석 사 학 위 논 문

네트워크 장비들을 위한 XML기반의 구성 관리

최 현 미 (崔 現 美)

정보통신학과 (네트워크 전공)

포항공과대학교 정보통신대학원

2003

네트워크 장비들을 위한 XML기반의 구성 관리

XML-based Configuration Management for IP Network Devices

XML-based Configuration Management for IP Network Devices

by

Hyoun-Mi Choi

Department of Computer and Communications Engineering

POSTECH Graduate School for Information Technology

A thesis submitted to the faculty of POSTECH Graduate School for Information Technology in partial fulfillment of the requirements for the degree of Master of Engineering in the Department of Computer and Communications Engineering.

Pohang, Korea

December 18, 2003

Approved by

____

Major Advisor

XML-based Configuration Management for IP Network Devices

최 현 미

위 논문은 포항공과대학교 정보통신대학원 석사 학위논문으로 학위논문 심사위원회를 통과하였음을 인정함.

2003년 12월 18일

학위논문심사 위원회 위원장 홍원기 (인)

위 원 김 종 (인)

위 원 서영주 (인)

MCC
20022623

최 현 미, Hyoun-Mi Choi, XML-based Configuration Management for IP Network Devices, 네트워크 장비들을 위한 구성 관리, Department of Computer and Communications, 2004, 42P, Advisor: J. Won-Ki Hong, Text in English.

ABSTRACT

As the Internet continues to grow, the tasks of operations and management of IP networks and systems are becoming more and more difficult. One of the most critical deficiencies of SNMP has been in the area of configuration management because of a limited information modeling and management operations. Also, SNMP over UDP provides the restrictive retrievals of bulk data. Recently, much attention has been given to the use of XML technology to network and service management as an alterantive or complementary approach to SNMP. The latest IETF's effort on configuration management, Netconf WG is in progress. In this thesis, we make some suggestions for improvement to the current Netconf protocol. Then we introduce XML-based configuration management and present the architecture and implementation of an XML-based configuration management system (XCMS). XCMS uses SOAP to take advantage of RPC interfaces and WSDL to notify the various management services of agents to the manager. As an addressing method to select the specific configuration, we chose XPath, which is redefined to support only the necessary expression from the agent aspect. We propose management information modeling with dependencies among managed objects. Moreover, we show the performance evaluations and validate the efficiency of the XCMS.

Table of Contents

1. Introduction 1

2. Related Work 5

 2.1. XML Related Technologies 5

 2.2. XML-based Network Management 8

 2.3. Overview of Netconf 12

  2.3.1. Netconf Configuration Protocol 13

  2.3.2. Netconf Transport protocol 17

3. Architecture for XML-based Configuration Management 19

 3.1. Problems and Proposed Solutions 19

 3.2. Management Information Model 21

 3.3. Management Protocol 22

 3.4. Architecture 26

4. Implementation 29

5. Comparative Analysis 33

 5.1. Performance Evaluation 33

 5.2. Comparison and Analysis 36

6. Conclusions and Future Work 39

References 40

List of Figures

Figure 1. High Level XCMS Architecture..3
Figure 2. Example of Netconf Protocol Message for <edit-config>.......17
Figure 3. Mapping of Netconf Protocol Message to SOAP Message 20
Figure 4. Management Information Model 22
Figure 5. WSDL Definition of <edit-config-replace> Operation............26
Figure 6. Architecture of XML-based Configuration Management System 28
Figure 7. Information Model Example..30
Figure 8. Message Size of Get...34
Figure 9. Exchanged Packets...35
Figure 10. Response Time of Get Request 36
Figure 11. Management Protocol Message 37

List of Tables

Table 1. Netconf Protocol Layers 14
Table 2. Examples of XPath expression 32

1. Introduction

The rapid pace of Internet evolution is currently witnessing the emergence of diverse network devices. Current IP networks are complex and are composed of various network devices. Efficient network management systems are necessary to manage these networks and devices. Since it was introduced in the late 1980s, the Simple Network Management Protocol (SNMP) [1] is the most widely used method for network management on the Internet. However, the SNMP management framework has some limitations concerning management information modeling, management protocols, and configuration management [2].

SNMP defines the management information base (MIB) using Structure of Management Information (SMI). The SNMP SMI is insufficient to present management information because it does not support such concepts as structured data types, objects, or methods. Also, the SNMP MIB is based on a simple hierarchy structure in the form of a tree, so it is difficult to present dependencies among objects. The SNMP protocol provides limited operations such as Get, Set, Trap and GetBulk, so it is difficult to improve management functionalities. Because SNMP transfers management information over UDP, it does not support bulk data retrievals in a reliable manner. The SNMP over UDP merely supports the transport of SNMP messages of a size up to 484 bytes, which is too small for bulk data transfers. Also, SNMP retransmits important data at the application level because of the unreliability in UDP. These weaknesses with SNMP especially influence configuration management which needs to retrieve or modify bulk data using the Object Identifier (OID), and provide complex dependencies between managed objects.

To overcome these weaknesses with SNMP, approaches have been attempted in recent years, but all have failed to be adopted as standards [2]. One such failure is the SMIng of IETF Network Management Research Group (NMRG), which was designed to improve the expressive power of the SMI language. To overcome the problem of unreliable transport, NMRG defined a transport mapping for SNMP over TCP to allow for larger SNMP messages to be transported in a reliable manner. However, this effort also failed to be adopted as a standard.

As the evolutionary approach to solve such problems with SNMP, Extensible Markup Language (XML) [3] technologies, such as the XML Schema [4], Document Object Model (DOM) API [5], XML Path Language (XPath) [6], Extensible Style-sheet Language (XSL) [7], Simple Object Access Protocol (SOAP) [8], Web Services Description Language (WSDL) [9], etc., are being applied to configuration management. A standardization process of configuration management for network devices using XML technologies is also in progress.

The Network Configuration (Netconf) working group (WG) [10] organized by IETF is responsible for this standardization work. In more detail, the standardization primarily focuses on a management protocol, which is a message format used between a manager and an agent. This entire process of standardization fulfills operational needs for the manager and agent on various network devices manufactured by different vendors and guarantees interoperability. Major network device vendors, such as Cisco and Juniper Networks, integrate the XML-based agent in their products and are actively participating in the Netconf standardization work.

The Netconf protocol uses the RPC mechanisim, so it defines the RPC operations with XML tags. The RPC mechansim easily provides additional operations except base operations to improve the management functionalities unlike the limited operations of SNMP. This means evry agent can serve the different management services and have various capabilities. Until now, little implemenatation work has been done using the Netconf protocol.

For the successful deployment of a standard management protocol, we need to demonstrate the feasibility and effectiveness of the new protocol to the operators and administrators. This thesis identifies problems in the current Netconf proposal and suggests possible solutions. This thesis hopes to demonstrate that by designing and implementing an XML-based Configuration Management System (XCMS) based on Netconf. Figure 1 is a high-level architecture of our XCMS, where the XML-based manager controls multiple network devices equipped with XML-based configuration management agents.

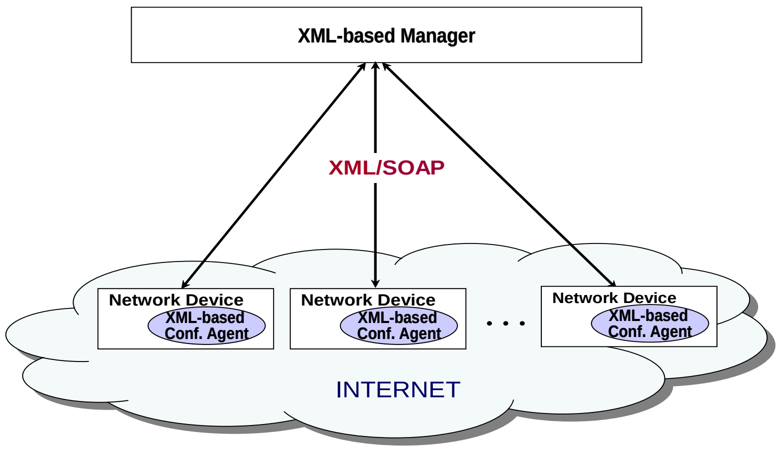


Figure 1. High Level XCMS Architecture

XCMS uses SOAP over HTTP to take advantage of RPC interfaces and WSDL to notify the various management services of agents to the manager. In addition, WSDL in the agent defines the manager’s RPC operations to receive asynchronous notification like the SNMP trap operation. As an addressing method to select the specific configuration, we choose XPath, which is redefined to support only the necessary expression from the agent aspect of configuration management. We propose management information modeling with dependencies among managed objects, by using the attribute. A dependency means that the modification of some objects influences certain other objects.

The organization of this thesis is as follows. Chapter 2 presents an overview of XML technologies, and introduces related work on XML-based network management. Also, it shows the details of Netconf protocol for configuration management. In Chapter 3, we describe the architecture for XML-based Configuration Management and propose several points for improvement of Netconf. Implement experience of XCMS for IP gateway (IP sharing device) is described in Chapter 4, and performance analysis results are presented in Chapter 5. Lastly, we summarize our work and discuss directions for future research in Chapter 6.

2. Related Work

In this chapter, we first explain XML related technologies that are applicable to network management. We also introduce related work on XML-based network management performed by others and present the details of Netconf protocol.

2.1. XML Related Technologies

- **DTD and XML Schema**: XML has two fundamental approaches to define the XML document structure: Document Type Definition (DTD) and XML Schema [4]. The DTD is used to specify a content model for each element. The content description is part of the element declaration in the DTD, and specifies the order and quantity of elements that can be contained within the element being declared. That is, the DTD is used to specify a property for each element in addition to the relationship between the elements. However, because the DTD does not support a complex information model, another modeling mechanism, the XML Schema, was proposed. The XML Schema substantially revised and extended the capabilities found in XML DTDs. The XML Schema is based on XML, so it can be parsed and manipulated in exactly the same manner as XML documents through the standard API. The XML Schema supports a variety of data types (44 kinds of basic types), while the DTD treats all data as strings or enumerated strings. The XML Schema also allows inheritance relationships between elements and supports namespace integration.

- **XSL and XSLT**: Extensible Stylesheet Language (XSL) [7] is a mark-up language designed for illustrating the method to display XML documents on the

Web. XML documents describe only the contents and the structure of the contents. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses a formatting vocabulary. That is, XSL enables XML to separate contents from presentation. XSL consists of two parts: a language for transforming XML documents, and an XML vocabulary for specifying formatting semantics. The style sheet technology to transform documents is XSL Transformation (XSLT) [11], which is a subset of XSL technology that fully supports the transformation of an XML document from one format into another, such as HTML or another custom XML document type. The reason for publishing the XSLT specification separately from XSL is that XML documents can be displayed, providing an easy display format for end users by transforming XML documents without a need for formatting semantics.

- **DOM and SAX**: The Document Object Model (DOM) [5] is a platform- and language-independent interface that allows programs and scripts to dynamically access and update the content, structure, and style of documents. The DOM is an API for valid HTML and well-formed XML documents. The Simple API for XML (SAX) [12] is an event-driven and serial-access mechanism for accessing XML documents. A DOM parser parses the XML document and creates a DOM tree which keeps the entire structure in memory at the same time, while SAX reads the XML document in sequential order and generates an event for a specific element. Therefore, if the application calls for sequential access to XML documents, SAX can be much faster than other methods without requiring much system overhead. 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 the event, applications can process operations, such as gaining the name and attribute of the element.

- **XPath**: XPath [6] is a language used to identify particular sections of an XML document. XPath has a compact, non-XML syntax to be used within URIs [37] and XML attribute values, and operates on the abstract, logical structure of an XML document. Each node in an XML document is indicated by its position, type, and content using XPath.

- **XQuery and Xupdate**: Xquery [13], a query language for XML, is designed to be broadly applicable across all types of XML data sources, such as structured and semi-structured documents, relational databases, and object repositories. Xquery uses XPath for path expression. Further, Xquery provides such features as filtering documents, joining across multiple data sources, and grouping the contents. Xupdate [14] 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.

- **SOAP**: Simple Object Access Protocol (SOAP) [8] 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 contents 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.

- **WSDL**: Web Services Description Language (WSDL) [9] is an XML-based language used to define Web Services 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 definitions 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. These endpoints 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.

2.2. XML-based Network Management

- **WBEM:** Web-based enterprise management (WBEM) [15] is an initiative of the DMTF that includes a set of technologies that enables the interoperable management of an enterprise network. WBEM defines an information model called the Common Information Model (CIM) which is an object-oriented schema for modeling managed objects. These managed objects are the representations of real resources, and the schema provides a single data description mechanism for all types of resources. WBEM provides an information standard that defines how data is represented, and a process standard that defines how the components interact. Obviously, a method for accessing CIM data is required. The DMTF defines the CIM to XML mapping and CIM operations over HTTP. The specification of the CIM to XML mapping defines the XML Schema used to describe the CIM object in XML for encapsulation over HTTP. Both CIM classes and instances must be valid XML documents for this schema. The CIM operations over HTTP allow implementations of CIM to operate in an open, standard manner. It describes how CIM operations are encoded in the HTTP payload using XML, and defines the syntax and semantics of the operation requests and their corresponding responses. WBEM uses XML only for object and operation encoding. WBEM is currently being updated to include emerging standards, such as SOAP.

- **WIMA:** J.P. Martin-Flatin was one of the first to propose using XML for integrated management in his work on Web-based integrated network management architecture (WIMA) [16]. WIMA showed that the push-based network management is more appropriate than pull-based network management. A WIMA-based research prototype, JAva MAnagement Platform (JAMAP), implemented a push-based network management using Java technology. WIMA provided 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, or BER-encoded

SNMP data. By separating the communication and information models, WIMA allows management applications to transfer SNMP, CIM, or other management data. WIMA showed that XML is especially convenient for distributed and integrated management, for dealing with multiple information models and for supporting high-level semantics.

- **XNM:** We proposed XML-based network management (XNM) using Embedded Web Server (EWS) [17]. This architecture has two key components: Web-Based Management (WBM) agent and WBM manager. It extended the use of EWS for element management to Web-based network management architecture platform by adding XML functionalities. XNM uses XML to transfer management information over HTTP between an agent and a manager. XNM also uses DOM to represent and process management data and XPath to access the specific part of management data.
- **XNAMI:** John, et al. proposed an XML-based architecture for SNMP management of network and application, called XNAMI [18]. This is an example of an XML-based management communication architecture that permits a manager system to extend the agent's MIB within the SNMP framework. SNMP GET and SET operations on extension objects are implemented in Java. In the XNAMI architecture, the XNAMI manager can transfer compressed Java bytecode to the XNAMI agent using an SNMP SET operation in order to add SNMP MIB. The agent maintains an explicit runtime representation of its SNMP MIBs and uses XML to represent managed object internally. An XML document describing the MIB is transferred and retrieved by the SNMP operation. In this architecture, XML is

used to represent the MIB definition and store it in a DOM tree at the agent, and to browse the MIB module at the manager. Upon receipt of an SNMP GET, the agent's SNMP server executes the GET method, which is added by XNAMI manager. With a Java bytecode, the XNAMI manager transfers an XML string specifying that a new OID should be created, and an existing OID deleted. The XNAMI manager and agent exchange management data via SNMP, while the XML data is transferred via HTTP.

- **Juniper Networks' JUNOScript:** Recently, Juniper Networks introduced JUNOScript [19] for its JUNOS network operating system. The JUNOScript, a part of its XML-based network management effort, uses a simple model that is designed to minimize both implementation costs and 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.
- **Cisco's Configuration Registrar:** The Cisco Configuration Registrar [20] is a Web-based system for automatically distributing configuration files to Cisco IOS network devices. The Configuration Registrar works in conjunction with Cisco Networking Services (CNS) configuration agents located at each device. The Configuration Registrar delivers the initial

configuration to the 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.

2.3. Overview of Netconf

The Network Configuration (Netconf) working group (WG) [10] was formed in May 2003. The Netconf WG attempts to standardize a protocol suitable for configuration management of network devices. The Netconf WG defines the Netconf protocol and transport mappings. Internet network operators, developers and researchers have participated in the Netconf WG and issued several Internet Drafts (I-D) [21, 22, 23, 24] and individual submissions. In this section, we introduce the Netconf protocol and transport mappings.

Netconf protocol [21] describes several requirements, which are summarized as follows:

- It distinguishes configuration data, state data, and system notification. The configuration data is read-write data, and state data and the system notification are read-only data.
- The configuration operations (e.g., get-config and edit-config) are provided. To prevent simultaneous modification, a mechanism for locking data is necessary. The transaction of the modification operations is provided optionally (e.g., stop-on-error, ignore-error).
- It is necessary to distinguish between the modification of a configuration and

the activation of configurations. The configuration is separated into three states: running, candidate, and startup. The implementation of 'running' state is mandatory, and the implementation of the other two states is optional.

- It can open three TCP connections (channels) in a single session: operation, management, and notification. The operation channel is mandatory, and the other two channels are optional.

2.3.1. Netconf Configuration Protocol

The Netconf protocol [21] uses XML for data encoding and a simple RPC-based mechanism to facilitate communication between a client and a server. The design goals of the Netconf protocol are as follows.

- Improve interoperability among the devices produced by different vendors.
- Provide a transport-neutral protocol based on TCP.
- Provide ease of implementation to developers using existing XML related tools.
- Provide the operations of Get and Edit full or partial configuration.
- Support actions, such as exec commands

As depicted in Table 1, the Netconf protocol can be conceptually partitioned into four layers: Content, Operation, RPC, and Transport.

The 'Content' layer presents the configuration content, which basically depends on the device vendors. This layer is currently outside the scope of Netconf but it is expected that a standard data definition language and standard content will soon be formulated. The state of configuration is divided into three phases: candidate, running, and startup. The ‘running’ is the complete configuration, which is currently active on the network device. The ‘candidate’ is the candidate configuration, which can be manipulated without impacting the device’s current configuration. The ‘startup’ is the copied configuration from the ‘running’ state when the running configuration is reliable.

Layer	**Example**
Content	Configuration data
Operations	<get-config>, <edit-config>, etc.
RPC	<rpc>, <rpc-reply>, etc.
Transport	SSH, BEEP, SOAP over HTTP, etc.

Table 1. Netconf Protocol Layers

The ‘Operations’ layer includes base and additional management operations. The base operations of the Netconf protocol are defined as follows:

- <get-config>: It retrieves all or parts of the specified configuration.
- <edit-config>: It requires three types of operation attributes: merge, replace, and delete. This operation message includes one of these attributes, which is inserted into the elements of the specified configuration. It merges or replaces all or parts of the specified configuration to the specified target configuration using the merge or replace attributes. It also deletes all or part of the specified configuration using the delete attribute. The transaction of the modification operations is provided optionally. The values for transaction processing are ‘stop-on-error’ (default) and ‘ignore-error’.
- <copy-config>: It creates or replaces an entire configuration file with the

contents of another complete configuration file.

- <delete-config>: It deletes configuration data.
- <kill-session>: It terminates the Netconf connections in channels within a single Netconf session, by using the session-ID.
- <lock>: It allows the manager to lock the configuration of a device.

A set of functionalities that supplements the base operations is called ‘capability’ in Netconf. The Netconf capability permits the agent to adjust its behavior to take advantage of features exposed by the device. The agent needs an XML document to describe its own capability which is performed by its optional operations. For example, operations for the Netconf capability are shown as follows:

- <notify>: It sends asynchronous notifications.
- <validate>: It validates the contents of the specified configuration.

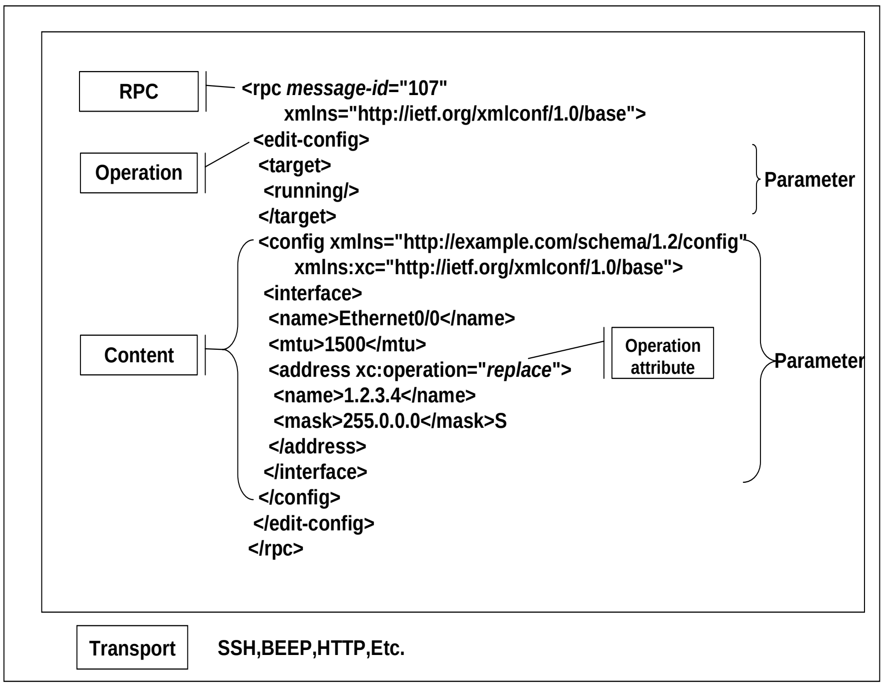
The ‘RPC’ layer presents the RPC-based communication model. This layer contains the message-ID which the manager can distinguish its own rpc message. The manager and the agent use <rpc> and <rpc-reply> elements to provide a transport-independent framing of protocol requests and responses. The elements in this layer are as follows:

- <rpc>: It expresses request messages of operations in the Netconf protocol. This element contains the message-ID as the attribute.
- <rpc-reply>: It presents a response message. The ‘<ok>’ element is sent in the <rpc-reply> message if no error occurs during the processing of an <rpc> request. Otherwise, the <rpc-error> element is delivered in it.

- <rpc-abort>/<rpc-abort-reply>: By using the message-ID, it terminates the RPC operations in progress and the response message is sent in <rpc-abort-reply>.
- <rpc-progress>: It sends a progress report of the RPC operations in progress before an <rpc-reply> that takes a long time to transmit can be generated.

The ‘Transport’ layer supports any transport protocol that is connection-oriented, requiring a persistent connection between the manager and agent. When the manager establishes a session with the agent, the unique session-ID is assigned by the agent. And, this session supports three channels of management, operation, and notification. The management channel manages on-going RPC operations on operation channels via ‘rpc-abort’ and notifies progress reports of the RPC operations via an ‘rpc-progress’ element. This channel also advertises the capability of the agent by spreading out an XML document to describe its capability. The operation channel performs most Netconf protocol operations including the <rpc> and <rpc-reply> elements. The notification channel transmits notification messages from the agent to the manager.

Figure 2 shows the example of the NETCONF protocol message to request the <edit-config> operation. This request message, which includes the message-id 107, replaces the values of the address descendants to ‘<name>1.2.3.4</name> <mask>255.0.0.0</mask>’. Because the Netconf protocol is difficult to select the specific configuration, this message contains the unmodified configuration information which is ‘<interface><name>Ethernet0/0</name><mtu>1500</mtu> </interface>’.



```
<rpc message-id="107"
      xmlns="http://ietf.org/xmlconf/1.0/base">
 <edit-config>
  <target>
   <running/>
  </target>
  <config xmlns="http://example.com/schema/1.2/config"
        xmlns:xc="http://ietf.org/xmlconf/1.0/base">
   <interface>
    <name>Ethernet0/0</name>
    <mtu>1500</mtu>
    <address xc:operation="replace">
     <name>1.2.3.4</name>
     <mask>255.0.0.0</mask>S
    </address>
   </interface>
  </config>
 </edit-config>
</rpc>
```

Figure 2. Example of Netconf Protocol Message for <edit-config>

2.3.2. Netconf Transport protocol

The Netconf protocol is currently considering three separate application protocol bindings for transport: Secure Shell (SSH) [22], Block Extensible Exchange Protocol (BEEP) [23] and SOAP over HTTP [24].

- **SSH:** Secure Shell (SSH) [22] is required by operators who are accustomed to the existing network device environment. SSH is more appropriate as a rapid-prototyping or debugging tool to examine the functionality of the Netconf protocol rather than an API for complex applications. SSH has a constraint on multiple channels within a session.

- **BEEP:** Block Extensible Exchange Protocol (BEEP) [23] is a peer-to-peer protocol, so the manager or the agent can initiate the connection. Because of the nature of this connection initiation, BEEP is superior to SOAP over HTTP in processing the asynchronous notification. BEEP also supports multiple channels within a single session. There are freely available software toolkits which support BEEP. However, the downside to BEEP is not widely deployed at this time, in either network devices or end user operating systems.

- **SOAP over HTTP:** Simple Object Access Protocol (SOAP) over HTTP [24] is a suitable application protocol because it supports its own RPC interface and has the ability to bind any application protocol for transport. SOAP tools and source codes provide an easy implementation environment that generates the skeleton and the stub for communicating. As HTTP is asymmetric with respect to client and server, it has some difficulties in sending asynchronous notifications from the agent to the manager. So, the manager can periodically poll requests for notification. This approach increases unnecessary traffic and the manager's processing overhead.

3. Architecture for XML-based Configuration Management

In this chapter, we describe the design and Architecture of an XML-based Configuration Management System (XCMS) in accordance with Netconf.

3.1. Problems and Proposed Solutions

To overcome limited management information and management protocol in SNMP, XML technologies currently are applied to the configuration management. We point the problems in Netconf protocol using SOAP, and illustrate methods for solving the problems.

The management information from diverse network devices developed by different vendors is difficult to define in a standard format. Although it realizes the importance of the ‘Content’ layer, the Netconf WG is not concerned with management information yet. We propose management information modeling to place guidelines for the presentation of management information into an XML format.

Also, Netconf does not provide an addressing mechanism for the selection of specific part of configuration information. In order to identify the specific part of a configuration, the ‘edit-config’ operation inserts one of the operation types (merge, replace, and delete) into the content of a specific configuration, so this operation is dependent on information modeling language of the content. Because the operation type exists in the content of the configuration, parsing of configuration data is required to determine the operation type. Therefore, we propose the XPath expression as an addressing method to present the accurate position of the configuration. By using XPath, the operation ‘edit-config’ can be separated into ‘edit-config-merge’, ‘edit-config-replace’, and ‘edit-config-delete’.

In the Netconf over SOAP Internet Draft (I-D) [24], the tags of the ‘RPC’ layer are acknowledged as SOAP RPC operations. The operation tags of the ‘Operation’ layer are passed as parameters of the SOAP message. This still requires parsing tags to retrieve the operation names and parameters as in BEEP and SSH. To reduce parsing overhead, it is necessary to map the Netconf protocol message to the SOAP message, as shown in Figure 3. The SOAP RPC Operation layer is a combination of the Netconf’s Operation layer and the RPC layer. The message-id in the RPC layer is mapped to the id in <SOAP:BODY>. The names of Operations in Netconf become SOAP RPC Operations. The necessary information is passed as parameters.

NETCONF Protocol Level	SOAP RPC Message Level	Example

Content	**Parameter**	//SOAP RPC Operation
		<SOAP-ENV:Body id= "50">
		<edit-config-replace>
		  <target>
		    <running/> Parameter 1
		  </target>
		  <xpath>
		   //interface Parameter 2
		  </xpath>
		  <config>
		    <interface>
		      <name>Ethernet</name>
		      <mtu>1500</mtu>
		      <address> Parameter 3
		        <name>1.2.3.4</name>
		        <mask>255.0.0.0</mask>
		      </address>
		    </interface>
		  </config>
		  <transaction>
		    rollback Parameter 4
		  </transaction>
		</edit-config-replace>
		</SOAP-ENV:Body>

Operation <get-config>, <edit-config>, <lock>, <kill-session>, etc.	**SOAP RPC Operation** <get-config>, <edit-config-replace>, <edit-config-merge>, <edit-config-delete>, <lock>, <kill-session>, <rpc-progress>, <rpc-abort>, etc.	
RPC <rpc>, <rpc-reply>, <rpc-progress>, <rpc-abort>		
Transport	**HTTP**	HTTP

Figure 3. Mapping of Netconf Protocol Message to SOAP Message

While SOAP can be bound to different underlying protocols such as HTTP, SMTP or BEEP, most existing SOAP implementations support HTTP. In the Netconf over SOAP I-D, it is not possible to send asynchronous notifications from the agent to the manager because the agent includes only the HTTP server. As a solution, the manager can periodically poll requests for notification. This approach increases unnecessary traffic and the manager's processing overhead for periodic requests. For notification delivery, the agent needs to equip the HTTP client as well as the HTTP server. The agent can send a notification message to the manager through the HTTP client.

The RPC mechanism exactly knows the following information: parameters of operation, data types of parameters, and return values of each RPC operation. However, the Neconf protocol message is not sufficient for specific operation description. Therefore, WSDL provided by SOAP tools is needed for describing Netconf operations and additional operations. The use of WSDL in the Netconf protocol is expected to be standardized for ease of implementation on agents and for integration with deployed systems. We also propose WSDL to describe the capability of an agent instead of the XML document.

3.2. Management Information Model

Information modeling presents the management information of the managed system. We define the configuration information of a network device using the XML Schema [4], which supports to define management information by adding new data types to a pool of data types. The configuration information in network device has relationships with each other. A relationship means that some objects of the configuration information can influence other objects.

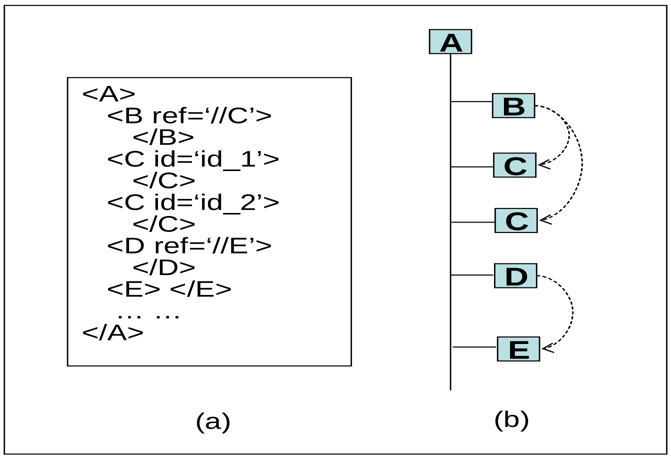


Figure 4. Management Information Model

Figure 4 shows the structure of configuration information. In Figure 4 (a), the method to represent the information is to use nested elements to present each name of a managed object, and the element's attributes indicates a dependency (relationship) among the objects or distinguishes the elements with the name of the same tag. For example, the name of the 'ref' attribute is used to present the referred relation with another object. If the value of the object 'B' is changed, the modified value can have influence on the object 'C'. The name of 'id' attribute implies an identifier of the elements with the same tag's name 'C'. In Figure 4 (b), the information is transformed into a tree structure where a dotted arrow line shows a referred relationship.

3.3. Management Protocol

We have selected to use the SOAP over HTTP as a transport protocol. With the absence of an RPC interface in BEEP and SSH, a parsing process on operation messages is required to retrieve the operation name and parameters. SOAP [8] provides an RPC interface, and its messages are easy to parse. SOAP tools automatically generate important codes for the skeleton and the stub to communicate. These tools also generate WSDL definitions, which contain useful information for RPC operations, such as the operation names and parameters. When a new operation is added to the agent's operation list, the new version of WSDL definition is generated. The updated WSDL definition is the specification to notify the manager with newly added operations. So, the WSDL is very useful for describing the various capabilities of the agents and the manager's RPC operations to receive the notification messages from the agent.

We have defined management operations as the SOAP RPC operations based on Netconf operations as follows.

- <get-config>: It retrieves configuration information from the agent to the manager.
- <edit-config-{operation}>: 'merge', 'replace' and 'delete' are inserted into the '{operation}' field. 'merge' is to add, 'replace is to replace, and 'delete' is to delete.
- <kill-session>: When the manager terminates the specific session, the 'kill-session' operation disconnects them using the session-ID.
- <lock>: It locks the configuration.
- <notify>: It processes transmitted notification messages from the agent. For example, the agent sends notification message when it abnormally terminates or reboots itself.

- <rpc-progress>: It is possible to receive reporting message from the agent. The agent calls this operation when some operations requested by the manager are deferred.
- <rpc-abort>: It terminates the RPC operation in progress, when the manager is desired to abort its RPC operation.

We define the additional operations except the base operations to improve the management functionality.

- <download><upload>: The manager downloads or uploads a configuration file as well as a WSDL file from/to the managed agent to/from the manager.
- <reboot><shutdown>: The agent reboots or shutdowns itself to apply the modified configuration information.

Another important issue with the management protocol is an addressing method to access the specific part of management information. An addressing mechanism in an XML document is provided using the XPath [6] expression, which provides explicit expressions to identify XML nodes. One role of XPath is to indicate a specific location path of configuration information with an unabbreviated or an abbreviated syntax. The location path in configuration information within the agent can be presented using only the abbreviated syntax of XPath.

The XML parser in the agent should be lightweight to support XPath because the agent is an embedded system. This requires removing unnecessary expression of XPath. It takes much effort to optimize an XML parser suitable for the configuration management of a network device. To minimize the size of the agent’s XML parser, the agent supports the XPath expression only using the following two factors: operators and node-set.

- Operators are used for selecting the nodes within a specific range.
- Node-set is an unordered collection of nodes without duplicates.

We have defined the SOAP RPC operations using WSDL [9]. As described in Figure 5, 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.
- service: A collection of related endpoints.

Figure 5 describes the WSDL definition of the <edit-config-replace> RPC operation. This operation includes the following parameters: ‘target’, ‘xpath’, ‘config’, and ‘transaction’. The parameter ‘target’ is a target configuration which is one of the states, such as candidate, running, and startup. The parameter ‘xpath’ is an XPath expression to access the specific part of the configuration for replacement. The parameter ‘config’ is the information to replace.

```
<?xml version="1.0" encoding="UTF-8"?>
 <definitions name="netconf“  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://ietf.org/netconf/1.0/soap"
  xmlns:xb="http://ietf.org/netconf/1.0/base" targetNamespace="http://ietf.org/netconf/1.0/soap"
  name="http://ietf.org/netconf/1.0/soap">
 <import namespace="http://ietf.org/netconf/1.0/base"  location="base.xsd"/>
 <message name="edit-config-replaceRequest">
   <part name="target" type="xsd:string"/>
   <part name="xpath" type="xsd:string"/>
   <part name="config" type="xsd:string"/>
   <part name="transaction" type="xsd:string"/>
 </message>
 <message name="edit-config-replaceResponse">
   <part name="rpc-reply" type="xsd:string"/>
 </message>
<portType name="netconfPortType">
   <operation name="edit-config-replace">
     <input message="tns: edit-config-replaceRequest"/>
     <output message="tns: edit-config-replaceResponse"/>
   </operation>
 </portType>
 <binding name="rpcBinding" type="tns:rpcPortType">
   <SOAP:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="edit-config-replace">
   <SOAP:operation/>
     <input>
       <SOAP:body use="encoded"  namespace="http://ietf.org/netconf/1.0/base"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
       <SOAP:body use="encoded" namespace="http://ietf.org/netconf/1.0/base"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>
 </binding>
 <service name="SoapInterfaceService">
  <port binding="SoapInterfaceSoapBinding" name="SoapInterface">
   <address location=" http://hostname:8080/"/>
  </port>
 </servcie>
</definitions>
```

Figure 5. WSDL Definition of <edit-config-replace> Operation

3.4. Architecture

In Figure 6 we illustrate the architecture of XCMS consisting of a manager and an agent. Both the manager and the agent contain the HTTP server/client modules for initiating a bidirectional connection. Each is composed of three modules: Management Operations, SOAP over HTTP, and a Repository. The management Operations module needs to process the various management functionalities, the SOAP over HTTP module provides to communicate between the manager and the agents, and the Repository module stores the management information. It is easy to apply XML technologies to these divided modules.

The Management Operations module is classified with several components and serves management needs. The RPC Operations component includes SOAP RPC operations with their own RPC interfaces. The Local Operations component is a set of internal operations that supplements the RPC operations. This component contains the operations for processing the transactions of the <edit-config-{operation}> RPC operations and logging the results of the transactions. The XML parser component is a set of packages with DOM APIs that allow us to read and write an XML document. The manager has an additional Web Interfaces component to provide a Web-based user interface. The XSLT sub-component transforms XML format into HTML format to display on the Web.

The SOAP over HTTP module contains the SOAP engine including the HTTP server/client, which is a set of packages providing SOAP binding APIs.

The Repository module has two storage types: the XMLDB [36] and a file. XMLDB is required by the manager, and it contains the topology and summary information of the managed devices. A variety of files exists in XCMS. The XSL files in the manager are used in XSLT [11] processing to display XML data on the Web. The notifications and transaction results of RPC operations are deposited into log files. In the manager, the configuration files are uploaded to the agent or downloaded from the agent. In the agent, the configuration files are divided into three steps: candidate, running, and startup. WSDL files are used to describe both base RPC operations and additional RPC operations. The agent has two WSDL files. One is its own WSDL and the other is the manager's WSDL which presents the manager's RPC operations to receive the notification message from the agent. By using the definitions in the WSDL file, the manager can know available RPC operations except the base operations of agents and design its own RPC operation for the notification message.

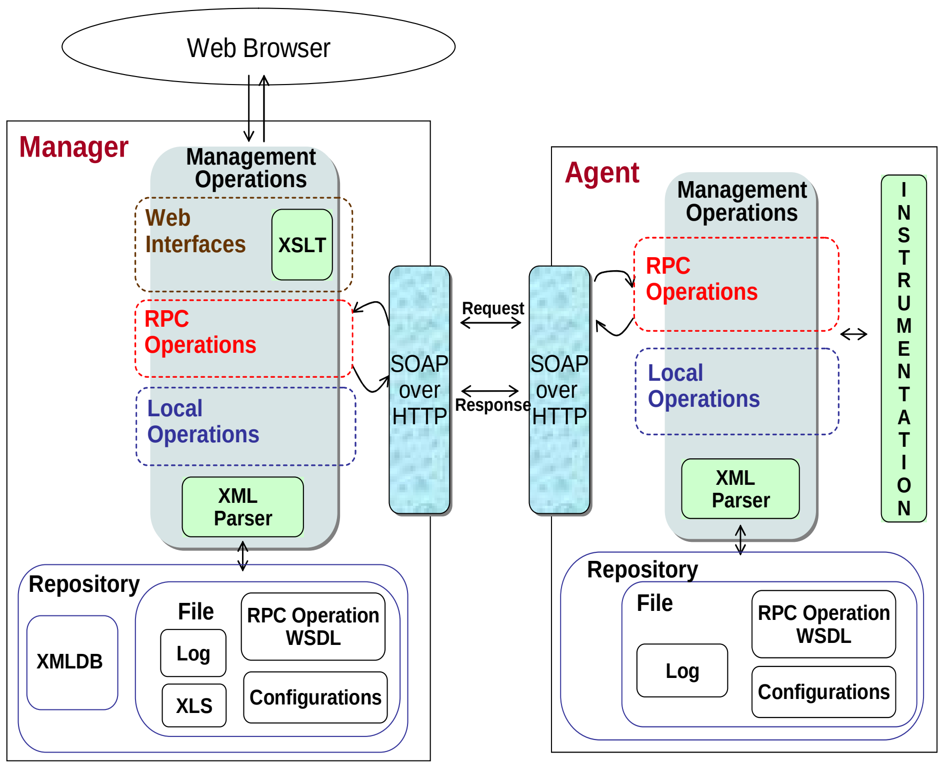


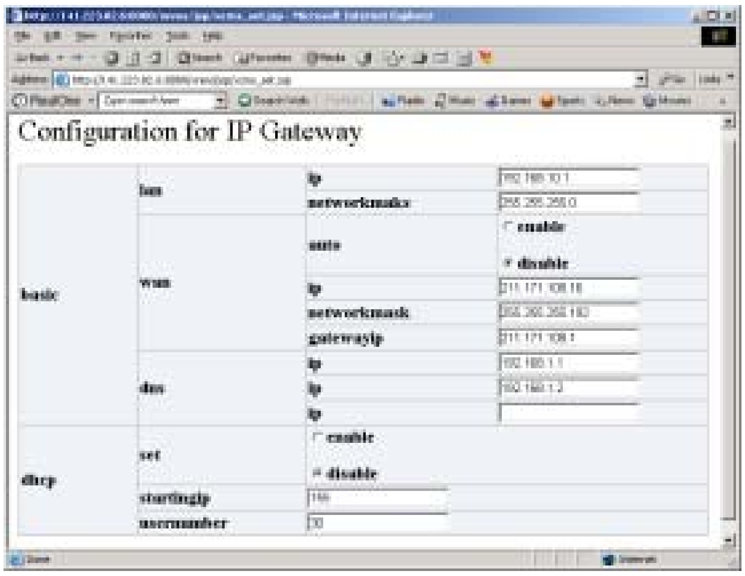
Figure 6. Architecture of XML-based Configuration Management System

4. Implementation

We have implemented an XCMS based on the proposed architecture. We added several operations, such as download, upload, reboot, shutdown, and log to improve the management functionality.

The manager uses various XML technologies to provide Web-based user interfaces and XMLDB. We referred to the Apache Project Group [25] that provides APIs implemented with JAVA. XCMS needs the following APIs: Xerces [26] as an XML parser [12, 30], Xalan [27] as an XPath handler and an XSLT processor, Xindice [28] as XMLDB and AXIS [29] as a SOAP engine to apply the SOAP communication method between the manager and the agents. Axis 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. XMLDB supports XML technologies, such as XPath [6], XQuery [13], and XUpdate [14] to directly handle XML documents via the DOM parser [30].

The XCMS agent is applied to the IP gateway (a commercial Internet sharing device). The agent uses gSOAP [31] implemented with C/C++ languages to exchange SOAP RPC messages. The gSOAP generates a stub, a skeleton, and a WSDL file using the header file which declares RPC operations. We select the libxml [32] which is the lightweight one among existing XML parsers as an XML parser in the agent. This deployment provides DOM APIs which supports to select the specified node, update the selected node value, and read management data.



Configuration for IP Gateway

basic	lan	ip	192.168.10.1
		networkmaks	255.255.255.0
	wan	auto	enable [✓ disable]
		ip	211.171.108.16
		networkmask	255.255.255.192
		gatewayip	211.171.108.1
	dns	ip	192.168.1.1
		ip	192.168.1.2
		ip	
dhcp	set	enable [✓ disable]	
	startingip	155	
	usernumber	30	

(a) Web Interface for Configuration Information of IP Gateway

```
<IPGW>
  <basic>
    <lan>
      <ip>192.168.10.1</ip>
      <networkmask>255.255.255.0</networkmask>
    </lan>
    <wan>
      <auto ref=‘//wanset’>disable</auto>
      <wanset>
        <ip>211.171.108.16</ip>
        <networkmask>255.255.255.192</networkmask>
        <gatewayip>211.171.108.1</gatewayip>
      </wanset>
    </wan>
    <dns>
      <ip name=‘primary’>192.168.1.1</ip>
      <ip name=‘second’>192.168.1.2</ip>
      <ip name=‘third’></ip>
    </dns>
  </basic>
  <dhcp>
    <set ref=‘//wan | //dhcpset’>disable</set>
    <dhcpset>
      <startingip>155</startingip>
      <usernumber>30</usernumber>
    </dhcpset>
  </dhcp>
</IPGW>
```

(b) XML Data for Configuration Information of IP Gateway

Figure 7. Information Model Example

Figure 7 shows an example of simple management information modeling from the Web browser display. Figure 7 (b) shows an XML document matching the Web interface in Figure 7 (a) based on XML Schema. The IP gateway column from the Web Interface represents basic configuration information, such as LAN, WAN, and DNS settings. The fields in this column correspond to each XML element.

The configuration information of a network device has a relationship with each other. The 'ref' attribute in Figure 7 (b) is an example of representing the dependencies among elements. If the automatic setting for WAN is disabled, the IP gateway requires information for WAN, such as the WAN IP address, network mask, and gateway IP address. In contrast, if the automatic setting for WAN is enabled, the information above is ignored. To present this dependency, the sub element 'auto' of 'wan' has the attribute 'ref' whose value is '//wanset'. Another use of the attributes is to identify the same name of the objects. The IP gateway contains IP information of a primary DNS server and the other two DNS servers. To distinguish the 'ip' tags from the 'dns' elements, the attribute 'name' is used as an identifier.

Table 2 shows the examples of XPath expression used in XCMS, and the expression indicates a specific location path with an abbreviated syntax composing of operators and node-set.

XPath	**Description**
//IPGW	Select all the *IPGW* descendants of the root document

//wan/auto	Select all the *auto* descendants of *wan* document
//wan | //dhcpset	Select all the *wan* and the *dhcpset* descendants of the root document
//ip[@name='primary']	Select all the *ip* descendants of the context node that has the attribute *name* with a value *primary*

Table 2. Examples of XPath expression

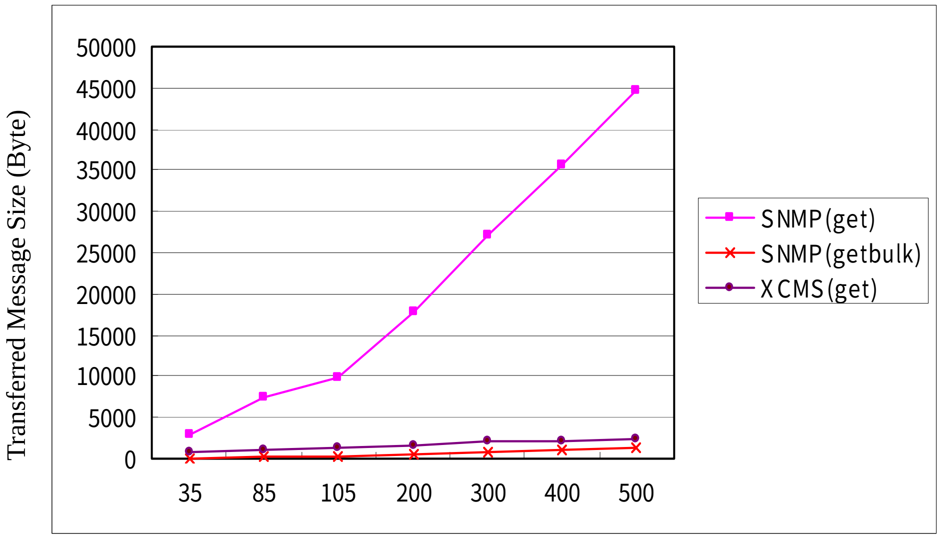
5. Comparative Analysis

In this chapter, the proposed XMML-based Configuration Management System (XCMS) is compared with the SNMP-based Configuration Management System. We also evaluated the latency performance, which is the roundtrip time for sending and receiving messages of Get between the manager and the agent.

5.1. Performance Evaluation

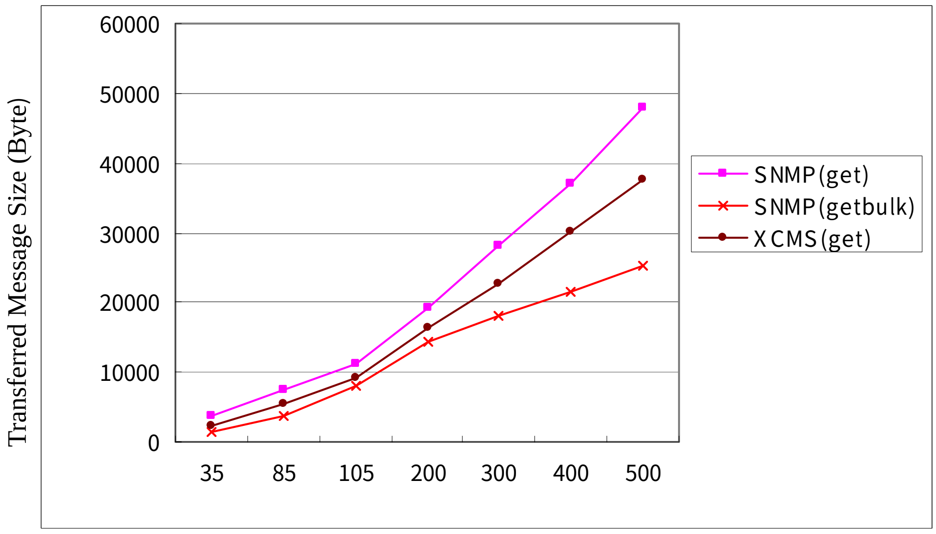
We verified the performance of the XCMS agent by comparing it with the SNMP agent on the same IP sharing device. We compared the generated network traffic between the manager and the agent. Moreover, the processing time of the XCMS agent compared to the traditional SNMP agent is measured with the response time of Get requests between the manager and the agent.

Figure 8 shows the network traffic of each agent between the SNMP agent and the manager, and between the XCMS agent and the manager. We captured packets and different sizes of each manager and agent by network traffic monitoring tool, Ethereal [33]. In the case of one node, 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 XCMS agent using the SOAP over HTTP protocol increases network traffic for each node access over SNMP. Conversely, as the number of the specified nodes is increasing, the XCMS agent produces less network traffic than the SNMP agent using the get operation because the SNMP agent using get operation requests several GetNext operations and receives responses for every node, while the XCMS agent retrieves multiple nodes in one request using SOAP over HTTP.



Number of MIB objects

(a) Get Request Message (Bytes)



Number of MIB objects

(b) Get Response Message (Bytes)

Figure 8. Message Size of Get

Figure 9 presents the same results as Figure 8 (a). The XCMS agent supports to retrieve the bulk data from a request, while the SNMP agent using the get operation should receive several GetNext requests and send the response.

Figure 10 shows the response time of the Get request. The response time is also measured by network traffic monitoring tool, Ethereal. The XCMS takes more time than the SNMP for accessing each leaf node. However, as the number of retrieved nodes is increasing, the response time of XCMS is improving over SNMP.

Although SNMP agent using the getbulk operation shows a better performance than the XCMS agent, the XCMS agent takes advantage to define various operations to improve management functionalities like a set operation for bulk data.

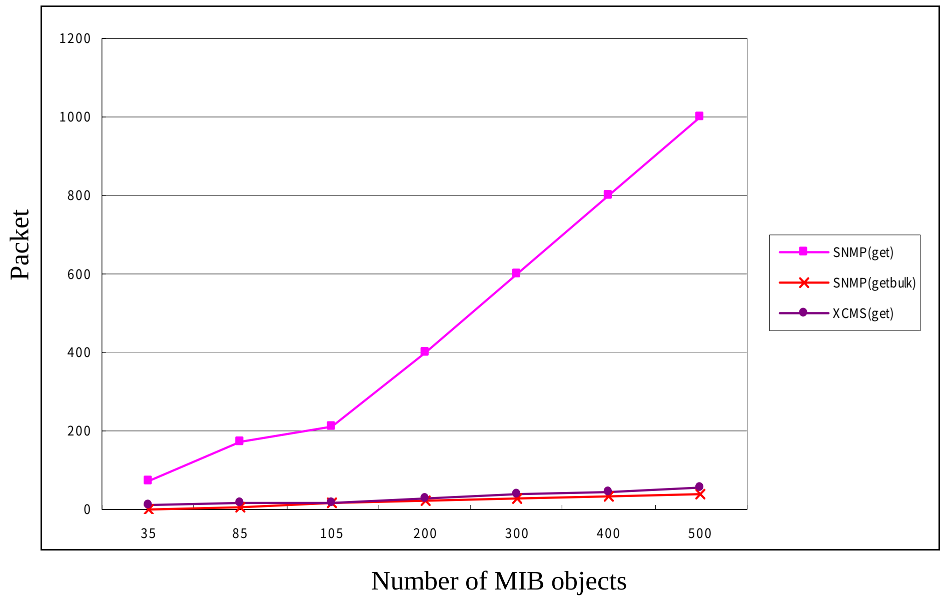


Figure 9. Exchanged Packets

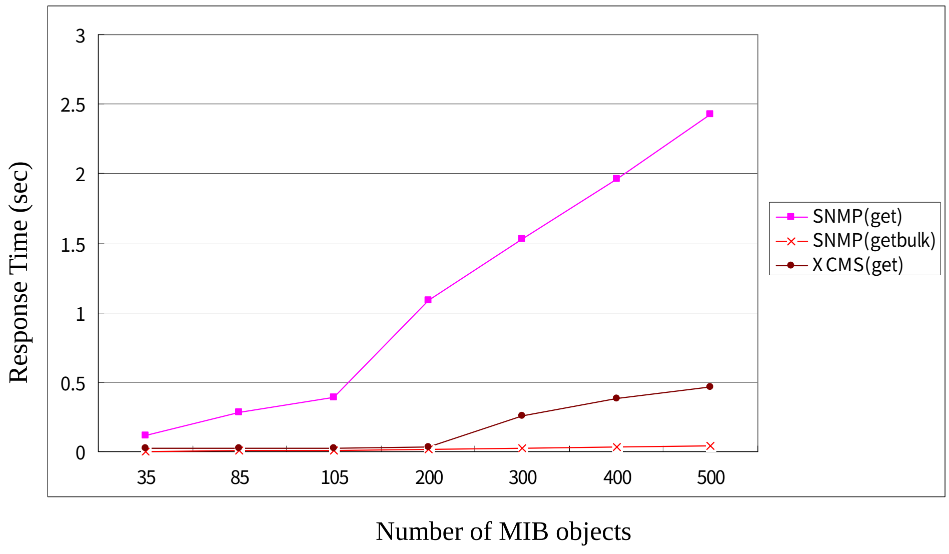


Figure 10. Response Time of Get Request

5.2. Comparison and Analysis

This section presents a comparison and analysis of three separated views: the management protocol, the information modeling, and the addressing method to access managed objects.

As depicted in Figure 11, the SNMP protocol data unit (PDU) is in a binary format with a fixed field size. However, XCMS is the SOAP-based protocol message which is defined in the readable textual format of XML. As compared with SNMP, the advantages of SOAP-based protocol are summarized as follows:

- SOAP message can be exchanged over connection-oriented protocols such as HTTP, which improves the efficiency of bulk data retrieval unlike SNMP over UDP.

<Request PDU>	PDU Type	Request-id	0	0	Variable bindings
<Response PDU>	PDU Type	Request-id	Error-status	Error-index	Variable bindings

<SNMP Message Formats>

```
<SOAP-ENV:Body id="107">
  <get-config>
   <target>
    <running>
  </target>
  <xpath>
    //admin
  </xpath>
 </get-config>
</SOAP-ENV:Body>
```

```
<rpc-reply>
<config>
  ……………………… (XML data)
</config>
</rpc-reply>
```

<SOAP-based Message Formats>

Figure 11. Management Protocol Message

- SOAP message provides Remote Procedure Call (RPC) interface for the direct operation call. If the agent adds a new operation for the configuration management, the manager should directly call the operation using SOAP. However, in the SNMP case, it analyzes the specific value of the defined field to call the operation because SNMP does not provide RPC interface. Therefore, the RPC mechanism is convenient to develop and extend the management operations.
- The SOAP tools generate the WSDL files that contain the definitions of the RPC Operations. By using the WSDL, the manager can easily learn the available RPC operations of the diverse agents except the base operations. Also, the use of WSDL is easy to newly update the existing RPC operations.

The management information modeling presents the management information of the managed system. The SNMP MIB is difficult to present dependencies among objects since MIB represents management information in only a simple hierarchy structure. An XML is possible to represent a certain dependency among nodes using elements and attributes.

The addressing method depends on the information modeling language. SNMP uses an object identifier (OID) expression to access the specific node of MIB. The OID using the position number expresses the defined path from the root to the specific node, so that the OID must present an absolute full path and map each node to its own absolute path. If the structure of the MIB tree is modified, the values of the OID are updated. However, XPath using the naming access can select the specific node from a relative path using an abbreviated syntax or a conditional expression. The XPath also select the multiple nodes from one XPath expression, while the OID points only one node.

Finally, on the configuration management aspect, the use of XML takes many advantages to management protocol, information modeling, and addressing method.

6. Conclusions and Future Work

XML has been viewed by many as a revolutionary approach to solve existing problems in network and systems management. One of the critical problems faced by the operators is the configuration management of multitudes of IP network devices. The effort by the Netconf WG attempts to apply the XML technology to solve the problem of configuration management. This thesis has given a brief overview of the Netconf protocol that is currently being standardized. We have designed and implemented an XML-based configuration management system (XCMS) based on the Netconf protocol with some of our proposed extensions. Also, XCMS guarantees interoperability using SOAP RPC operations defined by WSDL. Although this thesis demonstrated the feasibility of developing an XML-based configuration management system for network devices, we have also demonstrated the use of XML technology for configuration management of distributed systems [34].

We plan to optimize our XML-based agent so that we can embed it into any type of network devices with limited computing resources. We also plan to demonstrate the effectiveness of our approach through extensive performance and scalability tests. Another possible future work is to extend the configuration management to Web Services using WSDL, SOAP, and Universal Description, Discovery and Integration (UDDI) [35].

References

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

[2] J. Schonwalder, A. Pras, J.P. Martin-Flatin, “On the Future of Internet Management Technologies”, IEEE Communications Magazine, Oct. 2003, pp.90~97.

[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, “SOAP Version 1.2 Part 0: Primer”, W3C Working Draft, Dec. 2001.

[9] W3C, “Web Services Description Language (WSDL) Version 1.2” W3C Working Draft, Jul. 2002.

[10] IETF, “Network Configuration (Netconf)”, http://www.ietf.org/html.charters /Netconf-charter.html.

[11] W3C, “XSL Transformations (XSLT) Version 1.0”, W3C Recommendation, Nov. 1999.

[12] W3C, “Simple API for XML Version 2.0”, W3C Recommendation, Nov. 1999.

[13] W3C, “XQuery 1.0: An XML Query Language”, W3C Working Draft, Apr. 2002.

[14] XML:DB, “XUpdate”, http://www.xmldb.org/xupdate/xupdate-wd.html, Sep. 2000.

[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), October 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 the IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Florence, Italy, April 2002, pp. 5-18.

[18] A. John, K. Vanderveen and B. Sugla, “XNAMI-An extensible XML-based paradigm for network and application management instrumentation”, Proc. of IEEE International Conference on Network, 1999. pp. 115–124.

[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] Enns, R., “NETCONF Configuration Protocol”, draft-ietf-Netconf-prot-01 (work in progress), Oct. 2003,< http://www.ietf.org/internet-drafts/draft-ietf-Netconf-prot-01.txt>.

[22] Wasserman, M., “Using the NETCONF Configuration Protocol over Secure Shell (SSH)”, draft-ietf-Netconf-ssh-00 (work in progress), Oct. 2003, <http://www.ietf.org/internet-drafts/draft-ietf-Netconf-ssh-00.txt>.

[23] Lear, E., Crozier, K., Enns, R., “BEEP Application Protocol Mapping for NETCONF”, draft-lear-Netconfbeep-00 (work in progress), Oct. 2003, <http://www.ietf.org/internet-drafts/draft-ietf-Netconf-beep-00.txt>.

[24] Goddard, T., “NETCONF over SOAP”, draft-ietf-Netconf-soap-00 (work in progress), Oct. 2003, <http://www.ietf.org/internet-drafts/draft-ietf-Netconf-soap-00.txt>.

[25] Apache Group, “Apache”, http://www.apache.org/.

[26] Apache XML project, “Xerces Java parser”, http://xml.apache.org/xerces-j/.

[27] Apache XML project, “Xalan Java”, http://xml.apache.org/xalan-j/.

[28] Apache XML project, “Xindice”, http://xml.apache.org/xindice/.

[29] Apache XML project, “Axis”, http://xml.apache.org/axis/.

[30] W3C, “Document Object Model (DOM) Level 2 Core Specification”, W3C Recommendation, Nov. 2000.

[31] 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/.

[32] libxml, “The XML C parser and toolkit at Gnome”, http://www.xmlsoft.org/.

[33] Ethereal, http://www.ethereal.com/.

[34] H. M. Choi, M. J. Choi, and J. W. Hong, “Design and Implementation of XML-based Configuration Management System for Distributed Systems”, Accepted to appear in the Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, Apr. 2004.

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

[36] XML:DB, “XML:DB”, http://www.xmldb.org/.

[37] T.Berners-Lee, R. Fielding, and L. Masinter, “Uniform Resource Identifiers (URI): Generic Syntax”, IETF RFC 2396, August 1998.

요 약

현재 인터넷의 빠른 확산으로 다양한 네트워크 장치들이 출현하게 되었고, 거대한 네트워크가 형성되었다. 현재 네트워크 관리를 위해 사용되고 있는 SNMP(Simple Network Management Protocol) 기반의 관리시스템은 나날이 복잡해지고 거대해진 네트워크를 수용하기에는 관리 정보 모델과 관리기능 측면에서 취약함이 지적되고 있다. 또한 UDP를 기반으로 한정된 크기의 SNMP 메시지를 보내기 때문에 벌크 데이터(bulk data)를 처리 하는데 있어서 가장 큰 문제가 되고 있으며, 이는 구성관리(Configuration Management)의 어려움을 더욱 가중시키게 된다. 따라서 SNMP의 대체 방법인 XML(Extensible Markup Language) 기반의 네트워크 관리는 구성관리를 중심으로 IETF의 워킹그룹(WG)인 Netconf(Network Configuration)에서 표준화 작업이 활발히 진행되고 있다. 우리는 본 논문에서 Netconf에서 제시한 관리 프로토콜의 문제점을 지적하고 이를 개선한 XML 기반의 구성 정보 관리 시스템(XCMS)의 구조를 제안하였다. XCMS는 관리 프로토콜로써 RPC(Remote Procedure Call) interface를 제공하고, WSDL(Web Services Description Language)을 정의 할 수 있는 SOAP을 이용하고 있다. 그리고 본 논문은 임베디드 시스템인 XCMS의 에이전트 고려하여 관리 정보를 추출하기 위해 필요한 XPath 표현들을 제안하였고, 구성 정보 객체간의 연관성을 고려한 XML기반의 구성 정보 모델링 방법을 제시하였다. 또한 성능 평가를 통하여 기존의 SNMP 방법보다 XCMS가 구성관리 측면에서 효율성을 증명하였다. 우리는 현재 XCMS 의 성능 및 효율성 향상을 위하여 XCMS 에이전트를 최적화하는 작업을 하고 있다. 그리고 더 나아가 새로운 분산 환경을 제공하고 있는 Web Services 기술을 XCMS에도 적용해 보는 연구 또한 진행되고 있다.

감사의 글

석사 2년 동안 부족한 저를 지도해 주시고 돌봐주신 홍원기 교수님께 진심으로 고개 숙여 감사 드립니다. 교수님의 가르치심은 언제나 저에게는 삶의 밑거름이 될 것입니다. 더불어 저의 논문 심사를 위해 애써 주신 김종 교수님과 서영주 교수님께도 깊은 감사를 드립니다.

짧은 석사기간 동안이었지만 연구실 선후배 및 동료 여러분 그동안 고마웠습니다. 연구실 생활에 많은 도움을 주고 제 연구에 관심을 갖고 돌봐준 명섭선배 감사합니다. 좋은 논문 쓰시고 곧 졸업하시길 바라겠습니다. 매번 영어 논문 쓸 때 마다 함께 고생하고 제게 힘이 되어준 미정언니 너무나 감사합니다. 외국에서의 포스닥 생활도 지금처럼 즐겁고 힘차게 하시길 바라겠습니다. 포항의 후배들을 위해 끊임 없는 관심을 보여주시는 홍택 선배님과 석사 1년차 때 동기처럼 같이 어울렸던 효진이, 윤정언니, 훈정선배, 세희선배에게도 감사의 마음을 전합니다. 그리고 항상 따뜻하게 옆에서 응원해 준 착한 후배 소정이와 뒤늦게 우리 팀에 합류하여 많은 도움을 준 동현선배, 그리고 즐겁게 연구실 생활을 할 수 있게 도와준 후배 성철, 승화, 영준, 형조에게도 고마움과 함께 앞으로 남은 석사 생활 잘 하길 바랍니다. 2% 부족한 우리의 연구실 생활을 채워주는 은희 언니와 열심히 자신의 일을 하는 멋진 영미 언니에게도 감사합니다. Nice to meet you. Thank you, Deepali. I hope you have a good time in DPNM. 헤어지게 되어 아쉽지만, DPNM 연구실에서의 보낸 시간들은 제게 너무나도 소중하고 감사합니다.

작은 고민이라도 마음 터놓고 생활했던 여자 기숙사 2동 209호 여전사들 문경, 주영, 그리고 평생 내 옆에서 든든한 벗이 되어줄 수영, 미섭, 영란에게도 감사의 마음을 전합니다.

지금까지 뒷바리지 해준 부모님과 물심양면으로 힘이 되어준 큰오빠, 큰언니, 작은 오빠에게 그리고 곧 세상에 태어날 첫 조카에게도 감사하다는 글을 올립니다.

이 력 서

성 명 : 최 현 미

생 년 월 일 : 1978년 4월 23일

출 생 지 : 강원도 강릉시

주 소 : 충북 충주시 교현 2동 부강 아파트 101동 703호

학 력

1997.3 – 2002.2 : 홍익 대학교 컴퓨터 공학과 (B.S.)

2002.3 – 2004.2 : 포항공과대학교 정보통신대학원 정보통신학과 (M.S.)

학 술 활 동

♦Conference Papers

- Hyoun-Mi Choi, Mi-Jung Choi, James W. Hong, "Design and Implementation of XML-based Configuration Management System for Distributed Systems", Accepted to appear in the Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, April 2004.
- 최현미, 최미정, 홍원기, 박용석, “NETCONF 표준을 따른 XML 기반의 네트워크 구성관리 시스템”, Accepted to appear in the KNOM Review, Vol.6, No.2, December 2003.
- Hyoun-Mi Choi, Mi-Jung Choi James W. Hong, "XML-Based Configuration Management for Distributed System", Proc. of 2003 Asia-Pacific Network Operations and Management Symposium (APNOMS 2003), Fukuoka, Japan, October 1-3, 2003, pp. 599-600.
- 최현미, 최미정, 홍원기, "분산 시스템의 구성관리를 위한 XML 기반의 관리 시스템 설계 및 구현”, Proc. of KNOM 2003 Conference, Daejeon, Korea, May 22-23, 2003, pp. 330-337.

연구 활동

♦Projects

- Development of XML-based Network Management System (XNMS) (XML 기반의 네트워크 관리 시스템 개발), DPNM Project, 2003
- Development of XML-based Configuration Management System for IP Network Devices (XCMS) (네트워크 장비를 위한 XML 기반의 구성 관리 시스템 개발), DPNM Project, 2003