Integrating SNMP Agents and CLI with NETCONF-Based Network Management Systems

Baozhen Wu  
School of Computer Science  
Donghu College of Wuhan University  
Wuhan, China  
wbzyyjj@163.com

Yanan Chang  
Computer School of Wuhan University  
Wuhan, China  
cyn_2332@yahoo.com.cn

Abstract—With the development of Internet, the efficient network management of heterogeneous network devices from different vendors becomes more and more difficult. The Network Configuration Protocol (NETCONF) is presented as a new solution. But it can’t be widely used in current networks soon because most devices are based on Simple Network Management Protocol (SNMP) and Command Line Interface (CLI). Therefore this paper gives an approach that how to integrate SNMP and CLI with NETCONF-based network management systems to be compatible with nowadays network devices. We will focus on the conversion of data model between two different data modeling languages and mapping of messages in NETCONF/SNMP gateway and NETCONF/CLI gateway. The NETCONF manager can communicate with non-NETCONF devices by the implementation of proposed architecture.

Keywords—SNMP Agents; CLI; Network Configuration Protocol

I. INTRODUCTION

In order to meet the ever-increasing demands, it is therefore necessary to develop efficient network management tools. At present, the most popular tool is SNMP which has been widely used as an industrial standard in network management for its simplicity and flexibility [1]. After that SNMPv2 and SNMPv3 were proposed to solve the emerging problems with retrieval of large chunks of data and security of transmission [2]. Moreover, SNMP uses UDP to transfer messages, which results in unreliability and limited size of messages compared to stream-based bulk transfers [3].

Most of the configuration on real network devices is done with CLI because of the SNMP shortcomings for complex configuration demands. But CLI also has drawbacks [4]. It requires extra efforts by operators to learn different commands for different devices. In addition, by now there isn’t any formal description language to define all properties of the programmatic interface. Finally, CLI doesn’t have any structured error responses and any standard organization of management information.

Under such background, NETCONF offers opportunities to solve these problems. NETCONF, proposed by IETF in 2006, uses XML to exchange messages and store management information [5]. It has powerful function of configuration management to meet the diversity of configuration demands. NETCONF adopts the connection-oriented Simple Object Access Protocol (SOAP) over HTTP to ensure the security of the transmission [6]. It adds new characteristics of notification and access control. What’s more, many existing software tools of XML will make the development of NETCONF-based network management system more effective and efficient.

To the best of our knowledge, a few prototypes about NETCONF have been developed. And in the long run, the fully NETCONF-based network management system will offer great benefits. Nevertheless, those implementations may not be widely used in current networks because most devices only support SNMP or use CLI to do the network management. So it will be usually the situation that there is a NETCONF manager on one side and a non-NETCONF device on the other. But the NETCONF manager must communicate with non-NETCONF devices in order to do supposed operations, taking the monitoring or configuration for example. Therefore this paper aims to give the approach to bridge this gap which integrates SNMP agent and CLI-based devices with NETCONF-based network management systems.

The rest of this paper is organized as the below. The related work is introduced in Section 2. Section 3 presents the architecture of NETCONF-based integrated network management. NETCONF/SNMP gateway and NETCONF/CLI gateway are separately illustrated in Section 4 and Section 5. Finally Section 6 draws some conclusions.

II. RELATED WORK

SNMP uses Abstract Syntax Notation One (ASN.1) to describe MIBs while there isn’t a standard data modeling language proposed for NETCONF. NETCONF is based on XML, so XML Schema Definition (XSD) is considered as a better way to express NETCONF data model for its protocol-independence, diversity of data types, higher hierarchical data expressiveness and familiarity by people [7]. Reference [8] develop a SNMP MIB to XML translation algorithm. Until now, one of the popular tools converting MIBs to XSD files is smidump in LIMSMI [9]. Reference [10] depicts how the smidump converts a MIB to an XSD file. Until now, there have been several prototypes of NETCONF-based network management including NETCONF/SNMP gateway. For example, the EnSuite project of Madynes Research team and Instant NETCONF Manager of Tail-f Company both leave the interface with SNMP [11]. However, they don’t take into account the efficient expression of data model.
 Compared to SNMP, there aren’t many tools or studies about integrating CLI with NETCONF network management systems. Different kinds of devices from various vendors have different data organization and commands. Reference [12] and [13] show us how to make XML/CLI gateway of Juniper’s devices. Reference [14] gives a method to implement how the SNMP manager communicates with non-SNMP devices, which can provide some experience to our work. So in the following sections, a seamless integration of SNMP and CLI with NETCONF-based network management system will make up for the deficiencies.

III. ARCHITECTURE

NETCONF has done very well in network configuration and after that many new characteristics of monitoring, access control and security managements have been added. But now, most devices still support SNMP, CLI and many other tools. The NETCONF manager should have the ability to communicate with non-NETCONF agent. So this section, the architecture about NETCONF-based integrated network management will be shown in Figure 1 to solve the problems of incompatibility.

In Figure 1, the top level is the NETCONF manager. It can manage various devices from different vendors using different protocols by monitoring, configuration, fault treatment and so on. The second level is the most important part. The NETCONF agent implements the base protocol operations, some of the capabilities and notification. There are NETCONF/SNMP gateway and NETCONF/CLI gateway to translate messages and data organization between NETCONF manager and SNMP agent or CLI-based devices. Moreover, there still leave interfaces with others to be compatible with different kinds of devices as many as possible. Finally, the managed devices are in the bottom. This architecture is of good extensibility, which not only saves the funds for changing new devices, but also gives enough time for making network devices fully support NETCONF agent.

How to find devices more easily and quickly is also a problem, and there are two approaches to solve this problem. One is the traditional topology which is used widely in current network management systems. It can find all the managed devices on the internet by well-designed routing algorithm. The other one is using new web services technologies based on service-oriented architecture (SOA). Every agent can be considered as a service. The action of manager is regarded as the inquiry and request of services. All the agents should leave an interface with a Web Services Description Language (WSDL) generator. With WSDL generator, agent information can be generated in a formal format in WSDL files and thus the agent can publish its service information on Universal Description, Discovery and Integration (UDDI) register center in order that the manager can find them easily and quickly. When the manager finds the information about the agent such as IP address and agent type which are in the WSDL files.

![Figure 1. NETCONF-Based Integrated Network Management Architecture](image)

IV. NETCONF/SNMP GATEWAY

A. Conversion from MIB to XSD

Communication between SNMP and NETCONF needs the conversion of the data model between two kinds of data modeling languages. It will make no sense without the correct and effective conversion of data model. But the XSD files generated by smidump can’t be used here for reasons mentioned in related work. In this section we will make some improvement of smidump and then give the flow of the conversion of data model.

Figure 2 shows us the architecture of conversion from MIB to XSD. When a MIB comes, the first step is to do the lexical analysis. It should primarily do the scanning of words in the MIB file with the filtering of annotation, blank, ‘\t’ and ‘\n’. Then it will recognize a set of tokens with absolute meaning and translate them into the corresponding grammar expression which matches the grammatical rule. In addition, lexical check runs through the whole process of lexical analysis. Syntax analysis is the second step. Its main function is to check that whether the token sets get from the lexical analysis module is in line with the defined grammar rule. If the result is true, then a grammar tree will be generated. Otherwise it will be passed to the error handling module.

After lexical analysis and syntax analysis there comes the most important conversion part. With respect to the data model, the conversion includes four main parts: import declaration, data type translation, macro clause translation and structure change. The import declaration is the same with what smidump does. Every IMPORT/FROM clause in SNMP is divided into two parts in XSD: One is to make abbreviation declaration of namespace and the other is to import the foregoing namespace and its schema location. The data type translation is shown in reference [15]. It is an Internet draft written by myself and submitted to IETF this March. In the draft, the data types are defined on the base of restriction on build-in data types in XSD. It not only preserves the unambiguous of data types but also reduces the changes that vendors make to be suitable for translated data.
types. The macro clause translation is most commonly used in the whole conversion. There are nine kinds of macro clauses. We omit four kinds of conformance macro clauses because they only describe the smallest requirements in implementation on agent and don’t refer to any definition of new nodes. The translation of remained five macro clauses is similar.

The last part in conversion is structure change. Previous MIB tree has so many layers that some of the middle nodes are of little use which are only for the organization of its children. Based on the problems of smidump, we optimize the complex structure of MIB to a flattened and simple structure of four layers especially for NETCONF and its separation of configuration and state data. The new structure still reserves the relationship between leaf nodes and their parents. The details of the four layers are described as the below:

- The first level is the root of the document level which is the only entrance of the tree.
- The second level includes three branches of the root separately named “configuration”, “state” and “notification”. And the operations on these branches aren’t the same. The <get-config> operation retrieves configuration data only, while the <get> operation retrieves configuration and state data.
- The third level representing either the containers of scalar nodes or the entry of table nodes.
- The fourth level of all leaf nodes including scalars nodes or columnar nodes

After the translation is the OID generation which is used to locate nodes in message mapping between NETCONF and SNMP. With the last step of OID generation, we can show the complete XSD files converted from MIB.

B. Message Mapping

Currently, SNMP is the most popular tool among all the network management tools. So we give the detail design of NETCONF/SNMP message mapping shown in Figure 3.

In Figure 3, when the gateway receives a message, it will first take off the encapsulation ordered by SSL, HTTP and SOAP. Then the message is parsed to get the data of every part needed for SNMP by XML parser tools. Message mapping is divided into four parts: operation mapping, error type mapping, OID mapping and other mappings. The types of operation defined in NETCONF and SNMP are not completely the same. For example, the operation <lock> doesn’t exist in SNMP. So the mapping relationship of the operation is shown in Table 1.

<table>
<thead>
<tr>
<th>NETCONF</th>
<th>SNMPv2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;get&gt;</td>
<td>&lt;get-config&gt;</td>
</tr>
<tr>
<td>&lt;edit-config&gt;</td>
<td>&lt;copy-config&gt;</td>
</tr>
<tr>
<td>&lt;notification&gt;</td>
<td>&lt;set&gt;</td>
</tr>
<tr>
<td>None</td>
<td>&lt;get-next&gt;</td>
</tr>
<tr>
<td>&lt;lock&gt;</td>
<td>&lt;unlock&gt;</td>
</tr>
</tbody>
</table>

The error types also needed to be mapped between NETCONF and SNMP. There are 18 kinds of error status in NETCONF and 19 kinds in SNMPv2. Some of them can directly translate into each other. For instance, the error “access-denied” in NETCONF has the same meaning with “noAccess” in SNMP v2. Another part is mapping node name to OID. Because the messages from manager don’t include the OID which is the only identifier to locate the nodes in SNMP, we must generate OID in gateway. The namespaces in every message stands for a unique MIB and is used for the gateway to decide which MIB it will load, then through the translation algorithm described above, there is a XSD file corresponding to MIB generated. The OID is in form of annotation in the XSD file, and we can get the OID easily.

After the conversion of message, following module encapsulates message and sends it to SNMP agent. Similar to the process from the top down stated above, the order of flow from SNMP agent to NETCONF manager is just reversed. In addition, there also needs a conversion of Trap. The information includes managed objects, time stamp, trap type, IP address and network devices. Until now, there is still no
standard of notification-specific tagged content. Thus we generate the notification message in form of description text of every item in SNMP Trap PDU and send it to manager.

V. NETCONF/CLI Gateway

There are also many devices using CLI to do configuration management, and different kinds of devices from different vendors have different data organization and CLI commands, which increases our difficulty in developing NETCONF/CLI gateway. For example the data organization of Juniper’s devices isn’t the same with Cisco’s devices. This section gives a general frame of conversion in Figure 4. In the translation, every managed object has two mapping files, one of which stores its own specific commands mapping of NETCONF operation and the other stores the data organization in XML. For instance, if the manager wants to configure the router of Juniper, firstly a router module file should be generated. And we get the concrete path of management information by the module file. All these files are initialized before the gateway starts, and during the whole process, the user can add new files if necessary.

VI. CONCLUSION

This paper aims to give the approach to bridge this gap which integrates SNMP agent and CLI-based devices with NETCONF-based network management systems. It translates MIB module to XSD used in NETCONF and use NETCONF/SNMP gateway to access MIB data. In addition, it also proposed an approach that how NETCONF manager communicates with CLI-based devices. These methods are of great value and use in the condition that the configuration data model hasn’t been standardized and the interface for NETCONF hasn’t been widely used.

In the future, we will pay more attention on NETCONF itself. First of all, some new element should be added in NETCONF particularly in access control. The eXtended Access Control Markup Language (XACML) is an XML-based access control standard, with widespread acceptance from the industry and good open-source support. We should make studies to propose a profile that defines how to use XACML to provide fine-grain access control for NETCONF commands. On the other hand, Kalua, NCX and YANG are all new data modeling languages proposed for NETCONF recently in IETF. We should pay close attention to the study of these languages and focus on the character of configuration about data model.

REFERENCES