Architecture of Scalable Embedded Device Management System with Configurable Plug-In Translator

Hideya Ochiai
jo2lxq@hongo.wide.ad.jp

Hiroshi Esaki
hiroshi@wide.ad.jp

Graduate School of Information Science and Technology
The University of Tokyo

Abstract

Ubiquitous computer network shall include wide variety of devices. This means that the ubiquitous network certainly increases the heterogeneity of nodes connected to the Internet, while increasing the number of nodes. This paper proposes a system architecture to solve this architectural challenge by the introduction of two sub-systems: "Configurable Plug-In Translator System" and "Dynamic Management Script Resolver System". The translator system provides generalized interfaces to applications, by absorbing the heterogeneous low-level interfaces with script execution engine. Here, the behavior of every remote node is described by the script language. The resolver system achieves a global scale script repository, using DNS, inheriting the advantages of DNS, i.e., scalability, robustness and authority management capability. The integration of these two sub-systems autonomously provides application software with generalized interfaces to access any remote device node on the global Internet.

1 INTRODUCTION

In these days, many embedded electronic devices, such as sensor and actuator nodes, tend to have communication capability based on TCP/IP technology, in order to control and to manage these nodes through the Internet. However, we may observe that these device nodes have their own proprietary application interface for these remote nodes. As of today, we do not have any established "De-Facto" or "De-Jule" standardized protocol to communicate with these (embedded) device nodes.

In the era of ubiquitous networking with a lot of device nodes, it will be getting in general that some node on the Internet wants to access wide variety of device nodes and that a node wants to access the remote node whose device profile is unknown.

This paper discusses and proposes a system architecture, that can achieve generalized access to these device nodes. We propose two sub-systems in order to achieve this challenge. The one is "Configurable Plug-In Translator" and the other is "Dynamic Management Script Resolver System". The translator system mediates the communication between application and device nodes, parsing and evaluating the language-based script provided by device node owner through the resolver system. Here, the evaluation involves the translation of data and protocol. The resolver system (supported by DNS) inherits the advantage of DNS, such as scalability, robustness and management capability.

The rest of this paper is organized as follows. In the next two sections, we describe the proposed two subsystems: "Configurable Plug-In Translator" and "Dynamic Management Script Resolver System" respectively. We address the implementation and its evaluation in section 4, while we discuss the future work in section 5. Finally, section 6 gives the conclusion of this paper.

2 CONFIGURABLE PLUG-IN TRANSLATOR SYSTEM

In this section, we address the architecture of the proposed translator system. Figure 1 shows the operational concept of the proposed system. The interface of device node must be described with a certain script form provided by some authority (e.g., by device deployment organization). Then, the script obtains the portability. i.e., we can handle the device node with the same manner as an abstracted object, even with the different execution platform. The translator system, at the right side in the figure, parses and evaluates the corresponding description of the device interface in

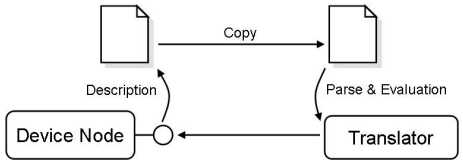

Figure 1. Description and Evaluation

the left side of the figure.

2.1 Description of device node interface

The proposed system must be able to deal with many types of interface and protocol. We realize that data-oriented description (e.g., Web Services Description Language[1]) is not appropriate because of the following reasons.

Data-oriented description :

- tends to be designed, assuming the particular underlying protocol
- cannot provide logical extension

Though the data-oriented description system may include many types of description scheme, it often requires another software to actually use them.

Based on the above observation, We adopted the process-oriented description (also called as the language-based description). Since the process-oriented description can express the processing sequence of data, the translator can communicate with many kinds of device nodes, absorbing the heterogeneous proprietary interfaces and protocols.

There are many kinds of process-oriented description. We can classify them into the following three categories.

1. Script Language
2. Intermedicate Code
3. Object Code

We evaluate these three description methodologies with the following two metrics.

1. Universal Runtime Environment
2. Description Modification Capability

In order to execute these descriptions, translator requires the corresponding runtime environment. From the runtime environment point of view, we realize that object code is not applicable in todays environment. This is because the object code can only run on the specific CPU and OS. As for the intermediate code, once the intermediate code is generated, we cannot modify it. On the other hand, the script language can be modified whenever you want, can be combined with other scripts and can be automatically generated by some services.

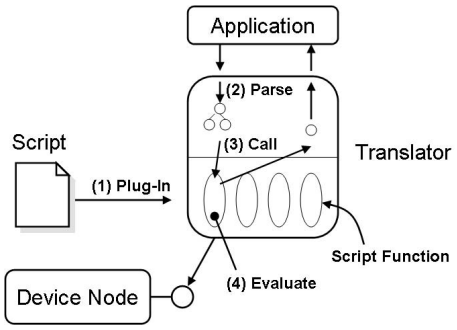

Figure 2. Translation Mechanism

Based on the above observation, we adopted the script language methodology for the process-oriented description in the proposed system.

2.2 Translation Mechanism

Translator mediates the communication between application and device node. Figure 2 shows the behavior of the proposed translator system.

1. Parsing the script of the target device node
2. Translator parses the application request
3. The translator calls the corresponging function described in the script language
4. The transation engine evaluates the function of the script, involving the communication with the device node, using TCP/IP sockets. This script evaluation performs the translation of application request and of device response.

3 DYNAMIC MANAGEMENT SCRIPT RESOLVER SYSTEM

The translator system must obtain the appropriate script for the target device node. In this section, we propose the scalable script management and destribution system so that the translator can obtain the appropriate script dynamically and automatically using the script resolution sub-system, whenever it wants to communicate with the device node.

We call this proposed system as "Global-Scale Script Resolution Service". The proposed script resolution system uses Domain Name System (i.e., DNS). We also

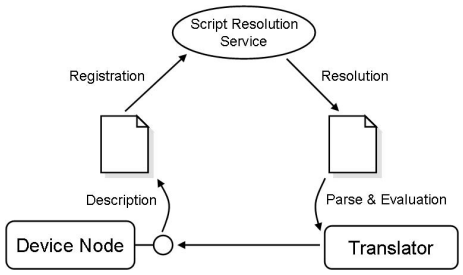

Figure 3. Overall System Architecture

show how the translator system described in the previous section and the resolution service system described in this section are integrated in the proposed system.

3.1 Global-Scale Script Resolution Service

In order to introduce the script resolution service easily, we propose to take advantage of DNS. By appending the script registry servers to DNS with SRV record[3], and pointing to these servers with NAPTR record[5] which is corresponded to each device node, the translator sub-system can resolve the appropriate script of the target device node via the following procedure.

1. Inquire the device FQDN to DNS by NAPTR
   The response message indicates the corresponding service location of script registry server(s) which manage the target script.
2. Inquire the translation script to one of these servers
   The response message includes the script.
3. Evaluate the posted application request(s) and the script. The translator can provide the communication with the target device node

3.2 Overall System Architecture

Figure 3 shows the integration of the two proposed sub-system, which is a kind of "SOA", Service-Oriented Architecture.

1. Device service provider registers the description of the service interface to the global registry.
2. Translator resolves the script from the registration system
3. Translator communicates with the target device node, mediating the communication between the application and the corresponding device node.

4 IMPLEMENTATION

In this section, we describe the implementation of the proposed system. The translator is implemented by JAVA. We use PICNIC[6], as a device node. PICNIC has the UDP interface for the Internet communication and we can access temperature sensor and 7 segment LED.

4.1 Script Design

The script of the prototype system consists of several command entries. Inside these command entries, the stub code which specifies the behavior of translator is described respectively. The stub code description is a functional language on XML.

The following is an example of the command entries.

```
<interface>
  <command name="config">
    <!-- stub code 1 -->
  </command>
  <command name="get">
    <!-- stub code 2 -->
  </command>
  <command name="display">
    <!-- stub code 3 -->
  </command>
</interface>
```

And the following is an example of the stub code section.

```
<dgRecv>
  <dgSend>
    <dgSocket>
      <text>203.178.135.86</text>
      <int>10001</int>
    </dgSocket>
    <getq name="myarray" />
  </dgSend>
</dgRecv>
```

These script means that the translator should send the contents of "myarray" to the node "203.178.135.86" with UDP port 10001 and return the response datagram.

4.2 Translator System

Translator system is mainly composed of three parts: (1) "Command Channel" which decomposes application requests, (2) "Script Resolver" which obtains the script of the target node, and (3) "Script Evaluator" which translates the application requests based on the script.

The implementation package diagram is shown in Figure 4 (generated by Eclipse UML[2]). Figure 5 shows the result of translation of application commands. It achieves the translation into UDP messages that can operate PICNIC device node.

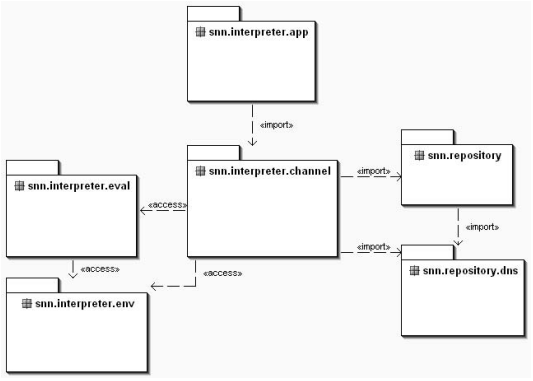

Figure 4. Translator Implementation

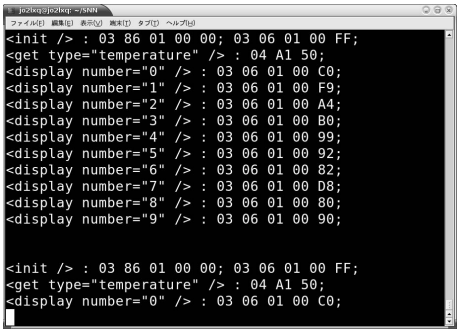

Figure 5. Application Command Translation

4.3 Script Resolution Service

Suppose that two device nodes, "node01.sample.org" and "node02.sample.org", are deployed on the Internet, and that the script of these nodes are managed under "csr.sample.org". We configure the zone file of BIND9[4] as followed:

```
node01 IN NAPTR 10 0 "S" "CSR+TCP" "" _csr._tcp.sample.org.
node02 IN NAPTR 10 0 "S" "CSR+TCP" "" _csr._tcp.sample.org.

_csr._tcp IN SRV 10 0 3072 csr.sample.org.
csr IN A 203.178.135.25
csr IN AAAA 2001:200:0:1cd1::25
```

This configuration shows that node01 are bound to the service "_csr._tcp" which is provided at TCP3072 of csr.sample.org (203.178.135.25 or 2001:200:0:1cd1::25).

5 DISCUSSION

The proposed system delegates the authority of providing the script to device deployment organizations. However, it is not practical for them to prepare all of the script. The script should be structured and should be divided into several parts: e.g., (1) service type part, (2) logical part and (3) service location part.

The fact of script division capability implies that these divided descriptions might be able to be provided by different authorities which have specific role(s) in the whole system. For example, (1) application authority might define the service type part, (2) device manufacturer might provide the logical source code, and (3) device deployment organization might add the service location part.

6 CONCLUSION

In order to provide a generalized access method for wide variety of proprietary device nodes on the global Internet, we proposed the "Scalable Embedded Device Management System". This system consists of two sub-systems; "Configurable Plug-In Translator" and "Dynamic Management Script Resolver System".

The translator is designed to parse and evaluate the description of device node interface. It mediates the communication between application and device node, translating the data and protocol between them.

The script resolver system distributes the registerd script for translator system. By taking advantage of DNS, it achieves scalability, robustness and management capability.

ACKNOWLEGEMENT

A part of this work was based on the part of the Research and Development Program of the Ubiquitous Network Authentication and Agent(2003, 2004, 2005), of the Ministry of Internal Affairs and Communications, Japan.

References

[1] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1. Technical report, W3C, mar 2001.

[2] Eclipse uml. http://www.omonodo.com/.

[3] A. Gulbrandsen, P. Vixie, and L. Esibov. A dns rr for specifying the location of services (dns srv). Technical report, IETF, 2000.

[4] Internet Systems Consortium. *BIND9 Administrator Reference Manual*, nov 2005.

[5] M. Mealling and R. Daniel. The naming authority pointer (naptr) dns resource record. Technical report, IETF, 2000.

[6] M. Ochiai. Picnic. *Transistor Gijyutsu*, pages 249–261, jan 2001.