XML-based Network Management

Mi-Jung Choi* and James W. Hong**

* School of Computer Science, University of Waterloo, Canada

** Dept. of Computer Science and Engineering, POSTECH, Korea

{mjchoi, jwkhong}@postech.ac.kr

Abstract

XML has been applied to numerous information processing application areas for various objectives and has earned excellent reputation for simplicity, extensibility, and openness. Researchers and developers in the network and system management area have begun to investigate the use of XML technologies in order to solve the problems that exist in Internet management standards such as SNMP. However, the methods on how to apply XML technologies to network management are not yet clearly defined. This chapter attempts to provide a guideline for applying XML technologies to developing XML-based management systems and to provide an architectural framework for XML-based network management. We first examine XML technologies that can be applied to network management and how they can be applied to common network management tasks. We then present a detailed architecture of the XML-based management that can be used in developing management systems. We have validated this architecture by implementing it and using it in managing network devices and systems.

1. What is XML-based Network Management?

In this section, we first explain the definition of XML-based network management (XNM) and illustrate merits and demerits of XNM. We also introduce previous research effort on XNM as a related work.

1.1. Definition of XNM

XML [1] is an extensible markup language, with which one can define own set of tags for describing data. It is also regarded as a meta-language, which describes information about data. XML enables data to carry its meta information in the form of an XML document. The meta information is used for searching, filtering, and processing efficiently. The main benefits of XML are simplicity, extensibility, and openness. The simplicity of XML relies on tree structured text-based encoding of data, so one can easily create, read and modify XML documents with even the most primitive text processing tools. Although simple, XML is extensible enough to express complex data structures. The extensibility of XML allows different applications to represent their own structures of data.

Because XML is an open standard, there is a wide selection of both freely available and commercial tools for handling and processing it. By supplementing XML with several advanced features such as

XML Schema [2], XSL [3], DOM [4], XPath [5], XQuery [6], and SOAP [7], XML technology is not only emerging as the language of choice, but also as the universal information-processing platform.

XNM is a kind of network management methods that applies XML technologies to network management. Therefore, XNM achieves the advantages of XML technologies and one can easily develop network management system using XML technologies. In the next subsection, advantages and disadvantages of XNM are examined.

1.2. Merits and Demerits of XNM

The application area of XML is very broad: data presentation on the Internet, multimedia presentation, data saving of specific programs, electronic commerce, academic uses, and etc. In all these areas, XML proves itself to be an excellent solution to solve technical challenges. Using XML in network management presents the following advantages [8]:

- The XML Schema defines the structure of management information in a flexible manner.
- Widely deployed protocols such as HTTP can reliably transfer management data.
- DOM APIs easily access and manipulate management data from applications.
- XPath expressions efficiently address the objects within management data documents.
- XSL processes management data easily and generates HTML documents for a variety of user interface views.
- Web Service Description Language (WSDL) [9] and SOAP [7] define web services for powerful high-level management operations.

However, the current XML-based network management has some weaknesses as well. The 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. Also the previous XML-based network management does not properly provide a general architecture of XML-based network management system (XNMS) from the aspect of manager and agent. Network bandwidth of transferring XML data is large because XML is text-based. The processing of XML is considered heavy, so it is not yet proven that XML is applicable to embedded systems. Many are curious whether the XML-based manager can process information delivered from multiple managed devices. The development experience of an XNMS is insufficient.

Integration efforts of XML-based network management with existing SNMP agents are the first step towards the integrated network management. The specification translation from SNMP SMI to XML DTD or XML Schema is widely in progress. The XML/SNMP gateway approach between an XML-based manager and SNMP agents is also considered to manage the legacy SNMP agents. Therefore, to provide an integrated network management of the XML manager and existing SNMP agents, the architecture and implementation of the gateway need to be provided.

1.3. Related Work

In the past several years, some research and development has been done in the area of XML-based network management. Much work has been done in the area of accommodating widely deployed SNMP devices, specifically in the specification and interaction translation for the gateways between XML-based managers and SNMP-based devices [10, 11, 18, 27]. Jens Müller implemented an SNMP/XML gateway as a Java Servlet that allows fetching of such XML documents on the fly through HTTP [12]. Avaya Labs has developed an XML-based management interface for SNMP enabled devices [13]. Y. J. Oh et. al, developed several interaction translation methods between an XML/SNMP gateway and an XML-based manager, based on the previously developed specification translation algorithm [11, 27].

Some work focused on extending Web-based management to exploit the advantages of XML technologies [14, 15, 16, 17, 18, 32]. Web-based Enterprise Management (WBEM) uses XML only for object and operation encoding, and is currently being updated to include emerging standards, such as SOAP [15]. Web-based Integrated Network Management Architecture (WIMA) showed that XML is especially convenient for distributed and integrated management, for dealing with multiple information models and for supporting high-level semantics [16]. J. H. Taek et. al, extended the use of embedded Web server for element management to Web-based network management architecture platform by adding XML functionalities [17]. L. R. Menten also showed his experience in application of XML technologies for device management ranging from small resource-limited data networking to high capacity server-hosted control plane monitoring systems [18]. He proposed an XML-based architecture of network management system and described the DOM-based management transactions to cover HTTP/SOAP client, SNMP manager, script client and CLI client. V. Cridlig et. al, proposed an integrated security framework for XML-based network management [32]. This framework is based on role-based access control, where requests are issued on behalf of a role, not of an user. M. H. Sqalli et. al, presented load balancing approaches to XML-based network management, to distribute the load across multiple parallel Java parallel virtual machine (JPVM) gateways [14].

While other work focused specifically on configuration management of network devices and systems [14, 19, 20, 21, 33, 34]. Juniper networks [19] and Cisco [20] manage the configuration information of their network devices equipped with their own XML-based agents. The Network Configuration (Netconf) [21] Working Group (WG) is chartered to produce a protocol suitable for configuration management of network devices. The Netconf protocol uses XML for data encoding and a simple RPC-based mechanism to facilitate communication between a manager and an agent. YencaP is a Python-based Netconf implementation by V. Cridlig et. al [33]. YencaP provides a Netconf management suite consisted of Netconf agent and manager. S. M. Yoo et. al, proposed a mechanism to manage configuration information more effectively using Netconf protocol and Web services technologies [34]. Using Universal Description, Discovery and Integration (UDDI) [35] helps a Netconf manager to quickly and intelligently recognize required parameters of network devices to operate configuration tasks.

1.4. Chapter Organization

Even though potential functionalities and advantages of XML are well known for various Internet applications, XML is not the favorable choice to use in the network management area yet. The lack of a clear guideline on how to apply XML technologies to network management is blamed for insignificant portions of XML usage. This chapter first reviews XML technologies that can be used in developing XML-based management systems, and then discusses on various methods to show the benefits of applying XML related technologies to network management. In order for the reader to easily understand the applicability, we have divided network management into the following areas: information modeling, instrumentation, management protocol, analysis and presentation. This chapter describes how XML technologies can be applied to each area.

That is, we present an architectural framework for the development of XML-based network management systems. The framework provides a detailed architecture for an XML-based manager, agent, and gateway. It also provides methods for interacting between a manager and agent. This framework can be used as a guideline for developing XML-based management systems. We have validated this architecture by implementing and using it in managing network devices and systems.

The organization of the chapter is as follows. Section 2 presents an overview of relevant XML technologies. Section 3 discusses methods for applying XML technologies to network management tasks. Section 4 presents our architectural framework for developing XML-based network management systems based on the agent, manager, and gateway. Section 5 describes our implementation experiences of XNMS. Finally, we summarize this chapter in Section 6.

2. XML Technologies

In this section, a set of XML related technologies that can be applied to developing network management systems is briefly introduced for better understanding of XML. The eXtensible Markup Language (XML) [1] is the World Wide Web Consortium (W3C) standard based on SGML. XML uses a text-based syntax that can be recognized by both computers and human beings and offers data portability and reusability across different platforms and devices. XML is rapidly becoming the strategic instrument for defining corporate data across a number of application domains. The properties of XML markup make XML suitable for representing data, concepts, and contexts in an open, platform-, vendor-, and language-neutral manner. The technology map of XML is depicted in Figure 1.

There are two fundamental approaches to define XML document structure: Document Type Definition (DTD) and XML Schemas [2]. Either approach is sufficient for most documents, yet each one offers unique benefits. DTD is in widespread use and shares extensive tool support while XML Schemas provide powerful advanced features such as open content models, namespace integration, and rich data typing.

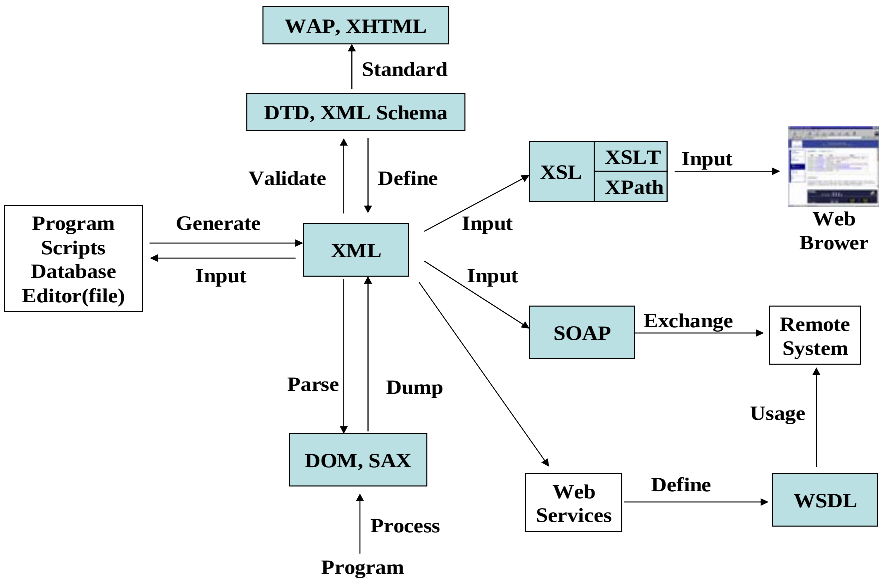

Figure 1. XML Technology Map

The Document Object Model (DOM) [4] specifies the means by which we can access and manipulate XML documents. Without DOM, XML is no more than a storage system for data that can be accessed by various proprietary methods. With DOM, XML begins to approach its promise as a truly universal, cross-platform, application-independent programming language. DOM allows reconstruction of the complete document from the input XML document, and also allows access to any part of the document. The Simple API for XML (SAX) [22] is an event-driven and serial-access mechanism for accessing XML documents. While a DOM parser parses the XML document and creates a DOM tree, keeping the entire structure 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 the other methods with heavy system overhead. However, it does not provide the hierarchical information that a DOM parser provides. A SAX parser generates events such as the start of an element and the end of an element, while accessing the XML document. By capturing these events, applications can process operations, such as gaining the name and attributes of the element.

The Extensible Stylesheet Language (XSL) [3] is a mark-up language to display XML documents on the Web. XML documents describe only the contents and their structure. An XSL style sheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document using a formatting vocabulary. XSL enables XML to separate the contents from the presentation. XSL consists of two parts: a language to transform XML documents, and XML vocabularies to specify formatting semantics. The style sheet technology to transform documents is the XSL Transformation (XSLT) [23], which is a subset of XSL technologies 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 XML documents without formatting semantics.

The XML Path Language (XPath) [5] is an expression language that is used to address parts of an XML document. The syntax used by XPath is designed for use in URIs and XML attribute values, which requires it to be very concise. The name of XPath is based on the idea of using a path notation to address an XML document. XPath operates under the assumption that a document has been parsed into a tree of nodes. XPath is a language to identify particular sections of an XML document. XPath has a compact, non-XML syntax to be used within URIs and XML attribute values, and operates on the abstract, logical structure of an XML document. Each node in the XML document is indicated by its position, type, and content using XPath.

The XML Query Language (XQuery) [6], a query language for XML, is designed to be broadly applicable to 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 [24] is an update language, which provides open and flexible update facilities to insert, update, and delete data in XML documents. The XUpdate language is expressed as a well-formed XML document, and uses XPath for selecting elements and conditional processing.

The SOAP [7] is a lightweight protocol for exchanging information in a distributed environment. It is an XML-based protocol that consists of three parts: an envelope that defines a framework for describing the content of a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and 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.

The Web Services Description Language (WSDL) [9] is an XML-based language used to define Web Services [25] and describe how to access them. A WSDL document is a collection of one or more service definitions. The document contains a root XML element named definitions, which contains the service definitions. The definition element can also contain an optional targetNamespace attribute, which specifies the URI associated with the service definitions. WSDL defines services as collections of network endpoints which are defined as ports in WSDL. WSDL separates the service ports and their associated messages from the network protocol binding. The combination of a binding and a network address results in a port, and a service is defined as a collection of those ports.

3. Applicability of XML Technologies to Management Tasks

In this section, we examine how XML technologies can be applied to network management. Network management involves the following essential tasks: modeling management information, instrumenting management information in managed resources, communicating between manager and agent, analyzing the collected data, and presenting the analysis results to users.

Network management systems consist of manager and agent systems. They perform various management tasks to process management information. Agent's tasks involve accessing attributes of managed objects, event reporting, and processing the management requests. Under normal conditions, agent's tasks are so simple that the management overhead of the agent can be negligible. On the other hand, manager systems perform complex management tasks to satisfy management objectives while they depend on management applications. The basic management tasks in a management system are depicted in Figure 2.

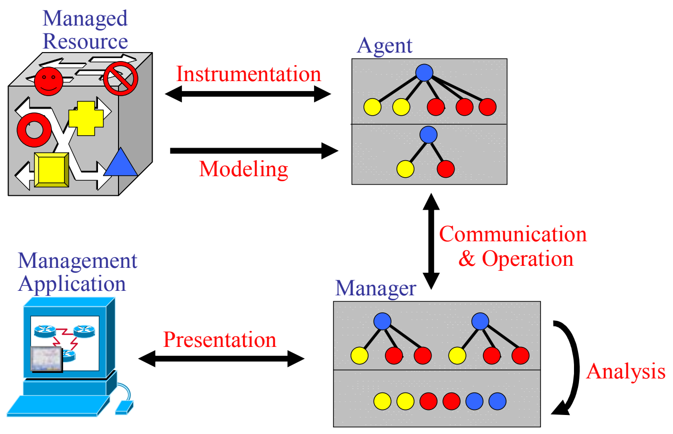

Figure 2. Basic Management Tasks in a Management System

3.1. Information Modeling

XML is a meta-language, a language to describe other languages, that enables applications such as network and system management to define its own tags. XML allows the user to define a customized markup language specific to an application. The expressive power of XML makes it possible to represent a wide range of information models presented in existing management standards and deployed solutions. The XML DTDs and Schemas define the data structure of XML documents. The XML Schemas is devised to overcome the limits of DTDs that cannot define a new data type. It is also flexible and extensible, allowing new tags to be added without changing the existing document structure. The XML Schema supports the presentation of management information by adding new data types to a variety of data types (44 kinds of basic types), and defining the document structure of management information. Therefore, the use of XML Schemas maximizes XML's key advantages in the management information model over other alternatives.

Compared with SNMP SMI and WBEM CIM, the XML Schema has many advantages in management information modeling. The XML Schema is easy to learn. Various powerful and convenient XML editor tools are available for developers. Another major advantage is that XML can be utilized for various purposes, including validation. By use of XML authoring tools, a developer can generate sample XML data, a database schema and other forms of data structure based on the XML Schema. The disadvantage of XML Schema for management information modeling is that there is no single widely adopted standard model. However, it is convenient to translate from the existing management information models such as SMI, CIM, GDMO to the XML Schemas. Research in this area is widely being conducted [10, 11, 12, 13].

3.2. Instrumentation

Tasks of management information instrumentation include operations to guarantee consistency between managed objects and managed resources in an agent system. The update operation from the state of managed resources to the variable of managed objects should be executed if the state of managed resources is changed. The mapping operation from the value of managed objects to the state of managed resource should be performed if a manager system modifies the variable of managed objects by management operations. The managed resources might be a hardware which has control or state register as interface mechanism, or execution program which has program interface as interface mechanism. They are implemented without aid from XML technologies. However, managed objects can rely on XML technologies. This means that the managed objects are in the form of XML document.

Management information instrumentation can be viewed as a group of processes for XML document interpretation, creation, and modification. When it comes to mapping, it is possible to access the managed resource after the interpretation of the XML documents, the variables of managed object. Also for the update, the state of managed resource is described in managed object with the modifications to XML documents. There are two standard XML document interpretation technologies: DOM and SAX. They make it easy to write programs that interpret, create, and modify XML documents.

Most network devices provide multiple management interface mechanisms, CLI, SNMP, Web, and etc. The management interfaces have been implemented separately, resulting no common interface or data format between managed objects and managed resources. XML can be used as middleware between managed objects and managed resources. By introducing a common interface and data format, the development cost can be reduced, and the consistency issue from multiple accesses to a single managed resource can be resolved.

3.3. Management Protocol

A management protocol must deal with the specifications of management operations and transport protocol for the exchange of management information. It should also define the syntax and semantics for the protocol data unit. With respect to management protocols, XML-based network management follows the model of transferring data over HTTP. Further, it uses XML as management information encoding syntax. This means management data is transferred over the HTTP payload in the form of an XML document. The management information in XML document format is distributed through HTTP over TCP. In contrast to UDP, TCP makes it possible to transmit reliable management data and large application-messages without limits to message size. A large amount of management data can be transferred between the manager and the agent such as an entire table. The XML data through HTTP can be compressed by setting some options in the HTTP header. By compressing data, a significant amount of network overhead can be reduced while keeping almost the same latency at the end-host for both compressed and decompressed data. The network overhead can also be minimized by sending a single message containing multiple sets of management information and by compressing messages with the HTTP header option. The drawback in using this method is the increase in latency from having to establish a TCP connection and perform compression and decompression.

Another important issue regarding the management protocol is in addressing managed objects. When a manager requests management information, it must specify a unique name of the managed object to be retrieved. XML-based network management uses the XPath standard for addressing managed objects. This solution offers the following advantages. First, XPath is a standard Web technology for addressing parts of an XML document. It is already used in conjunction with the XSLT [22] to identify pieces of an XML document targeted for transformations. XPath is also widely supported. Second, the manager can effectively send a query to the managed objects of the agent. XPath expressions are composed of element names, attributes, and built-in functions. Given the hierarchical nature of XML, one of the simplest expressions is to follow the tree structure to point at a particular element.

A management protocol defines the management operations. Management operations consist of creation, deletion, retrieval, modification, filtering, scoping, notification, and etc. These high-level management operations can be defined through WSDL and called via SOAP. For example, row creation and deletion in SNMP through Row Status objects can be considerately complicated. However, these tasks can be achieved by convenient higher-level operations in XML-based network management.

3.4. Analysis

The DOM and SAX APIs can be used to access management data for applications. Most XML parsers implement these standard APIs to access the contents of XML documents. With assist from DOM, one can access any element of the XML document, and modify its content and structure from XML input parameters. One can perform such management operations as analyzing statistical traffic data and filtering important events among the notifications by manipulating the XML document using the standard DOM interface. It is easy to extract the necessary management data and analyze them. Therefore, the management information can be analyzed using the DOM API. Items within management data documents can be addressed via XPath expressions.

Because XML data can be easily stored to and accessed from databases (DB) with the support of various tools, DB processing in network management functionality can be easily achieved. Data presented in the DOM tree can be stored to DB and simply transferred to applications. This method reduces the development cost and time, as well as the program overhead. Moreover, there are plenty of tools available for XML, such as XML parser, DOM, XPath, SAX, etc.

3.5. Presentation

After analyzing XML data, the next step is to present the analysis results to user. XML, unlike HTML, separates the contents of the document from the display. XML data which is validated by XML DTDs or XML Schemas is transformed to HTML or another XML document through XSL and XSLT. Using the software supporting the transformation from XML to HTML or other display format makes it possible to provide a Web-based management user interface (Web-MUI). Therefore, if management information is expressed in XML format, Web-MUI can be easily generated from the XML document.

4. XML-based Network Management Architecture

In this section, we present in-depth details of an XML-based network management architecture that can be used to develop a network or systems management system [26]. The architecture consists of an XML-based agent, an XML-based manager, and an XML/SNMP gateway.

4.1. Architecture of XML-based Agent

In this section, we first present the architecture of an XML-based agent and describe its components. The XML-based agent needs to define the management information then retrieve the management information from the managed resources, and transfer to the XML-based manager. In this section, we also explain the management tasks of the XML-based agent from aspects of information modeling, instrumentation, and management protocol.

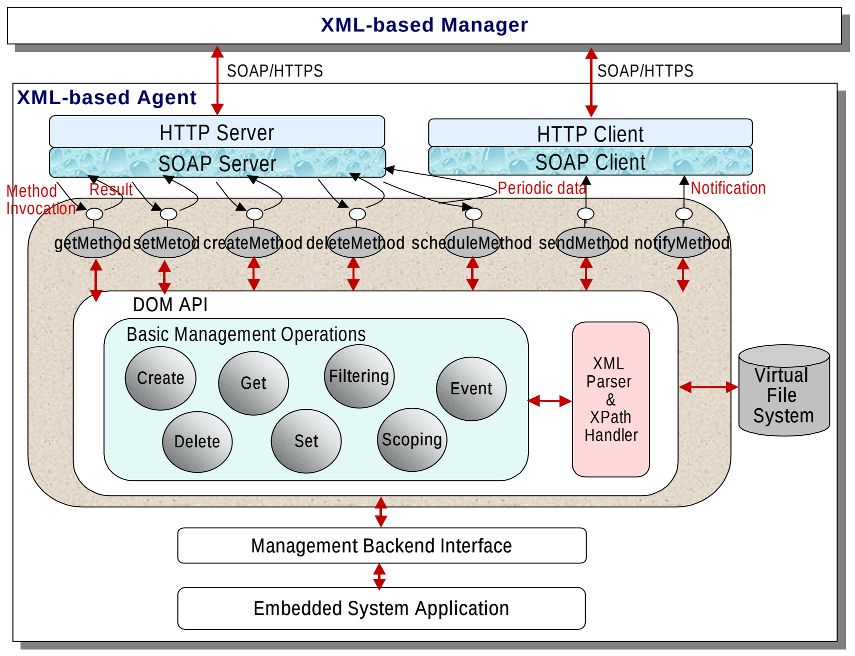

Figure 3. Architecture of XML-based Agent

Figure 3 illustrates the architecture of an XML-based agent. The XML-based agent includes an HTTP Server and SOAP Server as basic components. Other components are SOAP RPC operations, which contain DOM Interfaces performing basic management operations. The basic operations use the XML Parser and XML Path Language (XPath) Handler. The XML Parser module allows an agent to parse and access the contents of the XML message using an XPath expression. Also, the agent needs to send notifications to the manager, and the manager needs to receive and handle the notifications. To send notifications to the manager, the XML-based agent needs a SOAP Client module and an HTTP Client module. Consequently, the manager needs a SOAP Server module and an HTTP Server module.

The XML Parser parses an XML document, selects the specified node, and reads management data. In order to send up-to-date information, the XML-based agent gathers information from the Management Backend Interface. The XML Parser updates the selected node value through the Management Backend Interface before replying to the manager. Virtual File System stores management information in an XML format. The DOM API is used to process management information and instrument this information from real device resources through the Management Backend Interface.

The XML-based agent needs to define management information, retrieve the management information from the managed resources, and transfer the management information to the XML-based manager. In this section, we explain the management tasks of the XML-based agent on aspects of information modeling, instrumentation, and management protocol.

4.1.1. Information Modeling in Agent

We define management information using the XML Schema. An example of management information is SNMP MIB II that can be applied to all network devices. Each node of SNMP MIB converts into an element of the XML Schema: the name of object into XML tag, 'syntax' into the data type definition, and 'access' into the attribute for example. Figure 4 shows an example of an XML Schema in system group of MIB II, particularly the 'sysDescr' object of system group.

```
<xsd:element name="system">
   <xsd:complexType>
      <xsd:all>
         <xsd:element ref="sysDescr" minOccurs="0"/>
         <xsd:element ref="sysObjectID" minOccurs="0"/>
         <xsd:element ref="sysUpTime" minOccurs="0"/>
         <xsd:element ref="sysContact" minOccurs="0"/>
         <xsd:element ref="sysName" minOccurs="0"/>
         <xsd:element ref="sysLocation" minOccurs="0"/>
         <xsd:element ref="sysServices" minOccurs="0"/>
      </xsd:all>
   </xsd:complexType>
</xsd:element>
<xsd:element name="sysDescr">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:restriction base="DisplayString_0_255">
            <xsd:attribute name="access" type="xsd:string" use="fixed" value="read-only"/>
         </xsd:restriction>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

Figure 4. XML Schema of MIB II – System Group

4.1.2. Instrumentation in Agent

A DOM tree is a virtual repository of management data and provides a manipulation point to managed objects. A DOM parser provides a random access using an element name or an XPath expression and various manipulations of an XML document. The attributes of the DOM interface, such as Node and Text, are mapped to managed objects. When the XML-based agent receives a request message, the SOAP operation module selects specified nodes in the DOM tree using the XPath Handler. For the selected nodes, the agent retrieves management data from the DOM tree through the DOM interface, and sends the data to the XML-based manager. In order to send up-to-date information, the DOM tree updates the node value for the selected node through the Management Backend Interface before replying to the agent. This type of update of the DOM node value from real resources is called a pull-based update. The DOM tree requests a value to the embedded system application through the Management Backend Interface and receives a response.

For certain nodes, it is not necessary to update the DOM node value before replying because this value is up-to-date. In this case, the Management Backend Interface module is responsible for the update. Periodically, or when events occur, the Management Backend Interface updates the DOM tree node value. This type of update is called a push-based update. For frequently changing data, such as traffic measurement data, the pull-based update is more appropriate than the push-based update, and static or semi-dynamic information can benefit from using a push-based update rather than a pull-based one. In case of pull-based update, the DOM node is updated by replacing the text of the node value with the Processing Instruction (PI) node - a standard node type of DOM.

4.1.3. Management Protocol in Agent

The management protocol defines transport protocol, management operation and protocol data unit. First, the XML-based agent transfers XML data via HTTP. Next, the XML-based agent defines management operations and management data using SOAP. At the basic functionality level, SOAP can be used as a simple messaging protocol and can also be extended to an RPC protocol. The XML-based agent exchanges an XML-encoded message with an XML-based manager. This means that the request message from the manager and the response message from the agent are all formatted as an XML document. SOAP provides a standard method to transfer XML-encoded messages over HTTP between the manager and agent.

Management operations include managed object creation, deletion, retrieval, modification, filtering, scoping, notification, etc. These management operations can be defined using SOAP RPC operations, such as getMethod, setMethod, createMethod, deleteMethod, sendMethod, and modifyMethod. Brief explanations of these operations are as follows.

- *getMethod*: used to retrieve information from the management information XML file using the Get operation in DOM Interface.
- *setMethod*: used when the contents of the management information are modified without changing

the information structure. This uses the Set operation in the DOM Interface.

- *createMethod*: When a new management information is added, this method is invoked. This calls the Create operation in DOM Interface.
- *deleteMethod*: deletes managed objects and changes the structure of management information.
- *sendMethod*: The agent calls this method to send the periodic monitoring data to the manager.
- *notifyMethod*: The agent sends notifications to the manager via this method.

These operations can also be defined using WSDL. As described in Table 1, WSDL defines the main elements as follows:

- message: An abstract, typed definition of the data being communicated
- operation: An abstract description of an action supported by the service
- portType: An abstract set of operations supported by one or more endpoints
- binding: A concrete protocol and data format specification for a particular port type
- port: A single endpoint defined as a combination of a binding and a network address

(a) WSDL Elements

Definitions	**Name**	**Remarks**	
Message	getMethod	parameter sourceFile (type:string) parameter xpath (type:string)	
Message	getResponse	parameter getResponse (type:string)	
PortType	SoapInterface	operation get	input message getMethod
			output message getResponse
Binding	SoapInterfaceSoapBinding	operation get	input getMethod
			output getResponse
Service	SoapInterfaceService	port SoapInterface	address location http://hostname:8080/axis/SoapInterface.jws

(b) SOAP Request Message: getMethod

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xconf="http://tempuri.org">
<SOAP-ENV:Body id="1"   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <xconf:getMethod>
    <SourceFile>RFC1213-MIB.xml</sourceFile>
    <xpath>//sysDescr</xpath>
  </xconf:getMothod>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(c) SOAP Response Message: getMethod

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xconf="http://tempuri.org">
<SOAP-ENV: Body id="1"   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <xconf:getResponse>
    <getResponse><sysDescr access="read-write">IBM RISC System/6000 Machine Type: 0x0401 Processor id: 000106335900
     The Base Operating System AIX version: 03.02.0000.0000 TCPIP Applications version: 03.02.0000.0000</sysDescr>
    </getResponse>
  </xconf:getResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Table 1. WSDL Definition of getMethod Operation

The WSDL elements in Table 1 (a) describe the getMethod operations published by the XML-based agent as a service named SoapInterfaceService. Table 1 (b) describes a request message binding SOAP to call getMethod operation in the XML-based agent. Table 1 (c) is the response message to the request (b). The XML-based agent receives the getMethod message, it calls the SoapInterfaceService whose location is 'http://hostname:8080/axis/SoapInterface.jws'. This operation includes following parameters: 'sourceFile', and 'xpath'. The parameter 'sourceFile' is a source file which contains the management information. The parameter 'xpath' is an XPath expression to access the specific part of the management information to retrieve it.

4.2. Architecture of XML-based Manager

The XML-based manager needs to define management information added to the management information of the XML-based agent, retrieve the management information from the agent, then analyze the retrieved information, and present to the user. In this section, we explain the management tasks of the XML-based manager on the aspects of information modeling, management protocol, analysis, and presentation.

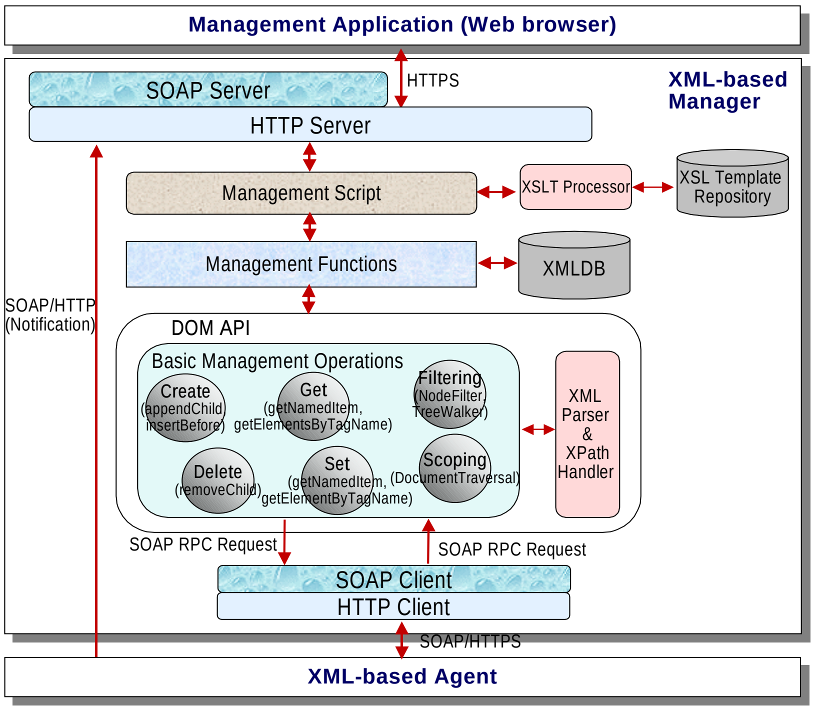
Management Application (Web browser)
HTTPS
XML-based Manager
SOAP Server
HTTP Server
Management Script
XSLT Processor
XSL Template Repository
Management Functions
XMLDB
SOAP/HTTP (Notification)
DOM API
Basic Management Operations
Create (appendChild, insertBefore)
Get (getNamedItem, getElementsByTagName)
Filtering (NodeFilter, TreeWalker)
Delete (removeChild)
Set (getNamedItem, getElementByTagName)
Scoping (DocumentTraversal)
XML Parser & XPath Handler
SOAP RPC Request
SOAP RPC Request
SOAP Client
HTTP Client
SOAP/HTTPS
XML-based Agent

Figure 5. Architecture of XML-based Manager

Figure 5 illustrates the architecture of an XML-based manager. The manager includes HTTP Server and Client, SOAP Server and Client, Management Script, Management Functions Module, DOM Interface Module, XSLT Processor, XMLDB, and XSLT Template Repository. The HTTP and SOAP Server/Client modules are necessary to communicate with XML-based agents and to perform management operations. The DOM API is necessary to analyze and process the management information, the XMLDB module is used to store long-term analysis data such as monitoring data. The XSLT processor combines XML document with Extensible Stylesheet Language (XSL) file and generates HTML document for the presentation.

An HTTP Server is used to provide administrators with a Web-MUI and for receiving requests from the management application and passing them to the Management Functions through the Management Script. The SOAP Server and HTTP Server are used to receive asynchronous messages for notifications from the devices via SOAP/HTTP. The SOAP Client and HTTP Client play a role in the interface module of the device and exchanges synchronous management information with the agent. The Management Functions such as object management, state management, event reporting, log control, alarm reporting, and etc. can be divided into several basic management operations such as creation, deletion, get, set, filtering, and scoping. The Management Functions module uses the DOM interface to implement the management application functions because management information is represented in XML data.

4.2.1. Information Modeling in Manager

XML Schema can be used to define management information. Figure 6 is drawn using an editing tool called XML Spy. It shows the management information of XNMS defined in an XML Schema format. The XML Schema presented in dotted lines in Figure 6 represents optional management information and the "0..∞" expression under the elements means that the number of elements ranges from zero to infinity. The basic cardinality is one when nothing is specified. A dotted box with no cardinality specification means an optional element, 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 server configurations, administrator lists, XML/SNMP gateway lists, and device topology information.

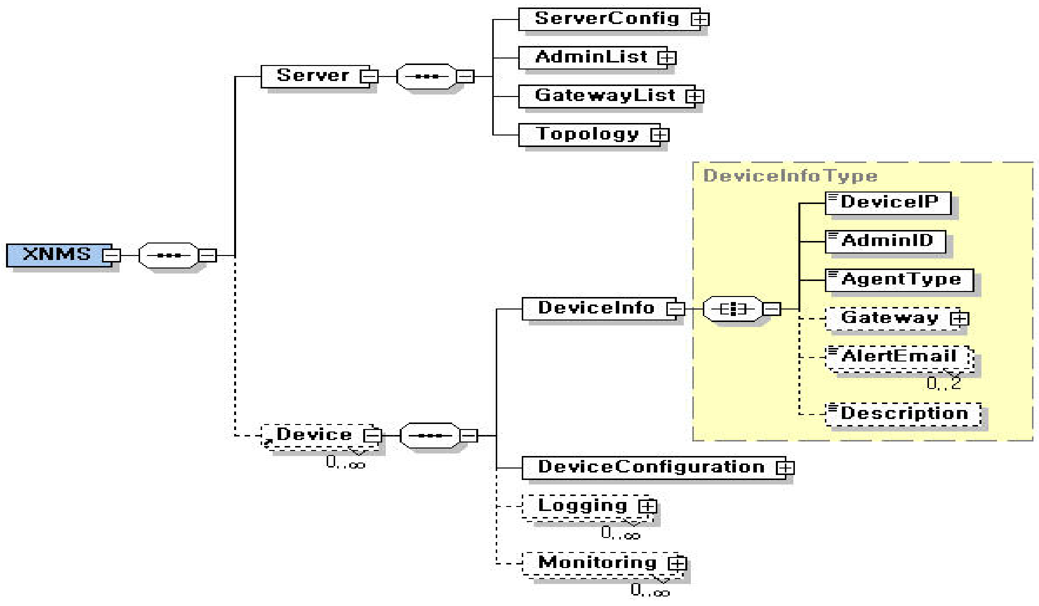

Figure 6. Management Information Model of XML-based Manager

The device part consists of device information, device configurations, logging, and monitoring. The

XML Schema has complex types containing elements and attributes and simple types containing none. XML Schema supports 44 kinds of built-in simple types, including string, integer, float, time, date, and 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 search. 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 as described in Section 4.1.1. This is for consistency of the 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, our MIB2XML translator [11] converts the private MIB definition to XML Schema and the translated XML Schema is added to XML Schema of XML-based manager as a child element of DeviceConfiguration in Figure 6.

4.2.2. Management Protocol in Manager

SOAP is a protocol to exchange XML-based messages over HTTP or SMTP. At the basic functionality level, SOAP can be used as a simple messaging protocol and can also be extended to an RPC protocol. The XML-based manager exchanges an XML-encoded message with communicating peers, such as an XML-based agent and an XML/SNMP gateway. This means that the request message from the manager and the response message from the gateway are all formatted as an XML document. SOAP provides a standard method to transfer XML-encoded messages over HTTP between the manager and the gateway and between the manager and agent.

We define several SOAP RPC messages between an XML-based manager and an XML/SNMP gateway and between an XML-based manager and an XML-based agent using WSDL as shown in Section 5.1.3. WSDL is an XML-based language to define Web Services and describe how to access them. SOAP is becoming the de facto standard protocol for Web Services. Therefore, we apply the WSDL Schema for service description instead of a proprietary XML structure using the XML Schema. The manager finds the definition of a method to invoke using this WSDL document.

4.2.3. Analysis in Manager

The Management Functions such as object management, state management, event reporting, log control, alarm reporting, and etc. can be divided into several basic management operations such as creation, deletion, get, set, filtering, and scoping. The Management Functions use the DOM interface to implement the management application functions because management information is represented in XML data.

The DOM Core defines 4 types, 20 fundamental interfaces, and 6 extended interfaces. In this section, we merely explain some fundamental interfaces to understand the following APIs for basic operations, such as creation, deletion, retrieval, and modification of objects. The DOM level 3 Core Specification includes interfaces such as Document, Node, Element, NamedNodeMap, and NodeList. Table 2 shows DOM interfaces corresponding to the management operations and functions.

Operations	**DOM Interfaces**
Creation	**interface Document : Node** • Element createElement(in DOMString tagName) → Creates an element of the type specified. Note that the instance returned implements the *Element* interface so attributes can be specified directly on the returned object. **interface Node** • Node appendChild(in Node newChild) raises(DOMException) → Adds the node *newChild* to the end of the list of children of this node. If the *newChild* is already in the tree, it is first removed. • Node insertBefore(in Node newChild, in Node refChild) raises(DOMException) → Inserts the node *newChild* before the existing child node *refChild.* If *refChild* is null, insert *newChild* at the end of the list of children.
Deletion	**interface Node** • Node removeChild(Node oldChild) raises(DOMException) → Removes the child node indicated by *oldChild* from the list of children, and returns it.
Navigation/ Retrieval	**interface Node** • readonly attribute Node parentNode • readonly attribute Node firstChild • readonly attribute Node lastChild • readonly attribute Node previousSibling • readonly attribute Node nextSibling • readonly attribute NodeList childNodes **interface NamedNodeMap** • Node getNamedItem(in DOMString name) → Retrieves a node specified by name. interface Document • NodeList getElementsByTagName(in DOMString tagname) → Returns a *NodeList* of all the Elements with a given tag name in the order in which they are encountered in a preorder traversal of the *Document* tree.
Setting values/ modification	**interface Node** • attribute DOMString nodeValue → The value of this node, depending on its type; When it is defined to be null, setting it has no effect. **interface Element : Node** • void setAttribute(in DOMString name, in DOMString value) raises(DOMException) → Adds a new attribute. If an attribute with that name is already present in the element, its value is changed to be that of the value parameter. This value is a simple string.

Table 2. DOM Interface for Management Operations

In addition, the DOM level 2 Traversal and Range Specification defines the framework for filtering and navigating in the filtered tree. It basically defines 4 interfaces: TreeWalker, NodeIterator, NodeFilter, and DocumentTraversal interfaces. Filtering conditions are defined in the concrete class implementing NodeFilter interface and DocumentTraversal returns traversal object such as TreeWalker or NodeIterator representing filtered-out nodes. This basic operation provides easy-to-use, robust, selective traversal of document's contents. Moreover, these DocumentTraversal API such as createNodeIterator() and createTreeWalker() and some other APIs of DOM core API provides scoping functionality basically with the root node of the containment tree.

We can analyze management data of XML document format using the DOM Interface. As mentioned in Section 3.4, we use native XML DB to store management information. This reduces the manipulation of the 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.

4.2.4. Presentation in Manager

After the management information is analyzed, the analysis result is presented to administrators. We adopted XSLT to accomplish the XML-based manager presentation. XSLT is an XML-based language that can be used 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 the XML-based manager. Static information such as 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.

The XML-based manager 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 the XML-based manager whenever the MIB is translated. Table objects defined in the MIB is presented as HTML table view using the template. The XML-based manager 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.

4.3. Architecture of XML/SNMP Gateway

An XML/SNMP gateway provides a method to manage networks equipped with SNMP agents by an XBM manager. The XBM manager transfers XML messages using SOAP/HTTP, and the SNMP agent passes SNMP messages using SNMP. The gateway converts and relays management data between the two management applications. In this section, we describe management tasks of an XML/SNMP Gateway from the specification translation and interaction translation perspectives.

Figure 7 illustrates the architecture of an XML/SNMP gateway, that is, a SOAP-based architecture of XBM manager and the gateway. In this architecture, the SOAP Client in the manager generates an XML-encoded SOAP request, such as get and set. Then the HTTP Client sends the HTTP POST request including the SOAP request in its body to the HTTP Server in the gateway. The SOAP Server parses the HTTP message into a properly formatted RPC call and invokes an appropriate method published by the gateway. This SOAP Server receives the result of the method and generates a well-formed SOAP response message. The response message backtracks to the SOAP Client in the manager. Finally, the manager application receives the result of the method invocation. Whenever the Trap Handler 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 SOAP Client in the gateway sends an asynchronous event message defined in the XML Trap Schema to the SOAP Server in the manager.

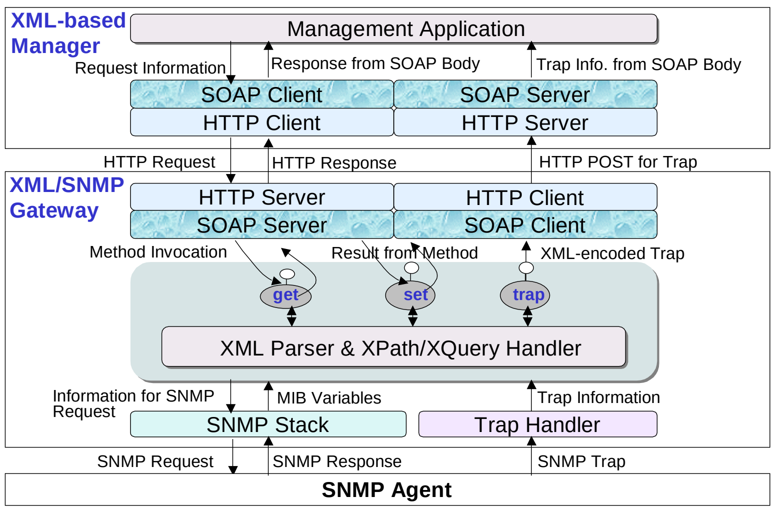

Figure 7. SOAP-based Architecture of Manager and Gateway

4.3.1. Specification Translation

To interact between the XML-based manager and the SNMP agent, the XML/SNMP gateway first translates management information of SNMP SMI to XML Schema. In this section, we describe an SNMP MIB to XML Schema translation algorithm [11] for specification translation of the gateway.

SNMP SMI	XML Schema	DOM Interface
MIB Module	XML document	Document
MIB Module name	Root Element name	Element::tagName
Leaf Node (macro definition)	element with child text node(s)	Element
Node name	element name	Element::tagName
Clauses of MIB node	Attributes of an element	Attr
Object Identifier (OID)	Attribute of type "ID"	Attr

Table 3. Document Structure Conversion

In the XML/SNMP gateway, the translator converts each node of SNMP MIB into an element of the XML Schema, and the name of the MIB node into the name of the element. Interior clauses of the MIB node such as "access", "status", etc. are translated into the attributes of the XML element. Table 3 shows the conversion of document structure between SNMP SMI and XML Schema. The example of a specification translation result in Table 4 shows how a MIB definition is translated into XML Schema. The <syntax> clause of each node is applied to the data type definition of the element in the XML Schema. We define an additional attribute of type "ID", which has an "oid" value of a node. An "oid" attribute of type "ID" enables a random access to a particular node by its "oid" value in the DOM tree.

MIB II	Translation Result
sysUpTime OBJECT-TYPE **SYNTAX** TimeTicks **ACCESS** read-only **STATUS** mandatory **DESCRIPTION** "The time..."	<xsd:element name="**sysUpTime**"><xsd:complexType> <xsd:simpleContent><xsd:restriction base="TimeTicks"> <xsd:attribute name="**oid**" type="xsd:string" use="fixed" value="1.3.6.1.2.1.1.3"/> <xsd:attribute name="**access**" type="xsd:string" use="fixed" value="read-only"/> <xsd:attribute name="**status**" type="xsd:string" use="fixed" value="mandatory"/> <xsd:attribute name="**description**" type="xsd:string" use="fixed" value= "The time....."/>

::= { system 3 }	</xsd:restriction></xsd:simpleContent></xsd:complexType></xsd:element>

Table 4. Example of Specification Translation

4.3.2. Interaction Translation

Y. J. Oh et. al, proposed three interaction translation methods [27] for an XML/SNMP gateway: XML Parser-based, HTTP-based, and SOAP-based. The DOM-based translation provides a method for XML-based manager to directly access and to manipulate management information through the standard DOM interfaces. The HTTP-based is a URI extension based translation applying XPath and XQuery, which enables to easily define detailed request messages and thus provides efficiency improvement in XML/HTTP communication between the manager and the gateway. In the SOAP-based translation, the gateway advertises its translation services to the manager using SOAP.

To integrate the XML-based manager with SNMP agents, we select the SOAP-based interaction translation method in this chapter. We define services into three big operations, namely get, set, and trap. Each operation has a parameter and a return value, which is also defined as an XML element between the manager and the gateway. In addition to the essential elements, such as get, set and trap, the SNMP GetBulk operation or other complex types of requests can be defined by extending the get or set operation with XPath or XQuery. As described in Section 2, XPath, XQuery and XUpdate provide an efficient means to indicate the managed objects to be retrieved. XPath, XQuery and XUpdate can also be applied in the SOAP request message as a parameter of each method.

5. Implementation Experience

We have implemented an XNMS based on the proposed architecture in Section 4. The manager uses various XML technologies to provide Web-based user interfaces, to communicate between manager and agent, and to process management information. We implemented the XNMS on a Linux OS using Java language. We used the Apache software [28] that provides APIs that were implemented with JAVA. XNMS uses the following APIs: Xerces as an XML parser, Xalan as an XPath handler and an XSLT processor, Xindice as XMLDB and AXIS as SOAP. XMLDB supports XML technologies, such as XPath, XQuery, and XUpdate to directly handle XML documents via the DOM parser.

The XML-based agent is applied to the IP sharing device. 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 with a 16 MB ROM. We used a powerpc-linux-gcc compiler. We implemented MIB-II information and basic configuration information in the IP sharing devices. The agent uses gSOAP [29] implemented with C/C++ languages to exchange SOAP messages. The gSOAP generates a stub, a skeleton and a WSDL definition using the header file which declares RPC operations. We selected the libxml which is more lightweight than most XML parsers in the agent because we considered the low computing resources of the IP sharing device.

The XML/SNMP gateway has been implemented on a Linux server. We used the Apache Tomcat 4.0

for the Web server and Servlet engine. We used Xerces for an XML parser, Xalan for an XPath/XSLT processor. We also used Innovation's HTTP Client for the HTTP Client and OpenNMS's joeSNMP 0.2.6 for the SNMP Handler and the Trap Handler. These are all Java-based. We used Apache Axis for the SOAP engine, which is the SOAP and WSDL compliant. We deployed the existing translation functions into Web Services using a Java Web Service (JWS) provided by Axis. This deployment method automatically generates a WSDL file and proxy/skeleton codes for the SOAP RPCs. In order to enable the existing Java class as a Web Service, we simply copy the Java file into the Axis Web application, using the extension ".jws" instead of ".java" We deployed the main class namely *SoapInterface* into Web Services as an interface to the SOAP-based XML/SNMP gateway. We installed the Axis SOAP engine both on the manager and the gateway.

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.

6. Concluding Remarks

In this chapter, we presented how XML technologies can be applied to network management tasks: modeling management information, instrumenting managed resources, management operations, communication between manager and agent, analyzing the data, and presenting the analysis result to users.

This chapter also presented an architectural framework for the development of XML-based network management systems. This framework can be used as a guideline for developing management systems. For validating our proposed architecture, we developed an XML-based network management system (XNMS) [26] consisting the XML-based agent, the XML-based manager, and the XML/SNMP gateway. Also, we developed an XML-based configuration management system (X-CONF) for distributed systems [30] and XML-based configuration management system (XCMS) for IP network devices [31]. Our XNMS fully utilizes XML technologies, such as the XML Schema, DOM, XPath, XQuery, and XSLT, to perform network management tasks. We were able to reduce the development cost of the management system through the support of standard APIs for processing XML documents.

XML technology is viewed by many as a revolutionary approach for solving the problems that exist in current standards and practices for network and systems management. More work is needed to prove this by developing not only standards but also applying to manage real networks and systems.

References

[1] W3C, "Extensible Markup Language (XML) 1.0", W3C Recommendation, Oct. 2000.

[2] W3C, “XML Schema Part 0,1,2”, W3C Recommendation, May 2001.
[3] W3C, “Extensible Stylesheet Language (XSL) Version 1.0”, W3C Recommendation, Nov. 2000.
[4] W3C, “Document Object Model (DOM) Level 2 Core Specification”, W3C Recommendation, Nov. 2000.
[5] W3C, “XML Path Language (XPath) Version 2.0”, W3C Working Draft, Apr. 2002.
[6] W3C, “XQuery 1.0: An XML Query Language”, W3C Working Draft, Apr. 2002.
[7] W3C, “SOAP Version 1.2 Part 0: Primer”, W3C Working Draft, Dec. 2001.
[8] J. Schonwalder, A. Pras, J.P. Martin-Flatin, “On the Future of Internet Management Technologies”, IEEE Communications Magazine, October 2003, pp.90~97.
[9] W3C, “Web Services Description Language (WSDL) Version 1.2” W3C Working Draft, Jul. 2002.
[10] F. Strauss, and T. Klie, “Towards XML Oriented Internet Management”, Proc. of IFIP/IEEE International Symposium on Integrated Network Management (IM 2003), Colorado Springs, USA, Mar. 2003, pp.505~518.
[11] J. H. Yoon, H. T. Ju and J. W. Hong, “Development of SNMP-XML Translator and Gateway for XML-based Integrated Network Management”, International Journal of Network Management (IJNM), Vol. 13, No. 4, July-August 2003, pp. 259-276.
[12] Frank Strauss, et al, “A Library to Access SMI MIB Information”, http://www.ibr.cs.tu-bs.de/projects/libsmi/.
[13] Avaya Labs., XML based Management Interface for SNMP Enabled Devices, http://www.research.avayalabs.com/user/mazum/Projects/XML/.
[14] M. H. Sqalli, and S. Sirajuddin, “Static Weighted Load-Balancing for XML-based Network Management Using JPVM”, Proc. of MMNS 2005, Barcelona, Spain, Oct. 2005, pp. 228~241.
[15] WBEM, “WBEM Initiative”, http://www.dmtf.org/wbem/.
[16] J.P. Martin-Flatin. “Web-Based Management of IP Networks and Systems”, Ph.D. Thesis, Swiss Federal Institute of Technology, Lausanne (EPFL), Oct. 2000.
[17] 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. of IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Florence, Italy, Apr. 2002, pp.1~14.
[18] L. R. Menten, “Experiences in the Application of XML for Device Management”, IEEE Communications Magazine, Vol. 42 No. 7, July 2004, pp.92~100.
[19] 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.
[20] Cisco Systems, Cisco Configuration Registrar, http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/ie2100/cnfg_reg/index.htm.
[21] IETF, “Network Configuration (Netconf)”, http://www.ietf.org/html.charters/Netconf-charter.html.
[22] W3C, "Simple API for XML Version 2.0", WC3 Recommendation, Nov. 1999.
[23] W3C, “XSL Transformations Version 1.0”, W3C Recommendation, Nov. 1999
[24] XML:DB, “XUpdate”, Working Draft - 2000-09-14, http://www.xmldb.org/xupdate/xupdate-wd.html.
[25] 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
[26] Mi-Jung Choi, James W. Hong and Hong-Taek Ju, “XML-based Network Management for IP Networks”, ETRI Journal, Vol. 25, No. 6, Dec. 2003, pp. 445-463.
[27] Y. J. Oh, H. T. Ju, M. J. Choi, J. W. Hong, “Interaction Translation Methods for XML/SNMP Gateway”, Proc. of DSOM 2002, Montreal Canada, Oct. 2002, pp. 54~65.
[28] Apache Group, “Apache”, http://www.apache.org/.
[29] Robert A., “gSOAP: Generator Tools for Coding SOAP/XML Web Service and Client Applications in C and C++”, http://www.cs.fsu.edu/~engelen /soap.htm/.
[30] H. M. Choi, M. J. Choi, and J. W. Hong, “Design and Implementation of XML-based Configuration Management System for Distributed Systems”, Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, Apr. 2004, pp. 831~844.
[31] M. J. Choi, H. M. Choi, H. T.Ju and J. W. Hong, “XML-based Configuration Management for IP Network Devices”, IEEE Communications Magazine, Vol. 42 No. 7, July 2004, pp. 84~91.
[32] V. Cridlig, R. State, and O. Festor, “An Integrated Security Framework for XML based Management”, Proc. Of the IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), Nice, France, May 2005, pp. 581~600.

[33] V. Cridlig, H. Abdelnur, J. Bourdellon, and R. State, “A Netconf Network Management Suite: ENSUITE”, Proc. of IPOM 2005, Barcelona, Spain, Oct. 2005, pp. 152~161.

[34] S. M. Yoo, H. T. Ju, and J. W. Hong, “Web Services Based Configuration Management for IP Network Devices”, Proc. of MMNS 2005, Barcelona, Spain, Oct. 2005, pp. 254~265.

[35] OASIS, “Universal Descritpion, Discovery and Integration (UDDI)”, http://www.uddi.org/.