A Design of IoT Device Configuration Translator for Intent-Based IoT-Cloud Services

Chaehong Chung*, Jaehoon (Paul) Jeong**

*Department of Electrical and Computer Engineering, Sungkyunkwan University, Republic of Korea

** Department of Interaction Science, Sungkyunkwan University, Republic of Korea

darkhong@skku.edu, pauljeong@skku.edu

Abstract— **The rapid growth of IoT (Internet of Things) makes life better, but it can be a burden for IoT users to manage heterogeneous IoT devices in a timely and appropriate manner. In this paper, we propose an IoT device configuration translator for IoT-cloud services that enables IoT users who do not have expertise in IoT environments to efficiently configure their IoT devices. The user's high-level configuration based on natural language is delivered to the translator in IoT-cloud platform using NETCONF (Network Configuration Protocol). The translator, which is designed with three components using automata theory and database mapping information, translates the high-level configuration into a low-level configuration applicable to the IoT devices. This allows IoT users to configure their IoT devices without considering which protocols and platforms they work on.**

Keywords— **IoT, Translator, IoT-cloud, NETCONF, YANG**

I. INTRODUCTION

The IoT (Internet of Things) has been received constant attention as it accommodates various things such as sensors, actuators and cellular devices. IoT is primarily focused on the connectivity of things, and IoT users monitor various environment from remote IoT devices and set up new configurations with feedback. According to Ericsson's report, the connectivity of cellular IoT is expected to grow by about 27% each year, and 4.1 billion cellular IoT connections will be expected by 2024 [1]. The growing number of IoT connections needs for management of dynamically changing situations. In order to manage such situations, IoT device configuration based on IoT-cloud services is required according to an IoT user's intent [2], [12].

The current IoT-cloud can provide IoT users with a variety of services. However, with the increase of IoT devices, IoT-cloud providers are also diverse, and support different protocols and services [3]-[5]. This can cause confusion for IoT users who are not experts on IoT environments, and they may not be able to set up the right configuration at the right time. As a result, several problems can occur. For example, let us consider a security camera as an IoT device. If the security camera is not able to set up the dynamic configuration in a timely manner, the blind spots can be exposed to crime and can allow damage to property or life. Also, if the configuration of the air conditioner is not properly set up, it can cause discomfort to the IoT users. Thus, Dynamic IoT device configuration can prevent IoT users from both small discomfort and large casualties.

In this paper, we propose an IoT device configuration translator for intent-based IoT-cloud services to solve the problems. IoT users can use the IoT device configuration translator to accurately and timely configure the desired devices regardless of which IoT-cloud services and platforms they use. When an IoT user requests an IoT device configuration in a natural language, a high-level configuration is composed of JSON or XML data in a REST (Representation State Transfer) format and transmitted to the translator in the IoT-cloud using the NETCONF (Network Configuration Protocol) [11]. The translator is based on the Automata theory, and it works as follows. The first step is data extraction from the high-level configuration using DFA (Deterministic Finite Automaton). The second step is data conversion to convert the extracted data into low-level data for IoT device mapping with a given required IoT service. The Third step is low-level configuration generation with converted low-level data using CFG (Context-Free Grammar). The low-level configuration is an XML format of YANG (Yet Another Next Generation) model [10]. In order to deliver the low-level configuration to the IoT device, our system uses delivery protocols, such as NETCONF/MQTT [6].

Note that the translator in this paper extends the idea of security policy translation from our previous research in order to apply the IoT device configuration [8].

The rest of this paper is organized as follows. Section 2 shows the IoT security configuration and security policy translation as related work. Section 3 describes the proposed architecture and data format. Section 4 explains the concept of IoT device configuration translator and the translation process. Finally, Section 5 concludes this paper along with future work.

II. RELATED WORK

A. IoT Security Configuration

As considering the rapidly increasing IoT connections, IoT security is getting important for the IoT device security. IoT device security needs to protect systems, networks, and data. There are several IoT security attacks with vulnerabilities in IoT devices. For example, there are communication attacks

that expose data exchanged between IoT devices, IoT device software attacks, and IoT device physical attacks. To protect from these attacks, IoT security configuration is one of the solutions. IoT devices should be configured when anyone who wants to use it for both services and security aspects [9]. Therefore, the on-demand security configuration technique was proposed, and in this paper, we expand this security configuration for general IoT device configuration using our translator.

B. Security Policy Translation

I2NSF (Interface to Network Security Functions) is a system to provide various Network Security Functions, such as web filters and firewalls, for users who are not experts in network security. The users do not consider any security policies because there is a security policy translation that automatically finds users security policies [8]. This research is motivated by the security policy translation. Also, the security policy translation is based on Automata theory, but there are several limitations. One of the main limitations is that it can only apply to I2NSF which is a specific and narrow environment. In this paper, we expand this translation mechanism into IoT environments for IoT device configuration.

III. PROPOSED ARCHITECTURE

In this section, we present the architecture and data format of the IoT device configuration translator. In IoT environment where many IoT devices use different protocols and platforms, our goal is to provide efficiency for IoT device configuration for IoT users. Using our translator, IoT users can configure their IoT devices without considering which protocols and platforms they work on.

A. IoT Device Configuration Translator Architecture

This subsection describes the overall architecture of our IoT device configuration translator. Figure 1 shows the architecture of our proposed IoT device configuration translator. The architecture consists of IoT user, IoT-cloud platform, IoT-cloud provider (e.g., Google Cloud Platform and Samsung ARTIK Cloud), and IoT endpoints (e.g., IoT devices and sensors).

An IoT user, who is not an expert, manages and uses IoT devices and sensors. This user does not have IoT knowledges but uses the IoT-cloud platform provided by IoT-cloud providers. The services and functions provided by each platform are diverse and difficult for non-expert IoT users to use. Therefore, most of IoT users cannot use IoT-cloud services properly and cannot even set up the desired IoT configurations.

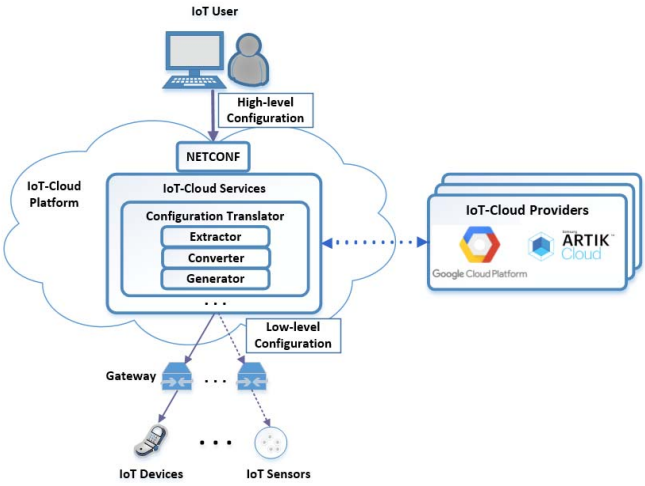

Figure 1. The Architecture of an IoT device configuration translator

To solve this problem, the users' high-level configuration is based on a YANG data model with the NETCONF. The NETCONF module in the IoT-cloud platform delivers the users' high-level configuration to the translator. The translator extracts the high-level configuration data through the "Extractor" and converts it to the low-level configuration data by referring to database mapping information in the IoT-cloud database using the "Converter". Finally, the "Generator" creates the low-level configuration for IoT endpoints. The low-level configuration is encoded in various protocol types such as NETCONF/MQTT and is transmitted to a gateway or IoT endpoints to be configured.

B. IoT Configuration Data

There are two types of IoT configuration data: high-level configuration and low-level configuration.

High-level configuration is the information for configuring an IoT user's IoT device, and it is represented as a JSON format. For the convenience of user's configuration, it enables natural language-based configuration of data as a "string" type. Figure 2 shows an example of a high-level configuration. In this example, the user sets up the current temperature in a natural language.

Low-level configuration is the information after the high-level configuration goes through the IoT device configuration translator. It is represented as an XML format, and it can be applicable to IoT devices. Figure 3 shows an example of a low-level configuration. In this example, we can see the specific IoT device configuration data such as a UUID (Universally Unique Identifier), command value and unit, and timestamp. The design of the translator and the its process are described in Section 4.

```
{
    "user": {
        "description": "User identified by userID",
        "userID": {
            "type": "string",
            "value": "testuser"
        }
    },
    "device": {
        "description": "Device identified by deviceName",
        "deviceName": {
            "type": "string",
            "value": "airconditioner"
        }
    },
    "configuration": {
        "description": "Configuration of IoT endpoints",
        "command": {
            "type": "string",
            "value": "nigth mode"
        },
        "time": {
            "type": "string",
            "value": "now"
        }
    }
}
```


Figure 2. Example of high-level configuration


```
<?xml version="1.0" encoding="UTF-8" ?>
<iot-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message
from="testuser">
  <data>
    <device-id>
      <uuid>39B1F4D-1A62-4A4F-89B6-E18F798D4ED8</uuid>
    </device-id>
    <command>
      <name="temperature" momentary="true" automaticReadout="true"
value="24.5" unit="celsius"/>
    </command>
    <timestamp>2019-10-01T19:09:05</timestamp>
  </data>
</iot-config>
```


Figure 3. Example of low-level configuration

IV. Design of IoT Device Configuration Translator

This section describes the structure of our proposed IoT device configuration translator and its translation process. This translator consists of three components: Extractor, Converter, and Generator. The following subsections explain these three components, respectively, along with the translation process.

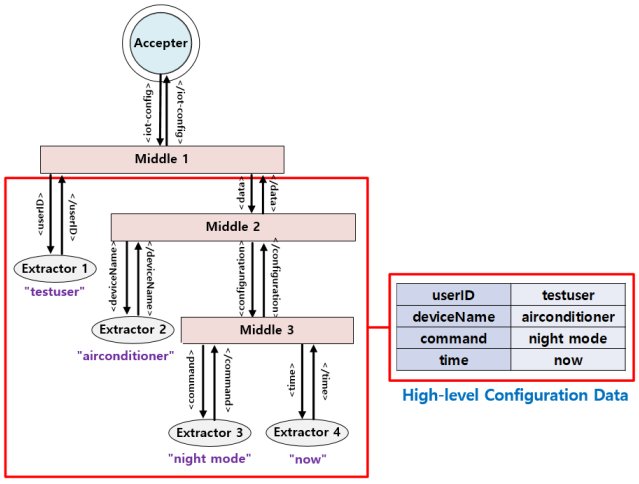

Figure 4. Example of Extractor schema

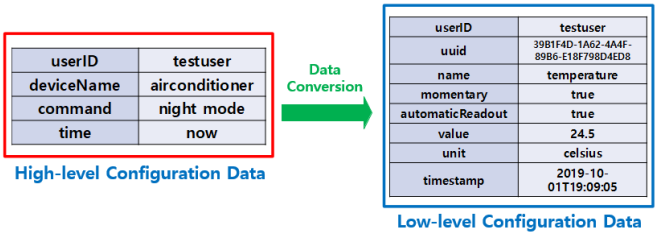

Figure 5. Example of converting data

A. *Extractor*

This subsection explains the extractor of the IoT device configuration translator. The extractor was designed based on DFA (Deterministic Finite Automaton). DFA is a finite state machine that accepts or rejects symbols by running through a state sequence determined by the input string [13]. In other words, the high-level configuration, which is the input string, is an application of a finite state. The first step is to change the JSON format of high-level configuration into the XML format to use an extractor. The characteristic of XML format is that each data element is composed of symbols between "<" and ">" (or "</" and ">"). Using this characteristic, it is possible to determine the start symbol "<" and the end symbol "</" of the XML structure. Then, it can extract the high-level configuration data. Figure 4 shows the extracting process using the high-level configuration example in Figure 2. Starting with "Accepter", the XML data is read sequentially to identify the "<" symbol, and then the state is moved. If the extractor reads the "</" symbol, it saves the data value and returns to the previous state. For example, in Figure 4, if the extractor reads "<userID>" and "</userID>", the data value "testuser" between them will be saved as an extractor, and then the current state will return to the "middle 1" which is the previous state. When the last state returns to "Accepter", the extraction is completed without any data structure errors.

B. Converter

This subsection explains the converter of the IoT device configuration translator. The converter converts high-level configuration data to low-level configuration data, based on mapping information in IoT-cloud database. In the ER model (Entity-relationship model) using "userID" as the primary key and "deviceName" as the foreign key in the high-level configuration data, the corresponding data is converted based on the mapping information. The current mapping information is somewhat static. The reason is that it is inevitable to do the initial mapping setup manually. However, when mapping information and conversion data are enough, automatic mapping will be attempted using the machine learning techniques in future. This is left as future work. Figure 5 shows the conversion from the high-level configuration data to the low-level configuration data using the mapping information in IoT-cloud database. According to the mapping information, "deviceName" of the high-level configuration data is converted to "uuid" of the low-level configuration data. In the same way, "command" of the high-level configuration data is converted to "name", "momentary", "automaticReadout", "value", and "unit" for the specific low-level configuration data.

C. Generator

This subsection explains the generator of the IoT device configuration translator. The generator is designed based on CFG (Context-Free Grammar). CFG is a certain type of formal grammar that consists of rules for generating nonterminal symbols and terminal symbols [13]. The low-level configuration data obtained through the converter is generated in the following production rules:

$$[config] \rightarrow [config][config], \quad (1)$$

$$[config] \rightarrow \langle tag\rangle[data]\langle /tag\rangle, \quad (2)$$

$$[data] \rightarrow \langle tag\rangle[rule1][rule2]\ldots[ruleN]\langle /tag\rangle, \quad (3)$$

$$[data] \rightarrow data1|data2|\ldots|dataN, \quad (4)$$

([rule] can be either nonterminal symbol or terminal symbol)

Production rule (1) indicates that a nonterminal symbol *[config]* can be used multiple times. Production rule (2) has *[data]*, which is a nonterminal symbol between <*tag*> and </*tag*> where *[config]* matches the XML format. Production rule (3) shows scalability by using *[rule]* several times in the same way with production rule (2). Production rule (4) means that CFG can be generated by assigning various data values to the terminal symbol *[data]*. In figure 6, it depicts the generation from the low-level configuration data to the low-level configuration XML format using CFG.

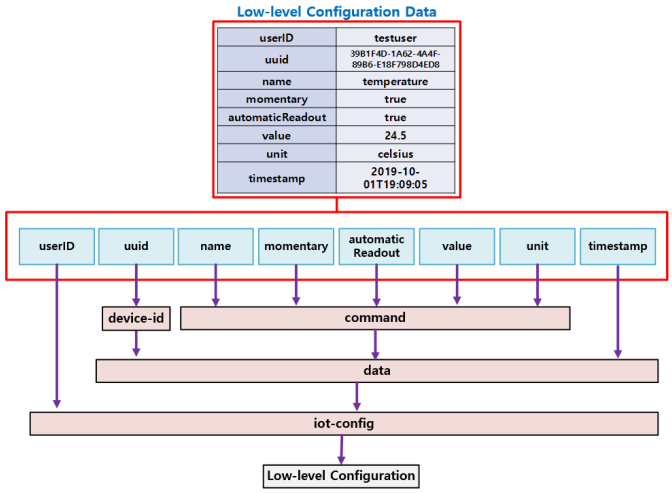

Figure 6. Example of generator schema

D. Process of IoT Device Configuration Translation

This subsection describes the overall translation process using the three components mentioned above. The translation process is as follows:

1) An IoT user sends a high-level configuration (i.e., Figure 2) in JSON or XML format to the IoT-cloud platform using the NETCONF.

2) The NETCONF module in the IoT-cloud platform receives the high-level configuration to verify the data, and it sends the configuration to IoT device configuration translator (i.e., Figure 1) in XML format.

3) The extractor component of the translator extracts the high-level configuration data using DFA (i.e., Figure 4), and it sends the data to the converter component (i.e., Figure 5).

4) The converter component of the translator converts the extracted high-level configuration data into low-level configuration data using the mapping information in the database, and it sends the low-level configuration data to the generator component (i.e., Figure 6).

5) The generator component of the translator generates a low-level configuration XML format (i.e., Figure 3) from the low-level configuration data using CFG.

6) The low-level configuration XML format is delivered with a target IoT device's protocol (e.g., NETCONF/MQTT) to configure the target IoT device.

V. Conclusions and Future Work

This paper proposes an IoT device configuration translator for intent-based IoT-cloud services. We designed a translator to provide convenience of IoT users. In the management of a large number of IoT devices, it can be necessary for IoT users to use this translator and configure their IoT devices without considering the environments.

As future work, we will attempt to use the machine learning techniques to make automatic mapping and converting the high-level configuration into the low-level configuration in the translator. Also, we will evaluate our translator to see which machine learning algorithms are suitable for the translator.

ACKNOWLEDGMENT

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A1B03035885). This work was supported in part by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2017-0-01633) supervised by the IITP (Institute for Information & Communications Technology Promotion).

REFERENCES

[1] Ericsson, "IoT connections outlook". [Online]. Available: https://www.ericsson.com/en/mobility-report/reports/november-2018/iot-connections-outlook (accessed on 10 October 2019).

[2] H. Yang, and Y. Kim, Design and Implementation of High-Availability Architecture for IoT-Cloud Services, Sensors, July. 2019.

[3] M. Zdravkovic, M. Trajancovic, J. Sarraipa, M. Lezoche, et al., Survey of Internet-of-Things platforms, ICIST 2016, Feb. 2016.

[4] T. Pflanzner, and A. Kertesz, A Survey of IoT cloud providers, MIPRO, May. 2016.

[5] P. P. Ray, A survey of IoT cloud platforms, Future Computing and Informatics Journal, vol. 1, pp. 35-46, Dec. 2016.

[6] T. Scheffler, and O. Bonneb, Manage resource-constrained IoT devices through dynamically generated and deployed YANG models, ANRW '17, July. 2017.

[7] S. Vijay, and M. K. Banga, Management of IoT Devices in Home Network via Intelligent Home Gateway using NETCONF, UBICNET 2017, pp. 196-207, Jan. 2018

[8] J. Yang, and J. Jeong, An Automata-based Security Policy Translation for Network Security Functions, ICTC 2018, Oct. 2018.

[9] B. Chung, J. Kim, and Y. Jeon, On-demand security configuration for IoT devices, ICTC 2016, Oct. 2016.

[10] M. Bjorklund, "YANG – A Data Modeling Language for the Network Configuration Protocol (NETCONF)", IETF RFC 6020, Oct. 2010.

[11] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)", IETF RFC 6241, June. 2011.

[12] I. Ullah, S. Ahmad, F. Mehmood, and D. Kim, Cloud Based IoT Network Virtualization for Supporting Dynamic Connectivity among Connected Devices, Electronics 2019, June. 2019.

[13] P. Linz, An Introduction to Formal Languages and Automata, Jones & Bartlett Learning; 5 edition, 2011.

Chaehong Chung is a M.S. Student in the Department of Computer Engineering at Sungkyunkwan University since 2019. His MS advisor is Professor Jaehoon (Paul) Jeong. He got a BS degree from department of Software at Sungkyunkwan University in 2019. His research interests include Internet of Things (IoT), Software Defined Networking (SDN), and Network Functions Virtualization (NFV).

Jaehoon (Paul) Jeong is an associate professor in the Department of Software at Sungkyunkwan University in Korea. He received his Ph.D. degree in the Department of Computer Science and Engineering at the University of Minnesota in 2009. He received his B.S. degree in the Department of Information Engineering at Sungkyunkwan University and his M.S. degree from the School of Computer Science and Engineering at Seoul National University in Korea in 1999 and 2001, respectively. His research areas are Internet of Things, Software-Defined Networking, Network Functions Virtualization, security, and vehicular networks. Dr. Jeong is a member of ACM, IEEE and the IEEE Computer Society.