Web Services Based Configuration Management for IP Network Devices*

Sun-Mi Yoo[1], Hong-Taek Ju[2], and James Won-Ki Hong[1]

[1]Dept. of Computer Science and Engineering, POSTECH
{sunny81, jwkhong}@postech.ac.kr
[2]Dept. of Computer Engineering, Keimyung University
juht@kmu.ac.kr

Abstract. The tasks of operating and managing diverse network devices from multiple vendors are getting more difficult and complicated. For that reason, IETF Netconf Working Group has been standardizing network configuration management. The Netconf protocol is an output of that standardization and can be used to effectively manage various devices on a network. However, it is still problematic for the manager to discover and manage configurable parameters in various devices. This paper proposes a method that can quickly discover devices and their parameters as well as methods to manipulate the configuration information on them. We present the architecture for XCMS-WS which meets this mechanism. Using the Web Services technologies, we have develioed a more powerful and flexible system than our previous XCMS. For validation, we have applied XCMS-WS for the configuration management of NG-MON, a distributed, real-time Internet traffic monitoring and analysis system.

1. Introduction

The rapid pace of Internet evolution is currently witnessing the emergence of diverse network devices from multiple vendors. Current networks are complex and are composed of various network devices. Efficient network management systems are necessary to manage these networks and devices effectively. Most configuration management systems for network devices are inefficient, because they use CLI depending on the proprietary operating systems of the vendors. To solve these problems, XML [1] technologies have been applied to configuration management. A standardization process of configuration management for network devices is also in progress. The standardization uses XML technologies and is consisted of protocol message and communication protocol. The IETF Network Configuration Working Group (Netconf WG) [2] is responsible for this standardization work, which attempts to fulfill operational needs for the manager and agent on diverse network devices manufactured by different vendors and guarantees interoperability. The Netconf standard protocol helps to send and receive configuration information between managers and agents, and to manage various network devices of different vendors.

Current networks are still hard to manage with only Netconf protocol in configuration information management. The network administrators need to perform

* This work was in part supported by the Electrical and Computer Engineering Division at POSTECH under the BK21 program of Ministry of Education, and the Program for the Training of Graduate Students in Regional Innovation of Ministry of Commerce, Industry and Energy of the Korean Government.

various configuration tasks for the newly added devices. For such configuration tasks, they should input appropriate parameters to the manager. More efficient configuration tasks are possible if each device can register their parameters to UDDI [3] on their own and the manager find the device parameters accordingly. A set of parameters only needs to be changed while performing a number of similar configuration tasks on single or multiple devices. We used WSDL [5] to minimize any redundancy of the tasks. In this paper, we propose a configuration management system and mechanisms to solve this inefficiency issue. The Netconf agents register and the Netconf manager operates configuration management with reference to the information registered. This mechanism applies Web Services technologies [3], namely UDDI, WSDL, and SOAP [6] to configuration management.

We have previously developed an XML-based Configuration Management System (XCMS) [7] which was based on the sixth draft version of Netconf protocol [10]. We have developed XCMS-WS which has extended XCMS by applying the Web Services technologies and the latest draft version of Netconf protocol [5]. The XCMS-WS system designed based on the above mechanism can communicate between a manager and agents using the Netconf protocol, quickly and effectively find device agents and provide instructions for the operations of devices. XCMS-WS can send and receive messages between a manager and agents and more effectively manage configuration information using Web Service technologies. We have verified our proposed mechanism by applying it to the NG-MON system [8].

2. Related Work

In this section, we explain Web Services technologies and Netconf protocol. We also introduce related work on XML-based network management by others and describe earlier our work.

2.1 Web Services Technologies

Web Services provide standard means to interoperate between different software applications and to run on a variety of operating system and hardware platforms. A Web service describes a collection of operations that are network accessible through standardized XML messaging. Web services fulfill a specific task or a set of tasks. A Web Service can be published and discovered through UDDI. It can also interoperate as XML message over the internet protocol (SOAP).

Web Services technologies [9] consists of certain functional areas, XML messaging (e.g., SOAP), transport (e.g., HTTP, FTP and SMTP) and description of application interactions (e.g., WSDL) and discovery (e.g., UDDI). XML describes a class of data objects called XML documents. SOAP is a lightweight protocol for the exchange of information between peer entities in a decentralized, distributed environment. It is an XML-based protocol and defines the use of XML and HTTP to access services, objects, and servers in a platform-and language-independent manner. WSDL is an XML format for describing web services as a set of endpoints operating on messages. WSDL describes services starting with the messages that are exchanged between the service requester and the provider. UDDI specifications define methods to publish and discover information about web services offerings.

Web Service is divided into three roles: the service provider, the service requester, and the service registry. The objects are the service and the service description. Moreover the operations performed by the actors on these objects are published, found and bound. A service provider creates a Web Service, defines the services and then publishes the service with a service registry based on a UDDI specification. Once a Web Service is published, a service requester may find the service via the UDDI interface. The UDDI registry provides the service requester with a WSDL service description and a URL pointing to the service itself. The service requester may then use this information to bind it directly to the service and invoke it.

2.2 IETF Netconf Protocol Overview

The IETF Netconf WG was formed in May 2003. It has attempted to standardize a protocol suitable for configuration management of multiple heterogeneous network devices. It has been defining the Netconf protocol and transport mappings.

The Netconf protocol [10] uses XML for data encoding and a simple RPC-based mechanism to facilitate communication between a manager and agents. The Netconf protocol defines messages in a well-defined XML format to manage the configuration information easily and to provide interoperability among the devices from different vendors. The Netconf protocol obtains configuration information from an agent and in reference provides modified configuration management information using the RPC mechanism through a structured XML message.

The state of configuration information is divided into three phases: candidate, running, and startup. The ‘running’ is a complete configuration, which is currently active on the network device. The ‘candidate’ is a candidate configuration, which can be manipulated without impacting the device’s current configuration. The ‘startup’ is a copied configuration from the ‘running’ state when the running configuration is reliable. Expressing management operations is not easy with only one transmission, data, and operation model because the environment of transmission of device information, management data, and management operation widely varies. Therefore, the transmission message is divided into 4 layers to satisfy the requirements of various environments [6]. Table 1 shows four layers defined in Netconf and examples of the contents in the layers.

Table 1. Netconf Protocol Layers

Layer	Contents
Application	Transmission protocol to provide a communication path between agent and manager: BEEP, SSH, SOAP over HTTP
RPC	A simple, transport-independent framing mechanism for encoding RPC.: <rpc>, <rpc-reply>, <rpc-error>
Operation	A set of base operations invoked as RPC methods with XML-encoded parameters.: <get-config>,<edit-config>,<copy-config>
Content	Configuration data

The Netconf protocol currently considers three separate application protocol bindings for transport: SSH [18], BEEP [19], and SOAP over HTTP [10]. The SOAP

over HTTP transport mapping uses WSDL for binding services and its specification proposes standardized WSDL.

2.3 University of Twente Research on Web services based management

Aiko Pras et. al have published several papers in which they suggested uses for Web services based management. As part of their research, the performance differences between SNMP and Web Services-based management have been investigated [16]. To compare performance, they investigated bandwidth usage, CPU time, memory requirements and round trip delay. To conduct tests, they implemented several Web services based prototypes and compared the performance of these prototype to various SNMP agents. That tests showed that there is a significant difference in the bandwidth requirements of SNMP and Web services. They concluded that SNMP is more efficient in cases where only a single object is retrieved but Web services is more efficient for larger number of objects. Web services management is more suitable for large scale networks such as an enterprise network.

2.4 XCMS

In our previous work, we had developed an XML-based configuration management system called XCMS which implemented the first draft IETF Netconf specification. XCMS proposed XPath which has been standardized since the fourth draft and used SOAP over HTTP as a transport mechanism. XCMS supports the fifth draft IETF Netconf protocol specification. In XCMS, a centralized manager controls the configuration information of network devices equipped with Netconf agents. XCMS works like Web Service with Netconf protocol. XCMS can manage configuration information of diverse devices. Yet, we found limitations of XCMS on networks where various network devices change dynamically. XCMS has difficulty when it searches for devices to operate configuration management and it uses configuration operations of unfamiliar devices. So, we have developed a new configuration management system called XCMS-WS by extending XCMS.

3. Proposed Method

We have mentioned the limitations of configuration management for network devices using only the Netconf protocol. Our proposed mechanism for solving these problems incorporates Web Services into configuration management.

3. 1 Registration and Search for Configuration

The Netconf protocol allows managing various devices from diverse vendors. However, the discovery of necessary parameters for configuration tasks is not trivial. If the Netconf manager can find device parameters which are managed by Netconf protocol in a repository, it can more easily and efficiently manage the devices.

Netconf protocol uses WSDL and SOAP with the ‘SOAP over HTTP’ transport mapping [20]. But, that does not provide a registry function. We apply UDDI as a registry for configuration management. The UDDI structure in our system consists of four factors: businessEntity, businessService, bindingTemplate, and tModel. The

businessEntity represents information about a business. Each businessEntity contains a unique identifier, the business name, a short description of the business, some basic contact information [12]. Each businessService entry contains a description of the service, a list of categories that describe the service, etc. The bindingTemplate provides technical description about how and where to access a specific service. A tModel includes descriptions and pointers to external technical specifications. XCMS-WS adapts the above structure; however the contents are different. businessService and bindingTemplate represent configuration parameters and the location of WSDL containing the descriptions of configuration parameters, respectively.

The Netconf manager sends out requests to discover parameters of network devices, which manage configuration information using the Netconf agent in the UDDI registry. The UDDI registry searches the business using a query on the structured data contained in a UDDI registry and then provides information of the business. This business is a set of parameters of the network device. This process brings business entry XML data into a Netconf manager. It finds the NETCONF_AGENT of businessService in the data and analyzes the entry list. The entry information includes the tModel instance info and the description of the Netconf agent service. The Netconf agent can be searched through the WSDL description which is pointed to by the overviewURL. The Netconf manager accesses the Netconf agent using information provided by the UDDI registry. Then, the manager operates configuration management to the device using the Netconf protocol with the parameters.

We use a private UDDI because private UDDI provides more valuable functions in a private setting. When using a public UDDI registry, it publishes local devices on the whole Internet creating some major security problems. However, a private UDDI registry is accessible by entities internal to an organization. This registry will be behind the organization's firewall and only accessible via the organization's intranet. The devices are managed by the administrator only. A private UDDI registry can also simplify organization's extranet operations.

3. 2 WSDL for Netconf Agents

When a new network device is added or a network device's software changes, operations are updated on network. Sometimes, on a network consisting of many network devices, the configuration management work can be done perfectly after configuration information of several devices linked to the work is modified. Moreover, the work may be repeated many times. For instance, the operation which changes and reverts to a network configuration for network upgrade is often needed. In the case of repeated configuration management, it is better to describe the task. There exist many approaches for describing configuration management. We use WSDL for the configuration management task.

WSDL can be used as a task description. Its operations and messages are described abstractly, and it provides a concrete protocol and message format to define a concrete endpoint. The Netconf Manager connects to the Netconf agent in a network device which is going to operate configuration task. This process needs a description that provides all the details necessary to communicate between a Netconf manager and a Netconf agent. So, the Netconf WG proposes the standardization of WSDL used on Netconf protocol. And then a Netconf manager operates configuration information of a network device. For correct configuration management of the network device, the

Netconf agent describes the message format, data types, operations and a location where the service called is binding in WSDL. The manager can perform configuration management by changing the parameters belonging to the WSDL defined tasks. Also, it would be more efficient to conduct multiple configuration tasks.

```
<message name="NetconfdRequest">
  <part name="req-msg" type="xsd:string" />
</message>
<message name="NetconfdResponse">
  <part name="rpl-msg" type="xsd:string" />
</message>
<portType name="NetconfdPortType">
   <operation name="Netconfd">
      <documentation>Service definition of function ns__Netconfd</documentation>
      <input message="tns:NetconfdRequest" />
      <output message="tns:NetconfdResponse" />
   </operation>
</portType>
<binding name="Netconfd" type="tns:NetconfdPortType">
  <SOAP:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
     <operation name="Netconfd">
        <SOAP:operation soapAction="" />
        <input>
           <SOAP:body use="encoded" namespace="urn:Netconfd"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </input>
       <output>
           <SOAP:body use="encoded" namespace="urn:Netconfd"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </output>
     </operation>
  </binding>
<service name="Netconfd">
      <documentation>gSOAP 2.7.0d generated service definition</documentation>
      <port name="Netconfd" binding="tns:Netconfd">
         <SOAP:address location="http://141.223.82.6:5455" />
       </port>
</service>
```

Fig. 1. WSDL of NETCONF agent

Fig. 1 is a fragment of the WSDL used by a XCMS-WS agent. An XCMS-WS manager can operate configuration information of Netconf agents as calling a service daemon, ‘Netconfd’. The manager can realize new information changed of the device as analyzing the WSDL and UDDI. The Netconf manager can recognize devices which are new installed or modify functions. And it can also know that devices software is changed or operations are updated. When the Netconf manager makes a request message for operating configuration task of a device, it can refer useless information from WSDL of the device. A Netconf agent operates configuration tasks based on request message from the Netconf manager.

4. Architecture for XCMS-WS

In this section, we present the design of XCMS-WS, XML-based Configuration Management System using Web Services. XCMS-WS consists of a manager, an agent and a private UDDI registry. Fig. 2 illustrates the detailed architecture of XCMS-WS, in which a centralized XML-based manager controls the configuration information of agents. A Netconf protocol request message is generated by the manager and sent to the agent. The agent analyzes this message and performs the configuration management function.

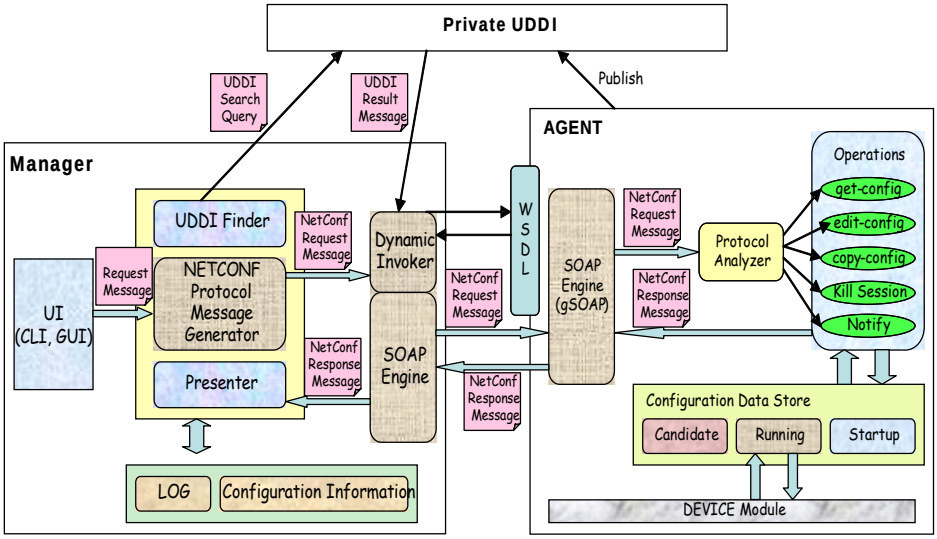

Fig. 2. Detailed Architecture of XCMS-WS Manager

A manager consists of five elements: UI, Protocol Message Generator, UDDI Finder, Dynamic Invoker, and SOAP Engine. The UI is used by the administrator to input request messages. A XCMS-WS manager provides CLI which is required essentially by Netconf standardization and also provides a GUI for the Web environment. In GUI, the administrator does not need to type the entire request message. A XCMS-WS manager semi-automatically creates a request message based on objects selected by the administrator using the GUI. The request message which is inputted by the administrator is translated into a Netconf protocol request message by the XML parser module of the manager. The XML parser module is a NETCONF Protocol Message Generator, which translates just an XML message to a Netconf request message using Netconf message formats. The Netconf protocol response messages are analyzed by the Presenter. Agents which can be managed by XCMS-WS are discovered and registered at the UDDI registry. The UDDI Finder module discovers device agents. If you get agent information from the UDDI Finder module, you can know the location of WSDL which is a description of the agent's service, and then you can do remote call through WSDL.

An agent is composed of the SOAP Engine module to deliver messages and the Protocol Analyzer module to parse messages. An agent also provides the service of processing the operation that is included in the configuration management message.

There is a Configuration Data Store which stores data corresponding to 'candidate', 'running' and 'startup' as the configuration information model. One of these, 'running' synchronizes with real device modules and this information describes the present running state. Processing Netconf protocol messages through the SOAP engine needs architecture like a message-queue style architecture because of the delay in processing messages as they arrive at the agent. Using a message queue can solve message loss problems that can occur because of multiple requests from several managers and processing delays. It also provides architecture for synchronizing configuration information. The protocol interpreter is a very important module related to the Netconf protocol, it extracts operation names described as XML tag and parameters which are needed for operating from the requirement message. Operations are called from this extracted information. When an agent executes operations, each operation reads and modifies information stored in the configuration information stores. The operating results are loaded on a Netconf protocol message and are transmitted to manager.

The configuration information of agents must be managed automatically by a manager in network management. Managers and agents synchronize state information of agent through Private UDDI. If an address of an agent is changed and the agent changes its address at UDDI, managers can automatically provide a layout changing the address.

5. Implementation

We have implemented an XML-based configuration management system based on the XCMS-WS design presented in Section 4. The distributed system applying the architecture of XCMS-WS is NG-MON which is a passive network monitoring system with a clustered and pipelined architecture for load balancing and distribution. This section explains the XCMS-WS implementation environment and describes about the manager, the agent, and UDDI registry in detail.

5.1 Implementation Environment

The manager of the XCMS-WS is on a Pentium IV 2.4 GHz CPU and 500 MB RAM server running Redhat Linux 9. The platform of the agent is different depending on the platform of the device agent. The manager uses various XML technologies to provide Web-based user interfaces and an XMLDB. We resorted to Apache that provides APIs implemented with JAVA. XCMS-WS needs the following APIs: Xindice as an XMLDB and AXIS as SOAP engine to apply the SOAP/HTTP communication method between the manager and the agents. The manager and the agents must setup a Web server for HTTP communications. Following the user interface implementation with Java Server Page (JSP), the manager installs TOMCAT provided in Jakarta Apache.

The manager composes a Dynamic Invocation environment with using WSDL when placing a service call. We used WSIF [13] for the Dynamic Invocation environment. Consequently, the manager can make a service call only with WSDL which is described as a service agent. XCMS-WS can call dynamically using WSDL which is discovered at the UDDI Netconf agent.

The agent uses a parser based on C compiler and gSOAP [14]. If the device agent has few resources, as in an embedded system, it must sparse resources. The gSOAP provides the smallest and most efficient environment among the SOAP/HTTP sources based on C/C++ language. It also guarantees interoperation with other SOAP implementation and can transmit SOAP RPC message between a manager and an agent. The gSOAP generates a stub, a skeleton, and a WSDL file using the header file which declares RPC operations. We select the libxml [15] which is lightweight compare to existing XML parsers in the agent. This deployment provides DOM APIs which supports to select the specified node, update the selected node value, and read management information. The XCMS-WS uses a private UDDI registry in the WSDK [15] package provided by IBM. We selected the private UDDI for security reasons.

5.2 Verification

NG-MON [8] consists of five subsystems: a packet capture, a flow generator, a flow store, a traffic analyzer, and the presenter of analyzed data. Each subsystem of NG-MON runs on a Linux server with Pentium III 800 MHz CPU and 256 MB RAM.

XCMS-WS consists of a manager, agents and private UDDI. The UDDI registry is located between a manager and agents for searching devices to be configured. The XCMS-WS agent manages the NG-MON module. The NG-MON module loads configuration information files from the NG-MON subsystems to manage the configuration information of the NG-MON system on the XCMS-WS agent, which it synchronizes with real configuration information of the NG-MON system. The XCMS-WS manager sends a request message to the XCMS-WS agent and the XCMS-WS agent sends this message to the NG-MON module.

```
<?xml version="1.0"?>
<ng-mon ip="141.223.82.144">
        <admin name="DongHyun Kim"
email="dhkim03@postech.ac.kr"></admin>
        <database user="root" password="********"/>
        <packetcapture>
                <device name="eth1"/>
                        ……
        </packetcapture>
        <flowgenerator interval="2">
                <subsysip = “141.223.11.3”/>
                   <subsysip = “141.223.11.4”/>
        </flowgenerator>
        <flowstore>
                <p2p file="p2p.xml"/>
        </flowstore>
</ng-mon>
```

Fig. 3. XML-based Structure of the Configuration Information

Fig. 3 is an XML-based structure of the configuration information of the NG-MON subsystems. The NG-MON module which manages the configuration information is loaded on the XCMS-WS agent in the NG-MON system. The manager then can manage the configuration information of the NG-MON.

The UDDI TREE of GUI connects to private UDDI registry with the administrator ID and shows the search results that are registered at UDDI. Using the Netconf protocol, we can easily select a list of manageable devices from all devices registered at the UDDI registry. While retrieving information registered at the UDDI registry into XCMS-WS, our system filters the necessary data and updates information automatically using the UDDI registry. Fig. 4 shows that two available devices including Netconf agent and a SNMP_XML_Gateway are registered at UDDI.

```
<?xml version="1.0" encoding="UTF-8"?>
<BusinessName bizID="DPNM">
  <Service serviceID="NETCONF_AGENT">
    <TModelInst devID="NETCONF_AGENT1"
            overViewDoc="http://141.223.82.6:8080/xcms-ws/Netconfd.wsdl"/>
    <TModelInst devID="NETCONF_AGENT2"
            overViewDoc="http://141.223.82.7:8080/xcms-ws/Netconfd.wsdl"/>
  </Service>
  <Service serviceID="SNMP_XML_GATEWAY">
    <TModelInst devID="SNMP_XML_GATEWAY"
    overViewDoc="http://141.223.82.6:8080/axis/services/SNMP_XML_GATEWAY?wsdl"/>
  </Service>
</BusinessName>
```

Fig. 4. UDDI Search Result

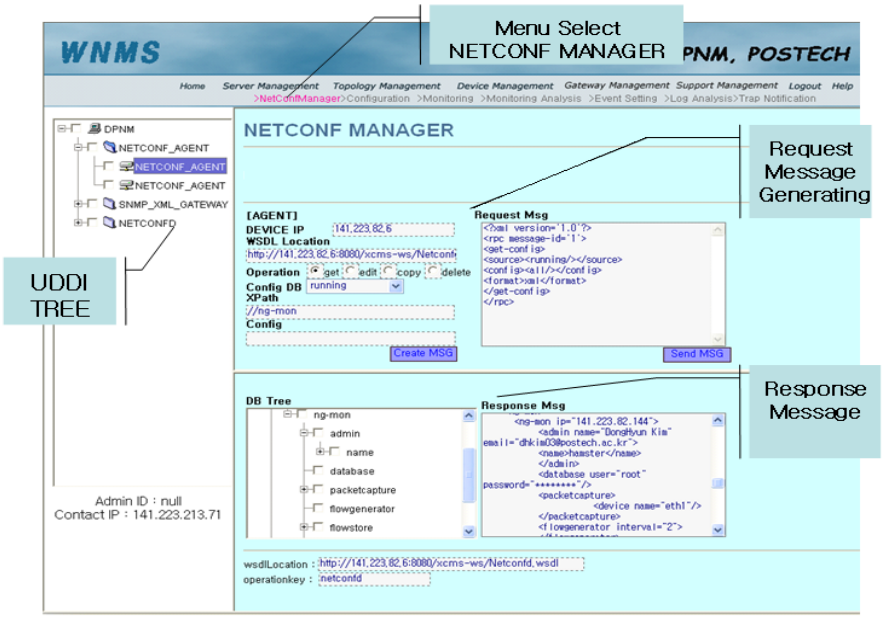

Fig. 5. Web-based user interface of XCMS-WS

The Web-based user interface of XCMS-WS as shown in **Fig.** provides the following functions: viewing, modifying and searching devices from UDDI registry,

and creating and sending request messages for configuration task. Moreover, it shows responses from Netconf agents and configuration information DB tree of the device.

6. Concluding Remarks

In this paper, we have proposed a mechanism to manage configuration information more effectively using the Netconf protocol and Web Services technologies. Using UDDI helps a Netconf manager to quickly and intelligently recognize required parameters of network devices to operate configuration tasks. A Netconf manager finds a parameter description of a device to manage configuration from a private UDDI registry. Using WSDL configuration tasks can be done efficiently and repeatedly to operate configuration tasks of several devices.

We have presented the design and implementation of XCMS-WS, which is based on the mechanisms above. The XCMS-WS effectively manages the configuration information using Web Services technologies. It automatically modifies configuration information of the related devices using the Netconf protocol when the configuration information shared with other devices is modified. We applied the XCMS-WS to the configuration management system used for NG-MON.

For future work, we plan to validate the flexibility and extendibility of the XCMS-WS as applied to other network systems. Finally, we will conduct performance tests on XCMS-WS in order to optimize it for adaptation to embedded systems.

References

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

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

[3] W3C, “Web Services Technologies”, WASP/D3.1, December 2002.

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

[5] W3C, “Web Services Description Language (WSDL) Version 2.0” W3C Working Draft 3, August 2004, http://www.w3.org/TR/wsdl20/.

[6] Enns, R., “NETCONF Configuration Protocol” draft-ietf-Netconf-prot-06, April 26, 2005, http://www.ietf.org/internet-drafts/draft-ietf-Netconf-prot-06.txt.

[7] W3C, “SOAP Version 1.2 Part 2: Adjuncts”, W3C Working Draft, Dec. 2001.

[8] Mi-Jung Choi, Hyoun-Mi Choi, Hong-Taek Ju and James W. Hong, "XML-based Configuration Management for IP Network Devices", IEEE Communications Magazine, Vol. 41, No. 7, July 2004. pp. 84-91.

[9] Se-Hee Han, Myung-Sup Kim, Hong-Teak Ju and James W.Hong, “The Architecutre of NG-MON: A Passive Network Monitoring System”, DSOM2002, Montreal, Canada, October, 2002, pp. 16-27.

[10] Goddard, T., “NETCONF over SOAP”, draft-ietf-Netconf-soap-05, October 11, 2005, http://www.ietf.org/internet-drafts/draft-ietf-Netconf-soap-05.txt.

[11] Enns, R., “NETCONF Configuration Protocol”, draft-ietf-Netconf-prot-06, October 27, 2005, http://www.ietf.org/internet-drafts/draft-ietf-Netconf-prot-06.txt.

[12] Simeon Semeonov, *Building Web Services with Java*, SAMS, 2001.
[13] Apache WebService project, WebService Invocation Frameworks “WSIF”, http://ws.apache.org/wsif/.
[14] 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/.
[15] IBM, “Web Services Development Kit (WSDK)”, http://www-106.ibm.com/developerworks /webservices/wsdk/.
[16] XMLSOFT, libxml, “The XML C parser and toolkit at Gnome”, http://www.xmlsoft.org/.
[17] A. Pras, T. Drevers, R. v.d. Meent, D. Quartel, “Comparing the Performance of SNMP and Web Services-Based Management,” IEEE eTNSM, V1, N2, Dec. 2004, pp. 1-11.
[18] Wasserman, M., “Using the NETCONF Configuration Protocol over Secure Shell (SSH)”, April 9, 2005, http://www.ietf.org/internet-drafts/draft-ietf-Netconf-ssh-04.txt.
[19] Lear, E., Crozier, K., Enns, R., “BEEP Application Protocol Mapping for NETCONF”, March 2005, http://www.ietf.org/internet-drafts/draft-ietf-netconf-beep-05.txt.
[20] Lear, E., Crozier, K., Enns, R., “BEEP Application Protocol Mapping for NETCONF”, draft-lear-Netconfbeep-05, March 2005, http://www.ietf.org/internet-drafts/draft-ietf-netconf-beep-05.txt.