

XML-Based Configuration Management for IP Network Devices

*Mi-Jung Choi, Hyoun-Mi Choi, and James W. Hong, POSTECH
Hong-Taek Ju, Keimyung University*

ABSTRACT

As the Internet continues to grow, the tasks of operations and management of IP networks and systems are becoming more difficult. Over the past few years, much effort has been given to improve the deficiencies of SNMP, but most have failed to be standardized. One critical deficiency of SNMP is in the area of configuration management. Recent work focuses on the use of XML technology for network and service management as an alternative or complementary approach to SNMP. This article presents the IETF's latest effort, Netconf, which is viewed by many as a promising revolutionary solution for configuration management. To validate this new effort, we present the design and implementation of an XML-based configuration management system based on Netconf. We also discuss our experience with XCMS and make some suggestions for improving the current Netconf protocol.

INTRODUCTION

The rapid pace of Internet evolution is currently witnessing the emergence of diverse network devices. IP networks are becoming larger and more complex as more people use the Internet and more enterprises rely on the Internet for their businesses. Efficient management techniques and tools are necessary to manage these networks. The Simple Network Management Protocol (SNMP) has been the most widely used method for network management on the Internet since it was introduced in the late 1980s. However, SNMP has been used mostly in monitoring for fault and performance management, but was hardly used for configuration management due to its limitations [1].

To overcome the shortcomings of SNMP, Extensible Markup Language (XML) technologies, such as the XML schema, XML Path Language (XPath), Extensible Style-Sheet Language (XSL), SOAP, and Web Services Description Language (WSDL), are currently being applied to configuration management [2]. The Internet Engineering Task Force's (IETF's) standardization process of configuration management for network devices using XML technologies is also in progress in the Network Configuration (Netconf) working group (WG) [3].

SNMP defines management information using Structure of Management Information (SMI) whose data modeling capability is limited to simple tables of scalar data types. However, network configuration data such as interface settings and routing tables requires a hierarchical information model. Thus, SNMP SMI is insufficient to represent network configuration data because it does not support a hierarchical information model. Netconf uses XML schema or Document Type Definition (DTD) instead of SNMP SMI [3]. The hierarchical representation of XML is better suited to network configuration data than SMI's relational model. The object identifier (OID), which is a naming mechanism of SNMP, is so simple and verbose that it is very inefficient in usage and implementation. Netconf has not yet specified any specific naming scheme. For simplicity, the Netconf protocol message includes the portions of configuration subtrees to indicate the target of management operations. In this article we propose to use a subset of the XPath to specify the target of management operations. The explicit and unique naming method of XPath is more efficient and powerful than the simple OID in SNMP [3].

Configuration tasks require several high-level management operations such as download, activation, rollback, and restoration. The SNMP Set operation can be used to realize such operations as side-effects, but it makes management applications very complicated. Therefore, with SNMP it is difficult to support various operations such as to load/restore configuration, activate a new configuration at a specific time, and roll back a configuration [3]. Netconf uses a remote procedure call (RPC) paradigm to define a formal application programming interface (API) for the network device. Also, Netconf defines these management operations invoked as RPC methods with XML-encoded parameters.

UDP is the preferred transport of SNMP for IPv4. The size of SNMP over UDP messages is usually limited by the size of the maximum transmission unit (MTU), which is insufficient for bulk configuration data transfers. Also, SNMP retransmits important data at the SNMP engine implementation or application code level because of unreliability in UDP. To overcome the weaknesses of SNMP, evolutionary

approaches have been attempted in the past few years, but all have failed to be adopted as standards [1]. Netconf, however, is connection-oriented, requiring a persistent connection between the manager and agent. This connection provides a reliable and sequential data delivery.

Command line interfaces (CLIs) have been widely used as an alternative to SNMP for configuration management. The CLI also has some drawbacks in the lack of a common data model, a proprietary interface, and insufficient access control. These drawbacks prohibit the CLI from being used to configure multiple devices in a heterogeneous environment. Netconf does not yet provide a solution for the common data model or access control. However, it provides a solution for the standard interface to guarantee interoperability in managing devices in a multivendor environment.

Major network device vendors, such as Cisco and Juniper Networks, already integrate their own XML-based agent in their products and actively participate in the Netconf standardization work. Netconf announced Internet drafts for a Netconf configuration protocol and transport bindings. Until now, little implementation work has been done using Netconf. For the successful deployment of a standard management protocol, we need to demonstrate the feasibility and effectiveness of the new protocol to operators and administrators. This article demonstrates this by designing and implementing an XML-based configuration management system based on Netconf. This article also identifies problems in the current Netconf proposal and suggests possible solutions.

The organization of this article is as follows. First, we present an overview of Netconf and introduce the XML-based configuration management system (XCMS). XCMS is applied to manage the IP sharing device, which is a small network address translation (NAT) appliance used in homes and small offices/home offices (SOHOs) in conjunction with high-speed Internet access for allowing up to 254 IP devices using private IP addresses with a single public IP address. Using our implementation experience with the IP sharing device, we suggest several improvements for Netconf. Last, we summarize our work and discuss directions for future research.

AN OVERVIEW OF NETCONF

The Netconf WG [3] was formed in May 2003. The Netconf WG is attempting to standardize a protocol suitable for the 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 [4–7] and individual submissions. In this section we introduce the Netconf protocol and transport mappings.

The features of Netconf protocol [4] are summarized as follows:

- Netconf uses an RPC paradigm to define a formal API for network devices. A manager encodes an RPC in XML and sends it to the agent using a secure connection-oriented session.
- A Netconf session is the logical connection

Layer	Example
Content	Configuration data (IP address, Admin ID/password, etc.)
Operations	<get-config>, <edit-config>, etc.
RPC	<rpc>, <rpc-reply>, etc.
Transport	SSH, BEEP, SOAP over HTTP, etc.

■ **Table 1.** *Netconf protocol layers.*

between a network administrator or network configuration application and a network device. A device must support at least one Netconf session, and may support more than one.

- It distinguishes between configuration data and state data. Configuration data is the set of read-write data, and state data is read-only data.
- Netconf is connection-oriented, requiring a persistent connection between the manager and agent. This connection must provide reliable and sequential data delivery.
- It is necessary to distinguish between the distribution of a configuration and the activation of configurations.

NETCONF CONFIGURATION PROTOCOL

The Netconf protocol [4] uses XML for data encoding and a simple RPC-based mechanism to facilitate communication between a manager and an agent. 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 operations for getting and editing full or partial configurations.
- 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 data such as IP address and administrator ID/password, 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 operations layer includes base and additional management operations. The base operations of the Netconf protocol are defined as follows:

- <get-config>: retrieves all or parts of the specified configuration.
- <edit-config>: modifies a configuration. The <edit-config> base operation has an argument that describes the details of the configuration change. An operation attribute, which is embedded in a configuration subtree, marks the point of the hierarchy at which to perform the operation determined by the value of the operation attribute. The operation attribute has one of three values: merge, replace, and delete. It merges or replaces either all or parts of the specified configuration to the specified

We define the configuration information of a network device using the XML Schema, which supports the definition of management information by adding new data types to a pool of data types.

target configuration. It also deletes all or part of the specified configuration. The transaction of modification operations is provided optionally. The values for transaction processing are stop-on-error (default) and ignore-error.

- **<copy-config>**: creates or replaces an entire configuration datastore with the contents of another complete configuration datastore.
- **<delete-config>**: deletes a configuration datastore.
- **<kill-session>**: terminates a Netconf session and releases all resources bound to the session.
- **<lock>**: allows the manager to lock the configuration of a device.
- **<unlock>**: releases a configuration lock previously obtained with a **<lock>** operation.
- **<get-all>**: retrieves configuration and state information from a device.

Netconf defines configuration datastores and allows configuration operations on them. There are some special configuration datastores for the **<running>** configuration currently running on the network device, the **<startup>** configuration used at the next reboot and a **<candidate>** configuration that is used temporarily to make and validate changes. Only the **<running>** configuration datastore is present in the base model.

A set of additional functionalities that supplements the base operations are called *capabilities* in Netconf. Capabilities augment the base operations of the device, describing both additional operations and the content allowed inside operations. The Netconf capability permits the client to adjust its behavior to take advantage of features exposed by the device.

There are capabilities such as manager, agent, writable-running, candidate, and validate. The manager capability is the manager's capability to manage the agent using the Netconf protocol. The agent capability indicates the agent's capability to be managed. The writable-running capability indicates that the device supports writes directly to the **<running>** configuration datastore. The candidate capability indicates that the device supports a candidate configuration datastore, which is used to hold configuration data that can be manipulated without impacting the device's current configuration. The validate capability consists of checking a candidate configuration for syntactical and semantic errors before applying the configuration to the device.

To support capabilities, more operations are defined:

- **<commit>**: commits the candidate configuration as the device's new configuration
- **<discard-changes>**: reverts the candidate configuration to the current committed configuration, if the manager decides that the candidate configuration should not be committed
- **<validate>**: validates the contents of the specified configuration

The RPC layer presents the RPC-based communication model. This layer merely uses RPC elements to define XML messages. Netconf peers use **<rpc>** and **<rpc-reply>** elements to

provide a transport-independent framing mechanism for encoding RPCs. The elements in this layer are as follows:

- **<rpc>**: expresses request messages of operations in the Netconf protocol.
- **<rpc-reply>**: presents the response message to an **<rpc>** request. The **<ok>** element is sent in **<rpc-reply>** messages if no error occurs during the processing of an **<rpc>** request. Otherwise, the **<rpc-error>** element is delivered in **<rpc-reply>** messages.

The transport layer supports any transport protocol that is connection-oriented, requiring a long-lived and persistent connection between the manager and agent. This allows the manager to make changes to the state of the connection that will persist for the lifetime of the connection. In addition, resources requested from the agent for a particular connection must be automatically released when the connection closes, making failure recovery simpler and more robust.

NETCONF TRANSPORT PROTOCOL

The Netconf protocol is currently considering three separate protocol bindings for transport: Secure Shell (SSH) [5], SOAP over HTTP [6], and Block Extensible Exchange Protocol (BEEP) [7]. The SSH transport binding uses SSH security methods for key exchange and authentication. However, the other approaches merely recommend the use of existing security infrastructures within transport such as TLS and HTTPS. Recently, Netconf has selected SSH [5] as a mandatory transport protocol because SSH is widely deployed. SSH is required by operators who are accustomed to the existing network device configuration environment (i.e., a command line shell environment).

SOAP over HTTP [6] is a natural application protocol for Netconf, essentially because it supports its own RPC interface. However, care must be taken in some cases, as HTTP is inherently synchronous and client-driven. Netconf finds many benefits in this environment: reuse of existing standards, ease of software development, and integration with deployed systems. SOAP tools and source codes provide an easy implementation environment that generates the skeleton and stub for communicating. The manager initiates an HTTP connection to an agent and drives the Netconf session via a sequence of SOAP messages over HTTP requests. The Netconf session state contains the following: authentication information, capability information, locks, pending operations, and operation sequence numbers. The Netconf SOAP binding relies on an underlying secure transport for integrity and privacy. Such transports are expected to include TLS and IPsec. HTTP and SOAP level authentication can be integrated with RADIUS to support remote authentication databases.

BEEP [7] allows either the manager or the agent to initiate the connection. Because of the nature of this connection initiation, BEEP is superior to SOAP over HTTP, especially in processing asynchronous notifications. However, the downside is that BEEP is not widely deployed at this time, in neither network devices nor end user operating systems.

AN XML-BASED CONFIGURATION MANAGEMENT SYSTEM

In this section we describe the design and implementation of an XML-based configuration management system (XCMS) for IP sharing devices in accordance with Netconf. Also, we propose improvements in the Netconf protocol based on our experience.

THE MANAGEMENT INFORMATION MODEL

Information modeling defines the management information of the managed system. We define the configuration information of a network device using the XML schema, which supports the definition of management information by adding new data types to a pool of data types.

Figure 1 shows an example of simple management information modeling from a Web browser display for an IPv4-based IP sharing device. Figure 1b shows an XML document matching the Web interface in Fig. 1a based on the XML schema. The IP sharing device 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 other information of the network device. We use attributes in an XML document to express relationship and identification. For example, the `ref` attribute in Fig. 1b represents dependencies among elements. The value of the `ref` attribute is an XPath expression. If the automatic setting for WAN is disabled, the IP sharing device 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 subelement `auto` of `wan` has the attribute `ref` whose value is `//wanset`. Another use of attributes is to identify the same name of objects. The IP sharing device 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.

MANAGEMENT PROTOCOL

We have selected SOAP over HTTP as the transport protocol. In 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. However, SOAP 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 [8] definitions, which contain useful information for RPC operations, such as operation names and parameters. When a new operation is added to the agent's operation list, the new version of a WSDL definition is generated. The updated WSDL definition is the specification to notify the manager with newly added operations.

We have defined management operations using SOAP RPC operations based on Netconf operations as follows. We added several opera-

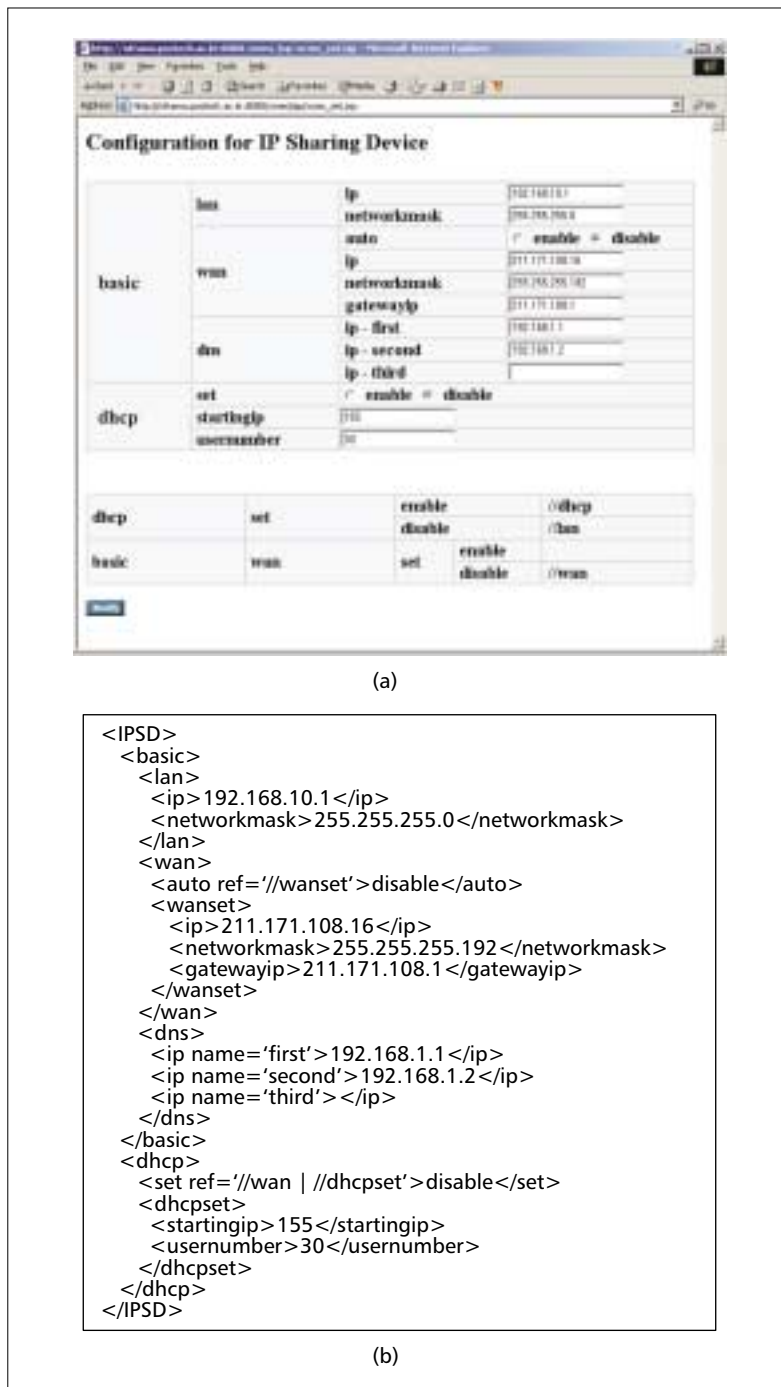
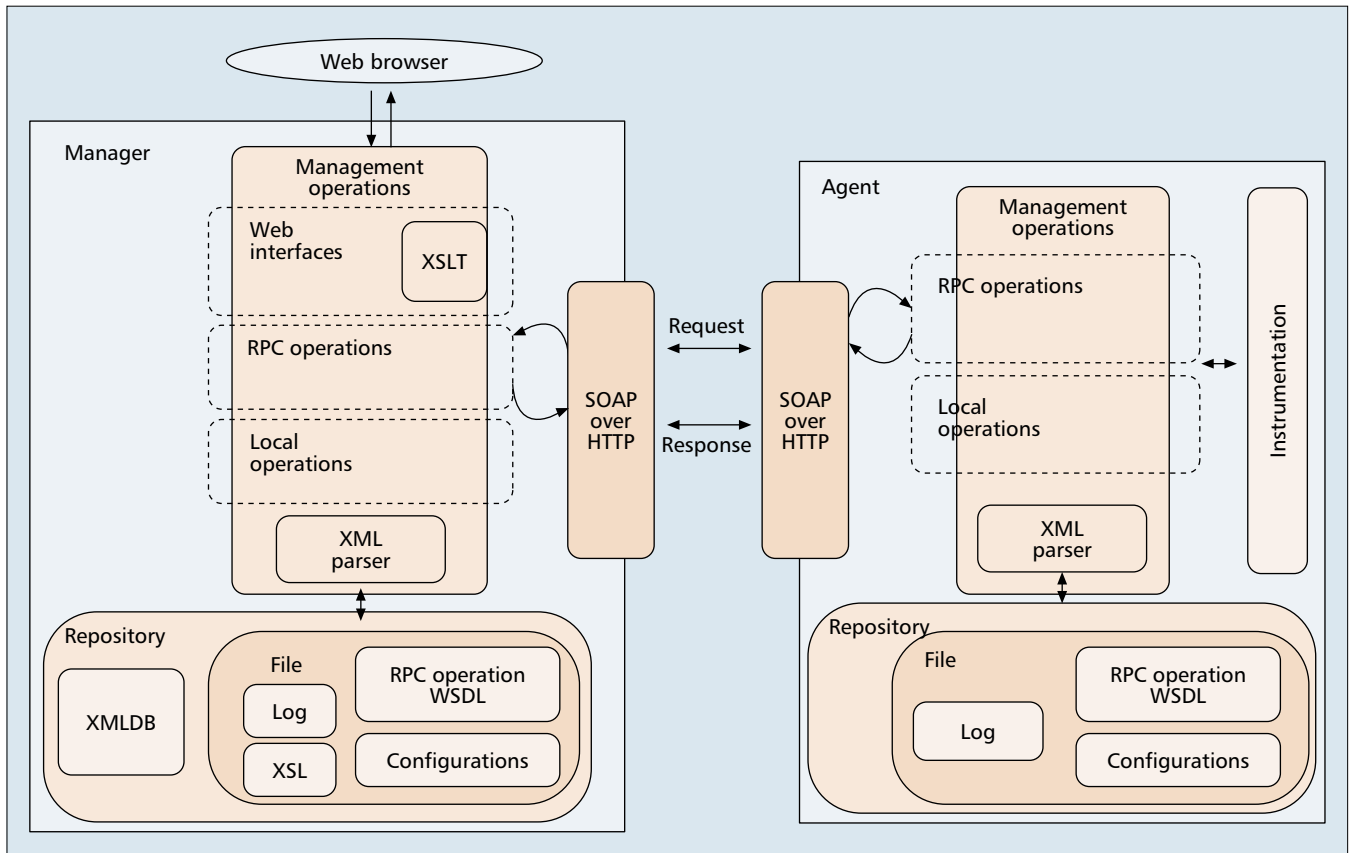


Figure 1. Information model example: a) the Web interface for configuration information of IP sharing device; b) an XML document for configuration information of an IP sharing device.

tions (e.g., reboot, shutdown, and log) to improve management functionality:

- `<get-config>`: retrieves configuration information.
- `<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 terminating a specific session, this operation disconnects all channels within the session.
- `<lock>`: locks the configuration.



■ **Figure 2.** The architecture of the XML-based configuration management system.

- `<unlock>`: releases previously locked configuration data.
- `<get-all>`: retrieves configuration and device state information.
- `<reboot>` `<shutdown>`: agent reboots or shuts itself down to apply the modified configuration information.
- `<log>`: logs notification and transaction results of the management operations.

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 expression, which provides explicit expressions to identify XML nodes. A role of XPath is to indicate a specific location path of configuration information with an abbreviated or unabbreviated 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. It requires removing an unnecessary expression in 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 using only 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 management services using WSDL. Table 2a describes the WSDL definition of the `<edit-config-replace>` RPC operation. Table 2b describes a request message binding SOAP to call `edit-config-replace` Request operation in the agent. Table 2c is the response message to the request shown in Table 2b). This operation includes the following parameters: `target`, `xpath`, `config`, `transaction`, and `sessionId`. The parameter `target` is a target configuration that is one of the states (e.g., 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.

ARCHITECTURE

Figure 2 illustrates the architecture of XCMS, consisting of a manager and agent. Both the manager and 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 consists of several components and serves management needs. The RPC Operations component includes the SOAP RPC operations with their own RPC interfaces. The Local Operations component is a set of internal operations that supplements the RPC operations. 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 the additional Web Interfaces compo-

(a) WSDL elements			
Definitions	Name	Remarks	Description
Request message	edit-config-replaceRequest	parameter target (type:string) parameter xpath (type:string) parameter config (type:string) parameter transaction (type:string) parameter sessionID (type:string)	
Response message	edit-config-replaceResponse	parameter rpc-reply (type:string)	
PortType	edit-config-replacePortType	operation edit-config-replace	input message edit-config-replaceRequest output message edit-config-replaceResponse
Binding	edit-config-replaceBinding	type edit-config-replacePortType operation edit-config-replace	input message edit-config-replaceRequest output message edit-config-replaceResponse
Service	edit-config-replaceService	port edit-config-replaceBinding name edit-config-replacePort	address location http://hostname:8080/axis/ edit-config-replaceService.jws

(b) SOAP request message (edit-config-replaceRequest)
<pre><?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/XMLSchemainstance" 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:edit-config-replaceRequest> <target><running/></target> <xpath>//lan</xpath> <config><lan><ip>1.2.3.4</ip><networkmask>255.255.255.0</networkmask></lan></config> <transaction>rollback</transaction> <sessionID>123</sessionID> </xconf:edit-config-replaceRequest > </SOAP-ENV:Body> </SOAP-ENV:Envelope></pre>

(c) SOAP response message (edit-config-replaceResponse)
<pre><?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/XMLSchemainstance" 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:edit-config-replaceResponse > <rpc-reply>OK</rpc-reply> </xconf:edit-config-replaceResponse> </SOAP-ENV:Body> </SOAP-ENV:Envelope></pre>

■ **Table 2.** The WSDL definition of `<edit-config-replace>` operation.

ment to provide a Web-based user interface. The XSLT subcomponent 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 that provides SOAP binding APIs. The Repository module has two storage types: the XMLDB and a file. The XMLDB is a necessary component in the manager, which contains the topology and summary information of the managed devices. A variety of files exist in XCMS. The XSL files in the manager are used in XSLT processing to display XML data on the Web. The notifications and transaction results of RPC operations are deposited in log files. In the manager, the configuration files are uploaded to or downloaded from the agent. In the agent, the configuration files are divided into three states: candidate, running, and startup.

WSDL files are used for describing additional RPC operations, as well as base RPC operations.

IMPLEMENTATION

We have implemented an XCMS based on the proposed architecture. The manager uses various XML technologies to provide Web-based user interfaces, communicate between manager and agent, and process management information. We used the Apache software [9] that provides APIs implemented with JAVA. XCMS 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 XCMS agent is applied to the IP sharing device. The agent uses gSOAP [10] implemented

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.

with C/C++ languages to exchange SOAP messages. The gSOAP generates a stub, a skeleton, and a WSDL definition using the header file that declares RPC operations. We selected libxml, which is more lightweight than most XML parsers in the agent in consideration of the low computing resources of the IP sharing device.

EXPERIENCE AND SUGGESTIONS

The management information from diverse network devices developed by different vendors is difficult to define in a standard format. Although the Netconf WG realizes the importance of the content layer, it is not concerned with management information yet. Our management information modeling provides guidelines for the presentation of management information in XML format.

Also, Netconf does not yet provide an addressing mechanism for the selection of a specific part of configuration information. 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 the specific configuration. 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 identification of the configuration information. By using XPath, the operation edit-config can be separated into edit-config-merge, edit-config-replace, and edit-config-delete. A detailed proposal about the use of a subset of the XPath was provided in the Netconf mailing list archives [3]. We implemented our system based on this proposal. The approach, which uses XPath for naming the target of a management operation, provides a more robust model for expressing management information. This approach results in a more efficient and reliable protocol in which transactions are expressed with precision and completeness, and can be accurately interpreted by examination without referring to the exact configuration state of the managed device [3].

In the Netconf over SOAP Internet draft [6], 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 Fig. 3. The SOAP RPC operation layer is a combination of Netconf's operation layer and the RPC layer. The names of operations in Netconf become SOAP RPC operations. The necessary information is passed as parameters.

While SOAP can be bound to different underlying protocols such as HTTP, SMTP, or BEEP, most existing SOAP implementations use HTTP. In the Netconf over SOAP Internet draft, 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 due to periodic requests. Periodic request

traffic is about 250 bytes, and response traffic excluding notification is about 210 bytes. If the monitoring period is short and no notification data exists, the traffic overhead increases. 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. Currently, a message exchange pattern (MEP) of SOAP 1.2 Part 2 and PAOS (reversed HTTP binding for SOAP) from the Liberty Alliance Project are doing research on asynchronous notification.

A session ID in every request message is the necessary information for processing operations, such as kill-session and lock. However, the Netconf protocol does not yet specify a standard mechanism to define a session ID in a message. In the Netconf over SOAP Internet draft, if the manager opens a new session with an agent, the manager sends a message with an empty session identifier. The agent must determine whether or not a session identifier is empty in the messages. Because there is no universal method to represent a session ID, the agent has difficulty in obtaining the session ID from the message. The session ID must exist within the Netconf protocol message.

It is possible to define complex management information such as a relationship definition by using the XML schema. In an SNMP MIB, the relationship is not expressed at all. Based on an efficient information model, our XCMS supports various operations, such as activation, rollback, and lock, that are typically not supported in SNMP-based network management. Also, the naming mechanism of XPath is sufficient to indicate a concise subtree of the management information.

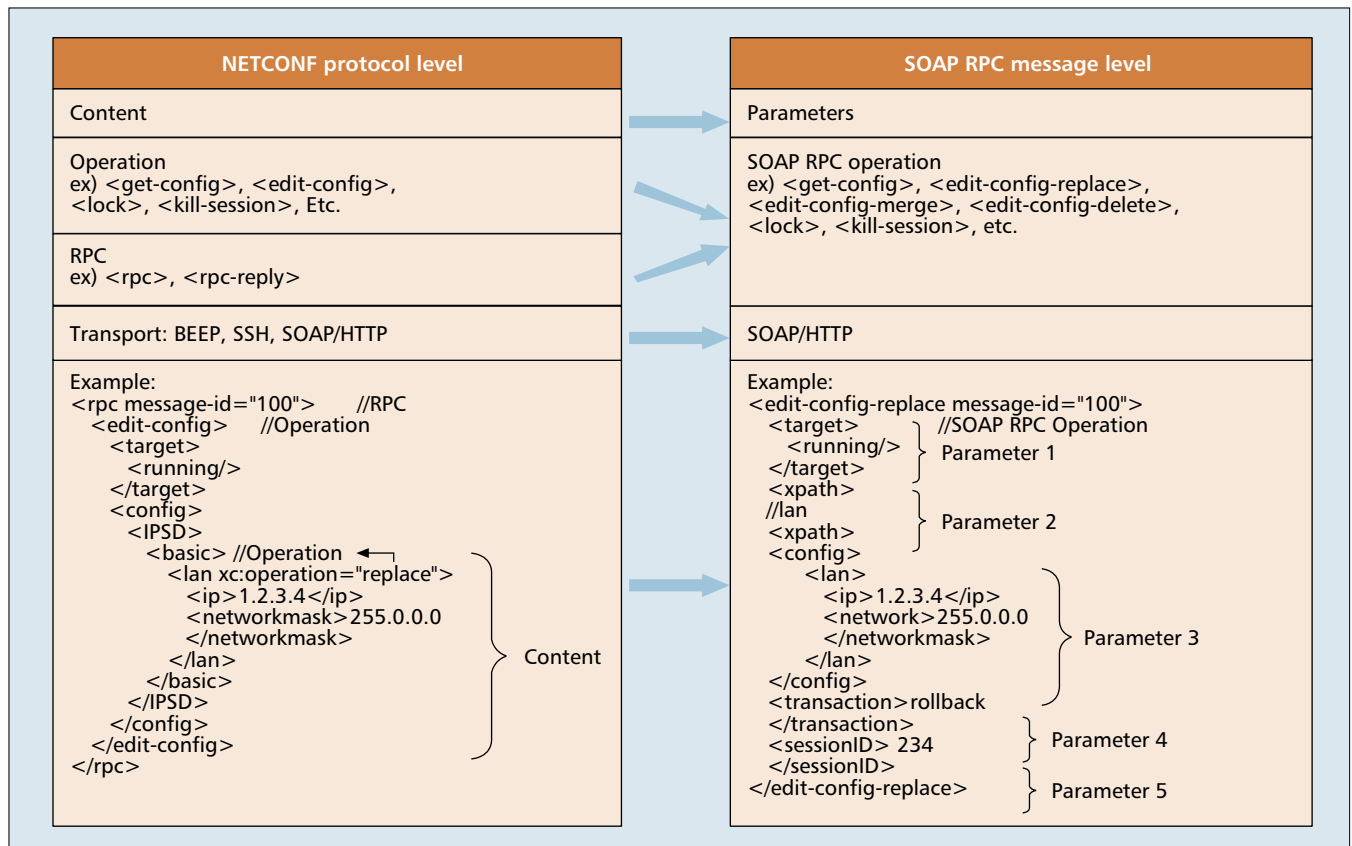
CONCLUDING REMARKS

XML has been viewed by many as a revolutionary approach to solving existing problems in network and systems management. One of the critical problems faced by operators is the configuration management of multitudes of IP network devices. The Netconf WG attempts to apply XML technology to solve the problem of configuration management. This article has given a brief overview of the Netconf protocol currently being standardized. We have designed and implemented an XCMS based on the Netconf protocol with some of our proposed extensions. This article has demonstrated the feasibility of developing an XML-based configuration management system for network devices. In a related work, we have demonstrated the use of XML technology for the configuration management of distributed systems [11].

We plan to optimize our XML-based agent so that we can embed it into any type of network device with limited computing resources. We also plan to demonstrate the effectiveness of our approach through extensive performance and scalability tests.

ACKNOWLEDGMENTS

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



■ Figure 3. Mapping of a Netconf protocol message to a SOAP message.

program of Ministry of Information and Communication, Korea.

REFERENCES

- [1] J. Schonwalder, A. Pras, and J. P. Martin-Flatin, "On the Future of Internet Management Technologies," *IEEE Commun. Mag.*, Oct. 2003, pp. 90–97.
- [2] M. J. Choi, J. W. Hong and H. T. Ju, "XML-Based Network Management for IP Networks," *ETRI J.*, vol. 25, no. 6, Dec. 2003, pp. 445–63.
- [3] IETF, "Network Configuration (Netconf)," <http://www.ietf.org/html.charters/netconf-charter.html>.
- [4] R. Enns, "NETCONF Configuration Protocol," draft-ietf-netconf-prot-02, Feb. 2004, work in progress.
- [5] M. Wasserman and T. Goddard, "Using the NETCONF Configuration Protocol over Secure Shell (SSH)," draft-ietf-netconf-ssh-00, Oct. 2003, work in progress.
- [6] T. Goddard, "NETCONF over SOAP," draft-ietf-netconf-soap-01, Feb. 2004, work in progress.
- [7] E. Lear and K. Crozier, "BEEP Application Protocol Mapping for NETCONF," draft-ietf-netconf-beep-00, Oct. 2003, work in progress.
- [8] E. Christensen *et al.*, "Web Services Description Language (WSDL) 1.1," W3C Note, NOTE-wsdl-20010315, Mar. 2001.
- [9] Apache Software Foundations, <http://www.apache.org/>
- [10] A. Robert, V. Engelen, and A. Kyle "The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks," *Proc. 2nd IEEE/ACM Int'l. Symp. Cluster Comp. and the Grid*, Berlin, Germany, May 2002, pp. 128–35.
- [11] H. M. Choi, M. J. Choi, and J. W. Hong, "Design and Implementation of XML-Based Configuration Management System for Distributed Systems," *Proc. IEEE/IFIP NOMS 2004*, Seoul, Korea, Apr. 2004, pp. 831–44.

BIOGRAPHIES

MI-JUNG CHOI [M] (mjchoi@postech.ac.kr) received her B.S. degree in computer science from Ewha Woman's University in 1998, and her M.S. and Ph.D. degrees in the Department of Computer Science and Engineering from Pohang University of Science and Technology (POSTECH) in 2000

and 2004, respectively. Currently, she is a postdoctoral student in the Department of Computer Science and Engineering, POSTECH. Her research interests include XML-based network management and policy-based network management. She is a member of KNOM.

HYOUN-MI CHOI (siwa@lge.com) received her B.S. degree in computer science from Hongik University in 2002 and her M.S. degree in computer science and engineering from Pohang University of Science and Technology (POSTECH) in 2004. Her research interests include XML-based management and Web services. Currently, she works at LG Electronics as a research engineer.

HONG-TAEK JU [M] (juht@postech.ac.kr) received his B.S. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1989, and his M.S. and Ph.D. degrees in the Department of Computer Science and Engineering from POSTECH in 1991 and 2002, respectively. From 1991 to 1997 he worked at Daewoo Telecom as a senior software engineer. Currently, he is a full-time lecturer in the College of Communication and Information, Keimyung University. His research interests include Web-based network management, network monitoring, and data synchronization over wireless mobile communications. He is a member of KNOM.

JAMES W. HONG [SM] (jwkhong@postech.ac.kr) is an associate professor in the Department of Computer Science and Engineering, POSTECH, Pohang, Korea. He has been with POSTECH since May 1995. Prior to joining POSTECH, he was a research professor in the Department of Computer Science, University of Western Ontario, London, Canada. He received a B.Sc. and M.Sc. degrees from the University of Western Ontario in 1983 and 1985, respectively, and a Ph.D. degree from the University of Waterloo, Canada, in 1991. He has been very active as a participant, program committee member, and organizing committee member for IEEE CNOM sponsored symposiums such as NOMS, IM, DSOM, and APNOMS. For the last few years, he has been working on various research projects on network and systems management, which utilize Web, Java, CORBA, and XML technologies. His research interests include network and systems management, traffic monitoring and analysis, and security management.