XML-Based Network Management for IP Networks

Mi-Jung Choi, James W. Hong, and Hong-Taek Ju

XML-based network management, which applies XML technologies to network management, has been proposed as an alternative to existing network management. The use of XML in network management offers many advantages. However, most existing network devices are already embedded with simple network management protocol (SNMP) agents and managed by SNMP managers. For integrated network management, we present the architectures of an XML-based manager, an XML-based agent, and an XML/SNMP gateway for legacy SNMP agents. We describe our experience of developing an XML-based network management system (XNMS), XML-based agent, and an XML/SNMP gateway. We also verify the effectiveness of our XML-based agent and XML/SNMP gateway through performance tests. Our experience with developing XNMS and XML-based agents can be used as a guideline for development of XML-based management systems that fully take advantage of the strengths of XML technologies.

Keywords: XML-based network management, XML, SNMP, XML/SNMP gateway, DOM, SAX, XML Schema, XPath, XSL, XSLT, XQuery, XUpdate, SOAP.

I. Introduction

The rapid pace of Internet evolution is currently witnessing the emergence of diverse network devices. The current complex IP networks are composed of these network devices. Efficient network management systems are necessary to manage these networks and network devices. Since its development in the late 1980s, the management of IP networks has exclusively relied on the simple network management protocol (SNMP) [1]. However, the SNMP management framework has some weaknesses related to configuration management and the application development process in managing large networks [2]. Today, technologies using Extensible Markup Language (XML) [3] seem to be a promising solution.

XML, which is a meta-markup language standardized by the WorldWide Web Consortium (W3C) for document exchange in the web, is widely used for business-to-business integration, data interchange, e-commerce, and the creation of application-specific vocabularies. It supports several standards such as XML Schema [4], document object model (DOM) application program interface (API) [5], XML path language (XPath) [6], Extensible Stylesheet Language (XSL) [7], XSL transformations (XSLT) [8], etc. Many efforts are in progress to apply XML technologies and their previously existing implementations to a wide range of network management. Using XML in network management presents many advantages [2]:

- The XML Schema defines the structure of management information in a flexible manner.
- Widely deployed protocols such as HTTP [9] reliably transfer management data.
- DOM APIs easily access and manipulate management data from applications.
- XPath expressions efficiently address the objects within

Manuscript received May 14, 2003; revised July 3, 2003.

This work was in part supported by the Electrical and Computer Engineering Division at POSTECH under the BK21 program of Ministry of Education and HY-SDR Research Center at Hanyang University under the ITRC program of Ministry of Information and Communication, Korea.

Mi-Jung Choi (phone: +82 54 279 5654, email: mjchoi@postech.ac.kr) and James W. Hong (email: jwkhong@postech.ac.kr) are with the DPNM Laboratory, POSTECH, Pohang, Korea.

Hong-Taek Ju (email: juht@kmu.ac.kr) is with the Computer Network Laboratory, Keimyung University, Daegu, Korea.

management data documents.

– XSL processes management data easily and generates HTML documents for a variety of user interface views.
– Web Service Description Language (WSDL) [10] and simple object access protocol (SOAP) [11] define web services for powerful high-level management operations.

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

Four possible combinations between managers and agents can be considered for XML-based integrated network management [12] (Fig.1). Figure 1(a) shows the current, widely deployed SNMP-based network management, and Fig. 1(d) shows an XML-based management scheme using an XML-based manager and XML-based agent. The gateways shown in Figs. 1(b) and 1(c) translate messages and operations between different management schemes, XML and SNMP. Figure 1(d) is the most ideal framework for gaining the maximum advantages of XML-based network management. However, most network devices currently deployed are equipped only with SNMP agents, and this architecture is hardly applicable to the current situation. Therefore, an XML/SNMP gateway is needed for the XML-based manager to manage SNMP-enabled network devices as well as devices equipped with XML-based agents. Figure 1(c) shows the most practical management framework using the XML/SNMP gateway, which translates and relays messages between an XML-based manager and an SNMP agent.

In this paper, we present the architecture of an XML-based manager, an XML-based agent, and an XML/SNMP gateway for XML-based integrated network management of IP-based networks. We also explain our XML-based network management system, called the XML-based network management system (XNMS), for managing network devices using XML technologies and methods of fully applying XML technologies to network management. Using performance tests, we verify the efficiency of XML-based network management systems. Our experience with developing XNMS, an XML-based agent, and an XML/SNMP gateway can be used as a good guideline for those who develop XML-based network management systems.

The organization of this paper is as follows. Section II describes the XML-based network management work carried out by other research groups [13]-[15] and vendors [16], [17]. In section III, 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 IV, we describe our implementation details of the proposed XNMS and XML-based agent. In section V, we discuss the validation and performance tests of the proposed solution. Finally, in section VI, we conclude our work and discuss directions for future work.

II. Related Work

In this section, we describe recent research on XML-based network management performed by other research groups. We also introduce several standard activities and industry efforts in this area.

1. Research on XML-Based Network Management

In this section, we describe related research work on XML-

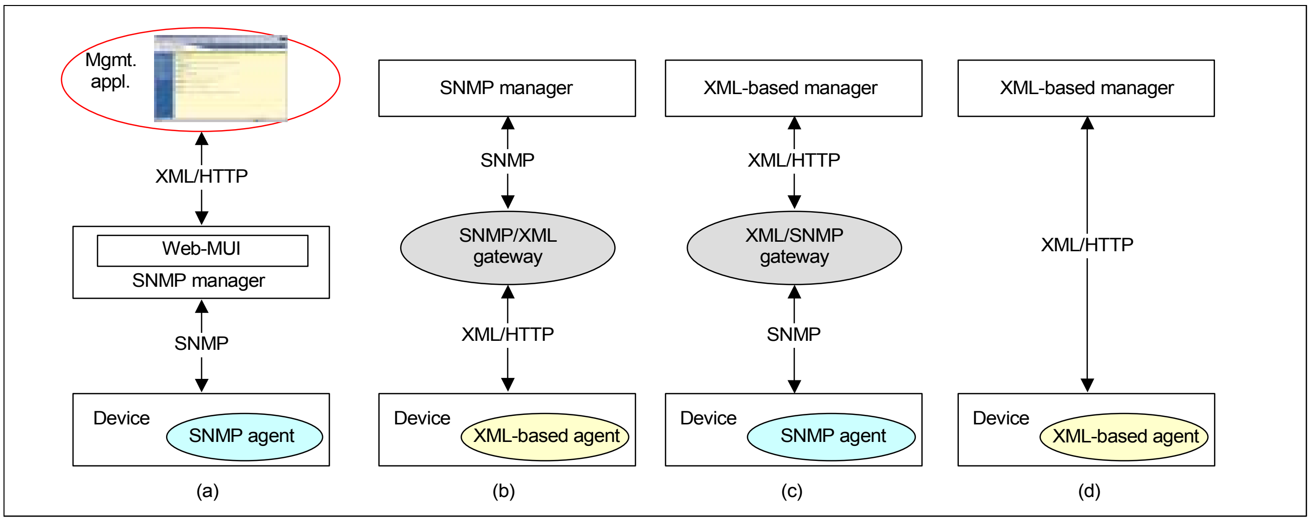


Fig. 1. Combinations of manager and agent.

SNMP integration, XML-based management architectures, and XML-based management using Web Services.

A. XML-SNMP Integration

- Specification translation

J.P. Martin-Flatin proposed the SNMP management information base (MIB) for XML translation models, namely model-level mapping and metamodel-level mapping [18]. In model-level mapping, the document type definition (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 presented a library to access structure of management information (SMI) MIB information, "libsmi" [19], which translates SNMP MIB to other languages, such as JAVA, CORBA, C, and XML. This library provides tools to check, analyze, dump, convert, and compare MIB definitions. The conversion tool, called "smidump," converts MIB modules to XML Schema.

In previous work on the XML/SNMP gateway, we developed an SNMP MIB to XML translation algorithm and also implemented an SNMP MIB to the XML translator using this algorithm [20]. For validation of the algorithm, we implemented an XML-based SNMP MIB browser using this SNMP MIB to the XML translator.

- XML/SNMP gateway

Jens Müller implemented an SNMP/XML gateway [19] as a Java Servlet that allows fetching of such XML documents on the fly through HTTP. MIB portions can be addressed through XPath(-like) expressions encoded in the URLs to be retrieved. The gateway works as follows. When an 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 is currently developing an XML-based management interface for SNMP enabled devices [15]. The prototype system consists of three parts:

– A tool for automatic generation of an XML Schema definition based on SNMP SMI information.
– A messaging protocol based on an XML-remote procedure call (RPC) 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 the XML Schema. This tool is an extension of a previously implemented tool for converting SNMP SMI to CORBA-IDL [21]. They are currently implementing the XML document adapter for SNMP MIB modules using Net-SNMP [22] and XML-RPC libraries [23].

In our previous work [24], [25], we proposed several interaction translation methods between an XML/SNMP gateway and an XML-based manager based on the previously developed specification translation algorithm [20]. First, we proposed a DOM-based translation method that enables the manager to directly access 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 that 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 improves efficiency in XML/HTTP communication, which is the most common form 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 the XML-based manager in a distributed environment. We have implemented and validated our XML/SNMP gateway for our XNMS.

B. Management Architecture

J.P. Martin-Flatin presented an idea to use XML for integrated management in his research on web-based integrated network management architecture (WIMA) [18]. WIMA provides a way to exchange management information between a manager and an agent through HTTP. HTTP messages are structured with a multipurpose Internet mail extensions (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, common information model (CIM), or other management data. A WIMA-based research prototype, java management platform [18], implemented push-based network management using Java technology.

We proposed an XML-based network management (XNM) using an embedded web server (EWS) in our previous research [26]. 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.

C. XML-Based Management Using Web Services

The Network Management Research Group (NMRG) [13] of the Internet Research Task Force (IRTF) is a forum for researchers to discuss and develop new technologies for improving Internet management. 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 [11], WSDL [12], and universal discovery description and integration [27], and compared them with SNMP. They also dealt with security in web services. NMRG's work in this area is in the early stage and has not yet produced any substantial results.

The Organization for the Advancement of Structured Information Standards (OASIS) [28] 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 [14] is to develop open industry standard management protocols 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) [29], XML, SOAP, DMTF CIM, and DMTF CIM operations. The management protocol technical committees (TCs) delivered a published management protocol specification on June 2003, but their work in this area is in the early stage.

2. Standardization Activities

In this section, we present the standardization efforts on XML-based network management within the Distributed Management Task Force (DMTF) and the Internet Engineering Task Force (IETF).

A. WBEM

Web-based enterprise management (WBEM) [30] is an initiative of the DMTF and includes a set of technologies that enables the interoperable management of an enterprise. WBEM consists of a CIM [31], a DTD to represent CIM in XML [32], and a specification for CIM operations over HTTP [33]. 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 [28] to sponsor a new management protocol technical committee and to develop open industry standard management protocols.

B. IETF XML Configuration (XMLCONF) BOF and Network Configuration (Netconf) Working Group

In the 54th IETF meeting in July 2002, a birds of a feather (BOF) session concerned with XML configuration (XMLCONF) was held. This BOF discussed the requirements for network configuration management and how the existing XML technologies, namely SOAP [11], WBEM [30], SyncML [34], and JUNOScript [16] could be used to meet those requirements. There are some Internet drafts [35], [36] that present basic concepts and the requirements for XML network configuration and provide guidelines for the use of XML within IETF standards protocols.

The Network Configuration (Netconf) Working Group [37] was formed in May 2003. The Netconf Working Group is chartered to produce a protocol suitable for network configuration. The Netconf protocol will use XML for data encoding purposes, because XML is a widely deployed standard that is supported by a large number of applications. XML also supports hierarchical data structures. The Netconf working group will take the XMLCONF configuration protocol as a starting point.

C. ITU-T X.3030 - Telecommunications Markup Language (tML) Framework

The Alliance for Telecommunications Industry Solutions Technical Subcommittee T1M1 (Internetwork Operations, Administration, Maintenance and Provisioning) is developing a Telecommunications Markup Language (tML) standard that would govern telecommunications network management. The tML is a language derived from XML and based on plain text tags that describe vocabulary used in the exchange of data between telecommunications entities. The goal of the tML framework is to guide the development of interoperable operations, administration, maintenance, and provisioning (OAM&P) interfaces using XML for the telecom domain, to apply to various telecommunications OAM&P functions, and to provide a common framework in developing network management specifications by different groups. This recommendation [38] is a framework containing rules, guidelines, and objectives for developing telecommunications industry standard tML schemas for OAM&P applications.

3. Industry Initiative

In this section, we introduce recent industry initiatives for XML-based network management.

A. Juniper Networks' JUNOScript

Recently, Juniper Networks introduced JUNOScript [16] for their JUNOS network operating system. The JUNOScript, a part of their XML-based network management effort, uses a simple model that is designed to minimize both 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.

B. Cisco's Configuration Registrar

The Cisco configuration registrar [17] 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. It 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.

III. Design of an XML-Based Network Management System

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

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. The web server is used to receive asynchronous messages for notifications from the devices via HTTP. The HTTP client plays a role in the interface module of the device and exchanges synchronous management information with the agent. The database (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 or analyzer, 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.

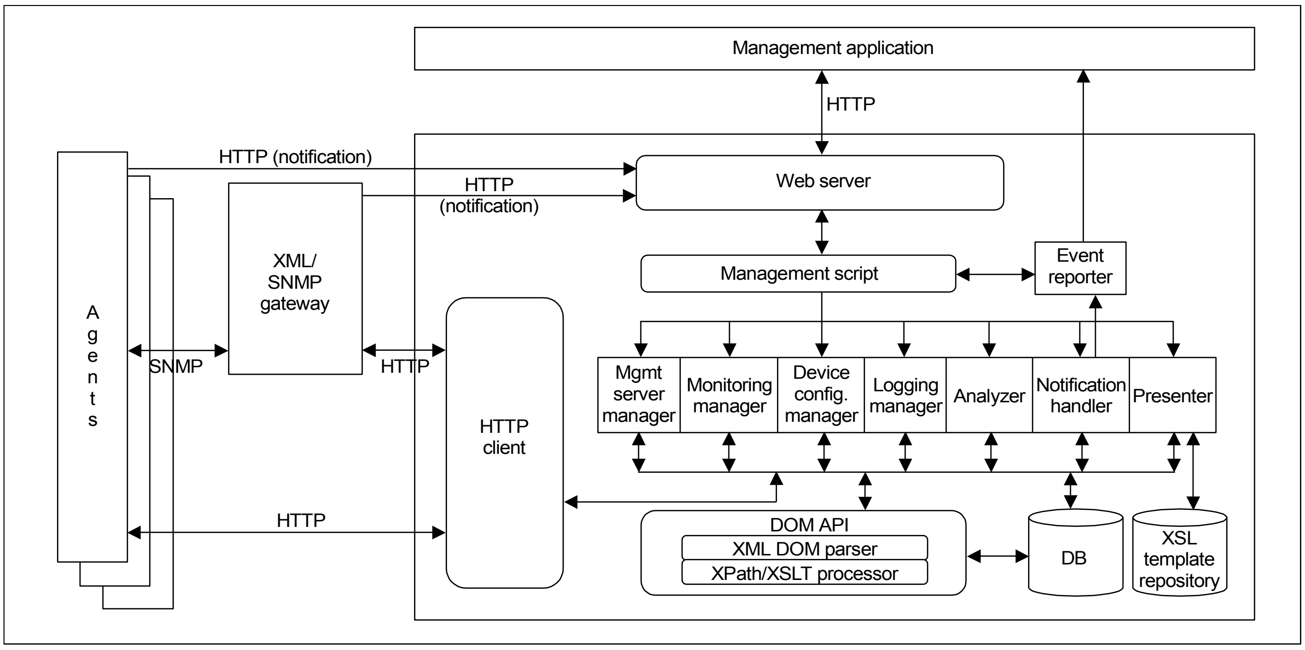


Fig. 2. Architecture of XML-based manager.

The basic components processing management functionalities [39] are management server manager, monitoring manager, device configuration manager, analyzer, notification handler, logging manager, presenter, and event reporter. The management server manager manages the configuration parameters for the management processing environment and handles the topologies of multi-level device/device group tree architecture. This component also manages an administrator account list. The device configuration manager gets and sets the configuration of a managed device; the monitoring manager obtains device monitoring information, such as device status and in/out traffic; and the Logging Manager logs the necessary data and analyzes the log data stored in the DB according to the administrator's request.

The Notification Handler receives notifications from the managed devices, stores the notifications in the DB tables, and sends a meaningful notification to the event reporter. The event reporter generates appropriate events and sends them to administrators by email, a pager, etc. The analyzer analyzes the collected management information. The presenter processes the XML document with XSLT and generates an HTML document for Web-MUI.

There are three typical information flows within the XML-based manager in Fig. 2. The first data flow is the management request from the management application to the web server. 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, which sends the request to an agent through the HTTP client, after which the result returns to the management application in reverse order. If the result needs to be stored for later analysis, the monitoring manager stores it in the DB. This flow is the same as the traditional SNMP Get operation. The flow of a Set method is the 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 an XML-based manager through HTTP. 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 the DB. This flow is the 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.

2. XML-Based Agent

Figure 3 illustrates the architecture of an XML-based agent. The basic component of the XML-based agent is the embedded web server (EWS) [40]. The components added to the EWS are the XML processor, including the SAX Parser (which is an XML parser) and the push scheduler, and the HTTP Client Engine. The SAX Parser does not support a write function; therefore, a Write Module is also necessary.

In our previous work [26], we used DOM and XPath for handling the XML document in the agent as well as in the manager. DOM and XPath are effective in processing XML documents for accessing and filtering. In our previous work, the device that embeds the XML-based agent was a Linux server, which posed no problems with resources. However, supporting DOM [5] and XPath [6] needs more memory resources in the device. To access a part of an XML document, the DOM tree of the whole XML document is loaded into memory. This wastes CPU and memory resources in the device. Moreover, the code size of basic libraries widely used for supporting DOM and XPath is large. Therefore, it is not suitable to apply DOM in an embedded system having few resources.

The code size and executable memory size of the Sax Parser [41] is smaller than DOM [42], [43]. As the SAX Parser is an event-driven mechanism for accessing and processing XML documents, there is no need to load the entire XML tree into the memory. Therefore, the SAX Parser is much lighter than the DOM from the perspective of functionalities and resources. While the SAX Parser can be lighter in resource usage because it reads the XML document in sequential order and generates an event for a specific element, the processing time for accessing an XML document of the SAX Parser is slower than that of DOM after the DOM tree is generated. However, the management information of most network devices is not so immense and can be defined by several small XML documents containing mutually related management data. Therefore, the processing time of the SAX Parser matters little. The access method of the SAX Parser is serial and read-only, so we added a Write Module as part of XML processor to provide a writing mechanism. Because we set a greater value on low resource requirements, we selected the SAX Parser instead of the DOM.

The SAX Parser parses the XML document, selects the specified node when parsing, and reads management data. In order to send up-to-date information, the agent gathers

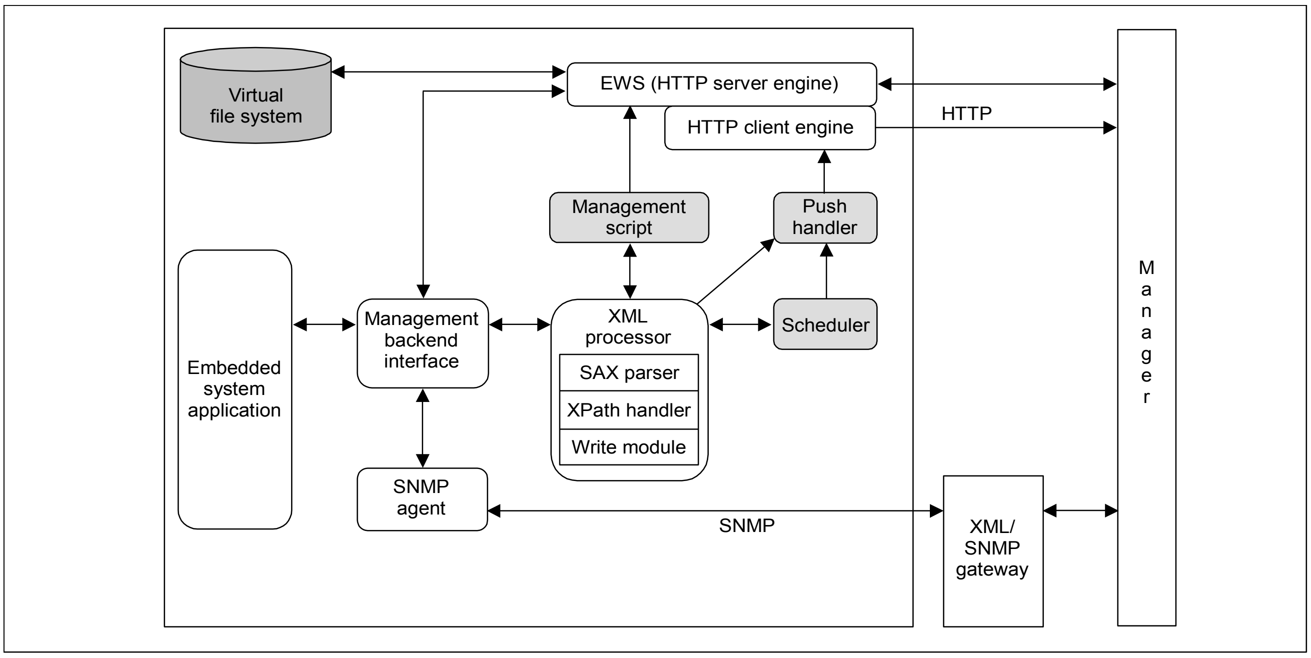


Fig. 3. Architecture of XML-based agent.

information from the Management Backend Interface. The Write Module updates the node value for the selected node through the Management Backend Interface before replying to the manager.

To send a notification to the XML-based manager, the XML-based agent needs a Push Handler and HTTP Client Engine as well as an XML Processor module. In addition, to send periodic management information to the manager at a fixed time with one schedule request, the XML-based agent needs a Scheduler. The HTTP Client Engine delivers asynchronous messages to the XML-based manager for reporting alarms and distributing management data according to the schedule. The Scheduler manages subscription information and the schedule for the distribution of the management information. Subscription information includes the subscriber's URL for receipt of subscriber information, a managed object's XPath expression (management item), and schedule information containing the start time, the end time, and the interval. 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. The Push Handler also sends a notification generated in the agent, which is sent to the manager through the HTTP Client. If the XML-based agent does not receive the 'HTTP OK' response to the notification from the manager, the Push Handler resends the notification message to the manager.

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.

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 the XPath/XUpdate expression in the request message. The Request Handler in the gateway receives and parses the XPath/XUpdate expression and delivers the request to the gateway application. The 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 as the reverse of those of the XML/SNMP gateway.

IV. Implementation and Experience

We implemented an XML-based network management system (XNMS), an XML-based agent, and an XML/SNMP gateway to validate our XML-based integrated management system. The XNMS is used to manage one or more types of

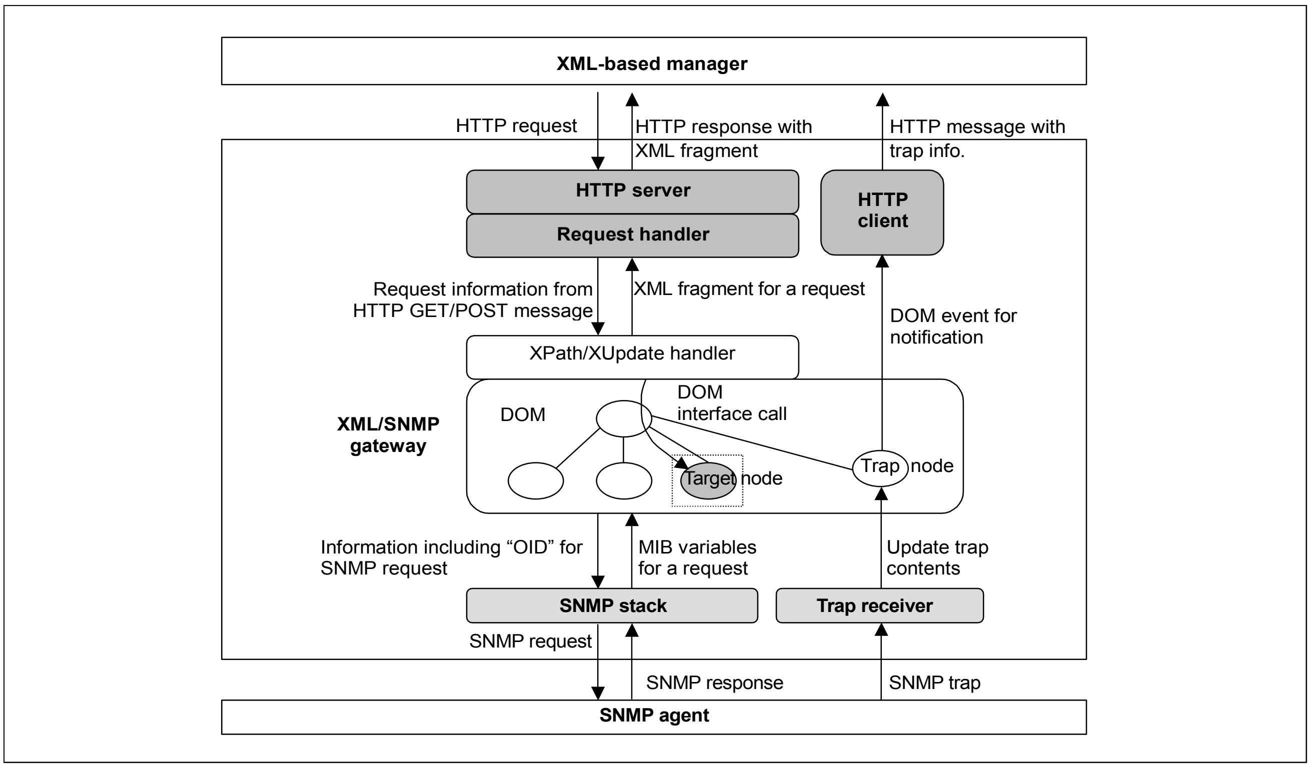


Fig. 4. Architecture of XML/SNMP gateway.

network elements scattered throughout the world. In this section, we explain our experience developing the XNMS and XML-based agent using XML technologies. We used the XML/SNMP gateway [20], [24], [25] approach for managing existing SNMP agents. The reasons the gateway approach is necessary are as follows:

- The configuration management in industry is already changing to an XML-based management. Therefore, having an XNMS to process XML is also necessary for integrated management.
- Management systems need to support managing network devices that are presently embedded with SNMP agents and that utilize maximally existing SNMP agents. Moreover, fault management and performance management are 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 a solution of efficient and atomic transfer of configuration data by applying XML related technologies.
- An SNMP agent is already embedded and manages most network devices. It is almost impossible to change the agents in devices that are already deployed. The latest requests concerning network management encourage developers to develop a new management system without modifying the existing agent. XML technologies provide many advantages in network management as mentioned in section I, and we can develop new management systems using XML technologies. Therefore, the gateway approach needs an XML-based manager to manage existing SNMP agents.

1. Implementation Details of XML-Based Manager

We use XML Schema to define the management information. Figure 5, drawn using an editing tool of XML Spy [44], shows the management information of XNMS defined in an XML Schema format. XML Schema presented in dotted lines in Fig. 5 represents optional management information and the "0..∞" expression under the elements means that the number of elements can range from zero to infinity. The basic cardinality is one when nothing is specified. A dotted box with no cardinality specification means an optional element, that is, the number of this element is zero or one. The management information of XNMS is divided into two parts: a server part and a device part. The server part consists of the server configuration, administrator lists, XML/SNMP gateway lists, and device topology information.

The device part consists of device information, device configuration, logging, and monitoring. XML Schema has complex types containing elements and attributes and simple

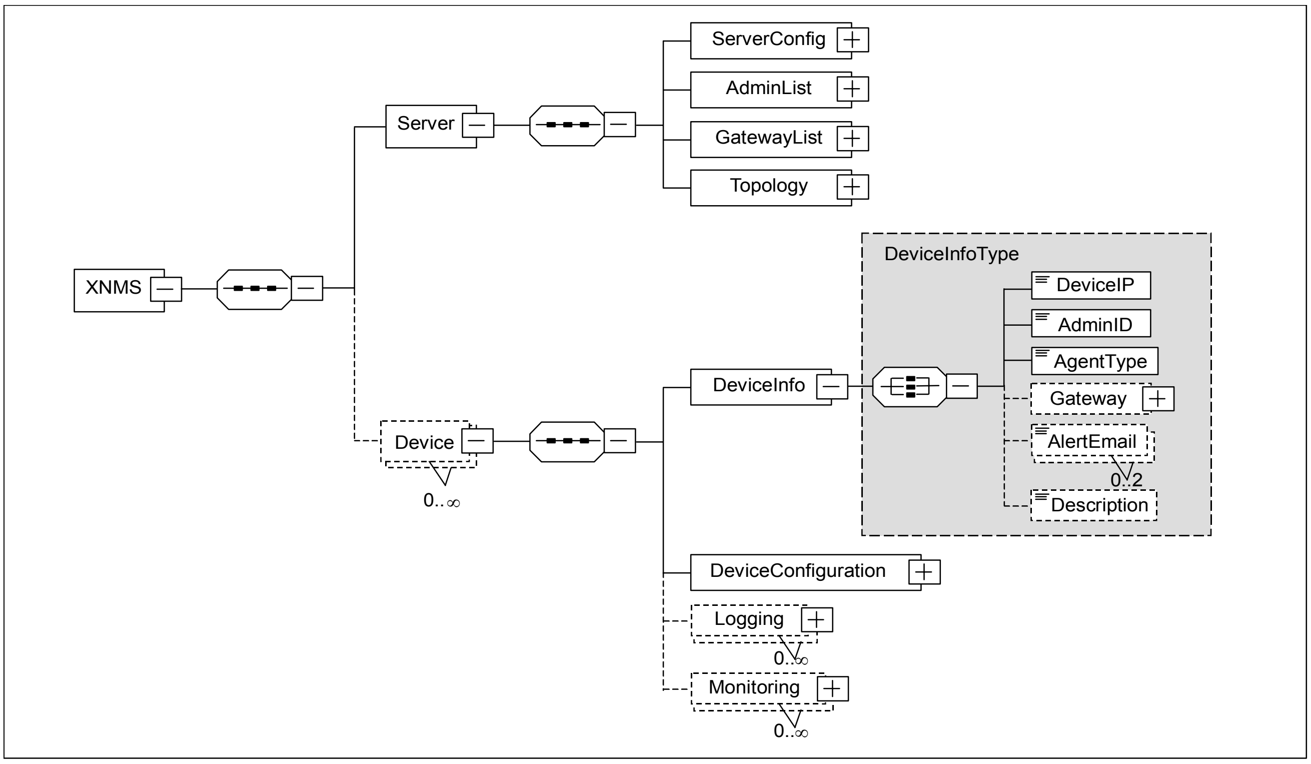


Fig. 5. XML Schema for XNMS.

types containing none. XML Schema supports 44 kinds of built-in simple types, including string, integer, float, time, date, etc. DeviceInfoType is defined as a complex type and the type of DeviceInfo is set to DeviceInfoType. Each element, such as DeviceIP, AdminID, or AgentType, has its own data type. We add at least one attribute to the device information as a primary key for searching. DeviceInfoType has the attribute string type and DeviceID unique key. If the type of an agent (AgentType) is an XML-based agent, the information from Gateway is not necessary.

The DeviceConfiguration section is defined as XML Schema converted from the MIB definition of each device. This is for consistency of management information between the XML-based manager and the SNMP agent interacting through the XML/SNMP gateway. We translate MIB II to XML Schema for basic management information. If a new device embedding an SNMP agent is added to XNMS, our MIB2XML translator [20] converts the private MIB definition to XML Schema and the translated XML Schema is added to XML Schema of XNMS as a child element of DeviceConfiguration in Fig. 5.

We considered three methods for efficient interaction between XML-based manager and the XML/SNMP gateway. We adopted the HTTP-based translation among interaction approaches [24], [25] between the XNMS 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.

In the HTTP communication between XNMS and the gateway or between XNMS and the XML-based agent, we define XML Schema for an HTTP Get/Post message. XNMS 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" [45] expression and Table 1 shows its example. 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. If the agent is an XML-based agent, the gateway information is not necessary, as the XML-based manager directly connects to the XML-based agent.

The XNMS sends an HTTP Post request to insert, delete, and update management information. An HTTP Post message contains request details in its message body. We use XUpdate [46] 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

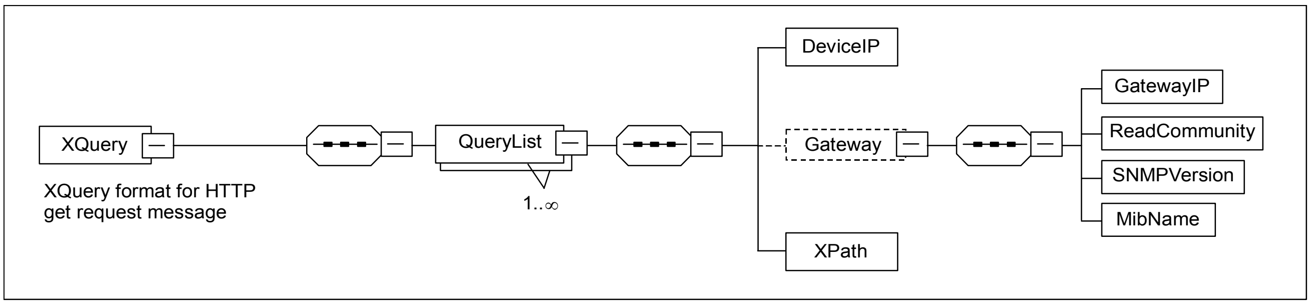


Fig. 6. XML Schema for “XQuery” in HTTP Get.

Table 1. XML example for “XQuery” in HTTP Get.

Between XNMS and the XML/SNMP gateway	Between XNMS and the XML-based agent
http://XNMS/monitoring.jsp?XQuery=<XQuery> <QueryList> <DeviceIP >141.223.82.121</DeviceIP> <Gateway><GatewayIP>141.223.82.56</GatewayIP> <ReadCommunity>public</ReadCommunity> <SNMPVersion>0</SNMPVersion> <MibName>RFC1213-MIB</MibName></Gateway> <XPath>//interfaces</XPath></QueryList></XQuery>	http://XNMS/monitoring.jsp?XQuery=<XQuery><QueryList> <DeviceIP >141.223.82.122</DeviceIP> <XPath>//interfaces</XPath></QueryList></XQuery>

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 an HTTP Post message.

To store management data for later analysis, a database is necessary. There are two practical DB formats for storing and retrieving XML content: native XML databases and relational databases (RDBMS) [47]. An 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 the storage of XML documents, the 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 having 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 a 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 the same as a row in the table. Any XML document can be added to a collection regardless of schema. We make collections in accordance with the XML Schema in Fig. 5. We make an XNMS collection, which contains the Device collection, which in turn contains the DeviceInfo collection in hierarchical order, as shown in Table 2. The cardinality of the AlertEmail element shown in Fig. 5 is 0 to 2, that is, the number of AlertEmail elements can be between zero and two. The device information for the SNMP agent needs the gateway information. However, the device information for the XML-based agent does not include the gateway information. Therefore, the cardinality of Gateway element is zero or one. The left column in Table 2 is for the device equipped with an SNMP agent that includes two AlertEmail and one Gateway items of information. The right column of the device is equipped with an XML-based agent that includes one AlertEmail and no Gateway information.

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 directly access the specified document. Therefore, we can easily

Table 2. XML document of DeviceInfo.

Device equipped with an SNMP agent	Device equipped with an XML- based agent

XNMS/Device/DeviceInfo
`<? xml version= "1.0" ?>`
`<DeviceInfoList DeviceID= "device1" >`
  `<DeviceIP>141.223.82.121</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>`
	XNMS/Device/DeviceInfo
	`<? xml version= "1.0" ?>`
	`<DeviceInfoList DeviceID= "device2" >`
	  `<DeviceIP>141.223.82.122</DeviceIP>`
	  `<AdminID>mjchoi</AdminID>`
	  `<AlertEmail>meanie@postech.ac.kr </AlertEmail>`
	  `<AgentType>2 (XML-based agent) </AgentType>`
	  `<Description>IP Sharing Device </Description>`
	`</DeviceInfoList>`

retrieve, update, and delete documents using an XPath expression. DB schema is consistent with XML Schema in a native XML DB, and storing management information to the DB is simple and there is no overhead for parsing the XML document. Therefore, we can easily process XML data using the native XML DB, which results in fast and easy development of the XNMS.

DOM [5] is the means of accessing and manipulating XML documents. DOM supports the 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 an 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 entire XML document is loaded in memory. This wastes memory and CPU resources and requires much processing time. Therefore, we make an effort to save resources. We use the DocumentFragment interface for updating an XML document. This interface provides a method to update a small portion of the document without constantly updating the NodeLists and NamedNodeMaps associated with the entire document. Updating the NodeLists can significantly slow down execution. As mentioned previously, we use a native XML DB to store management information. This reduces the manipulation of XML documents 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 [8] to accomplish the XNMS 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 XNMS. Static information such as XNMS server configuration data, which is not specific to managed devices, can be rendered using pre-defined XSLT. Another type of presentation is to generate a dynamic web page for device configuration, which has various items and styles to present according to devices and their MIBs.

XNMS maintains XML Schemas for the managed devices, and also XSL templates for the XML documents conforming to XML Schema. The XSLT template for each MIB is generated by the XSLT Generator of the XML/SNMP Gateway, and downloaded to XNMS whenever the MIB is translated. Table objects defined in the MIB is presented as HTML [48] table view using the template. XNMS 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.

2. Implementation Details of XML-Based Agent

The main function of the XML-based agent is to retrieve and update management information according to the manager's request of a Get/Set operation. The XML-based agent also delivers periodic monitoring data to the manager by performing the scheduler according to the manager's scheduling request. Figure 7 illustrates the function call for the flow of the Get/Set process of the XML-based agent.

The process of the Get operation in XML-based agent is as follows. The Get request from the manager calls the Get module through Management Script with two parameters: XML filename and XPath expression. The value of XML filename is used as the parameter of the xmlOpen() function for opening the appropriate XML document; the value of XPath is the parameter of the getXpath() function, which returns the structure type of XPath to utilize in the next phase. The XML document from xmlOpen() and XPath returned from getXpath() input into the parseXml() function as parameters. The parseXml() parses the XML document through the XPath grammar. The real management value is retrieved through the call of Management Backend Interface; then the XML document is updated with this value using the setXml() function. The parsed XML document is sent to returnXml() and backtracks to Management Script as string format. Finally, the xmlClose() function closes the previously opened XML document.

The process of the Set operation is almost equal to the Get operation. The Set module opens the XML document and retrieves the specific XML document part by applying an XPath expression as a Get operation, then modifies the XML document calling setXml(), and updates the real value through Management Backend Interface. Afterwards, it generates the XML document from the result of the Set operation and calls returnXml() function. The returned result to Management Script is the response of the Set operation, 'HTTP OK' message.

In Fig. 7, the getXpath() function is the XPath Handler module processing XPath. Currently, XPath supports various syntaxes. However, if the XPath expression is complex, the processing time is slow [49], and the XPath Handler supporting full XPath grammar [50] is heavy. This does not meet the requirements of low resource utility. Therefore, we implemented a part of XPath grammar sufficient for accessing the XML document of management information in the XML-based agent. Moreover, we implemented the XPath Handler to access the specific management information applying XPath expression during the parsing without loading the XML document into memory. Table 3 shows the XPath grammars that we implemented in our XML-based agent. We implemented the XPath Handler considering the extensibility supporting more XPath syntaxes. It is desirable to extract management information using simple XPath expression by considering the processing time.

In Fig. 7, the parseXml() function, which reads the XML document in sequential access and applies XPath expression, is the core module of the SAX Parser. This parses the XML document from the root element to its child element, retrieves the element name and attributes, and compares the retrieved values to XPath expression processed by the getXpath() function; then if the comparison result is equal, the values are sent to the next function. The setXml() function is for the Write Module. In the case of a Set request, if the Set operation needs to update the value, this function modifies the XML document after parsing the document that applies the XPath expression.

Figure 8 describes the process of the Scheduler for delivering periodic data to the manager at the scheduled time and the Trap for sending a notification to the manager. If the Scheduler receives a scheduling request from the manager, it updates the job list file. The Scheduler processes the job with thread functions. In the initial start, the Scheduler runs the getJob()

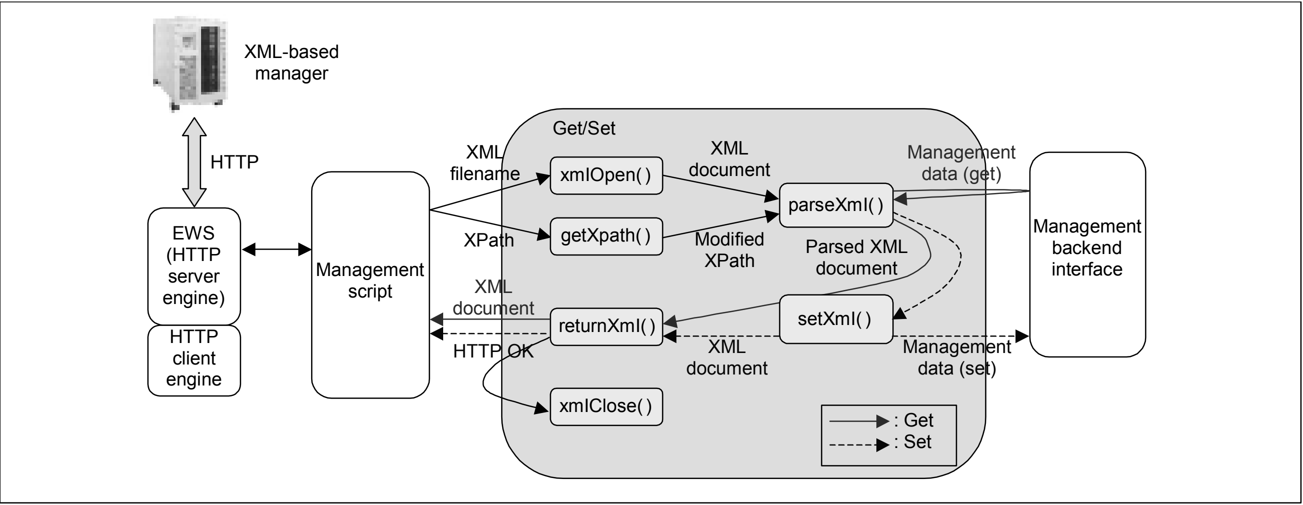


Fig. 7. Flow of get/set operations.

Table 3. Supported XPath grammar in XML-based agent.

Grammar	Explanation	Example
/	The basic XPath syntax similar to file system addressing	/AAA/BBB
//	All elements in the document which fulfill following criteria are selected	//BBB
*	All elements located by proceeding path	//*
@	Attributes are specified by @ prefix	//@id
[]	Filter	BBB[@id='b1']
=	Comparative operator	BBB[@id='b1']
|	Equals to logical OR	//AAA | // BBB
&	Equals to logical AND	//AAA & // BBB

function, and this function reads the job list file and retrieves the job contents. As mentioned in the scheduler part in section III.2, the job list file includes subscriber information, management items, and schedule information containing start time, end time, and interval. The checkJob() function receives the number of jobs and the job list from getJob() and compares the existing threads information to the new job list. According to the comparison result, the checkJob() generates the new job thread calling insertJob(), or destroys the existing job thread calling deleteJob(). The generated thread having the same time interval as the parameter checks the schedule calling timerHandler() and runs the runWget() function at the scheduled time. The runWget() function calls the Push Handler, then the processed management information is sent to the manager through an HTTP Client Engine called Wget [51]. A notification from the Management Backend Interface is sent to genXmlTrap() function in the Trap module. This function generates an XML document containing the trap information and calls the runWget() function. Trap information is delivered to the manager by this mechanism.

V. Validation and Performance Test

We implemented the XNMS on a Linux OS using Java language. The XML packages used in the XNMS are mostly from the Apache project [52]. We used the Apache Web server and OpenSSL [53] to communicate in the secure mode of HTTPS. We used Xerces [54] as a DOM parser and Xalan [55] as an XPath/XSLT processor. We used Xindice [56], a 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 [57] as the HTTP Client, which is also Java-based. A more detailed description of the implementation of the XML/SNMP gateway is found in our previous papers [20], [24], [25].

We implemented an XML-based agent to manage an IP sharing device based on the design presented in section III.2. Moreover, we focused on the implementation of an efficient and lightweight XML-based agent by considering such requirements as low resource utility (CPU usage and memory size). In addition, for portability to equip any type of embedded system, we used the C programming language throughout agent implementation and developed components for each module. To control the access of management information,

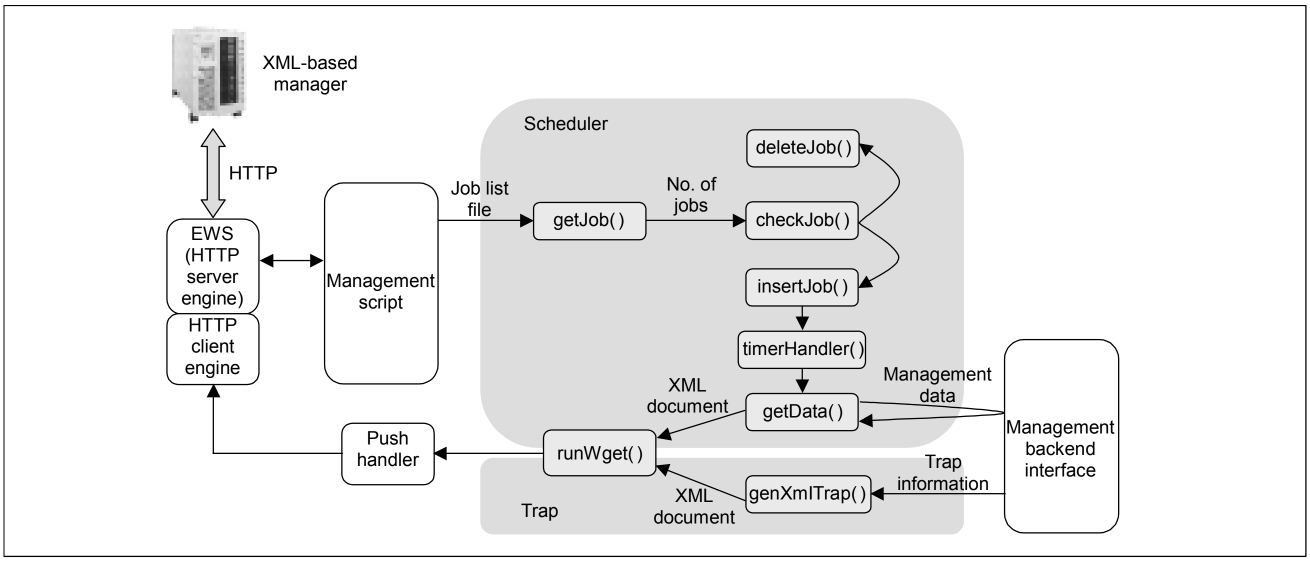
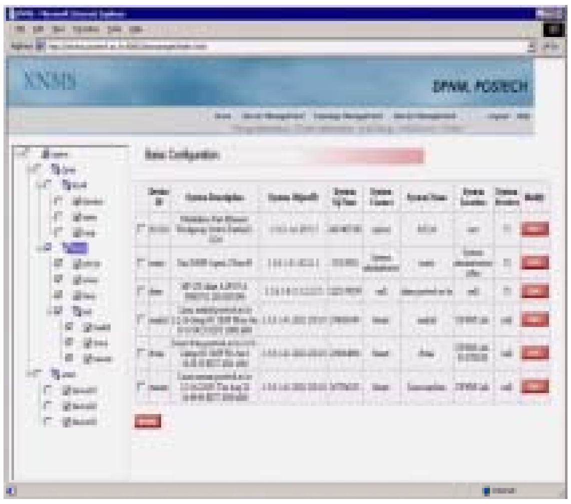
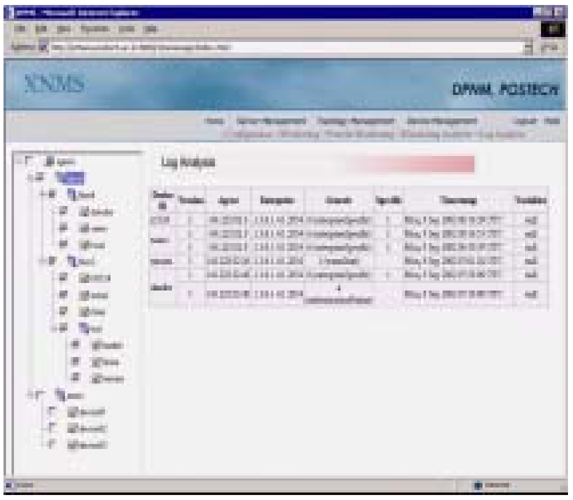


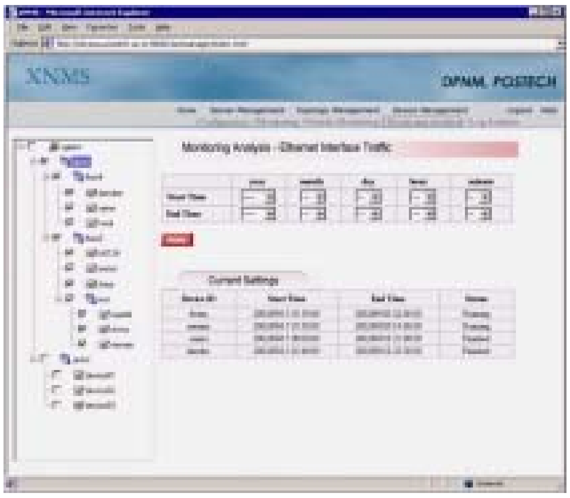
Fig. 8. Flow of Trap/Schedule operations.



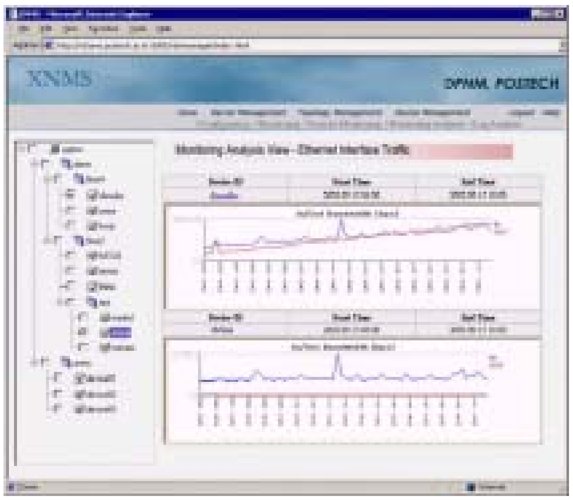
(a) Get/Set (MIBII-System Group)



(b) Log analysis from trap notification



(c) Configuration of periodic monitoring



(d) Result of monitoring analysis

Fig. 9. Examples of XNMS user interface.

access to the XML-based agent is permitted through authentication with ID and password in the initial contact. The IP sharing device equipped with our XML-based agent runs on an embedded Linux based on the linux2.2.13-7 kernel using Motorola's MPC850DE processor [58] with a 16 MB ROM. We used a powerpc-linux-gcc compiler.

Using the XNMS, an administrator has a unified management interface for multiple devices. Figure 9 shows examples of the XNMS user interface. Figure 9(a) shows the information of the system group of MIB II. First, one checks the device ID in the left tree and clicks the Configuration submenu of the Device Management menu in the upper frame in the Fig. 9 (a). The flow of management information of Fig. 9 (a) is the first flow (Get/Set flow) mentioned in section IV.1. Figure 9(b) shows the Trap information received from all devices. The flow of this information is the second flow (Trap flow) mentioned in section IV.1. Figure 9(c) shows the current setting of periodic monitoring. The two devices are in the process of monitoring, and the others have finished the periodic monitoring. If you check the device check-button in the tree panel and click the "Apply" button in Fig. 9(c), you get what is shown in Fig. 9(d). Figure 9(d) shows the monitoring analysis result of selected devices. The graph shows the in/out bandwidth of the ethernet interface during a specified time period. The flow of management information of Fig. 9(d) is the third flow mentioned in section IV.1.

The management functionalities of the XNMS 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 the XNMS 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 an XSLT template from an MIB definition. We used freely-available codes to process the XML.

We verify the performance [59] of our XML-based agent by comparing it with the SNMP agent on the same IP sharing device. We compare resource utilities, such as CPU load, run-time memory size, and executable code size. We also compare the generated network traffic [60] of each agent between the manager and the agent. Moreover, we measure how much network traffic is reduced by a scheduling and push mechanism for periodic monitoring. Finally, the processing time of our XML-based agent against the traditional SNMP agent is measured with the response time of Get/Set requests between the manager and the agent. We also measure the network traffic and response time between the XML-based manager and SNMP agent on the IP sharing device through the XML/SNMP gateway.

Table 4 compares the SNMP agent and the XML-based agent in terms of CPU load, run-time memory usage, and executable code size. This information is discovered by the Linux commands, such as *top, cpuload*, etc., and the status data in the proc directory generated during the run-time of the process daemon. Both the XML-based agent and the SNMP agent provide SNMP MIB II information. The SNMP agent extends the Net-SNMP [22] and supports only SNMPv1.

From Table 4, our XML-based agent takes more resources than the SNMP agent. However, the increase is insignificant. While the XML parser generally consumes many resources and may be insufficient in application to the embedded systems, our XML-based agent is lightweight enough to be equipped in network devices and perform network management functionality.

Table 4. Resource utility of the SNMP and XML-based agent.

Agent	CPU load	Run-time memory usage	Executable code size
SNMPv1	17 %	600 kB	400 kB
XML-based	20 %	700 kB	550 kB

Table 5 shows the network traffic of each agent between the SNMP agent and manager, between the XML-based agent and manager, and between the SNMP agent and XML-based manager through the XML/SNMP gateway in terms of system group and interface group of MIB II information. We captured packets and those sizes between each manager and agent by the network traffic monitoring tool, Ethereal [61]. The network traffic of the XML/SNMP gateway is the sum of traffic between the XML-based manager and the XML/SNMP gateway and between the XML/SNMP gateway and the SNMP agent.

In the case of requests for one object in Table 5, we can easily determine the smaller size of the Get request message and response message of the SNMP agent because it is aimed to serve the network management protocol. However, the XML-based agent uses the HTTP protocol, so it increases network traffic for each information access over the SNMP. Conversely, for a grouped request, the XML-based agent produces less network traffic than the SNMP agent because the SNMP agent requests several GetNext operations and receives responses for every node, while the XML-based agent retrieves the entire system group or interfaces group information in one request using an HTTP message. Moreover, the difference is outstanding in the case of the interfaces group because it includes more managed objects. Most cases of Get or Set

Table 5. Message size of Get.

Management property	Get request message (bytes)			Get response message (bytes)		
	SNMP	XML-based	XML/SNMP gateway	SNMP	XML-based	XML/SNMP gateway
sysDescr	82	238	396 (314 + 82)	145	240	401 (145 + 256)
sysContact	82	240	388 (316 +82)	103	190	309 (103 + 206)
system Group	572	241	889 (317 + 572)	722	440	1178 (722 + 456)
inOctets (2 interfaces)	169	240	485 (316 + 169)	175	252	443 (175 + 268)
outOctets (2 interfaces)	169	241	486 (317 + 169)	176	256	448 (176 + 272)
interfaces Group	3720	241	4037 (317 + 3720)	3818	1654	5488 (3818 + 1670)

operations for management information request multiple data at one time. Therefore, the XML-based agent produces less network traffic than the SNMP agent in this case.

If the XML-based manager manages the SNMP agent through the XML/SNMP gateway, the network traffic is the sum of traffic between the XML-based manager and the XML/SNMP gateway and between the XML/SNMP gateway and the SNMP agent. The message between the XML-based manager and the gateway adds the MIB information, such as MIB name and community name to the message between the XML-based manager and the XML-based agent. The network traffic between the XML-based manager and XML/SNMP gateway can be reduced using a compression mechanism of HTTP. Also, the XML/SNMP gateway can be implemented as an internal gateway, which implies the XML-based manager and XML/SNMP gateway run on a single system. This internal gateway does not generate network traffic. However, the computing overhead for implementing the manager and the gateway in a single system is complex. We compared the gateway performance of three different methods from the viewpoint of network traffic and response time in our previous paper [25], [62].

Regarding in/out octets to monitor periodic network traffic for devices with two interfaces, the SNMP creates a request message of approximately 169 bytes to each in/out octet information and response message of approximately 175 bytes. However, the XML-based agent receives a request message of 600 bytes to set schedule information in the initial time and sends only a message of 252 bytes periodically to the manager. If the period is shorter and the manager has many devices to monitor, the scheduler mechanism in the XML-based agent achieves a more effective network traffic reduction. The manager can reduce processing overhead to generate Get request messages for periodic monitoring.

Table 6 shows the message size of the Set operation. The Set response message size of the SNMP is the same as the Set request message size of the SNMP. The response message of XML-based is the “HTTP OK” message. The Set operation is processed by an HTTP post operation. The size of the post operation is longer than that of the Get operation in HTTP. As the Get operation, the Set operation can reduce network traffic when there is a request for multiple data all at once.

Table 7 shows the message of the Trap operation. The SNMP agent sends a notification to the manager, but the manager does not respond to the notification to the agent. Therefore, the Trap response message size of the SNMP is 0. However, this can cause a loss in important notifications. The notification message size of the XML-based agent is longer than that of the SNMP agent, and the XML-based manager generates the response message. However, the network traffic generated by the Trap message is scarce. The response message of the XML-based manager guarantees the safe delivery of the important Trap messages. For communication through the XML/SNMP gateway, the XML/SNMP gateway generates an XML-formatted Trap message from the SNMP Trap message, sends the message to the XML-based manager, and receives the response from the manager.

Table 8 shows the response time of each MIB object. The response time of the Get/Set request is measured by the time() function call from the request initiation to response reception. The XML-based agent takes more time than the SNMP agent for accessing each leaf node in the system group in MIB II. For

Table 6. Message size of Set.

Management property	Set request message (bytes)			Set response message (bytes)		
	SNMP	XML-based	XML/SNMP gateway	SNMP	XML-based	XML/SNMP gateway
sysDescr	104	350	550 (446 + 104)	104	210	330 (226 + 104)
sysContact	130	358	586 (456 + 130)	130	210	356 (226 + 130)

Table 7. Message size of Trap.

Management property	Trap notification message (bytes)			Trap response message (bytes)		
	SNMP	XML-based	XML/SNMP gateway	SNMP	XML-based	XML/SNMP gateway
coldStart trap	110	450	560 (110 + 450)	0	210	210
linkDown trap	127	520	647 (127 + 520)	0	210	210

retrieving group values at once, the SNMP agent takes more time than the XML-based agent. The response time through the XML/SNMP gateway adds the gateway processing time to the SNMP communication. The response time of the Set request is almost the same as that of the Get request. These data do not show a significant difference in the response time between the SNMP agent and the XML-based agent. Therefore, the XML-based agent achieves a performance good enough for processing XML documents. Also, the processing overhead of the XML/SNMP gateway approach is not large compared with the direct communication between the manager and agent.

Table 8. Response time of Get request.

Management property	Response time (ms)		
	SNMP	XML-based	XML/SNMP gateway
sysDescr	40	50	80
system Group	160	140	250
interface Group	980	800	1250

As a result, the XML-based agent is small and efficient as the SNMP agent in terms of resource utilization and processing time. Also, the XML/SNMP gateway approach does not generate the extreme processing overhead in terms of response time. In addition, the XML-based agent gives outstanding network traffic reduction to access much information in a limited time.

The XNMS 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 as a management system and the XNMS acts as the manager of managers. According to the response time, the gateway efficiently processes the translation, and quickly delivers the message between the manager and agent.

VI. Concluding Remarks

In this paper, we presented a survey of work performed in the area of XML-based network management. We also proposed an XML-based integrated network management system and explained our gateway approach for managing the existing legacy SNMP agent that uses the advantages of XML technologies. For validation, we designed and implemented an XNMS for managing network devices based on the proposed manager and gateway architecture. Our XNMS fully utilizes XML technologies, such as XML Schema, DOM, XPath, XQuery, and XSLT, to perform network management. We were able to reduce the development cost of the management system through the support of the standard API for processing XML documents. We also developed an XML-based agent for achieving a pure XML-based network management. We verified the functionality of our XML-based agent by applying it to a commercial Internet sharing device.

The performance result showed almost the same performance and resource usage compared with the existing management paradigm, the SNMP agent. Moreover, the XML-based agent is more efficient than the SNMP agent in terms of network traffic. The XML/SNMP gateway approach for integrated network management generates more network traffic and processing time. The network traffic overhead of the XML/SNMP gateway is unavoidable. However, the processing overhead of the XML/SNMP gateway approach is not large compared with the direct communication between the manager and agent.

We are actually monitoring our campus gigabit network using our management system. We are monitoring Cisco backbone routers and switches. The network devices we are monitoring comprise about two-dozen devices. So far, we have not run into any problems. We plan to increase the number of devices to be managed and plan to report the result in our future work. We will also validate the scalability and extensibility of our systems. Finally, we plan to extend management operations to web services by using WSDL [10] and SOAP [11].

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. Strauss and T. Klie, "Towards XML Oriented Internet Management," *Proc. IFIP/IEEE Int'l Symp. on Integrated Network Management (IM 2003)*, Colorado Springs, USA, Mar. 2003, pp.505-518.

[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] W3C, "XSL Transformations Version 1.0," *W3C*

Recommendation, Nov. 1999.

[9] 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, June 1999.

[10] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and 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.

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

[12] 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 Management Symp.*, Jeju, Korea, Sept. 2002, pp. 507-508.

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

[14] OASIS Management Protocol TC, "Management Protocol Specification," http://xml.coverpages.org/managementProtocol.html.

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

[16] 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.

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

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

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

[20] J.H. Yoon, H.T. Ju, and J.W. Hong, "Development of SNMP-XML Translator and Gateway for XML-based Integrated Network Management," *Int'l J. of Network Management* (IJNM), vol. 13, no. 4, July-Aug. 2003, pp. 259-276.

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

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

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

[24] Y.J. Oh, H.T. Ju, M.J. Choi, and J.W. Hong, "Interaction Translation Methods for XML/SNMP Gateway," *Proc. DSOM 2002*, Montreal, Canada, Oct. 2002, pp. 54-65.

[25] Y.J. Oh, H.T. Ju, and J.W. Hong, "Interaction Translation Methods for XML/SNMP Gateway Using XML Technologies," *Proc. of the Asia-Pacific Network Operations and Management Symp.*, Jeju, Korea, Sept. 2002, pp. 11-22.

[26] 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," *Proc. IEEE/IFIP Network Operations and Management Symp.* (NOMS 2002), Florence, Italy, Apr. 2002, pp.1-14.

[27] OASIS, *Universal Description, Discovery and Integration (UDDI)*, http://www.uddi.org/.

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

[29] 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.

[30] WBEM, *WBEM Initiative*, http://www.dmtf.org/wbem/.

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

[32] DMTF, *Specification for the Representation of CIM in XML Version 2.0*, DMTF Specification, July 1999.

[33] DMTF, "Specification for CIM Operations over HTTP Version 1.0," DMTF Specification, Aug. 1999.

[34] SyncML Initiative, http://www.syncml.org/.

[35] M. Wasserman, *Concepts and Requirements for XML Network Configuration*, Internet-Draft, http://www.ietf.org/internet-drafts/draft-wasserman-xmlconf-req-00.txt, June 2002.

[36] 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.

[37] IETF, *Network Configuration (Netconf)*, http://www.ietf.org/html.charters/netconf-charter.html.

[38] ITU-T Recommendation X.3030, *Telecommunications Markup Language (tML) framework*, May 2000.

[39] W. K. Hong, "An ATM Network Management System for Point-to-Multipoint Reservation Service," *ETRI J.*, vol. 24, no. 4, Aug. 2002, pp.299-310.

[40] 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," *Proc. IEEE/IFIP Network Operations and Management Symp.* (NOMS 2000), Hawaii, USA, Apr. 2000, pp. 187-200.

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

[42] Devsphere, *XML Parsing Benchmark*, http://www.devsphere.com/xml/benchmark/index.html.

[43] Nazmul Idris, *Should I Use SAX or DOM*, http://developerlife.com/saxvsdom/default.htm, May 1999.

[44] Altova, *XML Spy*, http://www.xmlspy.com.

[45] W3C, "XQuery 1.0: An XML Query Language," *W3C Working Draft*, Apr. 2002.

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

[47] Ronald Bourret, *XML and Databases*, Sept. 1999, http://www.rpbourret.com/xml/XMLAndDatabases.htm/.

[48] D. Raggett, A. Le Hors, and I. Jacobs, "HTML 4.01 Specification," *IETF HTML WG*, http://www.w3.org/TR/html401, Dec. 1999.

[49] ZVON Org, *XPath Tutorial*, http://www.zvon.org/xxl/XPathTutorial/General/examples.html.

[50] Georg Gottlob, Christoph Koch, and Reinhard Pichler, "XPath Query Evaluation: Improving Time and Space Efficiency," *Proc. 19th Int'l Conf. on Data Eng.* (ICDE 2003), Bangalore, India, Mar. 2003.

[51] GNU Wget, http://www.wget.org/.

[52] Apache XML Project, http://xml.apache.org/.

[53] Apache-SSL, http://www.apache-ssl.org/.
[54] Apache XML Project, *Xerces Java Parser*, http://xml.apache.org/xerces-j/.
[55] Apache XML Project, *Xalan Java*, http://xml.apache.org/xalan-j/.
[56] Apache XML Project, *Xindice*, http://xml.apache.org/xindice/.
[57] Innovation, *HTTPClient Version 0.3-3*, http://www.innovation.ch/java/HTTPClient/.
[58] Motorola, *MPC850: PowerQUICC™ Integrated Comm. Processor*, http://e-www.motorola.com/webapp/sps/site/prod_summary.jsp?code=MPC850.
[59] Y.H. Hwang, "A Performance Analysis of TMN Systems Using Models of Networks of Queues, Jackson's Theorem and Simulation," *ETRI J.*, vol. 24, no. 5, Oct. 2002, pp. 381-390.
[60] D.H. Heo, "The Effects of Management Traffic on the Local Call Processing Performance of ATM Switches Using Queue Network Models and Jackson's Theorem," *ETRI J.*, vol. 25, no. 1, Feb. 2003, pp. 34-40.
[61] Ethereal, http://www.ethereal.com/.
[62] Y.J. Oh, *Interaction Translation Methods for XML/SNMP Gateway*, Master Thesis, POSTECH, Dec. 2002.

Mi-Jung Choi received her BS degree in computer science from Ewha Womans University in 1998, MS degree in computer science and engineering from Pohang University of Science and Technology (POSTECH) in 2000. Currently, she is a PhD candidate in the Department of Computer Science and Engineering, POSTECH. Her research interests include XML-based network management and policy-based network management. She is a member of IEEE and KNOM.

James W. Hong is an associate professor in the Dept. of Computer Science and Engineering, POSTECH, Pohang, Korea. He has been with POSTECH since May 1995. Prior to joining POSTECH, he was a research professor in the Dept. of Computer Science, University of Western Ontario, London, Canada. Dr. Hong received the BS and MS degrees from the University of Western Ontario in 1983 and 1985, respectively, and the PhD degree from the University of Waterloo, Waterloo, Canada in 1991. 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 few years, he has been working on various research projects on network and systems management, which utilize Web, Java, CORBA and XML technologies. He is IEEE Comsoc CNOM Vice Chair and KICS KNOM Chair. His research interests include network and systems management, traffic monitoring and analysis, and security management. He is a member of IEEE, KICS, KNOM and KISS.

Hong-Taek Ju received his BS degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1989, MS and PhD degree in computer science and engineering from POSTECH in 1991, 2002 respectively. From 1991 to 1997, he worked at Daewoo Telecom as a senior software engineer. Currently, he is a fulltime lecturer in College of Communication and Information, Keimyung University. His research interests include web-based network management, network monitoring and data synchronization over wireless mobile communication. He is a member of IEEE and KNOM.