Towards XML-based Network Management for IP Networks

Mi-Jung Choi*, Yun-Jung Oh*, Hong-Taek Ju**, and Won-Ki Hong*
* Dept. of Computer Science and Engineering, POSTECH, Korea
** Dept. of Computer Engineering, Keimyung University, Korea
({mjchoi, bheart, juht, jwkhong}@postech.ac.kr)

Abstract

Recently, XML-based network management, which applies XML related technologies to network management, has been proposed as an alternative to existing network management. The use of XML in network management presents many advantages. However, most existing network devices are already embedded with SNMP agents and managed by SNMP managers. For integrated network management, we present four approaches towards XML-based network management. We also present the architectures of an XML-based manager, an XML-based agent, and a gateway. We explain our experience of developing XML-based Global Element Management System (XGEMS) through an XML/SNMP gateway. Our experience on developing XGEMS can be used as a good guideline of XML-based management system fully taking advantages and strengths of XML technologies.

Keywords: XML-based Network Management, XML, SNMP, XML/SNMP Gateway, DOM, SAX, XML Schema, XPath, XSL, XSLT, XQuery, XUpdate, SOAP.

1. Introduction

The rapid pace of Internet evolution is currently witnessing the emergence of diverse network devices. The current IP networks are much more complex and composed of these network devices. Therefore, efficient network management systems are necessary to manage these networks and network devices. The management of IP networks has exclusively relied on Simple Network Management Protocol (SNMP) [1], since the SNMP was developed in the late 1980s. However, the SNMP management framework has some weaknesses especially related to configuration management and application development process to manage current huge networks [2]. Today, Extensible Markup Language (XML) [3] related technologies seem to promise a good method out of this trouble.

The XML, which is a meta-markup language standardized by World-Wide Web Consortium (W3C) for document exchange in the Web, is widely used for business-to-business integration, data interchange, e-commerce, and creation of application-specific vocabularies. It supports several standards such as XML Schema [4], Document Object Model (DOM) API [5], XML Path Language (XPath) [6], Extensible Stylesheet Language (XSL) [7], etc. Many efforts of applying XML related technologies and their already existing implementations to a wide range of network management are in progress. The use of XML in network management presents many advantages, as

follows [2]:

- The XML Schema can be used to define the structure of management information in a more flexible way.
- Widely deployed protocols such as HTTP [8] can be used to transfer management data reliably.
- The DOM APIs can be used to easily access and manipulate management data from applications.
- XPath expressions can be used to efficiently address the objects within management data documents.
- XSL can be used to process management data easily and generate HTML documents for a variety of user interface views.
- WSDL [9] and SOAP [10] can be used to define Web Services for powerful high-level management operations.

However, these current efforts of applying XML to network management are accomplished in a limited network management area such as configuration management or done in a development process of network management system using very few and simple XML technologies.

In this paper, we present the work on XML-based network management carried out by other research groups [11, 12, 13] and vendors [14, 15]. We present four approaches towards XML-based integrated network management for IP-based networks. We also explain our XML-based network management system called XML-based Global Element Management System (XGEMS) to manage network devices using XML technologies and methods of fully applying XML technologies to network management. Our experience on developing XGEMS can be used as a good guideline for developers who develop XML-based network management systems.

The organization of this paper is as follows. Section 2 describes XML related technologies applied to network management, and XML-based network management research effort by other groups as related work. In Section 3, we explain and compare four approaches towards XML-based integrated network management. In Section 4, we present the design of an XML-based network management system composed of an XML-based manager, an XML-based agent and an XML/SNMP gateway. In Section 5, we describe our implementation details of XGEMS based on the proposed design. Section 5 also discusses the validation of the proposed solution. Finally, we conclude our work and give directions for future work in Section 6.

2. Related Work

In this section, we first explain XML related technologies that are applicable to network management, such as XML Schema [4], XSL [7], DOM [5], XPath [6], XQuery [16], and SOAP [10]. We also describe recent research on XML-based network management performed by other research groups. Finally, we introduce several standard activities and industry effort in this area.

2.1. XML Related Technologies

(1) DTD and XML Schema: XML has two fundamental approaches to define the XML document structure: Document Type Definition (DTD) and XML Schema [4]. The DTD is used to specify a content model for each element. The content description is part of the element declaration in the DTD and specifies the order and quantity of elements that can be contained within the element being declared. That is, the DTD is used to specify a

property for each element as well as the relationship between the elements. However, because the DTD does not support complex information model, another modeling mechanism, the XML Schema, was proposed. The XML Schema substantially revised and extended the capabilities found in XML DTDs. The XML Schema is based on XML, so it can be parsed and manipulated in exactly the same manner as the XML documents through the standard API. The XML Schema supports a variety of data types (44 kinds of basic types), while the DTD treats all data as strings or enumerated strings. The XML Schema also allows inheritance relationships between elements and supports namespace integration.

(2) XSL and XSLT: Extensible Stylesheet Language (XSL) [7] is a mark-up language designed for illustrating the method to display XML documents on the Web. XML documents describe only the contents and the structure of the contents. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary. That is, XSL enables XML to separate contents from presentation. XSL consists of two parts: a language for transforming XML documents, and an XML vocabulary for specifying formatting semantics. The style sheet technology that can transform documents is XSL Transformation (XSLT) [17], which is a subset of XSL technology that fully supports the transformation of an XML document from one format into another, such as HTML or another custom XML document type. The reason for publishing the XSLT specification separately from XSL is that XML documents can be displayed, providing an easy display format for end users by transforming the XML documents without formatting semantics.

(3) DOM and SAX: The Document Object Model (DOM) [5] is a platform- and language-independent interface. It allows programs and scripts to dynamically access and update the content, structure, and style of documents. The DOM is an API for valid HTML and well-formed XML documents. The Simple API for XML (SAX) [18] is an event-driven and serial-access mechanism for accessing XML documents. A DOM parser parses the XML document and creates a DOM tree, keeping the entire tree in memory at the same time. SAX reads the XML document in sequential order and generates an event for a specific element. Therefore, if the application calls for sequential access to XML documents, SAX can be much faster than other methods without requiring much system overhead. Whenever an event is generated, the method relevant to the event is processed. An XML processor using SAX does not create a data structure. Instead, while accessing the XML document, it generates events such as the start of an element and the end of an element. Applications can process operations such as gaining the name and attribute of the element by capturing the event. SAX is an interface to an XML parser rather than an API to one of the data structures that can be built using a parser.

(4) XPath: XPath [6] is a language used to identify particular sections of an XML document. XPath has a compact, non-XML syntax to be used within URIs [19] and XML attribute values. Also, XPath operates on the abstract, logical structure of an XML document. Each node in an XML document is indicated by its position, type, and content using XPath.

(5) XQuery and XUpdate: XQuery [16], a query

language for XML, is designed to be broadly applicable across all types of XML data sources, such as structured and semi-structured documents, relational databases, and object repositories. XQuery uses XPath for path expression. Further, XQuery provides such features as filtering documents, joining across multiple data sources, and grouping the contents. XUpdate [20] is an update language, which provides open and flexible update facilities to insert, update and delete data in XML documents. XUpdate language is expressed as a well-formed XML document, and uses XPath for selecting elements and conditional processing.

(6) SOAP: Simple Object Access Protocol (SOAP) [10] is a lightweight protocol for exchanging information in a distributed environment. It is an XML-based protocol that consists of three parts. The first is an envelope that defines a framework for describing the contents of a message and how to process it. The second is a set of encoding rules for expressing instances of application-defined data types. The last is a convention for representing remote procedure calls and responses. SOAP defines the use of XML and HTTP or SMTP to access services, objects, and servers in a platform- and language-independent manner.

2.2. Research on XML-based Network Management

In this section, we describe related research work on XML-SNMP integration, XML-based management architectures, and XML-based management using Web Services.

2.2.1. XML-SNMP Integration

- **Specification Translation**

– **J.P. Martin-Flatin's Management Information Model**: J.P. Martin-Flatin, proposed SNMP MIB to XML translation models, namely Model-level mapping and Metamodel-level mapping [21]. In Model-level mapping the DTD is specific to a particular SNMP MIB (set of MIB variables), and the XML elements and attributes in the DTD have the same names as the SNMP MIB variables. In Metamodel-level mapping the DTD is generic and identical for all SNMP MIBs.

– **F. Strauss's libsmi**: F. Strauss presented a library to access SMI MIB information, "libsmi" [22], which translates SNMP MIB to other languages, such as JAVA, CORBA, C, XML, etc. This library provides a tool for MIB dump (mibdump), which allows dumping the content of a MIB module into an XML document.

– **Our Work on MIB to XML Translation**: In our previous work on the XML/SNMP Gateway, we developed an SNMP MIB to XML translation algorithm, and also implemented an SNMP MIB to XML translator using this algorithm [23]. For validation of the algorithm, we had implemented an XML-based SNMP MIB browser using this SNMP MIB to XML Translator.

- **XML/SNMP Gateway**

– **F. Strauss's XML/SNMP Gateway**: Recently, F. Strauss implemented an SNMP-to-XML gateway [22] using mibdump. The gateway works as follows. When a MIB module to be dumped is passed to mibdump, an SNMP session is initiated, and then sequences of SNMP GetNext operations are issued to retrieve all objects of the MIB from the agent. Mibdump collects the retrieved data and the contents of these data are dumped in the form of an appropriate XML document with respect to the predefined XML Schema.

– **Avaya Labs Research**: Avaya Labs. is currently developing an XML-based management interface for

SNMP enabled devices [13]. The prototype system consists of three parts:

- A tool for automatic generation of XML Schema definition based on an SNMP SMI information.
- An XML-RPC based messaging protocol for retrieving and modifying MIB information in SNMP enabled devices. The messaging protocol defines XML Schema for a set of query commands (GET, SET, LIST, CREATE, DELETE) and identifies MIB variables using XPath-based identifiers.
- An adapter for retrieving and modifying device information in the form of XML data based on the information in the device's MIB.

They have already implemented a tool for mapping SNMP SMI information modules to XML Schema. This tool is an extension of previously implemented tool for converting SNMP SMI to CORBA-IDL [24]. They are currently implementing the XML document adapter for SNMP MIB modules using net-snmp [25] and XML-RPC libraries [26].

– **Our Work on the gateway**: In our previous work [27, 52], we proposed several interaction translation methods between an XML/SNMP gateway and an XML-based manager, based on the previously developed specification translation algorithm [23]. First, we proposed a DOM-based translation method, which enables the manager to directly access the DOM in the gateway using DOM interfaces in order to exchange management information with the SNMP agent. In the HTTP-based translation, we extended a URI string, which contains information on a request with XPath and XQuery. XPath and XQuery can be easily applied to URIs to express a location path of target objects and to provide a query language in the request message. This method provides efficiency improvement in XML/HTTP communication, which is the most common in the exchange of XML documents. Finally we proposed a SOAP-based translation method. Using SOAP, the gateway provides a flexible and standardized method for interaction with XML-based manager in a distributed environment. We have implemented and validated our XML/SNMP Gateway for our XGEMS. Detailed explanation on this implementation is described in Section 4.3.

2.2.2. Management Architecture

- **WIMA**: J.P. Martin-Flatin presented an idea to use XML for integrated management in his research on Web-based Integrated Network Management Architecture (WIMA) [21]. WIMA provides a way to exchange management information between a manager and an agent through HTTP. HTTP messages are structured with a MIME multipart. Each MIME part can be an XML document, a binary file, BER-encoded SNMP data, etc. By separating the communication and information models, WIMA allows management applications to transfer SNMP, CIM, or other management data. A WIMA-based research prototype, JAva MAnagement Platform (JAMAP) [21], implemented push-based network management using Java technology.
- **XML-based Network Management Architecture**: We proposed an XML-based network management (XNM) using EWS (Embedded Web Server) in our previous research [28]. In this work, we extended the use of EWS for element management to network management. XNM uses XML to transfer management information over HTTP between an agent and a manager. XNM also uses DOM for the representation of and the access to management data.

2.2.3. XML-based Management using Web Services

- **IRTF-NMRG meeting on Web Services for Internet management**: The Network Management Research Group (NMRG) [11] of the Internet Research Task Force (IRTF) is a forum for researchers to discuss and develop new technologies for improving the management of the Internet. Recently, NMRG organized a meeting to investigate the advantages and disadvantages of using Web Services technology for Internet management. In the meeting on Web Services, the participants discussed Web Services technologies, including SOAP, WSDL and UDDI, and compared them with SNMP. They also dealt with security in Web Services. NMRG's work in this area is in early stage and has not produced any substantial result yet.
- **OASIS Management Protocol**: OASIS [29] is a consortium that produces worldwide standards for security, Web Services, XML conformance, business transactions, electronic publishing, etc. The purpose of the OASIS Management Protocol Technical Committee [12] is to develop an XML-based management protocol specification for Web Services to provide a Web-based mechanism to monitor and control managed elements in a distributed environment based on industry accepted management models, methods, and operations, including, Open Model Interface (OMI) [30], XML, SOAP, DMTF CIM, and DMTF CIM Operations. The Management Protocol TC's work in this area is also in early stage but has proposed to deliver a published Management Protocol Specification by June 2003.

2.3. Standard Activities

In this section, we present standardization effort on XML-based network management within the DMTF and the IETF

2.3.1. WBEM

Web-Based Enterprise Management (WBEM) [31] is an initiative of the Distributed Management Task Force (DMTF), and includes a set of technologies that enables interoperable management of an enterprise. WBEM consists of Common Information Model (CIM) [32], a DTD for the representation of the CIM in XML [33] and a specification for CIM operations over HTTP [34]. CIM provides a comprehensive object-oriented information model and the CIM schemas are implemented not only for managing servers but also for network resources such as switches and routers. WBEM is currently being updated to include emerging standards, such as SOAP. DMTF is collaborating with OASIS [29] to sponsor a new management protocol technical committee and to develop open industry standard management protocols.

2.3.2. IETF XML Configuration BOF

In the 54th IETF meeting in July 2002, a BOF session concerned with XML configuration (XMLCONF) was held. This BOF discussed the requirements for a network configuration management, and how the existing XML technologies, namely SOAP, WBEM, SyncML [35] and JUNOScript [14] could be used to meet those requirements. There are some Internet-Drafts [36, 37, 38] that present basic concepts and the requirements for XML network configuration and provide guidelines for the use of XML within IETF standards protocols.

2.4. Industry Effort

In this section, we introduce recent industry

effort for XML-based network management.

2.4.1 Juniper Networks' JUNOScript

Recently, Juniper Networks introduced JUNOScript for their JUNOS network operating system. The JUNOScript is part of their XML-based network management effort and uses a simple model, designed to minimize both the implementation costs and the impact on the managed device. The JUNOScript allows client applications to access operational and configuration data using an XML-RPC. The JUNOScript defines the DTDs for the RPC messages between client applications and JUNOScript servers running on the devices. Client applications can request information by encoding the request with JUNOScript tags in the DTDs and sending it to the JUNOScript server. The JUNOScript server delivers the request to the appropriate software modules within the device, encodes the response with JUNOScript tags, and returns the result to the client application.

2.4.2 Cisco's Configuration Registrar

The Cisco Configuration Registrar [15] is a Web-based system for automatically distributing configuration files to Cisco IOS network devices. The Configuration Registrar works in conjunction with the Cisco Networking Services (CNS) Configuration Agents located at each device. The Configuration Registrar delivers the initial configuration to Cisco devices when starting up on the network for the first time. The Configuration Registrar uses HTTP to communicate with the agent, and transfers configuration data in XML. The Configuration Agent in the device uses its own XML parser to interpret the configuration data from the received configuration files.

3. Approaches for XML-based Integrated Network Management

To investigate carefully towards XML-based integrated network management, we have examined four approaches [41] of manager and agent: SNMP manager and agent, XML-based manager and SNMP agent, SNMP manager and XML-based agent, XML-based manager and agent. We compare these four approaches on the aspect of three management models: organization model, information model, and communication model. Also, we describe the classification of the related work into each approach.

3.1. Four Approaches for Integration

As depicted in Figure 1, four possible combinations between managers and agents can be considered [41]. The first and last combinations are typical in the SNMP and XML-based management framework, respectively. The gateways in the second and third combinations translate messages and operations between different management schemes.

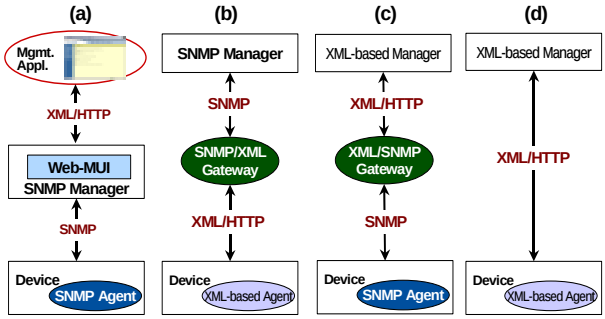


Figure 1. Combinations of Manager and Agent

In the first approach in Figure 1 (a), the XML complements SNMP-based management on the side of management user interface, namely, the presentation part. SNMP does not specifically mention a display mechanism of management information. The existing way is to provide a proprietary and a device specific GUI. The SNMP management information is translated to an XML document, and the XSL transforms the display

format from the XML document to HTML, and the HTML provides the Web-based management user interface (Web-MUI).

The second approach in Figure 1 (b) illustrates that an SNMP manager manages devices equipped with XML-based agents through an SNMP/XML gateway. This type of combination is not practical and difficult to implement, because most of the existing devices are already equipped with SNMP agents, and it is also difficult to translate management information defined in XML Schema into SNMP SMI. Whereas, the third approach of XML-based manager and SNMP agent, shown in Figure 1 (c), is the most possible in real management framework, because this combination provides methods to manage network devices, which are already equipped with legacy SNMP agents in worldwide networks in XML-based integrated management.

The last approach in Figure 1 (d), which replaces the SNMP agent and the SNMP manager with the XML-based agent and the XML-based manager, is the most ideal case to utilize the maximum advantages of XML-based network management. This architecture, however, can be applied to limited situations, which eliminates the interoperability with existing SNMP-based network management systems.

3.2. Comparison of Four Approaches

Table 1 compares four approaches on the aspect of three management models: organization model, information model, and communication model. The organization model of all four approaches is manager and agent paradigm. The second and third approaches need an SNMP/XML gateway and an XML/SNMP gateway, respectively. The gateway must provide both specification translation and interaction translation between the two management applications.

The XML Schema has many advantages in defining management data compared with SNMP SMI and WBEM CIM. The XML Schema is flexible and extensible, allowing new tags to be added without changing the existing document structure [42]. Therefore, the XML Schema can be used as the presentation of management information by adding new data types, and defining the document structure of management information. Moreover, it is easy to learn, and can be also utilized for various purposes, including validation. By using XML authoring tools, developers can easily generate XML documents, a database schema, and other forms of data structure based on the XML Schema. The management information of the XML document format is distributed through HTTP. The operations of HTTP are Get and Post. The HTTP Get operation gathers management information from the device, and the HTTP Post updates the management information. The XML-based manager receives a notification from an XML/SNMP gateway or an XML-based agent by a Push

Table 1. Comparison of Four Approaches

Features	**SNMP Manager & Agent**	**SNMP Manager & XML-based Agent**	**XML-based Manager & SNMP Agent**	**XML-based Manager & Agent**
Organization Model	Manager-Agent	Manager-Agent (SNMP/XML Gateway)	Manager-Agent (XML/SNMP Gateway)	Manager-Agent
Information Model	SNMP SMI	XML Schema (Specification translation)	SNMP SMI (Specification translation)	XML Schema
Communication Model	SNMP, Get, Set, Trap, MIT with OID	SNMP & HTTP (Interaction translation)	HTTP & SNMP (Interaction translation)	HTTP, Get, Post, XPath expression

mechanism [21] using a Post operation. As the XML-based network management performs the basic management functionality through the HTTP operations Get and Post, it is not necessary to develop a new management protocol.

XPath is used for addressing managed objects in XML-based applications. The syntax used by XPath is designed for use in URIs [19] and XML attribute values. The main premise behind XPath is the traversal of an XML document to arrive at a specified node. Therefore, a manager can query effectively managed objects of an agent using XPath. In a Get operation, either a single value can be retrieved by parameters of XPath, or multiple values can be simultaneously retrieved. Also, we can retrieve specific information with conditioning and filtering by XPath expression.

The research work of 'libsmi' of F. Strauss [22], Avaya Labs [13], and our gateway [23, 27, 52] all focuses on the third approach for the integrated management of SNMP-based network management. Industry effort of Juniper [14] and Cisco [15] is the fourth approach of XML-based manager and agent, where an XML-based manager manages their own network devices with a proprietary XML-based agent involving the capability of processing XML documents for configuration management.

4. Design of XML-based Management System

In Section 3, we investigated four approaches towards XML-based network management. To replace SNMP-based NMS with XML-based NMS, an XML-based manager, an XML-based agent, and a gateway are all necessary. In this section, we present detailed architectures of XML-based manager, an XML-based agent, and an XML/SNMP gateway.

4.1. XML-based Manager

Figure 2 illustrates the architecture of an XML-based manager. A *Web Server* is used to provide administrators with a Web-MUI and for receiving requests from the *Management Application* and passing them to the management components through the *Management Script*. Also, the *Web Server* is used for receiving asynchronous messages for notifications from the devices over HTTP. The *HTTP Client* plays a role in the interface module of device and exchanges synchronous management information with the agent. The *DB* is used to store management information for long-term analysis. The *XSL Template Repository* stores XSL files for generating HTML documents from XML documents. The management components such as the *Device Configuration Manager*, *Analyzer*, etc., use the *DOM Interface* to implement the management application functions because management information is represented in XML data. These functions include filtering, logging, and collecting data from multiple agents.

The basic components processing management functionalities are *Management Server Manager*, *Monitoring Manager, Device Configuration Manager*, *Analyzer*, *Notification Handler*, *Logging Manager, Presenter and Event Reporter*. The *Management Server Manage*r manages the configuration parameters for management processing environment, and handles the topologies of multi-level device/device group tree architecture. Also, this component manages administrator account list. The *Device Configuration Manager* is a module to get and set the configuration of a managed device, the *Monitoring Manager* is a module to obtain device monitoring information such as device status and in/out traffic, the *Logging Manager* logs the necessary data and analyzes the log data stored in DB per the administrator's request.

The *Notification Handler* receives notifications from the managed devices and stores the notifications to the DB tables and sends a meaningful notification to the *Event Reporter*. The *Event Reporter* generates appropriate events and sends them to administrators by an email or a pager, etc. The *Analyzer* is a module to analyze the collected management information. The *Presenter* processes the XML document with XSLT and generates HTML document for Web-MUI.

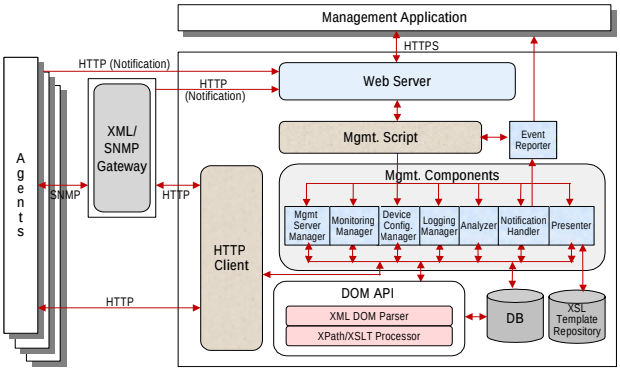


Figure 2. Architecture of XML-based Manager

There are three typical information flows within the XML-based manager in Figure 2. The first data flow is the management request from the *Management Application* to the *Web Server*, and the *Web Server* calls the *Management Script* and selects the proper management module. If the management function is to monitor a device, the procedure is as follows. The *Management Script* calls the *Monitoring Manager*, and then the *Monitoring Manager* sends the request to an agent through the *HTTP Client*, then the result returns to the *Management Application* in reverse order. If the result needs to be stored for later analysis, the *Monitoring Manager* stores them in the *DB*. This flow is same as the traditional SNMP Get operation. The flow of a Set method is same as the SNMP Get method. The agent can be an XML-based agent or an SNMP agent.

Secondly, when the agent sends a notification to the administrator, the information travels in the following order. The agent sends an alert message to the *Web Server* of XML-based manager through HTTP. Then, the *Web Server* receives the notification, and calls the *Notification Handler* through *Management Script*. The *Notification Handler* sends the specific event to the *Event Reporter* for generating the appropriate event and stores the notification for later analysis to *DB*. This flow is same as the SNMP Trap operation.

The last data flow from the *Management Application*, through the *Web Server*, the *Management Script*, and the *DOM Interface*, to the *DB* is used to generate a long-term analysis report. For example, the *Web Server* first calls the *Analyzer* through the *Management Script*, then the *Analyzer* searches for data from the *DB* using the *DOM Interface*. After processing the data by filtering, sorting, and correlating, it finally deduces the result. The analysis result is then sent to the *Management Application*.

4.2. XML-based Agent

Figure 3 illustrates the architecture of an XML-based agent. The XML-based agent includes an *Embedded Web Server (EWS)* [43] as a basic component. The components added to the *EWS* are the *XML Processor* having *SAX Parser*, the *Push Scheduler*, and the *HTTP Client Engine*.

In our previous work [28], the device embedded with XML-based agent, which uses DOM [5] and XPath [6] for handling the XML document, was Linux server. So there was no problem with the resource. However, supporting DOM and XPath needs a lot of memory resources in the device. To access a part of an XML document, the DOM tree of the whole XML document is loaded into the memory. This wastes the resources in the device. The code size and executable memory size of SAX [17] is smaller than DOM. As SAX is an event-driven mechanism for accessing and processing the XML document, there is no need to load the whole XML tree

to the memory. Therefore, SAX is much lighter than DOM from the perspective of resources. However, the access method of SAX is serial and read-only, so we add a *Write Module* as part of XML processor for providing a writing mechanism. Because we set more value on low resource requirements, we selected the SAX.

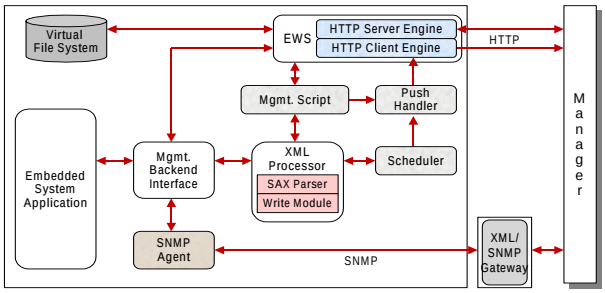


Figure 3. Architecture of XML-based Agent

The *SAX Parser* parses the XML document, and selects the specified node when parsing, and reads management data. In order to send up-to-date information, the agent gathers information from the *Management Backend Interface*. SAX has no writing functionality, so the *Write Module* updates the node value for the selected node through the *Management Backend Interface* before replying to the manager. The *Scheduler* manages subscription information and the schedule for the distribution of the management information and the *Push Handler* receives the request from the *Scheduler* and sends the scheduled data to the manager through the *HTTP Client* at the scheduled time. Also, the *Push Handler* sends a notification generated in the agent, which is sent to the manager through the *HTTP Client*.

If an *SNMP Agent* is also available in the managed network device, the same *Management Backend Interface* can be used. The XML-based manager communicates with the *SNMP Agent* through the XML/SNMP gateway.

4.3. XML/SNMP Gateway

Figure 4 illustrates the architecture of an XML/SNMP gateway, and the HTTP request translation into SNMP requests on the basis of XPath/XUpdate expression in the request message. The *Request Handler* in the gateway receives and parses XPath/XUpdate expression, and delivers the request to the gateway application. *XPath/XUpdate Handler* in the gateway application analyzes the expression, and transfers a list of target nodes addressed by the XPath expression, or the result for the XUpdate request.

Whenever the *Trap Receiver* receives a notification message from the SNMP Agent, it invokes the DOM event for the Trap nodes in the DOM tree. For notification delivery, the *HTTP Client* in the gateway sends an asynchronous Post message defined in the XML Trap Schema to the XML-based manager. The XML-based manager interacts with the gateway through the XML message over HTTP. The interaction translation and the architecture of an SNMP/XML gateway are almost the same with the reverse of those of the XML/SNMP gateway.

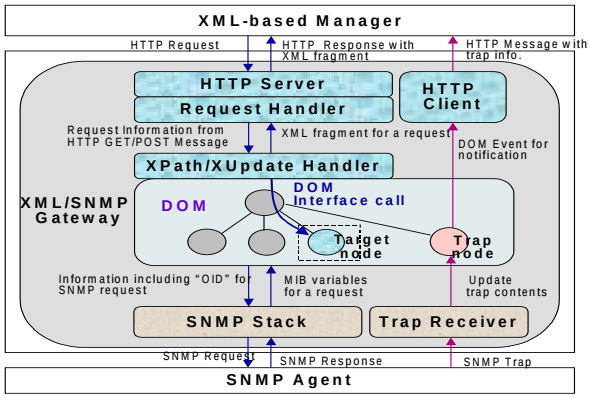


Figure 4. Architecture of XML/SNMP Gateway

5. Implementation and Experience

We have implemented an XML-based Global Element Management System (XGEMS) as a validation of our XML-based integrated management system. XGEMS is used to manage one or more types of network elements scattered possibly throughout the world. In this section, we explain our experience of developing the XGEMS using XML technologies. We have used the

XML/SNMP gateway [23, 27, 52] approach for managing existing SNMP agents. The reasons why the gateway approach is necessary are as follows:

- The configuration management in industry is already changing to XML-based management. Therefore, the XGEMS to process XML is also necessary for integrated management.
- Management systems need to support managing network devices now embedded with SNMP agents and utilizing maximally existing SNMP agents. Moreover, fault management and performance management is already well performed by SNMP agents. However, the SNMP agent reveals its limitation in the configuration management with the SNMP Set operation. Juniper and Cisco suggest the solution of efficient and atomic transfer of configuration data by applying XML related technologies.
- SNMP agent is already embedded and manages most network devices. It is almost impossible to change the agents in already deployed devices. The latest requests on the network management incite developers to develop a new management system without modifying the existing agent. XML technologies provide many advantages in network management mentioned in Section 1 and we can develop new management systems using XML technologies. Therefore, the gateway approach needs for XML-based manager to manage existing SNMP agents.

5.1. Implementation Details

We use the XML Schema to define the management information. Figure 5, drawn using an editing tool of XML Spy [53], shows the management information of XGEMS defined as an XML Schema format. The XML Schema presented in dotted lines in Figure 5 means optional management information. The management information of XGEMS is divided into two parts: a server part and a device part. The *Server* part consists of server configuration, administrator lists, XML/SNMP gateway lists, and device topology information.

The *Device* part consists of device information, device configuration, logging, and monitoring. The XML Schema has complex types containing elements and attributes, and simple types containing none. The XML Schema supports 44 kinds of built-in simple types including string, integer, float, time, date, etc.

DeviceInfoType is defined as complex types and the type of *DeviceInfo* is set to *DeviceInfoType*. Each element such as DeviceIP, AdminID, AgentType, etc. has its own data type.

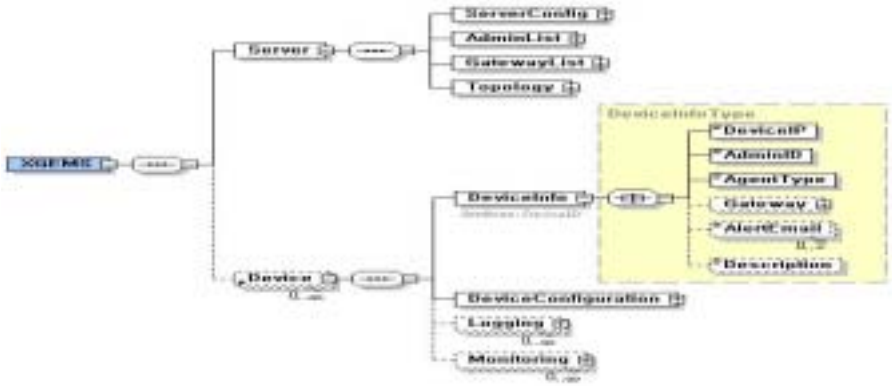


Figure 5. XML Schema of XGEMS

And we add at least an attribute to device information as a primary key for searching. *DeviceInfoType* has an attribute of string type, *DeviceID*, as a unique key.

The *DeviceConfiguration* section is defined as the XML Schema converted from MIB definition of each device for consistency when interacting the XML/SNMP gateway with XGEMS. We translate MIB II to XML Schema for basic management information. If a new device embedding an SNMP agent is added to XGEMS, our MIB2XML translator [23] converts the private MIB definition to XML Schema and the translated XML Schema is added to XML Schema of XGEMS as a child element of *DeviceConfiguration* in Figure 5.

We considered three methods for efficient interaction between XML-based manager, XGEMS, and the XML/SNMP gateway. We adopted the HTTP-based translation among the interaction approaches [27, 52] between the XGEMS and the XML/SNMP gateway. This method enables one to easily define complex request messages and improve communication efficiency over HTTP, which is the most common approach to exchange XML documents.

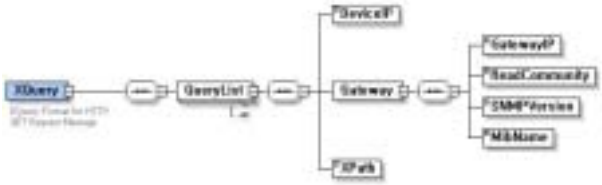


Figure 6. XML Schema for "XQuery" in HTTP Get

In the HTTP communication between XGEMS and the gateway, we define an XML Schema for an HTTP Get/Post message. XGEMS requests management information using the HTTP Get request message. An HTTP Get request has a parameter named "XQuery" to describe the detailed request. Figure 6 shows the XML Schema for "XQuery" expression and Table 2 shows its example. The XML Schema includes definitions of elements, which are "*DeviceIP*" for device identification, "XPath" for addressing portions of management information, and several elements for SNMP communications.

Table 2. XML Example for "XQuery" in HTTP Get

http://XGEMS/monitoring.jsp?XQuery=<XQuery><QueryList>
<DeviceIP >141.223.82.122</DeviceIP>
<Gateway><GatewayIP>141.223.82.56</GatewayIP>
<ReadCommunity>pubic</ReadCommunity>
<SNMPVersion>0</SNMPVersion>
<MibName>RFC1213-MIB</MibName></Gateway>
<XPath>//interfaces</XPath></QueryList><XQuery>

The XGEMS sends HTTP Post request to insert, delete and update management information. An HTTP Post message contains request details in its message body. We use XUpdate [20] expression for HTTP Post message content. An update in the XUpdate language is expressed as a well-formed XML document. XUpdate uses the expression language defined by XPath. These XPath expressions are used in XUpdate for selecting nodes for processing afterwards. An update is represented by a "Modifications" element in an XML document. An asynchronous trap message from an SNMP agent is sent through the HTTP Post message via the XML/SNMP Gateway. The gateway generates an XML document, which contains the trap message, and delivers it to the manager using HTTP POST message.

To store management data for later analysis, a DB is necessary. There are two practical database formats for storing and retrieving XML content: native XML databases and relational databases (RDBMS) [44]. RDBMS does not support up-to-date XML technologies such as XPath and XUpdate. Therefore, an approach is required to map the relational DB or move it to an XML document and schema. Although many RDBMSs support to store XML document, a mapping of XML documents to relational models is not only difficult but often results in incompatible schema. Moreover, mapping the structure of the XML document to a relational schema can heavily degrade performance because it always

needs to parse the XML document.

The native XML DB stores the data structured as XML without the need to translate the data to a relational or object database structure. This is especially valuable for complex and hierarchical XML structures that would be difficult or impossible to map to a more structured database. Therefore, we use a native XML database instead of a traditional RDBMS.

First, we need to know the definition of collection and document in the native XML DB. The collection is the container in which the XML document is stored. The document is an intact XML document used in a collection. Compared to a relational database, the collection is roughly equivalent to a table and the document is same as a row in the table. Any XML documents can be added to a collection regardless of schema. We make collections in accordance with the XML Schema in Figure 5. We make *XGEMS* collection, the collection of *XGEMS* contains *Device* collection, and *Device* collection contains *DeviceInfo* collection in hierarchical order shown in Table 3.

Table 3. XML Document of *DeviceInfo*

XGEMS/Device/DeviceInfo

```
<?xml version="1.0"?>
<DeviceInfoList DeviceID=”device1” >
      <DeviceIP>141.223.82.56</DeviceIP>
      <AdminID>mjchoi</AdminID>
        <AlertEmail>mjchoi@postech.ac.kr</AlertEmail>
      …
        <AlertEmail>siwa@postech.ac.kr</AlertEmail>
      <AgentType>1 (SNMP agent)</AgentType>
      <Gateway >
         <GatewayIP>141.223.82.77</GatewayIP>
         <ReadCommunity>public</ReadCommunity>
         <WriteCommunity>private</WriteCommunity>
         <MIBName>RFC1213-MIB</MIBName>
      </Gateway>
      <Description>Linux Machine</Description>
</DeviceInfoList>
```

We can query *DeviceInfoList* using the same pattern of XPath in the native XML DB. We divide collections in minimum size related to the same information for fast DB operation. The native DB provides the unique key of each document in the collection, which makes it easier to access directly to the specified document. Therefore, we can easily retrieve, update, and delete documents using an XPath expression. DB Schema is consistent with XML Schema in native XML DB, and storing management information to DB is simple and there is no overhead to parse the XML document. Therefore, we can easily process XML data using native XML DB and result in fast and easy development of XGEMS.

The DOM [5] is the means of accessing and manipulating XML documents. The DOM supports reconstruction of XML documents, access to any part of the documents, and the method of manipulations, additions, and deletions to the document. We can analyze management data of XML document format using the DOM Interface. We use the fundamental interfaces of the DOM Core interface [5]: *Node*, *Document*, *DOMImplementaion*, *NodeList*, *NamedNodeMap*, *Attr*, and *Element*.

To access a part of an XML document, the DOM tree of the whole XML document is loaded in memory. This wastes memory and CPU resources and takes a lot of processing time. Therefore, we make an effort to save resources. We use *DocumentFragment* interface for updating XML document. This interface provides a method to update a small portion of the document without constantly updating the *NodeList*s and *NamedNodeMap*s associated with the entire document. Updating the *NodeList*s can significantly slow down the execution. As mentioned before, we use native XML DB to store management information. This reduces the manipulation of XML document using the DOM Interface for inserting management information to the DB, because the XML document can be directly inserted into the native XML DB. When an analysis is requested, we extract the data from the DB by filtering and scoping using the DOM Interface and calculating the data.

After the management information is analyzed, the

analysis result is presented to administrators. We adopted XSLT [17] to accomplish XGEMS presentation. XSLT is an XML-based language that enables us to transform one class of XML document to another. An XML document can be transformed so it can be rendered on a variety of formats fitting different display requirements.

We classify types of presentation in XGEMS. Static information such as XGEMS server configuration data, which is not specific to managed devices, can be rendered using pre-defined XSLT. Another type of presentation is generating a dynamic web page for device configuration, which has various items and styles to present according to devices and their MIBs.

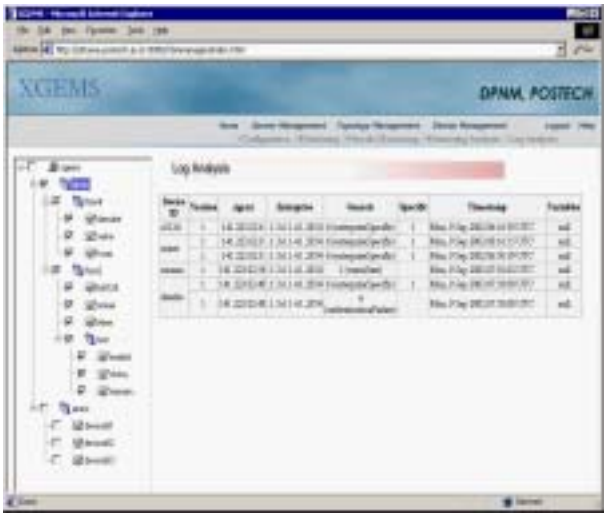
XGEMS maintains XML Schemas for the managed devices, and also XSL templates for the XML documents conforming the XML Schema. The XSLT template for each MIB is generated by XSLT Generator of the XML/SNMP Gateway, and downloaded to XGEMS whenever the MIB is translated. Table objects defined in the MIB is presented as HTML [45] table view using the template. XGEMS reduces the in-line code to control presentation logic and HTML code for work iterating whenever a device is added or an MIB module is recompiled.

5.2. Validation

XGEMS has been implemented on Linux OS using Java language. The XML packages used in XGEMS are mostly from the Apache project [46]. We used the Apache Web server and OpenSSL [47] to communicate in secure mode of HTTPS. We used Xerces [48] as a DOM parser, Xalan [49] as an XPath/XSLT processor. We used Xindice [50], native XML DB, to store the XML document of management data. These all belong to the Apache project and are Java-based. We used Innovation's HTTP Client V0.3-3 [51] as HTTP Client, which is also Java-based. More detailed implementation of the XML/SNMP gateway is found in our previous papers [23, 27, 52].

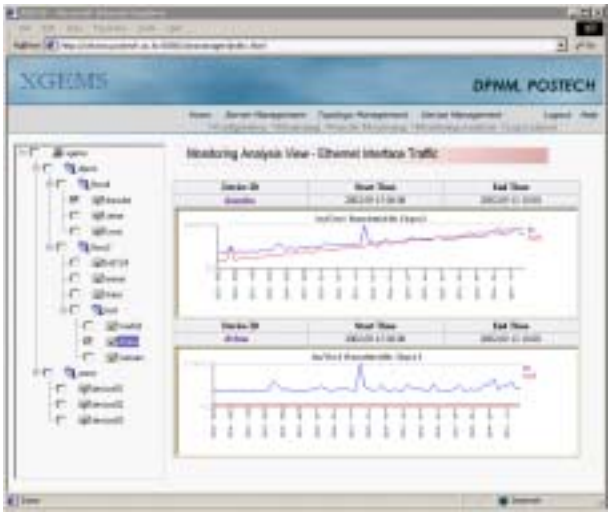
Using XGEMS, an administrator has a unified management interface for multiple devices. Figure 7 shows examples of XGEMS user interface. Figure 7 (a) shows the information of system group of MIB II. First, you check the device ID in left tree and click the *Configuration* submenu of *Device Management* menu in upper frame in (a) figure. The flow of management information of (a) is the first flow (Get/Set flow) mentioned in Section 4.1. Figure 7 (b) shows trap information received from all devices. The flow of this information is the second flow (Trap flow) mentioned in Section 4.1. Figure 7 (c) shows the current setting of periodic monitoring. The two devices are in progress in monitoring, and the others finished the periodic monitoring. If you check the device check-button in tree panel and click "Apply" button in Figure 7 (c), Figure 7 (d) is shown. The Figure 7 (d) shows the monitoring analysis result of selected devices. The graph shows the in/out bandwidth of Ethernet interface during a specified time period. The flow of management information of Figure 7 (d) is the third flow mentioned in Section 4.1.

(a) Get/Set (MIBII – System Group)



(b) Log Analysis from Trap Notification

(c) Configuration of Periodic Monitoring



(d) Result of Monitoring Analysis

Figure 7. Examples of XGEMS User Interface

The XGEMS solves the scalability problem in the manager's processing capacity because the overhead for processing management functionality is distributed to the XML/SNMP gateway. If the number of agents to be managed increases, we can increase the number of gateways. The architecture of the management system is hierarchical and the gateway acts a management system and the XGEMS acts a manager of managers. Because all communications between the XGEMS and the SNMP agents are through the gateway, the gateway must process the translation efficiently, so it must not delay the message transmission between the manager and agent.

The management functionalities of XGEMS can be easily and quickly developed from the support of the standard API and the database. We were able to easily develop analysis management functionalities using standard DOM Interfaces. DOM Interface results in fast and easy development of XGEMS, and saves development time and cost. Building a user interface composed of a number of HTML pages is repetitive and time-consuming work in an application development. We reduced a significant amount of designing Web-MUI using reusable XSLT and generated XSLT template from an MIB definition. We used freely-available libraries processing XML.

6. Concluding Remarks

In this paper, we presented a survey of work done in the area of XML-based network management. We also suggested four approaches towards XML-based integrated network management and explained our gateway approach for managing the existing legacy SNMP agent using the advantages of XML technologies. For validation, we designed and implemented XGEMS for managing network devices based on the proposed manager and gateway architecture. Our XGEMS fully utilizes the XML technologies such as XML Schema, DOM, XPath, XQuery, and XSLT to network management. We were able to reduce the development

cost of the management system through the support of standard API for processing XML document.

Our future work is to test the performance of the XGEMS and the XML/SNMP gateway, then, validate the scalability and extensibility of our systems. We are also working on the development of XML-based agent for achieving a pure XML-based network management. Finally, we plan to extend management operations to Web Services by using WSDL [9] and SOAP [10].

References

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

[2] F. Straus, and T. Klie, "Towards XML Oriented Internet Management", Submitted to the 2003 IFIP/IEEE International Symposium on Integrated Network Management (IM 2003), Cobrado Springs, Mar. 2003.

[3] Tim Bray, Jean Paoli and C. M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", W3 Recommendation REC-xml-19980210, Feb. 1998.

[4] W3C, "XML Schema Part 0,1,2", W3 Consortium Recommendation, May 2001.

[5] W3C, "Document Object Model (DOM) Level 1 Specification", W3C Recommendation, Oct. 1998.

[6] W3C, "XML Path Language (XPath) Version 2.0", W3C Working Draft, Apr. 2002.

[7] W3C, "Extensible Stylesheet Language (XSL) Version 1.0", W3C Recommendation, Nov. 2000.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, IETF HTTP WG, Jun. 1999.

[9] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI", IEEE Internet Computing , Vol. 6, No. 2, Mar.-Apr. 2002, pp.86 –93.

[10] W3C, "SOAP Version 1.2 Part 0: Primer", W3C Working Draft, Dec. 2001.

[11] Network Management Research Group, http://www.ibr.cs.tu-bs.de/projects/nmrg/.

[12] OASIS Management Protocol TC, http://www.oasis-open.org/.

[13] Avaya Labs., XML based Management Interface for SNMP Enabled Devices, http://www.research. avayalabs.com/user/mazum/Projects/XML/.

[14] P. Shafer and R. Enns, JUNOScript: An XML-based Network Management API, http://www.ietf.org/internet-drafts/draft-shafer-js-xml-api-00.txt, Aug. 27, 2002.

[15] Cisco Systems, Cisco Configuration Registrar, http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/ie2100/cnfg_reg/index.htm.

[16] W3C, "XQuery 1.0: An XML Query Language", W3C Working Draft Ap. 2002.

[17] W3C, "XSL Transformations Version 1.0", W3C Recommendation, Nov. 1999.

[18] W3C, "Simple API for XML Version 2.0", WC3 Recommendation, Nov. 1999.

[19] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax". IETF RFC 2396, Aug. 1998.

[20] XML:DB, "XUpdate", Working Draft - 2000-09-14, http://www.xmldb.org/xupdate/xupdate-wd.html.

[21] J.P. Martin-Flatin. "Web-Based Management of IP Networks and Systems", Ph.D. Thesis, Swiss Federal Institute of Technology, Lausanne (EPFL), Oct. 2000.

[22] Frank Strauss, "A Library to Access SMI MIB Information", http://www.ibr.cs.tu-bs.de/projects/libsmi/.

[23] J. H. Yoon, H. T. Ju, and J. W. Hong, "Development of SNMP-XML Gateway for XML-based Integrated Network Management", Accepted to appear in the International Journal of Network Management (IJNM), Wiley, 2002.

[24] S. Mazumdar, CORBA/SNMP Gateway, http://www1.bell-labs.com/project/CorbaSnmp/.

[25] NET-SNMP, http://net-snmp.sourceforge.net/.

[26] First Peer, Inc., XML-RPC for C and C++, http://xmlrpc-c.sourceforge.net/.

[27] Y. J. Oh, H. T. Ju, M. J. Choi, J. W. Hong, "Interaction Translation Methods for XML/SNMP Gateway", Accepted to be presented at DSOM 2002, Montreal Canada, Oct. 2002.

[28] H. T. Ju, M. J. Choi, S. H. Han, Y. J. Oh, J. H. Yoon, H. J. Lee, and J. W. Hong, "An Embedded Web Server Architecture for XML-based Network Management", In Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Florence, Italy, Apr. 2002, pp.1~14.

[29] Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/.

[30] WebMethods, Inc. and Hewlett-Packard Company, Open Management Interface Specification 1.0, http://www.oasis-open.org/committees/mgmtprotocol/Docs/OMISpecification_1.0rev1_OASIS.pdf.

[31] WBEM, "WBEM Initiative", http://www.dmtf.org/wbem/.

[32] DMTF, "Common Information Model (CIM)", http://www.dmtf.org/standards/standard_cim.php.

[33] DMTF, “Specification for the Representation of CIM in XML Version 2.0”, DMTF Specification, Jul. 1999.
[34] DMTF, “Specification for CIM Operations over HTTP Version 1.0”, DMTF Specification, Aug. 1999.
[35] SyncML Initiative, http://www.syncml.org/.
[36] M. Wasserman, Concepts and Requirements for XML Network Configuration, Internet-Draft, http://www.ietf.org/internet-drafts/draft-wasserman-xmlconf-req-00.txt, Jun. 2002.
[37] T. Goddard, Towards XML Based Management and Configuration, http://www.ietf.org/internet-drafts/draft-goddard-xmlconf-survey-00.txt, Jun. 2000.
[38] S. Hollenbeck, et. al, Guidelines for the Use of XML within IETF Protocols, http://www.ietf.org/internet-drafts/draft-hollenbeck-ietf-xml-guidelines-06.txt, Aug. 2002.
[39] SSH Communications Security, SSH Secure Shell, http://www.ssh.com/products/ssh/.
[40] Alan O. Freier, et al, “The SSL Protocol Version 3.0”, Internet-Draft, Nov. 1996.
[41] M. J. Choi, H. T. Ju and J. W. Hong, “Towards XML and SNMP Integrated Network Management”, Proc. of the Asia-Pacific Network Operations and Managemet Symposium, Jeju Korea, Sep. 2002, pp. 507-508.
[42] DMTF, “XML As a Representation for Management Information - A White Paper. Version 1.0”, September 1998, http://www.dmtf.org/spec/xmlw.html/.
[43] M.J. Choi, H.T. Ju, H.J. Cha, S.H. Kim, and J.W.K. Hong, “An Efficient and Lightweight Embedded Web Server for Web-based Network Element Management”, In Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2000), Hawaii, USA, Apr. 2000, pp. 187~200.
[44] Bouret, Ronald, “ XML and Databases”, Sep. 1999, http://www.rpbourret.com/xml/XMLAndDatabases.htm/.
[45] D. Raggett, A. Le Hors, I. Jacobs, “HTML 4.01 Specification”, IETF HTML WG, http://www.w3.org/TR/html401, Dec. 1999.
[46] Apache XML Project, http://xml.apache.org/.
[47] Apache-SSL, http://www.apache-ssl.org/.
[48] Apache XML project, “Xerces Java parser”, http://xml.apache.org/xerces-j/.
[49] Apache XML project, “Xalan Java”, http://xml.apache.org/xalan-j/.
[50] Apache XML project, “Xindice”, http://xml.apache.org/xindice/.
[51] Innovation., “HTTPClient Version 0.3-3”, http://www.innovation.ch/java/HTTPClient/.
[52] Y. J. Oh, H. T. Ju, J. W. Hong, “Interaction Translation Methods for XML/SNMP Gateway Using XML Technologies”, Proc. of the Asia-Pacific Network Operations and Management Symposium, Jeju Korea, Sep. 2002, pp. 11-22.
[53] Altova, “XML Spy”, http://www.xmlspy.com.

최 미 정

1998 이화여자대학교, 컴퓨터공학과 학사
1998~2000 포항공과대학교, 컴퓨터공학과 석사
2000~현재 포항공과대학교, 컴퓨터공학과 박사 과정
<관심분야> XML 기반의 네트워크 관리, Agent 기술, 정책 기반의 네트워크 관리

홍 원 기

1983 Univ. of Western Ontario, 전산학 학사
1985 Univ. of Western Ontario, 전산학 석사
1985 ~ 1986 Univ. of Western Ontario, 전산학과 강사
1986 ~ 1991 Univ. of Waterloo, 전산학 박사
1991 ~ 1992 Univ. of Waterloo, Post-Doc fellow
1992 ~ 1995 Univ. of Western Ontario, 연구교수
1995 ~ 현재 포항공과대학교 컴퓨터공학과 부교수
<관심분야> 분산처리, 네트워크 트래픽 모니터링, 네트워크 및 분산 시스템 관리, CORBA, Internet 관리