Towards XML-based Configuration Management for Distributed Systems

Hyoun-Mi Choi, Mi-Jung Choi and James W. Hong
Graduate School for Information Technology
Pohang University of Science and Technology
{siwa, mjchoi, jwkhong}@postech.ac.kr

Young-Myoung Kim
Technology Group, KT
ymkim001@kt.co.kr

Abstract

Today, we are seeing more distributed systems on enterprise networks and on the Internet. It is difficult to effectively manage configuration information of these distributed systems because they are distributed in nature and they may be deployed with different software components and running on heterogeneous computing platforms. However, a centralized configuration management system for distributed systems is necessary to meet critical requirements as the representation of complex relations of configuration information of computers and their communication. This paper proposes the design of X-CONF (Xml-based CONFiguration management system) to meet these requirements using XML technologies. XML is a data modeling language used to define configuration information, which supports to represent more complex structures including relations of information rather than simple lists. X-CONF automatically recognizes changes of configuration information among distributed system components because of a formal structure of configuration information specification. We have designed a flexible and interoperable configuration management system by applying Simple Object Access Protocol (SOAP) as a communication method. Also, X-CONF provides 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, real-time Internet traffic monitoring and analysis system.

Keywords
Configuration Management, Distributed Systems Management, XML-based Configuration Management, XML, XML Schema, SOAP

1. Introduction

As high-speed networks are emerging, most large-scale software systems are composed of a number of computers in a distributed computing environment. These systems are implemented with many computers to distribute the processing. In this paper, 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 a management overhead 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 environments of all subsystems are various, as well as partial relations existing between these subsystems. A partial relation means that some parts of the configuration information of a subsystem can be shared with other subsystems.

For example, NG-MON [23] is a distributed, 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. All subsystems have common configuration information such as administrative information and the user ID and password of database. In the case of the packet capture subsystem, it has configuration information...
such as IP address of flow generator and the time interval of the flow generation, which is shared with the flow generator subsystem. That is, there exists a partial relation between the packet capture and the flow generator. Also, the packet capture has configuration information such as IP address of flow store and p2p traffic information, which is shared with the flow store. The p2p information is also shared with the traffic analyzer. In addition to this configuration information, the packet capture subsystem has its own configuration information. In this manner, the configuration information of the packet capture subsystem has complex relations with other subsystems in the NG-MON.

If configuration information in a subsystem is changed, all subsystems that had relations with it must be automatically replaced with new information. The larger the distributed systems are, the more complex the relations of the configurations and the more various implementation environments among member subsystems become. A centralized configuration management system is required to consider the relationship to maintain consistency of configuration information and communication methods.

Simple Network Management Protocol (SNMP) [1] is the most widely used method for network management on the Internet. However, retrieving large volumes of information via simple operations of SNMPv1 is impossible. Only one value of an object defined by MIB is accessed through an SNMP v1 operation. Moreover, SNMP MIB is based on a simple hierarchy structure of management information, so it is difficult to present the dependency among managed objects. XML [7] 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 SNMP protocol, which complements the constraint mentioned previously.

In this paper, we propose the design 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 XCONF, where XML-based manager controls multiple subsystems equipped with an XML-based configuration management agent.

*Figure 1. High-Level X-CONF Architecture*

Our configuration management system is concerned with how to define the configuration information with XML Schema [4, 5, 6], which provides powerful and extensible modeling. The configuration information specification must be defined in a formal structure to explicitly represent close relations concerning configuration information shared with other subsystems as well as to easily apply XML [7] technologies such as XSL [8], XSLT [9], and XPath [10]. The formal structure must be so logical and effective as to easily apply to any other systems. X-CONF is also concerned with how to exchange messages between managers and agents in subsystems. HTTP communication is the most common in the exchange of XML messages. However, we apply Simple Object Access Protocol (SOAP) [11] which is accepted as a standard protocol for XML. The SOAP provides a better solution than HTTP because the XML-based manager can directly call basic functions in the subsystem via SOAP RPC [12]. The main task of exchanging messages in the management system is not to simply deliver the messages but to execute a management function in the agent. We selected SOAP as an effective communication method because there is no need for additional process in the
agent. In addition, SOAP is platform independent and thus it places no restriction on the 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 an overview of XML-related technologies 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 XCONF system. Finally, we conclude our work and discuss directions for future work in Section 6.

2. Related Work

In this section, we first explain XML and its related technologies, such as XSL, XSLT, XMLDB and SOAP. We also describe related work on XML-based configuration management.

2.1 XML Related Technologies

In this subsection, we overview XML and XML related technologies

2.1.1 XML

Extensible Markup Language (XML) [7] is a meta-markup language, which is the W3C standard for document exchange on the Web. Information held in an XML document is self-describing. XML, unlike HTML, separates contents from presentation. The presentation of an XML document is provided by a standard stylesheet language, XSL (Extensible Stylesheet Language) [8]. XML supports powerful modeling features by DTD or XML Schema [4, 5, 6], which defines the structure, and validates the contents of a document. Presently, XML has become pervasive in various application areas because it is easy to learn, simple to use, and inexpensive. In addition, XML represents explicitly and simply a complex structure of a document having some relations using various tags.

2.1.2 XSL and XSLT

Extensible Stylesheet Language (XSL) [8] is a mark-up language designed for illustrating the method to display XML documents on the Web. XML documents describe only the contents and the structure of the contents. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses a formatting vocabulary. That is, XSL enables XML to separate contents from presentation. XSL consists of two parts: a language for transforming XML documents, and an XML vocabulary for specifying formatting semantics. The style sheet technology to transform documents is XSL Transformation (XSLT) [9], which is a subset of XSL technology that fully supports the transformation of an XML document from one format into another, such as HTML or another custom XML document type. The reason for publishing the XSLT specification separately from XSL is that XML documents can be displayed, providing an easy display format for end users by transforming the XML documents without formatting semantics.

2.1.3 XMLDB

A XML database (XMLDB) [13] is a special database designed for only XML document, which stores intact XML document and partially control the contents of the XML document. The XMLDB provides a better solution than a relational DB as a 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. In addition, XMLDB provides various XML related technologies such as XPath [10], XQuery [14], and XUpdate [15], so that it directly processes the contents of an XML document parsed by XML parser.

It is necessary to identify the terms of a collection and a document used in the XMLDB.
Collection is the container storing the XML document. 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 is like fields in a row which does not have a null value. XMLDB includes variable XML documents and another collection in a collection.

2.1.4 SOAP

Simple Object Access Protocol (SOAP) [11] is a lightweight protocol for exchanging information in a distributed environment. It is an XML-based protocol that consists of three parts: an envelope that defines a framework for describing the contents of a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP defines the use of XML and HTTP or SMTP to access services, objects, and servers in a platform- and language-independent manner.

2.2. XML-based Configuration Management

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

2.2.1 IETF XML Configuration BOF

In the 54th IETF meeting in July 2002, a BOF session concerned with XML configuration (XMLCONF) was held. This BOF discussed the requirements for a network configuration management, and how the existing XML technologies, namely SOAP, WBEM [16], SyncML [17] and JUNOScript [3] could be used to meet those requirements. There are some Internet-Drafts [18, 19, 20] that present the basic concepts and the requirements for XML network configuration and provide guidelines for the use of XML within IETF standards protocols.

2.2.2 Cisco's Configuration Registrar

The Cisco Configuration Registrar [2] 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 Agents located at each device. The Configuration Registrar delivers the initial configuration to Cisco devices when starting up on the network for the first time. 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 [3] 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 the 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.

3. Requirements Issues

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 them. That is, configuration information is shared among the component subsystems. To effectively manage the configuration information of each subsystem, requirements of configuration
management system for the distributed system are as follows.

1) Show, delete, modify the configuration information in the subsystem selected by an administrator
2) Add a new subsystem and delete subsystems that an administrator selects.
3) Provide Web-based user interface for a 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 is shared among the subsystems. Also, the communication between a manager and agent in each subsystem must be generated without any other constraints.

4) Sufficiently present complex relations among subsystems in the description of configuration information.
5) Propose a formal structure to describe configuration information of a subsystem. This formal structure of configuration information must be easily extended to other subsystems and should allow an XML-based manager to apply XSL and XSLT for the user interface.
6) Maintain the consistency of the configuration information among remote subsystems. A manager automatically permits to advertise to all subsystems sharing the changed configuration information. As well, whenever a subsystem is added or deleted, the manager also reflects the new changed information into the configuration information of other subsystems related to it. Then, the subsystems must reboot themselves in order to apply the new changed configuration information.
7) Communicate between a manager and an agent in each subsystem regardless of the implementation environment.

Configuration management systems for distributed systems must satisfy the above requirements. The configuration information specification defines a formal structure using XML Schema. XML as a data modeling language enables to present the hierarchical as well as more complex structures including relations between objects. Therefore, XML meets requirements 4) and 5).

XMLDB [23] satisfies the requirement 6). XMLDB in a manager includes all configuration information of subsystems to manage. If the configuration information shared with other subsystems is modified, the manager accesses all the changed configuration information in the XMLDB using XPath [9] expression and thoroughly replaces them with the new configuration information. Then the manager sends the new configuration information to all subsystems related to it. Finally, SOAP provides a solution of the last requirement because it can access services, objects and servers in a platform-independent manner.

4. Design of X-CONF

In this section, we define XML Schema of the configuration information to meet the requirements mentioned in Section 3. Then we explain the structure for the data to be stored in the XMLDB in the manager and agent of X-CONF.

Figure 2. XML Schema of the Configuration Information

Figure 2 shows the description of configuration information of the each subsystem presented by XML Schema. The ‘1..8’ expression under the group element means that one or more group
elements exist. A subsys element also has one or more elements. A group element can therefore have several subsys elements. The subsystems that perform the same work have almost the same configuration information, so they can be classified into an identical group. An all_info element, a child element of the Configuration element, contains the configuration information shared with all subsystems. Any element can be a child element of all_info element according to the configuration information. A group_info element describes the configuration information shared with the same group. A subsys element contains the configuration information independent of other subsystems. A manager must know the meaning of this formal structure. By identifying the parent element to which the changed configuration information belongs, the manager decides where to deliver the changed configuration information.

Figure 3 is XML Schema of the data stored in the XMLDB of the manager. SubsysList of Figure 3 (a) enumerates the subsystems for a manager to monitor and ConfigList of Figure (b) contains all the configuration information of each subsystem. When an administrator adds a new subsystem belonging to a certain group, if the group does not exist, a new group is created; otherwise the existing group is used. A manager adds the new subsystem into the SubsysList and it also adds its configuration information into the ConfigList. Then the manager inserts the new configuration information into the existing configuration information of other subsystems related to it. As a manager separately manages the list of the subsystems in XMLDB, it is easy for the manager to explore, add, or delete subsystems.

![SubsysList diagram](image1)

(a) Subsystem list of the XMLDB

![ConfigList diagram](image2)

(b) Configuration list of the XMLDB

**Figure 3. XML Schema of Data in the XMLDB**

The configuration information of all subsystems is stored into XMLDB, which improves the overall performance for manager to search and retrieve the configuration information of the selected subsystem. When some subsystems are corrupt, all the configuration information in the ConfigList can be used as a backup data.

Figure 4 illustrates the detailed architecture of X-CONF, in which a centralized XML-based manager controls the configuration information of subsystems equipped with an XML-based configuration agent. A manager is divided into four modules: XMLDB handler, XSL/XSLT processor, SOAP client, and Operation module. The XMLDB handler module retrieves and modifies the configuration information from XMLDB and inserts the new configuration information into the XMLDB. The XMLDB supports XML technologies such as XPath, XQuery and XUpdate to directly handle the XML documents via DOM parser [21]. The XSL/XSLT processor module transforms XML form into HTML form to offer Web-based user interface. The SOAP client module connects to SOAP server to deliver the SOAP RPC message. The last module, the Operation module, has three methods: getMethod, modifyMethod, and setMethod. This module is used for retrieving, modifying, and saving the configuration information of the XMLDB in the manager as well as that of the subsystem.
The XML-based configuration management agent illustrated in Figure 4 contains a SOAP server module, which defines the methods to be called from a manager, the Operation module as the applications of a subsystem and an XML parser module. The Operation module reads and writes the configuration information. The XML Parser module allows for an agent to directly handle not only all of the configuration information but also a portion of it. The agent parses the exiting configuration information, and accesses the contents of the configuration information by simple XPath expression. We consider a SAX parser [22] as an XML parser in the agent. The SAX parser invokes the callback methods whenever it parses because of event driven interface. The disadvantage of the SAX parser does not exceed than that of the DOM parser [21]. The embedded agent concerns the consumption of the resources, so that the DOM parser is not suitable because it requires more memory than the SAX parser.

For example, the subsystem A has relations both with the subsystem B and the subsystem C. The configuration information of the subsystem A and the subsystem B is almost the same because of performing the same actions. The subsystem A and the subsystem C partially share the configuration information. In this case, the subsystem A and B is bound in a group and the portion of the configuration information of the subsystem C, which is shared with the system A, is encountered in the configuration information of the subsystem A.

If an administrator modifies the configuration information shared with other subsystems in the distributed system, the X-CONF manager automatically processes as following order. First, if the configuration information of the subsystem is not in the XMLDB, the manager brings the configuration information from the subsystem and stores the configuration information into the XMLDB. The manager shows the administrator the configuration information obtained from XMLDB after passing XSL & XSLT processing module. Then the manager checks the parent element of the changed configuration information. If the changed information is a child element of all_info, the manager informs all subsystems of the changed configuration information; if the changed element is a child of group_info, the manager delivers the changed information to all the subsystem in the group; otherwise, the manager modifies only the configuration information of the own subsystem. The message that the manager sends to a subsystem is two kinds. One is the message including all the configuration information of the own subsystem and the others contains only the new changed configuration information. The manager sends these messages to maintain the consistency of the configuration information between the XMLDB and the subsystems after processing the data of the XMLDB.
5. Implementation

We have implemented an XML-based configuration management system based on the X-CONF design presented in Section 4. The distributed system applying the architecture of X-CONF is NG-MON [23] which is a passive network monitoring system with a cluster architecture for the load distribution. This section explains X-CONF implementation environment and describes about the manager, the agent, and user interface in detail.

The manager of the X-CONF is a Linux 7.2 server with Pentium III 800 MHz CPU and 256 MB RAM and the platform of the agent is different according to the platform of the subsystem. Each subsystem of NG-MON runs on a Linux server with Pentium III 800 MHz CPU and 256 MB RAM. X-CONF is implemented with XML related technologies. Therefore, we referred to Apache Project Group [24] which provides Application Program Interface (API) implemented with JAVA to support related XML technologies. X-CONF needs following APIs: XML Xerces [25] as an XML parser [21, 22], 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.

SOAP is standardized by W3C. SOAP communication method delivers SOAP messages over HTTP, FTP, SMTP, etc. In this paper, we choose HTTP. The reasons for selecting HTTP are that it is currently the most "firewall friendly" protocol for remote method call operation, has a great record of successful interoperability and has excellent security features for non-repudiation and confidentiality. The manager and the subsystems must setup Web server for HTTP communication. Considering user interface implementation with Java Server Page (JSP), the manager installs TOMCAT provided in Jakarta Apache [29].

SOAP is a standard protocol for XML, which is a platform- and language-independent. The manager can manage the configuration information of any subsystem which has an agent embedded with a SOAP module and management application functions.

The NG-MON consists of five subsystems: packet capture, flow generator, flow store, traffic analyzer, and presenter of analyzed data. Also, each subsystem such as packet capture, flow generator, and traffic generator may be composed of multiple computers. The packet capture is capturing all packets on the network link. The flow generator splits 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 stored in the flow store and then stores according to the various analysis scopes. The presenter provides the Web-based user interface to the users. All subsystems in the NG-MON have common configuration information such as administrative information and the user ID and password of database. In the case of the packet capture subsystem, the packet capture has configuration information such as IP address of the flow generator and the time interval of the flow generation, which is shared with the flow generator. Also, the packet capture has configuration information such as IP address of flow store and p2p traffic information, which is shared with the flow store. The p2p information is also shared with the traffic analyzer. In addition to this configuration information, the packet capture has private configuration information such as its own database information.

Figure 5 is an example of an XML-based formal structure of the configuration information of a subsystem (packet capture) in the NG-MON [22]. The root element (configuration) has three attributes: name (ng-mon), ip (141.223.11.1), and target (packetcapture). The name attribute is the name of the distributed system. The ip attribute is the IP address of the subsystem. The target attribute is the group name. If a manager asks the subsystems in the same group to simultaneously perform the same management action, the manager can make very simple query using the group name. An instance is when the manager wants all subsystems in the group to reboot.
Figure 5. XML-based Formal Structure of the Configuration Information

Table 1 (a) is the list of the configuration information of all subsystems in the XMLDB. When the manager requests the configuration information, it obtains the configuration information from the list in the XMLDB. Through Figure 5 and Table 1, the subsystem (packet capture) of the IP address (141.223.11.1) has a relation with the subsystems of the IP address from 141.223.82.2 to 141.223.82.8. The system (141.223.11.1) belongs to the packetcapture group and the subsys name is packetcapture1 as shown in Table 1 (b).

Table 1 (b) shows the lists of the subsystems in the XMLDB. The subsys element has two attributes: the ip attribute presents the IP address for the manager to distinguish subsystems with a unique value and the name attribute for the administrator to discriminate the subsystems.

<table>
<thead>
<tr>
<th>(a) Configuration list managed by manager</th>
<th>(b) Subsystem list managed by manager</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;ConfigList&gt;</code></td>
<td><code>&lt;SubsysList&gt;</code></td>
</tr>
<tr>
<td><code>&lt;configuration name=&quot;ng-mon&quot;</code></td>
<td><code>&lt;group name=&quot;packetcapture&quot;</code></td>
</tr>
<tr>
<td>ip=&quot;141...1&quot;`</td>
<td><code>&lt;/group&gt;</code></td>
</tr>
<tr>
<td>target=&quot;packetcapture&quot;&gt;</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td><code>&lt;configuration name=&quot;ng-mon&quot;</code></td>
<td><code>&lt;subsys name=&quot;packetcapture1&quot; ip=&quot;141...1&quot;/&gt;</code></td>
</tr>
<tr>
<td>ip=&quot;141...8&quot;`</td>
<td><code>&lt;subsys name=&quot;packetcapture2&quot; ip=&quot;141...2&quot;/&gt;</code></td>
</tr>
<tr>
<td>target=&quot;presenter&quot;&gt;</td>
<td><code>&lt;/subsys&gt;</code></td>
</tr>
<tr>
<td>...</td>
<td><code>&lt;/group&gt;</code></td>
</tr>
<tr>
<td>&lt;/ConfigList&gt;`</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;group name=&quot;flowstore&quot;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/group&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;subsys name=&quot;flowstore1&quot; ip=&quot;141...5&quot;/&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;subsys name=&quot;flowstore2&quot; ip=&quot;141...6&quot;/&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/subsys&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/group&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/SubsysList&gt;</code></td>
</tr>
</tbody>
</table>

Table 1. List of the XMLDB

5.1 Manager

A manager proposed in Section 4 has four modules: XMLDB handler, XSL/XSLT processor, SOAP client, and Operation module. The XMLDB handler module is implemented based on API provided in the Xindice. The methods that this module provides are in Table 2: xQuery, xUpdate, xInsertElement, addDocument, and xDelete.
<table>
<thead>
<tr>
<th>Methods &amp; parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XMLDB Document handler</strong></td>
<td><strong>Method</strong></td>
</tr>
<tr>
<td>xQuery</td>
<td>Retrieve all the data or the portion of data, and return it.</td>
</tr>
<tr>
<td>xUpdate</td>
<td>Set the new changed data.</td>
</tr>
<tr>
<td>xInsertElement</td>
<td>Insert new element into existing data. According to the number of attribute of new element, the number of the parameters of this method is different.</td>
</tr>
<tr>
<td>xDelete</td>
<td>Delete all or the portion of data.</td>
</tr>
<tr>
<td>addDocument</td>
<td>Add the whole contents of the XML document into the XMLDB.</td>
</tr>
<tr>
<td><strong>parameters</strong></td>
<td><strong>xPath</strong></td>
</tr>
</tbody>
</table>

Table 2. Methods of the XMLDB Handler

Table 3 shows the example of XPath expression used in the manager. The expression in Table 3 (a) is used to obtain the IP address to access the subsystems in the same group. Table 3 (b) is the expression that the manager examines what is the parent of the element selected by the administrator. The manager decides how to process after confirming the name of the parent element. The expression in Table 3 (c) is used when the manager wants to know the group name of the selected element.

(a) Request the IP address of the subsystems in same group.
```
//configuration[@ip="ip_address"]//subsys[@name="group_name"]/host/@ip
```

(b) Check what is the parent of the selected element.
The manager must know that the configuration information of the selected element is shared with all subsystems or with a group, otherwise, the selected element is the private element needed in only one subsystem.
```
/*[name(//selected_element/parent::*='all_info')]
/*[name(//selected_element/parent::*='group_info')]
/*[name(//selected_element/parent::*='subsys')]
```

(c) Request the group name of the selected element.
```
//selected_element/parent::group_info/parent::group/@name
```

Table 3. Examples of XPath Expression Used in the Manager

The XSL/XSLT processing module transforms an XML document by applying XSL stylesheet. The result of the process is the string value of the transformed data. This module transforms an XML document into an HTML document to easily provide Web-based user interface to an administrator.

(a) SOAP Request
```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:xsd=http://www.w3c.org/....
 xmlns:SOAP-ENV=http://schemas....
 xmlns:xsi="http://www.w3c.org/...." >
 <SOAP-ENV:Body>
   <ns1:get
 xmlns:ns1=http://soapinterop.org/>
   <arg0 xsi::type="xsd:string">//configuration</arg0>
   </ns1:get>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(b) SOAP Response
```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:xsd=http://www.w3c.org/....
 xmlns:SOAP-ENV=http://schemas....
 xmlns:xsi="http://www.w3c.org/...." >
 <SOAP-ENV:Body>
   <ns1:getResponse
 xmlns:ns1=http://soapinterop.org/>
   <result xsi::type="xsd:string">
     <configuration name="ng-mon">
       ip="141.223.1.1"
       target="packetcaputre"
     </configuration>
   </result>
 </ns1:getResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Table 4. SOAP Messages
The SOAP client module defines the declaration for the RPC service to call the method in the agent. Table 4 is an example of the SOAP request message (a) and SOAP response message (b). This message is used when the manager requests the entire configuration information in the agent.

The Operation module is implemented with Java Server Page (JSP). This module is divided into three methods: `getMethod`, `modifyMethod`, and `setMethod`. These methods are composed of the functions of the other modules of the manager mentioned previously. This module provides Web-based user interface to an administrator.

### 5.2 Agent

An agent is composed of a SOAP server module, an XML parser module and Operation module. The SOAP server module is implemented using Java Web Services (JWS) provided in AXIS [28]. This module declares the methods which are called by a manager. The declared methods are a `getMethod` and a `setMethod` in the agent. The XML parser module checks the XPath expression and parses the XML document of the configuration information in the subsystem with the XPath expression. The XPath expression used in the agent is not as complex as in the manager. The Operation module reads and writes the configuration information in the agent. Table 5 shows the methods of the agent.

<table>
<thead>
<tr>
<th>Functions moduler</th>
<th>Methods &amp; parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>getMethod</code></td>
<td>Retrieve all the data, and return it.</td>
</tr>
<tr>
<td></td>
<td><code>setMethod</code></td>
<td>Modify or delete the new changed data.</td>
</tr>
<tr>
<td></td>
<td><code>parameters</code></td>
<td><code>XPath</code> Present the position of the information to access using XPath expression. e.g., <code>//Configuration //group[@name='name']/p2p</code></td>
</tr>
</tbody>
</table>

Table 5. Methods of the Agent

### 5.3 User Interface

Web-based user interface provides on the Web as follows: viewing, modifying, and saving the data of the configuration information. Figure 6 (a) shows the subsystems list and Figure 6 (b) presents the configuration information of the subsystem through passing the XSLT processing module using the XSL stylesheet.

![Figure 6. Web-based User Interface](image_url)
6. Conclusion and Future Work

In this paper, we proposed the design and implementation of the X-CONF, which effectively manages the configuration information using the SOAP communication between the XML-based manager and the XML-based configuration agents. X-CONF automatically advertises modified configuration information to the related subsystems via the defined XML form when the configuration information shared with other subsystems is modified. This maintains the consistency of the configuration information among the subsystems.

We applied the X-CONF to the configuration management system for NG-MON. For the future work, we will validate the flexibility and extendibility of the X-CONF by developing configuration management system for other distributed systems. Finally, we are going to propose a general and formal XML Schema for the configuration information specification to be easily applied to the configuration management of other distributed systems.

References