Design and Implementation of XML-based Configuration Management System for Distributed Systems

Hyoun-Mi Choi, Mi-Jung Choi and James W. Hong
Dept. of Computer Science and Engineering, POSTECH, Pohang, Korea
{siwa, mjchoi, jwkhong}@postech.ac.kr

Abstract

Today, we are witnessing more distributed systems on enterprise networks and on the Internet. In general, a distributed system is composed of many subsystems. It is difficult to effectively manage the configuration information of distributed systems because they may be deployed with different software components and run on heterogeneous computing platforms. In addition, the configuration information of a subsystem has complex relations with the information of other subsystems, so it is difficult to provide automatic reconfiguration of related subsystem. To overcome the difficulties, we propose a management information model that considers the relations among subsystems and the Simple Object Access Protocol (SOAP) as a communication method. This paper presents the design and implementation of X-CONF (Xml-based CONFiguration management system) for a distributed system. For validation, we have developed the X-CONF for NG-MON, which is a distributed and real-time Internet traffic monitoring and analysis system.

Keywords

XML-based Configuration Management, XML, XML Schema, SOAP.

1. Introduction

Today, most large-scale software systems are composed of a large number of computers in a distributed computing environment. These systems are usually implemented with many computers to distribute the processing. In this paper, the components of a distributed system are called subsystems. The implementation and execution environments of subsystems may be various. Also, the relation of configuration information exists among these subsystems. A relation means that some parts of the configuration information of a subsystem can be shared with, have influence on, and be inherited by other subsystems. Therefore, a configuration management system for the distributed system needs to understand the relationship among subsystems for automatic reconfiguration and to provide the communication method in a platform- and language-independent manner.

The Simple Network Management Protocol (SNMP) [1] is the most widely used method for network management on the Internet. Also, SNMP is used in configuration management. However, retrieving large volumes of information via

Get/GetBulk operations of SNMP is not facile because of SNMP over UDP. SNMP MIB using SMI is insufficient to present a set of interrelated tables considering the relations among managed objects. Recently, much attention has been given to the use of XML [2] technology to configuration management as an alternative approach to SNMP. A current standardization of configuration management using XML technologies is not concerned with management information yet. Also, there is very little implementation work on XML-based configuration management for distributed systems.

In this paper, we present the design and implementation of X-CONF for distributed systems. Figure 1 is a high-level architecture of X-CONF, where the XML-based manager controls multiple subsystems equipped with XML-based configuration management agents.

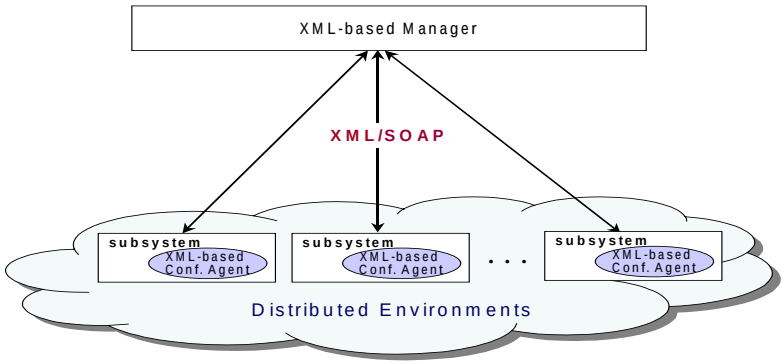


Figure 1. High-Level X-CONF Architecture

We define the management information model with the XML Schema [3]. The management information model presents configuration information, which contains wide configuration information of the distributed system, and relationship information, which represents relations among the subsystems using any tags (element and attribute). The configuration information is applicable to multiple subsystems by transforming it using the relationship information. We apply SOAP [4] to communicate, which can bind any connection-oriented protocol like HTTP for the transport, so there is no limit to process bulk data. Based on XML and SOAP, the XML-based manager can directly call management operations in the agent via SOAP RPC [5]. SOAP is very convenient to develop and extend the management operations. In addition, it is platform-independent, which places no restrictions on endpoint implementation technology choices.

The organization of this paper is as follows. In Section 2, we list related work on XML-based configuration management. In Section 3, we discuss the requirements of X-CONF. Section 4 explains the design of a manager and an agent in X-CONF. In Section 5, we explain the implementation details of a prototype X-CONF system. Finally, we conclude our work and discuss directions for future work in Section 6.

2. Related Work

In this section, we describe related work on XML-based configuration management of standard activities and industrial efforts.

2.1 Cisco's Configuration Registrar

The Cisco Configuration Registrar [6] is a Web-based system for automatically distributing configuration files to Cisco IOS network devices. The Configuration Registrar works in conjunction with the Cisco Networking Services (CNS) Configuration Agent located at each device. The Configuration Registrar delivers the initial configuration to Cisco devices during the initial startup on the network. The Configuration Registrar 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.

2.2 Juniper Networks' JUNOScript

Recently, Juniper Networks introduced JUNOScript [7] for their JUNOS network operating system. The JUNOScript is part of their XML-based network management effort and uses a simple model, designed to minimize both implementation costs and the impact on the managed device. The JUNOScript allows client applications to access operational and configuration data using an XML-RPC. The JUNOScript defines the DTDs for the RPC messages between client applications and JUNOScript servers running on the devices. Client applications can request information by encoding the request with JUNOScript tags 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.

2.3 IETF Network Configuration

The Network Configuration (Netconf) [8] Working Group (WG) was formed in May 2003. The Netconf WG is chartered to produce a protocol suitable for configuration management of network devices. The Netconf WG defines the Netconf configuration protocol and transport mappings. The Netconf protocol uses XML for data encoding, and RPC as a communication mechanism. The Netconf protocol is conceptually partitioned into four layers: 'content', 'operations', 'RPC', and 'transport' and defines management operations and message formats. The Netconf protocol considers three separate application protocol bindings for the transport such as Secure Shell (SSH) [9], Block Extensible Exchange Protocol (BEEP) [10], and SOAP over HTTP [11].

3. Requirements

Generally, a large-scale software system is composed of multiple subsystems to

perform different tasks and to distribute the load. The subsystems have close relations with their own configuration information. That is, configuration information is shared among the component subsystems and the configuration information of one subsystem affects the other subsystems. To effectively manage the configuration information of each subsystem, the functional requirements of configuration management system for distributed systems are as follows.

(1) Show, delete, and modify the configuration information of the subsystem.
(2) Add and delete single or multiple subsystems.
(3) Provide a Web-based user interface for ubiquitous access.

Requirements (1), (2), and (3) are fairly simple without considering the relations of the subsystems. More requirements are needed to maintain consistency in the configuration information, which has various relations with the subsystems. Also, the effective communication mechanism between the manager and agents must be supplied. The additional requirements are as follows.

(4) Provide a management information model to describe the configuration information and the relationship information of subsystems. The management information model represents complex relations among subsystems. The kinds of relations are shared, referred, and inherited.
(5) Maintain consistency in configuration information among subsystems. When the configuration information of a subsystem is added, deleted, or modified, the automatic reconfiguration must be provided to other related subsystems using the relationship information.
(6) Configuration activities can cause one or more state changes in a subsystem. In addition, a configuration activity in one subsystem can cause one or more state changes in other related subsystems. It is critical that the configuration system must treat the overall change operation atomically in a subsystem or multiple subsystems. The goal is for a change request either to be completely executed or ignored. This is called transactional integrity, which makes it possible to develop reliable configuration systems that can invoke transactions and keep track of the subsystems' overall state and work in the presence of error states.
(7) Store the history of interaction messages between a manager and agents into a log file. The messages are the notification about the system state of the agents, and the result of the agent's management operations invoked by the manager.
(8) Provide communication methods between a manager and agents regardless of the implementation environment.

4. Design of X-CONF

In this section, we define a management information model and an interaction operation model. We propose a general management information model that can be applied to any other distributed systems. By using a SOAP-based interaction

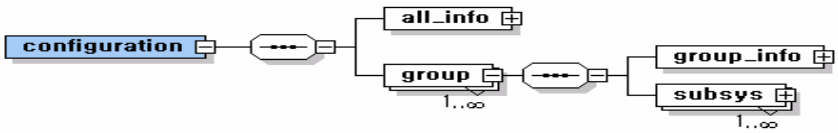
operation model, the agent easily extend new operations. This section also shows examples according to each model and describes the architecture of X-CONF.

4.1 Management Information Model

In order to use automatic reconfiguration of related subsystems, we propose a configuration information model, which contains wide configuration information of a distributed system, and a relationship information model which focuses on the dynamic relationships of subsystems.

Table 1 shows the configuration information model using the XML Schema. The subsystems that perform the same work have almost the same configuration information, so they are classified into an identical group. The sub-elements of the elements such as *all_info*, *group_info* and *subsys* are to present the specific configuration information in the subsystems. An *all_info* is defined as a collection of configuration information needed in all managed subsystems. A *group_info* is a collection of configuration information shared with subsystems in the same group. A *subsys* is a collection of configuration information used by only one subsystem. The names and attributes of sub-elements (*all_info*, *group_info* and *subsys*) are not static but dynamic. Therefore, the name of the element (*anyElement*) and that of the attribute (*anyAttribute*) can be defined in any configuration information.

XML Schema of Configuration Information



Example

```
<configuration name="ng-mon">
  <all_info>
      <admin email=mount@postech.ac.kr name="mount"/>
      <database password="password" user="root"/>
  </all_info>
  <group name="packetcapture">
     <group_info>
        <device name =" eth1"/>
        <data type=" all"/>
     </group_info>
     <subsys ip="141.223.11.1"/><subsys ip="141.223.11.2"/>
  </group>
  <group name="flowgenerator" inheritance="packetcapture">
      <group_info>
         <time interval="2"/>
     </group_info>
     <subsys ip="141.223.11.3">
         <database user="siwa" passwd="123"/>
     </subsys>
     ...
</configuration>
```

Table 1. Configuration Management Information Model

The management information within a group can be inherited from other groups. In this case, we use an *inheritance* attribute in the management information model. The *inheritance* attribute represents that a whole or a part of configuration information belonging to the *group_info* in a parent group is inherited to a child group. Due to the growing complexity of distributed systems, multiple inheritances can occur. In this case, we add the character (':') among group names to distinguish each group name. To describe various relationships of configuration information among subsystems, we propose an XML Schema for the relationship information model.

The relationship information model explicitly explains complex relations between groups. Complex relations mean that the configuration information of a subsystem shares, influences, and inherits a total or a part of the configuration information under the *group_info* in other subsystem groups.

XML Schema of Relationship Information

Example

```
<relation name = "ng-mon">
   <group name =" packetcapture">
    <sharedInfo name = "flowstore">
       <element>p2p</element>
    </sharedInfo>
   </group>
   <group name = "flowgenerator">
     <inheritance name    = "flowstore" />
     <sharedInfo name = "packetcapture">
        <element>data</element>
     </sharedInfo>
     <referInfo name = "packetcaputre">
       <element>device</element>
     </referInfo>
   </group>
   <group name = "flowgstore">
     <sharedInfo name = " flowgenerator">
        <element>time</element>
     </sharedInfo>
           ...
</relation>
```

Table 2. Relationship Information Model

Table 2 is the relationship information model using an XML Schema. In the relationship information model, we define three element tags: *inheritance*, *sharedInfo*, and *referInfo*. The *inheritance* element needs only the group name if all the sub-elements under the *group_info* are inherited. If the information is partially inherited, the *inheritance* element requires both the group name and the

specific element names inherited in the *group_info*. The *inheritance* is used when the child node modifies the inherited information independent of the parent node. However, the *shareInfo* is used when the information changes occurring in a subsystem are delivered to the other subsystems in the related groups. The difference between the *sharedInfo* and the *referInfo* is the possibility to change the value of the related information. In the case of *sharedInfo*, any groups including the *sharedInfo* can change each value of *sharedInfo* elements. This change is reflected to the other groups with the same *sharedInfo*. However, the group including *referInfo* cannot change the value in *referInfo* elements but can reflect the changed value of *referInfo* elements in the original group. In summary, the *inheritance* and the *sharedInfo* are the read-write data, and the *referInfo* is read-only data.

The log information stored in the log file is made of two types of information, such as operation results information and notification information. Table 3 (a) presents the result of each transaction at the agent side. It contains diverse information as attributes. They are an agent IP address, a group name, a type of operation, data, time, and result. The operation is *modify*, or *load* at the agent and *modify*, *add* or *delete* at the manager. The data is the XPath [12] expression of the configuration information. The result is the result of the operation: *success* or *fail*. If the result is *fail*, the manager must rollback to guarantee the consistency of the configuration information. The rollback is the basic method when the transaction fails in our X-CONF. Also, the error message is stored into the log file.

(a) Example of Transaction Information Model

```
<transaction name = "ng-mon">
  <subsystem ip="141.223.82.1" group="packetcapture" operation="modify"
    xpath="//group[@name= ”packetcapture”]/group_info/device/@type"
    time="2003.08.10:12:06:56" result="success" />
  <subsystem ip="141.223.82.3" group=”packetcapure”
        …
</transaction>
```

(b) Example of Notification Information Model

```
<notification name = "ng-mon">
  <subsystem ip="141.223.82.1" group="packetcapture" state="reboot"
        time="2003.08.10.12.06.56" />
  <subsystem ip="141.223.82.2" group="packetcaputure" state="reboot"
        time="2003.08.10.12.06.50" />
        …
</notification >
```

Table 3. Log Information Model

Table 3 (b) represents the records of notification messages made by the agent. If the agent terminates normally, the value of the state is ‘*stop*’. When the agent reboots in order to apply changed configuration information, the state is ‘*reboot*’. The manager checks the received time, adds the time information, and stores the notification into a log file.

4.2 Interaction Operation Model

We use an RPC-based paradigm as a communication protocol. Specifically, SOAP [4] has been chosen for this. SOAP can access services, objects, and servers in a platform-independent manner. This is connection-oriented, so this connection provides reliable and sequential data delivery. In our X-CONF system, the result of the operation mentioned in Section 4.1 is important in order to maintain consistent configuration information. If the result from a subsystem is *fail*, the manager retries the operation several times. If the result status remains *fail*, the manager must rollback the changed configuration information of all subsystems.

(a) Operation Request Message

```
<modify messageID="192">
        <xconf:subsystem>
            <systemName>mg-mon</systemName>
            <group>flowgenerator</group>
            <subIP>141.223.82.3</subIP>
            <fileName>ng-mon_flowgenerator_141.223.82.3</fileName>
        </xconf:subsystem>
        <xconf:parameter>
            <xpath>//all_info/admin/@email</xpath>
            <data>siwa@postech.ac.kr</data>
            <xpath>//group[@name="trafficanalyzer"]/group_info/p2p/@file</xpath>
            <data>test.xml</data>
        </xconf:parameter>
</modify>
```

(b) Operation Result Message

```
<modifyResponse messageID="192">
        <xconf:subsystem>
            <systemName>ng-mon</systemName>
            <group>flowgenerator</group>
            <subIP>141.223.82.3</subIP>
            <fileName>ng-mon_flowgenerator_141.223.82.3</fileName>
        </xconf:subsystem>
        <xconf:result>
            <result>ok</result>
        </xconf:result>
</modifyResponse >
```

(c) Notification Message

```
<notify messageID = "0923-567b">
        <xconf:subsystem>
            <systemName>ng-mon</systemName>
            <group>flowgenerator</group>
            <subIP>141.223.82.3</subIP>
        </xconf:subsystem>
        <xconf:state>
            <action>reboot</action>
        </xconf:state>
</notify>
```

Table 4. The Interaction Operation Message Examples

The interaction messages between the manager and the agent are also made in an XML document format. Table 4 shows the examples of the interaction message and simultaneously represents the operation description corresponding to the operation model. By using the unique *messageId*, the manager can distinguish the operation message. Each subsystem has its own configuration file. The name of the configuration file must be unique in our X-CONF. The name is made by integrating three factors (*distributed_system_name, group_name, subsystem ip address*).

Table 4 (a) describes a request message binding SOAP to call management operations in the agent. Table 4 (b) is the response message corresponding to the request message (a). "<xconf:subsystem>" contains the subsystem information and the configuration file name for the manager to distinguish the subsystem, and "<xconf:parameter>" in the message includes the information of parameters used in the operation. In Table 4 (b), "<xconf:result>" is the result of the operation transaction. If the manager receives the error result message, the manager must rollback the corresponding operations. The agent sends the notification message in the format of Table 4 (c) if the subsystem is rebooted or shutdown.

4.3 X-CONF Architecture

Figure 2 illustrates the architecture of X-CONF, in which a centralized XML-based manager controls the configuration information of subsystems equipped with XML-based configuration agents. The manager is divided into five modules: *XMLDB, XMLDB Handler, XSL/XSLT Processor, SOAP Server&Client,* and *Management Operation*. The manager possesses the list of subsystems and the configuration information of each subsystem in the XMLDB [13]. XMLDB is a special database designed only for XML documents, stores intact XML documents and partially controls the contents of the XML documents. The *XMLDB Handler* module processes information in XMLDB. The *XSL/XSLT Processor* module transforms XML form into HTML form to offer a Web-based user interface. The *SOAP Server* module receives the notification message from the agent. The *SOAP Client* module calls the SOAP RPC methods in the agent. The *Management Operation* module has five methods: *getMethod*, *addMethod*, *delMethod*, *modifyMethod* and *createMethod*.

- ***getMethod*:** This method is used for retrieving management information.
- ***addMethod*:** When the configuration information or relationship information is added, the manager invokes this method. This means that a subsystem belonging to the existing group is added into the distributed system or a new group is added. The manager takes every IP address of the related agents over relationship information, and then sends the RPC message to the selected agents to call *loadMethod*, a management operation of the agent.
- ***delMethod*:** Unlike *addMethod*, this method deletes subsystems or groups. When the manager changes the structure of configuration information or relationship information, it delivers the RPC messages to all related agents to

call *loadMethod* in them.

- ***modifyMethod*:** This method is used when the contents of the configuration information is modified without a change in the structure of the configuration information or relationship information. Also, the manager sends the RPC message to all related agents to call *modifyMethod* in them.
- ***createMethod*:** The manager possesses the entire configuration management information and relationship information. At first, subsystems do not possess the configuration information. The manager automatically creates the configuration information of each subsystem using the configuration information and relationship information. Then, the manager loads the configuration information to each subsystem using *loadMethod* in the agent.
- ***notifyMethod*:** As the SOAP RPC method, the manager receives and processes notifications from the agents and stores them into a log file via this method.

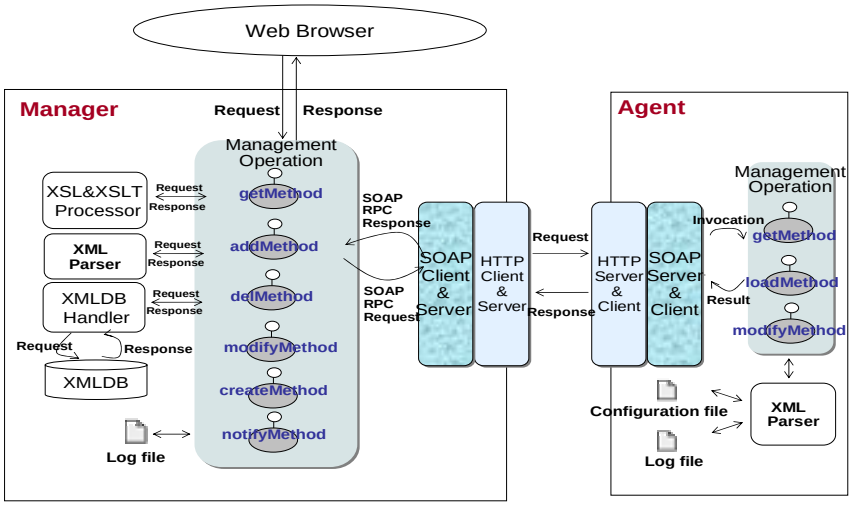


Figure 2. Architecture of X-CONF

The X-CONF agent illustrated in Figure 2 contains *SOAP server&client*, *XML parser* and *Management Operation* modules. The *XML Parser* module allows for an agent to parse and access the contents of the XML message using XPath expression. The *Management Operation* module as SOAP RPC methods is as follows: *getMethod*, *loadMethod*, and *modifyMethod*.

- ***getMethod*:** This method is used to retrieve the information from the configuration XML file in the subsystem and to show the information to the administrator.
- ***loadMethod*:** After the manager changes the structure of the configuration information or relationship information, it newly generates the configuration information of every related subsystem via the *createMethod* operation. The manager calls the *loadMethod* to send the new information to the related

subsystems.

- ***modifyMethod*:** This method is used when the contents of the configuration information is modified without a change in the information structure.

5. Implementation

We have implemented an XML-based configuration management system. Therefore, we referred to the Apache Project Group which provides Application Program Interface (API) implemented with JAVA to support related XML technologies. X-CONF needs following APIs: XML Xerces [14] as an XML parser, Xalan [15] to transform XML document into other forms, Xindice [16] as an XMLDB, which stores the management information defined in the management information model, and AXIS [17] as a SOAP engine to apply the SOAP communication method between the manager and the agents.

5.1 User Interface

Figure 3 shows the information transformed from an XML document format to an HTML document format by using an XSL [18] and XSLT [19] file to show or to modify the configuration information. The *sharedInfo* is the read-write data, and the *referredInfo* data is read-only fields.

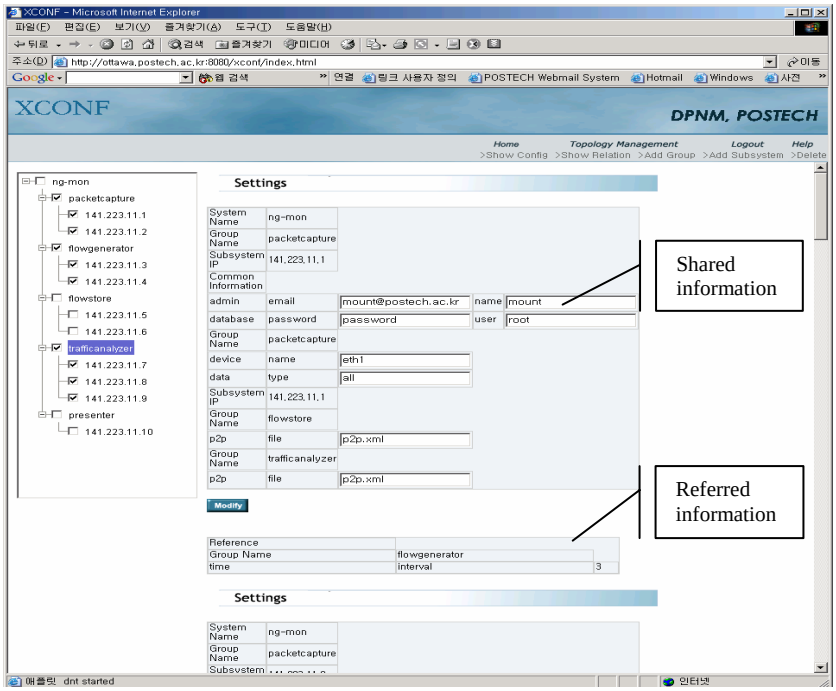
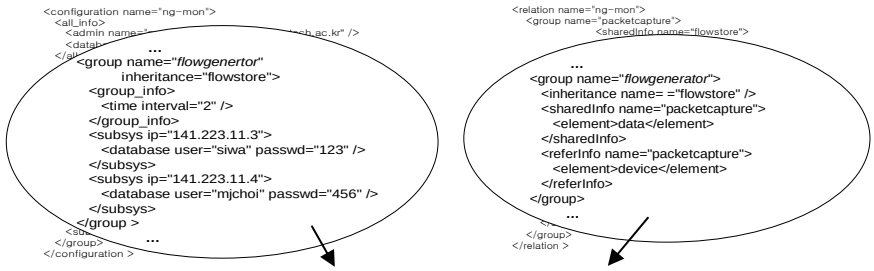


Figure 3. Web-based User Interface

5.2 Example

We have applied X-CONF to the configuration management system for NG-MON [20], which is a distributed and real-time Internet traffic monitoring and analysis system composed of five subsystems: packet capture, flow generator, flow store, traffic analyzer, and presenter of analyzed data. Each subsystem may be composed of multiple computers. The packet capture captures all packets on the network link. The flow generator sorts the captured packets into the flow containing the same 5-tuple: source IP address, destination IP address, protocol number, source port, and destination port. The flow store stores the flow data into the DB. The traffic analyzer queries the flow data to the flow store and then stores it according to the various analysis scopes in the own DB. The presenter provides a Web-based user interface to the administrators.



(a) Flow Generator's Configuration (b) Flow Generator's Relationship

```
<configuration name="ng-mon" ip="141.223.11.3" target="flowgenerator">
  <all_info>
    <admin email="mount@postech.ac.kr" name="mount" />
    <database password="password" user="root" />
  </all_info>
  <group name="packetcapture">
    <group_info>
      <data type="all" />
    </group_info>
  </group>
  <group name="flowgenerator">
    <group_info>
      <time interval="2" />
    </group_info>
    <subsys ip="141.223.11.3">
      <database passwd="234" user="siwa" />
    </subsys>
  </group>
  <group name="flowstore">
    <group_info>
      <p2p file="p2p.xml" />
    </group_info>
  </group>
  <referInfo name="packetcapture">
    <device name="eth1" />
  </referInfo>
</configuration>
```

(c) Configuration in Subsystem (141.223.11.3)

Figure 4. Example of Management Information

Figure 4 is an example of management information: (a) is configuration information in a manager, (b) is relationship information, and (c) is configuration information in a subsystem. Figure 4 (c) shows that the manager automatically creates the configuration information of the subsystem (*ip* = “141.223.11.3”) by using both Figure 4 (a) and (b).

In Figure 4 (a), the *flowgenerator* group consists of two subsystems. These subsystems have their own information because they include the sub-elements (*subsys*). The *flowgenerator* group has the *inheritance* attribute related with the *flowstore*. This means that the subsystems in the *flowgenerator* group inherit the entire configuration information of the *group_info* in the *flowstore* group. In this case, the inherited information is the value of the *p2p* element.

In Figure 4 (b) , the *flowgenerator* group has the relationship with the specific information of the *packetcapture* group. This group shares the *data* information in the *packetcapure* group and can modify it. However, the flowgenerator group refers to the *device* information in the *packetcapture* group. The *referInfo* cannot be modified in the referred group. If the *device* information is modified in the *packetcapure*, the modified value is transferred to the *flowgenerator*.

Finally, Figure 4 (c) shows the automatically generated XML document from Figure 4 (a) and (b) by the manager. The root element shows that the configuration information in Figure 4 (c) belongs to the subsystem (*ip* = “141.223.11.3”) in the *flowgenerator* group. If this system has the modified information, the manager processes the automatic reconfiguration of other related subsystems. In addition, the result of every process in these subsystems is sent to the manager. Also if rebooting the agent is necessary, the agent reboots itself and delivers the *reboot* notification to the manager.

6. Conclusion and Future Work

In this paper, we presented the design and implementation of X-CONF, which effectively manages configuration information using SOAP communication between the XML-based manager and the XML-based configuration agents. We have presented a general management information model that can be applied to the configuration management of distributed systems using an XML Schema. By using relation expressions such as *all_info*, *group_info*, *inheritance*, *referInfo*, *shareInfo*, there is no need to specify the same management information to represent shared properties. This avoids redundancies that are often found in configuration management among subsystems. X-CONF automatically transfers modified configuration information to the related subsystems when the configuration information is modified. If one of the related subsystems fails in the transaction, X-CONF rollbacks the operation. Therefore, X-CONF provides the consistency of the configuration information among the subsystems.

We applied the X-CONF to the configuration management system for NG-MON. For future work, we will validate the flexibility and extendibility of the X-CONF by applying it to other distributed systems.

7. References

[1] J. Case, R. Mundy, D. Partain, and B. Stewart, “Introduction and Applicability Statements for Internet Standard Management Framework”, RFC 3410, Dec. 2002.

[2] W3C, “Extensible Markup Language (XML) 1.0”, W3C Recommendation, Oct. 2000.

[3] W3C, “XML Schema”, W3C Recommendation, May 2001.

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

[5] W3C, “An XML Schema Document for SOAP RPC V1.2”, http://www.w3.org/2001/12/soap-rpc.

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

[7] Juniper Networks, “JUNOSCRIPT API SOFTWARE”, http://www.juniper.net/support/junoscript.

[8] 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.

[9] 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.

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

[11] 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.

[12] W3C, “XML Path Language (XPath) Version 2.0”, W3C Working Draft, Apr. 2002.

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

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

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

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

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

[18] W3C, “Extensible Stylesheet Language (XSL) Version 1.0”, W3C Recommendation, Oct. 2001.

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

[20] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju and James W. Hong, “The Architecture of NG-MON: A Passive Network Monitoring System”, 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2002), Montreal, Oct. 2002, pp. 16-27.