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
{siwa, mjchoi, jwkhong}@postech.ac.kr

Abstract

Today, we are seeing 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 are distributed in nature and may be deployed with different software components and run on heterogeneous computing platforms. The configuration information of a subsystem has relations with information of other subsystems in a distributed system. Therefore, a centralized configuration management system for distributed systems is necessary to meet requirements as the representation of complex relations of configuration information among subsystems and automatic reconfiguration of subsystems. This paper presents the design and implementation of X-CONF (Xml-based CONfiguration management system) to manage the configuration information of a distributed system using XML technologies. We use the XML Schema to define configuration information and relationship information among distributed system components. X-CONF automatically recognizes the changes of configuration information among distributed system components by referring to relationship information. We have developed a flexible and interoperable configuration management system by applying Simple Object Access Protocol (SOAP) as a communication method. Also, X-CONF provides a Web-based user interface to administrators for ubiquitous access. For validation, we have developed an XML-based configuration management system for NG-MON, which is a distributed and real-time Internet traffic monitoring and analysis system.

Keywords
Configuration Management, Distributed Systems Management, 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. A configuration management system is needed to efficiently manage the configuration information in each subsystem. It is tedious and time consuming for administrators to configure each subsystem through a direct console connection. Ideally, a configuration management system uses a centralized method in which a manager system can control all subsystems. The implementation and execution environments of all subsystems in the distributed system are 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 to those of other subsystems.

The larger the distributed systems are, the more various the relations of the configurations and the more diverse implementation environments among member subsystems become. A centralized configuration management system is required to consider the relationship to maintain consistency of
configuration information and to provide the communication method in a platform- and language-independent.

The Simple Network Management Protocol (SNMP) [2] 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. Only one object value defined by a MIB is accessed through an SNMP Get operation. The GetBulk operation must check the limitation of message size that can be retrieved in one request. SNMP MIB is based on a simple hierarchy structure of management information, because it is difficult to present dependencies among managed objects. XML [3] can represent the complex structure of management information using any tag (element and attribute). An XML-based protocol has been proposed as an alternative to the SNMP protocol, which complements the constraints mentioned previously.

In this paper, we present the design and implementation of X-CONF (Xml-based CONFiguration management system), which uses XML technologies to implement the configuration management system 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**

X-CONF is first concerned with how to define the configuration information with the XML Schema [4, 5, 6], which provides a powerful and extensible modeling capability. The management information model through the XML Schema defines configuration information to explicitly represent close relations among the subsystems. The management information model must be general and effective in order to be easily applicable to the configuration management of various distributed systems.

X-CONF is also concerned with how to exchange messages between a manager and agents in subsystems. HTTP communication is the most common in the exchange of XML messages. However, we apply the Simple Object Access Protocol (SOAP) [7] which is accepted as a standard communication protocol for XML. It is cumbersome to build a remote communication mechanism based solely on XML and HTTP. Based on XML and SOAP, the XML-based manager can directly call management operations in the agent via SOAP RPC [8]. Also, SOAP supports a standard description of operations using Web Services Description Language [32]. Therefore, it is very convenient to develop and extend the management operations. SOAP is generally accepted as a sufficient transport for Netconf [9]. Thus, we have chosen SOAP over HTTP as an effective communication method. In addition, SOAP is platform-independent and this places no restrictions on endpoint implementation technology choices. SOAP messaging is therefore interoperable with any platform and any device.

The organization of this paper is as follows. In Section 2, we present existing configuration management with SNMP and examine 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. In Section 6, we analyze the performance test result of our management system. Finally, we
conclude our work and discuss directions for future work in Section 7.

2. Related Work

In this section, we first explain existing configuration management with SNMP performed by a standard working group (snmpconf). We also describe related work on XML-based configuration management of standard activities and industrial efforts.

2.1 Configuration Management With SNMP

The snmpconf working group [10] outlined effective methods for using the SNMP framework to accomplish configuration management. This group attempted to define management information for device-specific as well as network-wide configuration. The group was also chartered to write any MIB modules necessary to facilitate configuration management. Specifically, they wrote MIB modules which described network entities capabilities and capacities which can be used by management entities making policy decisions at a network level or a device-specific level [11].

Snmpconf proposed guidance in the effective use of SNMP for configuration management [12]. This information is relevant to vendors that build network elements, management application developers, and those that acquire and deploy this technology in their networks. As proof of the concept, the working group also defined a MIB module [13] which described management objects to control the differentiated services policy in coordination with current efforts in the Differentiated Services Working Group [33].

The configuration management with SNMP revealed a weakness in management information modeling. A MIB module is insufficient to define relations and operations of configuration information. Also, the current configuration management with SNMP is bounded to configure network devices, namely, the element level. This is inappropriate to apply to configuration management of distributed systems.

2.2. XML-based Configuration Management

In this subsection, we introduce related work on XML-based configuration management.

2.2.1 IETF XML Configuration and Network Configuration

In the 54th IETF meeting in July 2002, a BOF session was held to discuss XML-based configuration management (XMLCONF). This BOF discussed the requirements for a network configuration management, and how the existing XML technologies, namely SOAP [7], WBEM [14], SyncML [15] and JUNOScript [16] could be used to meet those requirements. There are some Internet-Drafts [17,18] that present basic concepts and the requirements for XML network configuration and provide guidelines for the use of XML within IETF protocols.

The Network Configuration (netconf) Working Group [9] was formed in May 2003 to produce a protocol suitable for network configuration. The Netconf protocol will use XML for data encoding purposes. The working group suggests the NETCONF Configuration Protocol [34] as a starting point.

2.2.2 Cisco’s Configuration Registrar

The Cisco Configuration Registrar [19] 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.
2.2.3 Juniper Networks’ JUNOScript

Recently, Juniper Networks introduced JUNOScript [16] 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 in the DTDs and sending it to the JUNOScript server. The JUNOScript server delivers the request to the appropriate software modules within the device, encodes the response with JUNOScript tags, and returns the result to the client application.

The current configuration management with XML of standard activities and industrial efforts is also bounded to configure network devices like that with SNMP. This is inappropriate to apply to configuration management of distributed systems, which are our target systems.

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 configuration information of themselves. 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 of 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 the 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. Also, 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 for the configuration management to meet the requirements specified in Section 3. The modeling language chosen in our work is XML because XML enables us to present hierarchical as well as more complex structures, including relations between objects. We propose a general management information model that can be applied to any other distributed systems. This provides flexibility to our X-CONF to manage distributed systems. By using the interaction operation model, the agent can easily extend new operations. This provides the operational extensibility of our agents. This section also shows examples according to each model and describes the architecture of our X-CONF in detail.

4.1 Management Information Model

In order to use automatic reconfiguration of related subsystems, we propose a configuration information model and a relationship information model that focuses on the dynamic relationships of configuration management information of subsystems in a distributed system. The definitions of the core terms in the management information model are introduced as follows.

Table 1 shows the configuration management 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 element (anyElement) and that of attribute (anyAttribute) can be defined in any configuration management information.

<table>
<thead>
<tr>
<th>XML Schema of Configuration Information</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;xs:element name=&quot;configuration&quot;&gt;</td>
<td>&lt;configuration name=&quot;ng-mon&quot;&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;all_info&quot; minOccurs=&quot;0&quot;&gt;</td>
<td>&lt;all_info&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;anyElement&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;admin email=mount@postech.ac.kr name=&quot;mount&quot; /&gt;</td>
</tr>
<tr>
<td>&lt;xs:sequence maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;database password=&quot;password&quot; user=&quot;root&quot; /&gt;</td>
</tr>
<tr>
<td>&lt;/xs:sequence&gt;</td>
<td>&lt;/all_info&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;group&quot; maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;group name=&quot;packetcapture&quot;&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;anyElement&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;device name=&quot;eth1&quot; /&gt;</td>
</tr>
<tr>
<td>&lt;xs:sequence maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;data type=&quot;all&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/xs:sequence&gt;</td>
<td>&lt;/group_info&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;subsys&quot; maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;subsys ip=&quot;141.223.11.1&quot; /&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;anyElement&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;subsys ip=&quot;141.223.11.2&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;xs:attribute name=&quot;anyAttribute&quot; type=&quot;xs:xsSimpleType&quot; use=&quot;required&quot;/&gt;</td>
<td>&lt;/group&gt;</td>
</tr>
<tr>
<td>&lt;/xs:element&gt;</td>
<td>&lt;group name=&quot;flowgenerator&quot; inheritance=&quot;packetcapture&quot;&gt;</td>
</tr>
<tr>
<td>&lt;/xs:element&gt;</td>
<td>&lt;time interval=&quot;2&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/configuration&gt;</td>
<td>&lt;/group_info&gt;</td>
</tr>
<tr>
<td>&lt;/xs:element&gt;</td>
<td>&lt;subsys ip=&quot;141.223.11.3&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;subsys&quot; maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;database user=&quot;siwa&quot; passwd=&quot;123&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;anyElement&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;/subsys&gt;</td>
</tr>
<tr>
<td>&lt;xs:attribute name=&quot;anyAttribute&quot; type=&quot;xs:xsSimpleType&quot; use=&quot;required&quot;/&gt;</td>
<td>&lt;subsys ip=&quot;141.223.82.4&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;xs:sequence maxOccurs=&quot;unbounded&quot;&gt;</td>
<td>&lt;database user=&quot;mjchoi&quot; passwd=&quot;456&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;xs:element name=&quot;ip&quot; type=&quot;xs:string&quot; use=&quot;required&quot;/&gt;</td>
<td>&lt;/subsys&gt;</td>
</tr>
<tr>
<td>&lt;/xs:element&gt;</td>
<td>&lt;/group&gt;</td>
</tr>
<tr>
<td>&lt;/xs:sequence&gt;</td>
<td>...</td>
</tr>
<tr>
<td>&lt;/xs:element&gt;</td>
<td>&lt;/configuration&gt;</td>
</tr>
</tbody>
</table>

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 presents 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. 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 sharedInfo 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.

<table>
<thead>
<tr>
<th>XML Schema of Relationship Information</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;xs:element name=&quot;relation&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xs:element name=&quot;group&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xs:element name=&quot;inheritance&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xs:element name=&quot;element&quot; type=&quot;xs:string&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot; use=&quot;required&quot;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xs:element name=&quot;sharedInfo&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xs:element name=&quot;element&quot; type=&quot;xs:string&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot; use=&quot;required&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xs:element name=&quot;referInfo&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot;&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;xs:element name=&quot;element&quot; type=&quot;xs:string&quot; minOccurs=&quot;0&quot; maxOccurs=&quot;unbounded&quot; use=&quot;required&quot;&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Relationship Information Model

The log information stored in the log file is made of two kinds of information, such as operation results information and notification information. Table 3 (a) presents the results of each transaction at two sides, a manager and an agent. 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 get, modify, or load at the agent and modify, add or delete at the manager. The data is the XPath [20] 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.

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.

<table>
<thead>
<tr>
<th>(a) Transaction Information Model</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Schema of Transaction Information</td>
<td>Example</td>
</tr>
</tbody>
</table>
| <xs:element name="transaction ">
  <xs:element name="manager" minOccurs="0" maxOccurs="unbounded">
    <xs:attribute name="operation" type="xs:string" use="required" />
    <xs:attribute name="time" type="xs:string" use="required" />
    <xs:attribute name="result" type="xs:string" use="required" />
  </xs:element>
| <transaction name = "ng-mon">
  <manager operation="modify" time="2003:08:10:12:05:02" result="success" />
  <subsystem ip="141.223.82.1" group="packetcapture" operation="modify" data="/group[@name= "packetcapture"]/group_info/device" time="2003.08.10:12:06:55" result="success" />
  <subsystem ip="141.223.82.3" group="packetcapture" state="reboot" time="2003.08.10:12:06:56" result="success" />
  ... |
| </xs:element> |
| (b) Notification Information Model | Example |
| XML Schema of Notification Information | Example |
| <xs:element name="notification">
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="subsystem" minOccurs="0" maxOccurs="unbounded">
      <xs:attribute name="ip" type="xs:string" use="required" />
      <xs:attribute name="group" type="xs:string" use="required" />
      <xs:attribute name="state" type="xs:string" use="required" />
      <xs:attribute name="time" type="xs:string" use="required" />
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
| <notification name = "ng-mon">
  <subsystem ip="141.223.82.1" group="packetcapture" state="reboot" time="2003.08.10:12:06:56" />
  <subsystem ip="141.223.82.3" group="flowgenerator" state="reboot" time="2003.08.10:12:06:53" />
  <subsystem ip="141.223.82.2" group="packetcapture" state="reboot" time="2003.08.10:12:06:50" />
  ... |
| </xs:element> |

Table 3. Log Information Model

4.2 Interaction Operation Model

We use an RPC-based paradigm as a communication protocol. Specifically, SOAP [7] has been chosen for this. SOAP can access services, objects, and servers in a platform-independent manner and 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 is still fail, the manager must rollback the changed configuration information of all subsystems.

Also, the interaction messages between the manager and the agent are made in an XML document format. Table 4 shows the examples of the interaction message and simultaneously represents the operation description corresponding to operation model suggested by the Netconf IETF working group [9]. By using the unique messageld, 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 result message respondent to the request message (a). In Table (a) and (b), the value of the operation can take one of the agent’s management operation (get, load, modify). If the agent
receives the request message, it parses the value of the operation using an XML Parser and calls the 
operation. “<xconf:parameter>” includes the information of parameters used in the operation. In 
Table 4, “<xconf:subsystem>” contains the subsystem information or the configuration file name for 
the manager to distinguish the subsystem. 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.

### Table 4. The Interaction Operation Message Examples

<table>
<thead>
<tr>
<th>(a) Operation Request Message</th>
</tr>
</thead>
</table>
| `<perform-request messageID="1923-567y" operation = "modify">`  
| `<conf:subsystem>`  
| `<systemName>mg-mon</systemName>`  
| `<group>flowgenerator</group>`  
| `<subIP>141.223.82.3</subIP>`  
| `<fileName>mg-mon_flowgenerator_141.223.82.3</fileName>`  
| `<conf:parameter>`  
| `<xpath>//all_info/admin/@email</xpath>`  
| `<data>siwa@postech.ac.kr</data>`  
| `<xpath>//group[@name="trafficanalyzer"]//group//p2p/@file</xpath>`  
| `<data>test.xml</data>`  
| `<xpath>//subsystems[@ip="141.223.82.3"]//database/@user</xpath>`  
| `<data>mjchoi</data>`  
| `</conf:parameter>`  
| `</perform-request>`  
|  
| (b) Operation Result Message |  
| `</perform-result messageID="message id of targeted request" operation = "modify">`  
| `<conf:subsystem>`  
| `<systemName>mg-mon</systemName>`  
| `<group>flowgenerator</group>`  
| `<subIP>141.223.82.3</subIP>`  
| `<fileName>mg-mon_flowgenerator_141.223.82.3</fileName>`  
| `<conf:result>`  
| `<result>ok</result>`  
| `<error />`  
| `</conf:result>`  
| `</perform-result>`  
|  
| (c) Notification Message |  
| `<notify messageID = "0923-567b">`  
| `<conf:subsystem>`  
| `<systemName>mg-mon</systemName>`  
| `<group>flowgenerator</group>`  
| `<subIP>141.223.82.3</subIP>`  
| `<conf:state>`  
| `<action>reboot</action>`  
| `<conf:state>`  
| `</notify>`  

#### 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 Client**, and **Management Operation**. The manager possesses the list of subsystems and the 
configuration information of each subsystem in the XMLDB [21]. XMLDB is a special database designed only for XML documents, stores intact XML documents and partially controls the contents of the XML documents. XMLDB provides a better solution than the relational DB as the database for an XML document because it is difficult or impossible for complex and hierarchical XML structures to map into the simple structure of the relational DB. The **XMLDB Handler** module performs to process information in XMLDB. The **XSL/XSLT Processor** module transforms XML form into
HTML form to offer a Web-based user interface. The SOAP Client module connects to SOAP server to deliver the SOAP RPC messages. The last module, the Management Operation module, has five methods: getMethod, addMethod, delMethod, modifyMethod and createMethod. A brief summary of these operations is as follows.

- **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 the 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 information needed to a subsystem using the configuration management information and relationship information. Then, the manager calls loadMethod, a management operation in the agent.
- **notifyMethod**: The manager receives, processes notifications from the agents, and stores them into a log file via this method.

![Figure 2. Architecture of X-CONF](image)

The XML-based configuration management agent illustrated in Figure 2 contains a SOAP server module, an XML parser module and a Management Operation module. If the agent receives the request message, it parses the value of the operation using an XML Parser and calls the operation. 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 has three methods: getMethod,
loadMethod, and modifyMethod. A brief explanation of these operations is as follows.

- **getMethod**: This method is used to retrieve the information from the configuration XML file in the subsystem and to send 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 the 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.

Also, the subsystems need to send notifications to the manager, and the manager needs to receive and handle the notifications. To send the notification to the manager, the subsystems need a SOAP Client module and an HTTP Client module, and the manager needs a SOAP Server module and an HTTP Server module.

5. Implementation

We have implemented an XML-based configuration management system based on the X-CONF design presented in Section 4. The manager of X-CONF is a Linux 7.2 server with a Pentium III 800 MHz CPU and 256 MB RAM. X-CONF is implemented with XML related technologies. Therefore, we referred to the Apache Project Group [22] which provides Application Program Interface (API) implemented with JAVA to support related XML technologies. X-CONF needs following APIs: XML Xerces [23] as an XML parser [24, 25], Xalan [26] to transform XML document into other forms, Xindice [27] as an XMLDB and AXIS [28] as a SOAP engine to apply SOAP communication method between the manager and the agents.

The manager can manage the configuration information of any subsystem which has an agent embedded with a SOAP module and management operations. XMLDB supports XML technologies such as XPath [20], XQuery [29] and XUpdate [30] to directly handle the XML documents via DOM parser [24].

5.1 Information Repository

X-CONF includes XMLDB and XML files as an information repository in order to store all management information. In XMLDB, it is necessary to identify the terms collection and document. The collection is the container storing the XML document. The document is an intact XML document stored in a collection. Compared to a relational database, a collection is roughly equivalent to a table and a document resembles fields in a row which do not have a null value. Our X-CONF distinctly defines the name of each distributed system and creates the collection name with that name. For example, the distributed system, NG-MON creates an identical collection called NG-MON. XMLDB in the manager has several collections in the identical collection that stores the management information defined in the management information model. The configuration information of every subsystem is stored in the XML file, whose unique name is made using three pieces of information: distributed system name, group name, and subsystem IP address.

5.2 User Interface

A Web-based user interface is composed of three frames: the menu frame, the tree frame, and the main frame. Figure 3 shows the information transformed from an XML document format to an HTML document format by using an Extensible Stylesheet Language (XSL) [31] file to show or to modify the configuration information. XSL is a mark-up language designed to display XML documents on the Web. Figure 3 (a) presents the configuration information of each subsystem chosen by the administrator. If the administrator presses the modify button, Figure 3 (b), the result page, is displayed. The sharedInfo is the read-write data so that the sharedInfo data in Figure 3 (b) are the text fields.
(a) Show Configuration Information

(b) Modify Configuration Information

Figure 3. Web-based User Interface
5.3 Example

We have applied X-CONF to the configuration management system for NG-MON [1], 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.

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 whole 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 packetcapture 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 packetcapture, the modified value is transferred to the flowgenerator.

Finally, Figure 4 (c) shows 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. Performance Analysis

This section presents a performance analysis of X-CONF. We tested the performance of get operation in the agent. We classified the test with two groups: HTTP vs. SOAP and RDB (Relational DataBase) vs. XMLDB. Basically, XML format data is transferred via HTTP. We examined the overhead of SOAP compared with HTTP.

Table 5 analyzes the response time of the XMLDB query and the RDB query. To evaluate, we simply mapped the XML structure into RDB table fields and executed the select query statement from the command line interface regardless of processing time to the user interface. Actually, the number of the XML documents in XMLDB is 10. Also, the number of rows in the RDB table is 10. The first expression in Table 5 means that the manager retrieves all the configuration information of each subsystem in the DB. The second and third expression is more complex conditional statements to retrieve portions of the configuration information. The XPath expressions of XMLDB are replaced using the select and where statements in RDB.

<table>
<thead>
<tr>
<th>Query expression of RDB</th>
<th>Select *</th>
<th>Select data Where A=a</th>
<th>Select data Where A=a &amp; B=b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Time (ms)</td>
<td>10</td>
<td>21</td>
<td>30</td>
</tr>
<tr>
<td>XPath expression of XMLDB</td>
<td>/A[. =a]/data</td>
<td>//A[.='a']/B[.='b']/data</td>
<td></td>
</tr>
<tr>
<td>Response Time (ms)</td>
<td>43</td>
<td>42</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 5. Response Time of DB Query (RDB vs. XMLDB)

Table 6 compares the performance of HTTP and SOAP methods. We implemented the HTTP-based get operation in the agent for the performance test. The get operation brings the configuration information to a manager, the packet capture to the manager. Table 6 presents the exchanged message size and the latency between the manager and the agent. The overhead of SOAP is not severe compared with HTTP.

<table>
<thead>
<tr>
<th>Management Protocol</th>
<th>Latency (ms)</th>
<th>Request Message Sizes (bytes)</th>
<th>Response Message Sizes (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>18</td>
<td>15</td>
<td>56</td>
</tr>
<tr>
<td>SOAP</td>
<td>25</td>
<td>30</td>
<td>74</td>
</tr>
</tbody>
</table>

Table 6. Response Time of Get (HTTP vs. SOAP)

7. 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 in the configuration management system of other distributed systems.

References