "xsd:unsignedInt"> </ xsd:restriction> </ xsd:simpleType>

Table 4. XML Schema definition of SMIv2data types

MIB definition	XML Schema definition	Notes
Enumerated INTEGERs (ex) : SYNTAX INTEGER { Up (1) Down (2) Testing (3) }	<xsd:complexType> <xsd:simpleContent> <xsd:restriction base = "xsd:int"> <xsd:enumeration value = "1" /> <xsd:enumeration value = "2" /> <xsd:enumeration value = "3" /> </ xsd:restriction> </ xsd:simpleContent> </ xsd:complexType>	Have one value among listed values.
DisplayString (SIZE (0 .. 255))	<xsd:simpleType name = "DisplayString_0_255"> <xsd:restriction base = "xsd:string"> <xsd:minLength value = "0" /> <xsd:maxLength value = "255" /> </ xsd:restriction> </ xsd:simpleType>	Assignment of string length.

Table 5. XML Schema definition of user-defined data types

Translation of macro—SNMP MIB is defined by macro definition which is defined in SMI. Thus, first we should convert the macro definitions into XML definitions. There are OBJECT-TYPE macro and TRAP-TYPE macro in SMIv1.

The OBJECT-TYPE macro is a representative macro that defines the table node or the data node of MIB. Through the conversion method of this macro, we can understand a method to change the MIB's node to XML's element. We describe the general style of MIB node expression that uses the OBJECT-TYPE macro, as follows.

```
NodeName OBJECT-TYPE
  SYNTAX "SyntaxType"
  ACCESS "AccessType"
  STATUS "StatusType"
  DESCRIPTION "DescriptionText"
  REFERENCE "ReferenceType"
  INDEX "IndexList"
  DEFVAL "DefaultValue"
: : = {parentNodeName nodeNumber}
```

The above general MIB node expression can be presented in the XML Schema as follows.


```
<xsd:element name = "NodeName">
<xsd:complexType>
<xsd:simpleContent>
<xsd:restriction base = "xsd:string">
<xsd:sequence>
  (lower part node definition part)
</ xsd:sequence>

<xsd:attribute name = "oid" type =
  "xsd:string" use = "fixed" value =
  "OidValue" />
<xsd:attribute name = "Access" type =
  "xsd:string" use = "fixed" value =
  "AccessType" />
<xsd:attribute name = "Status" type =
  "xsd:string" use = "fixed" value =
  "StatusType" />
<xsd:attribute name = "Description"
  type = "xsd:string" use = "fixed"
  value = "DescriptionText" />
<xsd:attribute name = "Reference"
  type = "xsd:string" use = "fixed"
  value = "ReferenceType" />
<xsd:attribute name = "Index" type =
  "xsd:string" use = "fixed" value =
  "IndexList" />
<xsd:attribute name = "Defval" type =
  "xsd:string" use = "fixed" value =
  "DefaultValue" />
</ xsd:restriction>
</ xsd:simpleContent>
</ xsd:complexType>
</ xsd:element>
```

We changed the 'NodeName' corresponding to the name of the node into an element name in XML, and interior paragraphs, such as 'ACCESS', 'STATUS' etc. are defined into attributes of the element which correspond to their meaning. We defined the new attribute of 'oid' and, the value of 'oid' is absolute OID value of relevant node. This 'oid' value is used by the key to indicate each node. Oid value of 'system' node in MIBII is '1.3.6.1.2.1.1'.

TRAP-TYPE macro is used for the definiton of 'trap' function, where an agent reports events to manager. The general node form of TRAP-TYPE macro is as follows.

```
NodeName TRAP-TYPE
  ENTERPRISE "EnterpriseName"
  VARIABLES "VariableType"
  DESCRIPTION "DescriptionText"
  REFERENCE "ReferenceType"
: : = trapNumber
```

If we convert the above MIB definition into an XML Schema, the result is as follows.

```
<xsd:element name = "NodeName">
<xsd:complexType>
<xsd:simpleContent>
<xsd:restriction base = "xsd:string">
<xsd:attribute name = "Trapnumber"
  type = "xsd:int" use = "fixed" value
  = "TrapNumber" />
<xsd:attribute name = "Enterprise"
  type = "xsd:string" use = "fixed"
  value = "EnterpriseName" />
<xsd:attribute name = "Description"
  type = "xsd:string" use = "fixed"
  value = "DescriptionText" />
<xsd:attribute name = "Reference"
  type = "xsd:string" use = "fixed"
  value = "ReferenceType" />
</ xsd:restriction>
</ xsd:simpleContent>
</ xsd:complexType>
</ xsd:element>
```

This conversion changes the name of node into the name of element, and interior clauses into the attributes of the element.

SNMPv2 SMI has various types of macros in addition to the OBJECT-TYPE macro that extends the previous macro. The MODULE-IDENTITY macro has information about the module. If we define the node that uses this macro according to the macro definition, a general form can appear as follows.

```
NodeName MODULE-IDENTITY
  LAST-UPDATED "Last-updated"
  ORGANIZATION "Organization"
  CONTACT-INFO "Contact-info"
  DESCRIPTION "Description"
    REVISION "Revision"
    DESCRIPTION
      "Revision_description"
: : = {parentNodeName nodeNumber}
```

This node has module information performing a function as a group node. The function as group

node can change into an element with only one attribute: 'oid'. Its XML Schema definition is as follows.

```
<xsd:element name = "ParentNodeName">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref = "NodeName" />
    </ xsd:sequence>
    <xsd:attribute name = "oid" type
      = "xsd:string" use = "fixed"
      value = "Parent_oid" />
  </ xsd:complexType>
</ xsd:element>
<xsd:element name = "NodeName">
  <xsd:complexType>
    <xsd:sequence>
      (...lower part element
        definition ...)
    </ xsd:sequence>
    < xsd:attribute name = "oid" type
      = "xsd:string" use = "fixed"
      value = " parent_oid.
      NodeNumber " />
  </ xsd:complexType>
</ xsd:element>
```

The general style of an MIB node using an OBJECT-TYPE macro defined in SMIv2 is as follows.

```
NodeName OBJECT-TYPE
  SYNTAX "SyntaxType"
  UNITS "UnitsType"
  MAX-ACCESS "AccessType"
  STATUS "StatusType"
  DESCRIPTION "DescriptionText"
  REFERENCE "ReferenceType"
  INDEX "IndexList"
  DEFVAL "DefaultValue"
: : = {parentNodeName nodeNumber}
```

The conversion method is similar to the OBJECT-TYPE macro of SMIv1, and a converted XML Schema definition is as follows.

```
<xsd:element name = "NodeName">
<xsd:complexType>
<xsd:simpleContent>
<xsd:restriction base = "xsd:string">
<xsd:sequence>
  (lower part node definition part)
</ xsd:sequence>

<xsd:attribute name = "oid" type =
  "xsd:string" use = "fixed" value =
  "OidValue" />
<xsd:attribute name = "Units" type =
  "xsd:string" use = "fixed" value =
  "UnitsType" />
<xsd:attribute name = "Access" type =
  "xsd:string" use = "fixed" value =
  "AccessType" />
<xsd:attribute name = "Status" type =
  "xsd:string" use = "fixed" value =
  "StatusType" />
<xsd:attribute name = "Description"
  type = "xsd:string" use = "fixed" value
  = "DescriptionText" />
<xsd:attribute name = "Reference" type
  = "xsd:string" use = "fixed" value =
  "ReferenceType" />
<xsd:attribute name = "Index" type =
  "xsd:string" use = "fixed" value =
  "IndexList" />
<xsd:attribute name = "Defval" type =
  "xsd:string" use = "fixed" value =
  "DefaultValue" />
</ xsd:restriction>
</ xsd:simpleContent>
</ xsd:complexType>
</ xsd:element>
```

The general node form which uses the NOTIFICATION-TYPE macro is as follows.

```
NodeName NOTIFICATION-TYPE
  OBJECTS "ObjectsType"
  STATUS "StatusType"
  DESCRIPTION "DescriptionText"
  REFERENCE "ReferenceType"
: : = {parentNodeName nodeNumber}
```

This node acts as an event reporter, without values. Therefore, the above node form can appear as follows using the XML Schema.

```
<xsd:element name = "NodeName">
<xsd:complexType>
<xsd:attribute name = "oid" type =
  "xsd:string" use = "fixed" value =
  "OidValue" />
```

```
<xsd:attribute name = "Objects" type =
  "xsd:string" use = "fixed" value =
  "ObjectsType" />
<xsd:attribute name = "Status" type =
  "xsd:string" use = "fixed" value =
  "StatusType" />
<xsd:attribute name = "Description"
  type = "xsd:string" use = "fixed" value
  = "DescriptionText" />
<xsd:attribute name = "Reference" type
  = "xsd:string" use = "fixed" value =
  "ReferenceType" />
</ xsd:complexType>
</ xsd:element>
```

The name of the node is used as the name of the element, and the interior paragraphs of node is defined by attributes of the element. The 'oid' attribute has the OID value of the node as the key value for communication.

A general node using OBJECT-IDENTITY macro is as follows.

```
NodeName OBJECT-IDENTITY
  STATUS "StatusType"
  DESCRIPTION "DescriptionText"
  REFERENCE "ReferenceText"
: : = {parentNodeName nodeNumber}
```

The node expresses a group node that does not have values but child nodes. If the above general node expression is displayed by the XML Schema, the result is as follows.

```
<xsd:element name = "NodeName">
  <xsd:complexType>
    <xsd:sequence>
      (......children element
        definition ...)
    </ xsd:sequence>
    <xsd:attribute name = "oid" type =
      "xsd:string" use = "fixed" value
      = "OidString" />
    <xsd:attribute name = "Status"
      type = "xsd:string" use =
      "fixed" value = "StatusText" />
    < xsd:attribute name =
      "Description" type =
      "xsd:string" use = "fixed" Value
      = "DescriptionText" />
    < xsd:attribute name = "Reference"
      type = "xsd:string" use =
      "fixed" Value = "ReferenceText"
      />
  </ xsd:complexType>
</ xsd:element>
```

Module name and group node—The SNMP MIB module defines management information between 'BEGIN' and 'END' with 'DEFINITION' keyword according to the ASN.1 rule.

```
moduleName DEFINITIONS ::= BEGIN
......management information definotion
  ...
END
```

Because an XML document must have one root element enveloping the whole document in XML grammar, the model name of '*moduleName*' is changed into a top-level element (root element). The XML Schema expression follows.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd =
      "Http://www.w3.org/2000/10/
      XMLSchema" elementFormDefault
      = "Qualified">
<xsd:element name = "moduleName">
    <xsd:complexType>
      <xsd:sequence>
        (...management information
          definition ...)
      </ xsd:sequence>
      <xsd:attribute name = "Version"
        type = "xsd:string" use =
        "required" />
    </ xsd:complexType>
      </ xsd:element>
</ xsd:schema>
```

The 'OBJECT IDENTIFIER' node, which is a method to express the group node, has child nodes without interior clauses. This node corresponds to the middle node in a tree structure and may have multiple child nodes. The middle nodes of SNMP MIB do not have values. An example defined as RFC1213 13 follows.

```
Mib-2 OBJECT IDENTIFIER : : = {mgmt 1}
```

This expression means that the mgmt node is the first child node of the mib-2 node. If this

expression is displayed by an XML Schema, the converted result follows.

```
<xsd:element name = "Mgmt">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref = "Mib-2" />
    </ xsd:sequence>
    <xsd:attribute name = "oid" type
      = "xsd:string" use = "fixed"
      value = "1.3.6.1.2" />
  </ xsd:complexType>
</ xsd:element>

<xsd:element name = "Mib-2">
  <xsd:complexType>
    <xsd:sequence>
      (......child element definition
        ...)
    </ xsd:sequence>
    <xsd:attribute name = "oid" type
      = "xsd:string" use = "fixed"
      value = "1.3.6.1.2.1" />
  </ xsd:complexType>
</ xsd:element>
```

To summarize the general conversion method we described thus far, each node of MIB is changed into each relevant element of XML Schema, and the node name is used as the relevant element name, which is the 'model-level mapping' method named by J.P. Martin-Flatin.[4] Further, all clauses of each node are changed into attributes of the relevant element of XML Schema definition. Finally, all elements have the 'oid' attribute which marks the OID value of the relevant MIB node.

—Implementation of Translator—

The translator translates the SNMP MIB definition into an XML Schema definition automatically without loss of information about managed objects. Input is allowed from a network URL or a local file. The output of this translator is the XML Schema file and the XML DOM tree structure. The XML Schema file is used for validation of each instance of the XML document, and DOM is used for generating and processing XML documents which contain management data.

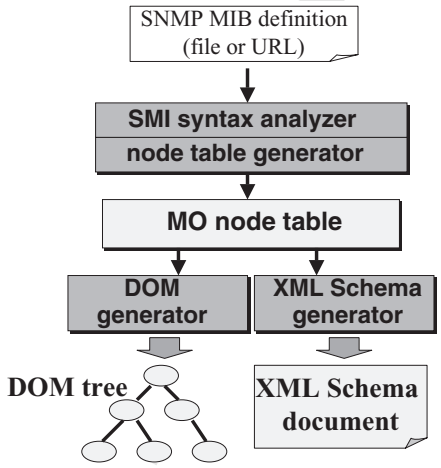

Figure 2. Structure of SNMP MIB to XML translator

The translator can change the DOM tree in runtime, and must have functions to manipulate DOM, such as adding elements or deleting elements etc. The structure of the translator is depicted in Figure 2.

The SMI syntax analyzer reads the SNMP MIB definition, deletes comments, divides by a series of tokens, and examines whether the MIB definition is defined within the rule of SMI and ASN.1. If an error occurs, the translator returns an error code. The node table generator generates a table of managed objects. Each node is mapped into one Java class, and the table contains a series of classes. In case there is a user-defined data type, the type is converted into a Java class, and is stored into a data type table. The XML Schema generator generates an XML Schema definition corresponding to the source MIB definition by using information from the MO table. The XML Schema corresponds to a W3C standard.

The DOM generator creates a DOM tree using management information that is stored in the MO table. The created DOM tree is used as intermediate storage for management information when the manager and agents communicate with each other. For handling the DOM tree, we use standard APIs for XML DOM. We used JAVA programming language to implement the translator and the JDK1.3 package and Xerces[20] which is a JAVA package for processing DOM offered free by the Apache group.

The gateway is necessary for communication between an XML-based manager and an SNMP-based agent in order to translate the two protocols, HTTP and SNMP.

SNMP-XML Gateway

We explain the implementation procedure for the SNMP-XML gateway developed to manage SNMP agents with the XML-based manager directly. Acting as a relay for management data, the gateway translates SNMP messages into XML documents and delivers the XML documents to the XML-based manager, and vice versa. The gateway is necessary for communication between an XML-based manager and an SNMP-based agent in order to translate the two protocols, HTTP and SNMP.

Figure 3 shows an application of the gateway. The gateway relays between HTTP and SNMP.

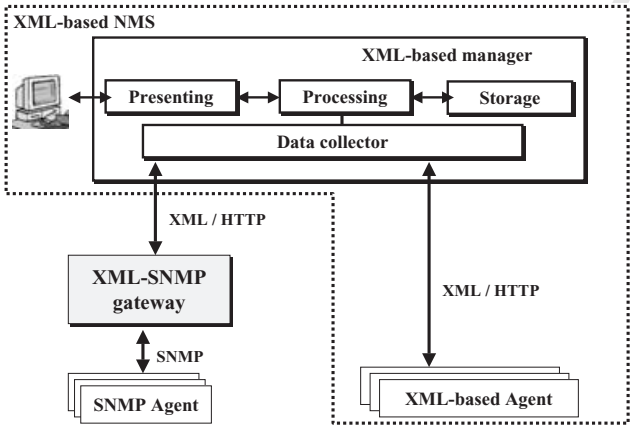

Figure 3. An application of SNMP-XML gateway

—Design—

We must consider both interaction translation and specification translation to define the action of the gateway. We described specification translation in the previous chapter in detail. For interaction translation, we define operation mapping methods here.

Interaction translation stands for converting the expression and transferring methods for actual data. In the SNMP protocol, the operations of SNMP are divided into three types: get, set, and trap. The Get operation is the action where the SNMP manager brings data from the agent. The set operation is the action where the SNMP manager sends data to an agent and changes the corresponding node value of the agent. If a particular event happens in the agent, the trap operation occurs. The trap operation is the action where the SNMP agent automatically reports the event information to the manager. Table 6 shows mapping methods of these SNMP operations between SNMP and HTTP.

When the manager requests information from the agent by HTTP, the HTTP message is composed of 5 parameters of host, operation, community, xpath, value. The 'host' argument displays the address of the SNMP agent. The 'operation' argument means operation type, 'set' for set action, and 'get' for get action. The 'community' argument delivers the name of the community defined in the SNMP agent. The 'path' argument delivers the name of a node which has data which

SNMP action	HTTP action	Note
Get	`http://(gateway_address)/SNMP?host = agent_name &community = community_name&operation = get &path = node_name`	HTTP GET
Set	`Http:// (gateway_address)/SNMP ?Host = agent_name & community = community_name&operation = set & xpath = node name&value = value_string`	HTTP GET / HTTP POST
Trap	`http://(manager-address)/NOTIFICATION/SNMP?H ost = host_name`	HTTP POST

Table 6. Operation mapping between SNMP and HTTP

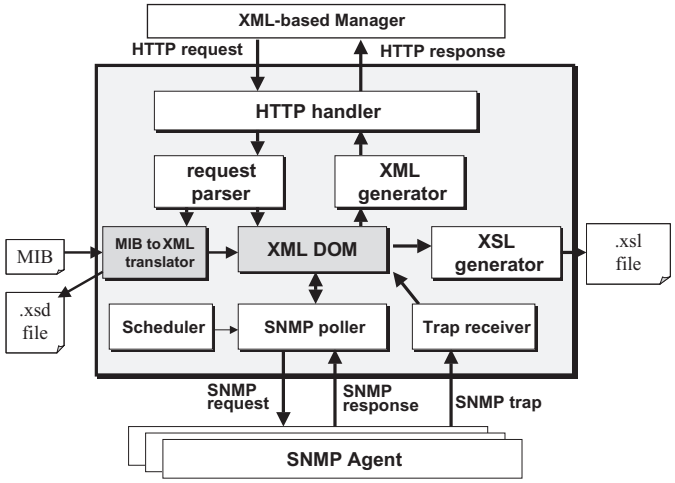

Figure 4. Architecture of SNMP-XML gateway

the mamager wants to know. The 'value' argument stores data that the manager sends to the agent in case of an SNMP set operation. Figure 4 shows the structure of this gateway.

The function of each module of the gateway is as follows.

- HTTP handler is for communication with the XML-based manager. This module receives an HTTP request from the manager and delivers it to the request parser, and delivers the XML document generated by the XML generator module to the manager.
- The request parser analyzes the HTTP request and call functions corresponding to the request.
- XML DOM is data structure for data conversion, which is created by the MIB to XML converter.
- XML generator generates XML documents on the basis of DOM and delivers them to the HTTP handler.
- MIB to XML translator is a specification translator, which generates the XML Schema and the DOM tree from the MIB file used in the SNMP agent.
- SNMP poller is an SNMP polling module. This module sends an SNMP request message to the SNMP agent and updates the DOM tree with received data from the agent.
- A trap receiver receives an SNMP trap message from agents, and updates the DOM with received data.
- The scheduler is for changing schedule for polling periods and target agents.
- The XSL generator creates an XSL file that is a style sheet for user-friendly presentation. This style sheet is used when the gateway is used as an MIB browser.

—Action Scenarios—

The action scenarios of the gateway which translates and delivers messages between an XML-based manager and an SNMP-based agent is divided into two models.

Request/Response model—The Request/Response model is a general model for data teansfer. The manager requests messages from an agent, and the agent sends the requested data to the manager as a respose. The action processed in this way is as follows.

(1) When there is a request from the manager, this is received through the HTTP handler and is passed by the request parser.
(2) Contents of the request are analyzed by a request parser. If the request is 'get' action, the SNMP poller finds out the relevant OID with the DOM and the requested node name, and receives data from the SNMP agent whose address is known by an HTTP request.

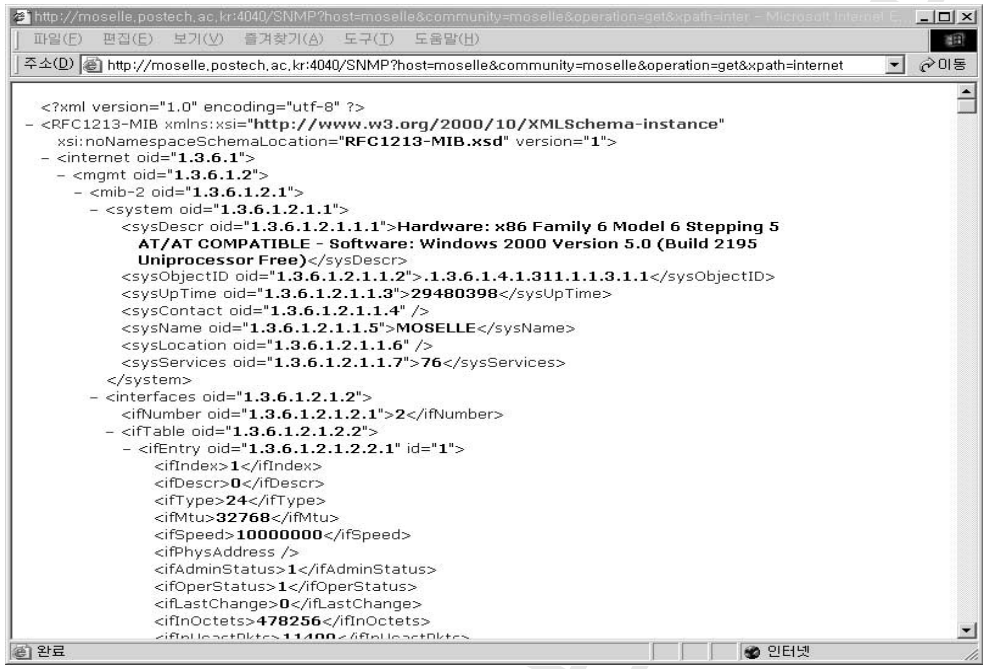

Figure 5. Gateway operation 1

(3) Returned value is stored at the indicated element of the DOM, and converted into an XML document by an XML generator when the DOM is completely updated.
(4) Generated XML document is sent to the manager through an HTTP handler as an HTTP response message.

Notification model—A notification model enables the agent to notify to manager voluntarily when the predeterminate event happens. An action processed in this way is as follows.

(1) If an event happens in the SNMP agent, this is passed to the trap receiver of the gateway as a trap message.
(2) The trap receiver updates the DOM tree which received a trap message which includes OID, value, occurrence time etc.
(3) The XML generator makes out an XML document corresponding to the event delivered from the agent, and delivers the document to the manager through the HTTP handler.

—Implementation of Gateway—

To implement the gateway, we used the JAVA programming language based on JDK1.3. For specification translation, we used the MIB to XML translator, as explained above. We used an openNMS's joeSNMP package 24 that is Java-based and supports SNMPv1 and SNMPv2c, to implement the SNMP Poller module and the Trap receiver module. These two modules include functions for updating DOM. Also, we used an Xerces Java package 20 from the Apache sroup to handle DOM and to generate XML documents. An HTTP handler module communicated with an XML-based manager embodied as a simple Web-server by using functions offered within JDK1.3. For changing DOM when the agent is changed, a request parser calls functions for changing DOM in run-time.

—Examples of Gateway Operation—

Figure 5 shows the XML document received from the gateway as a response to an HTTP

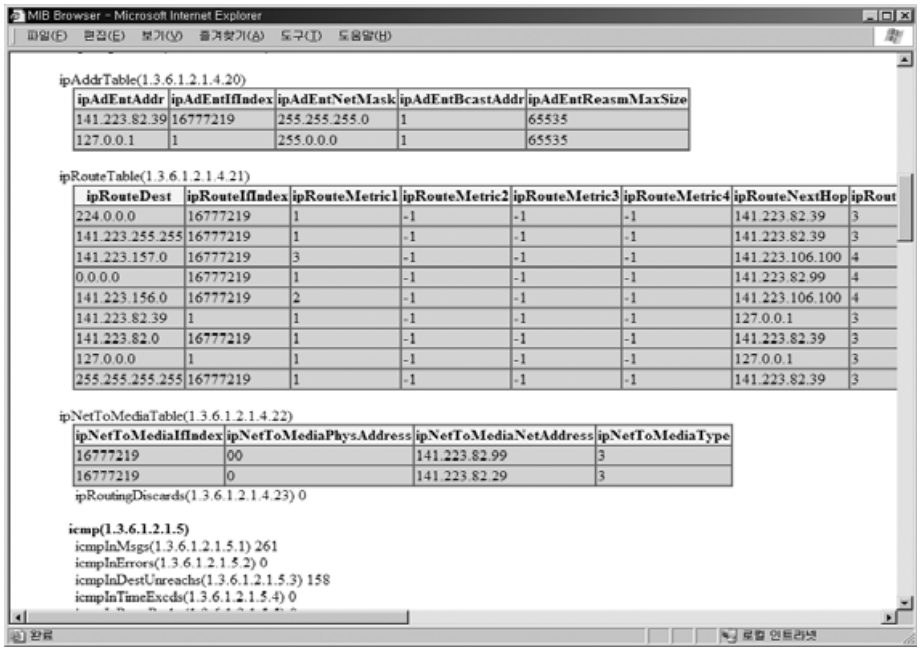

Figure 6. Gateway operation 2

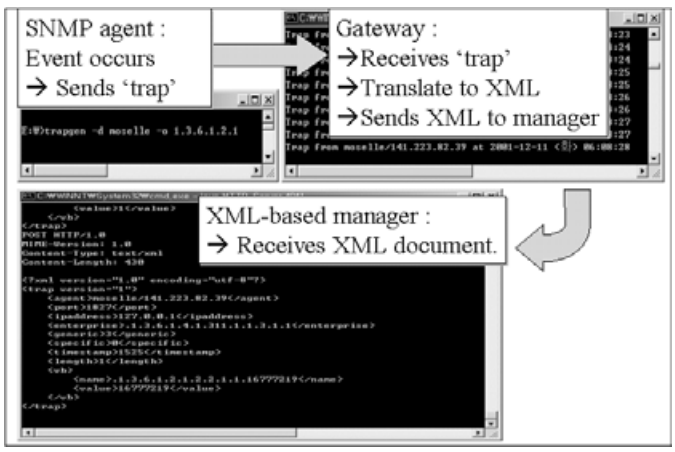

Figure 7. Transferring notification message

request sent by the Web-browser directly. The document includes all nodes of MIB II.

We also developed an XSL generator for user-friendly presentation of XML documents. Figure 6 shows an output of the gateway when the style sheet is adapted. Using this function, the gateway can be used as an MIB Browser. We can see any management data of any SNMP agent with only a common Web-Browser.

Figure 7 shows the process of transferring an SNMP trap message as a notification operation. A trap message generated by an agent is delivered to

the gateway. The gateway translates the trap message into an XML document, and delivers it to the manager. The XML-based manager receives the XML document.

Conclusion and Future Work

We developed a gateway which translates messages between SNMP and XML/HTTP. For this gateway, we proposed a translation algorithm which changes SNMP MIB into the XML Schema as a method of specification translation, and inplemented an MIB to XML translator which embodied the algorithm. Also, we defined the operation translation methods for interaction translation.

SNMP has limits in scalability and efficiency when managing increasingly large networks. Research on XML-based NMS is evolving to solve these shortcoming of SNMP-based NMS. XML-based NMS uses XML in network management to pass management data produced in large networks. XML-based NMS delivers management data in the form of an XML document over the HTTP protocol. This method is efficient for transferring large amounts of data. However, an XML-based NMS cannot manage the legacy SNMP agent directly. If a manager cannot communicate with an SNMP agent, it is not practical in the real world where SNMP is used worldwide. Because most Internet devices are equipped with an SNMP agent, and network namagement was performed by the agent, we studied how to manage the legacy SNMP agent using the advantage of XML-based network management simultaneously.

Because of the excellent compatibility and user-friendly features of XML, integration of data into XML is expected to accelerate in the future.

Because of the excellent compatibility and user-friendly features of XML, integration of data into XML is expected to accelerate in the future. Specifically, in order to use XML as middleware for information transmission between different systems, a standard translation method to change SNMP MIB to XML within transmission of information for network and system management is necessary.

In future work, we need to enhance the translation algorithm through a performance evaluation of the algorithm. For enlarging scalability, we need to study how one manager can manage many SNMP agents distributed to large networks such as enterprise networks. Distributed processing is the method presented here. For example, one XML-based manager governing several distributed SNMP-XML gateways through networks can expand the scope of management.

References

1. The World Wide Web Consortium, http://www.w3c.org
2. Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, 3rd edition; Addison-Wesley: Reading, MA, 1999.
3. Ju HT, Choi M-J, Han S, Oh Y, Yoon J-H, Lee H, Hong JW. An embedded web server achitecture for XML-based network management. *Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2002)*; April, 2002, Florence, Italy.
4. Martin-Flatin JP. *Web-Based Management of IP Networks and Systems*; Ph.D. Thesis, Swiss Federal Institute of Technology, Lausanne (EPFL), Oct 2000.
5. Kim YD *et al*. Web-based network management using translation SNMP SMI to XML, *Proc. of APNOMS 2001 Conferences*; Sydney, Sept., 2001, 292–302.
6. W3C, Extensible Markup Language (XML) 1.0, W3C Recommendation, October 2000, http://www.w3.org/TR/REC-xml
7. W3C, HTML 4.0 Specifications. Internet Draft, HTML Working Group, April 1998.
8. International Organization for Standardization. ISO 8879: Standard Generalized Markup Language (SGML), 1986.
9. Network Sorcery. Inc., ICMP, Internet Control Message Protocol. http://www.networksorcery.com/enp/protocol/icmp.htm
10. Mazumdar. S. CORBA/SNMP gateway. Bell Labs. http://www.bell-labs.com/project/CorbaSnmp
11. Strauss F. A library to access SMI MIB information. http:// www.ibr.cs.tu-bs.de/projects/libsmi/
12. Imamura, T, Maruyama H. Mapping between ASN.1 and XML. *Applications and the Internet, 2001. Proceedings, IEEE*; 2001, 57–64.
13. McCloghrie K. Rose M. Management information base for network management of TCP/IP-based Internets MIB-II *IETF*, *RFC1213*, March 1991.

14. McCloghrie K *et al*. 'Structure of management information version 2 (SMIv2)'. *IETF*, *RFC2578*, April 1999.
15. Case J *et al*. Management information base for version 2 of the simple network management protocol (SNMPv2). *IETF*, *RFC1907*, January 1996.
16. Case J. 'Introduction to Version 3 of the Internet-standard network management frameworks', *IETF*, *RFC2570*, April 1999.
17. W3C. Extensible Stylesheet Language (XSL) Version 1.0. Recommendation, Oct. 2001. http://www.w3c.org/Style/XSL/
18. W3C. Document Object Model (DOM) Level 2 Core Specifications. Nov. 2000, http://www.w3.org/TR/DOM-Level-2 - Core/
19. Simple API for XML 2.0, 1999. http://www.saxproject.org/
20. Apache XML project. Xerces Java parser. http://xml.apache.org/xerces-j/
21. W3C. XML Schema Part 0 : Primer, Recommendation, May 2001. http://www.w3.org/TR/xmlschema-0/
22. W3C. XML Linking Language (XLink). Version 1.0, W3C Proposed Recommendation, Dec. 2000, http:// www.w3.org/TR/xlink/
23. W3C. XML Pointer Language (XPointer). Version 1.0, W3C Last Call Working Draft, Jan. 2001, http://www.w3.org/TR/xptr
24. OpenNMS. http://www.opennms.org/
25. Ju HT. *Embedded Web Server Architecture for Web-based Element and Network Management*. Ph.D thesis, Pohang University of Science and Technology, Korea, Oct. 2001. ■

