CCDM: Central Controller-based Device Management Architecture and Method to Split Management Scripts

Akihiro Sugiyama, Hideya Ochiai, and Hiroshi Esaki
The University of Tokyo
Graduate School of Information Science and Technology
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
sugi@hongo.wide.ad.jp

Abstract

This paper presents CCDM, a new architecture for management of devices, which collect environmental information and give effects to the environment. CCDM provides easy management of devices and distributed execution of gateways for sudden network disruptions. We separate the management system to the control plane and the data plane. CCDM's easy management is achieved by a logically centralized controller. The controller handles the control plane. All the policy of device management and network information are stored to 2 kinds of scripts; service scripts and resource scripts. The controller transforms the policy to work distributedly at each gateway. Data plane is the data communication between devices and gateways. Gateways, which operate devices directly, works at distributed manner based on the policy on the controller.

1. Introduction

More and more embedded devices, (i.e., sensors and actuators, have been connected to the Internet via gateways. Embedded devices have their own unique protocols. When one wants to get information from a sensor and operate an actuator by the information, gateways need to take a role in translating a device specific protocol to another. The role is not just in protocol translation; gateways provide data processing and routing for devices. There being many devices and gateways, distributed operation is desirable because of the durability to the network disruptions or traffic reduction. However, this distributed situation makes it difficult to provide easy management way because a user must know all the network information, device protocols and device-to-gateway relations and decide where to allocate and execute applications.

We propose Central Controller-based Device Management(CCDM) to tackle on those problems. We design the control plane as centralized and the data plane as distributed. The central controller-based architecture allows managers to operate devices through a simple interface and applications to be executed in a distributed manner. CCDM provides command set such as arithmetic operations, branch, loop and so on for system operators, which are necessary in data processing. Additional command set for device abstraction and communication between gateways is prepared, too.

We adopted the controller-based architecture by Ethane[4], OpenFlow[1] and Cleanslate 4D[2]. Ethane makes enterprise networks more manageable and more secure by a controller-based approach. A controller in Ethane has the global network policy to govern enterprise networks. OpenFlow also adopts controller-based approach. It aims to provide easy way for network researchers to run experimental protocols on campus networks. Active network [6, 7] looks alike from the perspective of programmable networks. A mobile agent[3] migrates its software and data from one computer to another autonomously and keeps its execution. Our point is in the adoption fo the controller-base architecture to device management.

Though CCDM's controller-based architecture is similar to architectures of these research, our policy description languages is completely different from them because our target is in the management of devices. The way to describe management policies in CCDM is based on [5]. The high level script language provides command set suitable for device management.

2. Controller-based System

2.1. System Requirement

We assume that various kind of devices are installed in buildings or campus networks. The management sys-

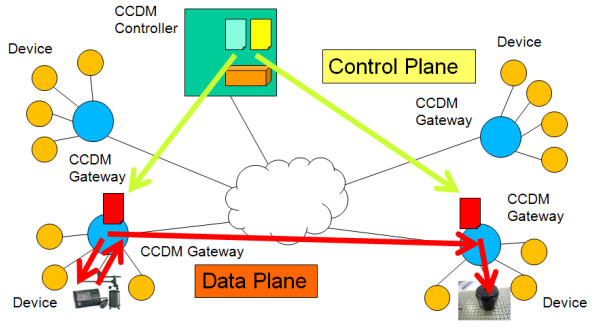

Figure 1. Overview of CCDM System

tem must satisfy the following 5 requirements. (1)protocol translation: Devices, which have different protocols from device to device, must communicate with each other. That is, protocol translation should be carried out. (2)data processing: Data aggregation, conditional branch and any data processing must be programmable. (3)communication chain: From devices to devices procedure must be done in distributed environment. (4)easy management: To enable users to deploy services without considering underlying network and device protocols. (5)fault tolerance and traffic balancing: Data plane must avoid single point of failure and traffic congestion.

2.2. CCDM Architecture

Figure 1 shows the architecture of CCDM. CCDM has 3 components; a CCDM controller, CCDM gateways and devices.

A CCDM controller is a logically-centralized server. The controller has service programs, which decide from which device data come from and how to manage it, and resource information, which is consisted of IP address of gateways, device communication protocols, device-to-gateway binding and so on. The controller has two functions. One is to generate execution scripts for each CCDM gateway from the resource information and service program. One is to deploy those execution scripts to each CCDM gateway.

CCDM gateways are the gateways of each device. All devices are bound to gateways. It is not a controller but a gateway that operates devices in CCDM. Gateways work under the direction of the controller with execution scripts. Execution scripts shows the operation policy of each gateway. They derive from service programs and resource information stored at the controller. After deployment of execution scripts, a gateway gets data from sensors, calculate it and give it to actuators.

Devices are set of sensors and actuators. They only throw their data to a gateway and receive data from it according to their own protocols.

```
1  <progn>
2    <chunk>
3      <INPUT from="WeatherSensor"/>
4      <post channel="A" name="INPUT"/>
5    </chunk>
6
7    <chunk>
8      <catch channel="A" name="temperature"/>
9      <if>
10       <lt>
11         <getq name="temperature"/>
12         <int>20</int>
13       </lt>
14       <progn>
15         <OUTPUT to="Buzzer">
16           <ARG>
17             <byte>1</byte>
18           </ARG>
19         </OUTPUT>
20       </progn>
21     </if>
22   </chunk>
23 </progn>
```

Figure 2. Sample Service Script

2.3. Policy Description Languages

CCDM defines 3 scripts; service scripts, resource scripts and execution scripts. All the scripts are in the form of XML languages, but they have different functions.

Service scripts are the programming language of CCDM. A user writes an application to a service script. Applications in CCDM include alert notification, data delivery, data aggregation, and so on. Command set of service scripts are arithmetic operations, logical operations, branch, loop, abstract device communication, etc. The command set provide the appropriate way for users to manage devices.

Resource scripts store the information of the network; IP address of gateways, device protocols, etc. Resource scripts are defined to complement service scripts in compilation of them. It is because service scripts cover up the underlying network information, device specific communication protocols and device-to-gateway bindings: the service scripts need to complement these detailed information.

Execution scripts are created by the compilation of service scripts and resource scripts. Execution scripts are the instruction for CCDM gateways. An execution script works at each CCDM gateway.

2.4. Programming by Service Scripts

Consider, for example, Bob wants to sound a buzzer when the room temperature becomes above 20 degree. Bob writes the program to a service script. Figure 2 shows the example. The program is divided into 2 parts by the chunk commands in the line 2 and 7. Generally a chunk can contain stateful data processing. The program gets data from a WeatherSensor by INPUT command in the line 4. IN-

```
1  <host name="CCDM_A">
2    <ip>192.168.1.2</ip>
3    <port>
4      <begin>10000</begin>
5      <end>20000</end>
6    </port>
7    <devicelist>
8      <device name="WeatherSensor" func="INPUT">
9        <progn>
10          ........  [Device Specific Protocol]
11       </progn>
12     </device>
13   </devicelist>
14 </host>
```

Figure 3. Sample Resource Script

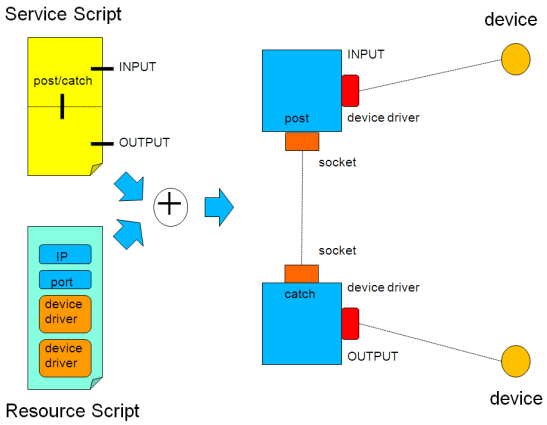

Figure 4. Mechanism of CCDM Compilation

PUT/OUTPUT commands provide generalized device communication API. In the line 4, the data is sent to the channel named A by post command. Post/catch commands enable a program to exchange data between chunks. A channels is the communication channel between chunks.

The other chunk receives the data from the first chunk by channel A in the line 8. It checks whether the data is above 20 or not in the line 9-20. If the data is over 20, the buzzer makes a sound by the OUTPUT command in the line 17. If not, it does nothing.

2.5. CCDM Compilation of Service Script and Resource Script

Figure 4 shows the mechanism of the compilation of scripts.

A service script is divided into several parts. The unit of the partition is a chunk. The controller allocates each chunk to gateway. The allocation mechanism is based on device-to-gateway bindings. Because a device is always bound to a gateway, a chunk with a device can be bound to the gateway which is bound to the device. A chunk with no relation to any devices can be allocated to any gateways.

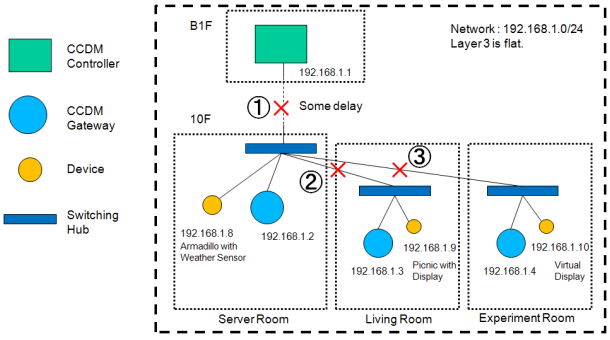

Figure 5. Experimental Network

INPUT/OUTPUT need to be replaced because they only represent the generalized device interface. They are replaced to the device specific protocols described at the resource script at Figure 3.

Post/catch are replaced to UDP datagram socket. The destination IP address and port are decided by their channel and resource information.

3. Experiment and Analysis

3.1. Experimental Network Environment

We implemented CCDM prototype. To test and evaluate the prototype, we built the experimental network Figure 5 . It is based on the real network at the Bld 2 of Engineering Department, The University of Tokyo.

We prepared 7 test case applications. The applications are (1)Weather Display, (2)Multicast Data Delivery, (3)Weather Threshold Notification, (4)Data Aggregation, (5)Several Source Data Aggregation, (6)Largest Value Update Notification and (7)Updated Data Delivery.

3.2. Network Management Cost

Figure 6 shows the code size of the test case applications. The summation of the code size of 7 applications with CCDM system is 1074. On the other hand, one without CCDM system is 4539. The code size becomes 23.7% with CCDM system. It shows that programming with CCDM make it easy to deploy new application.

3.3. Durability to Network Failures

We assumed that 3 points of the network at Figure 5 were disconnected suddenly.

Table 1 shows the continuity of the Weather Display Service when each 3 point of Figure 5 was disconnected. Even

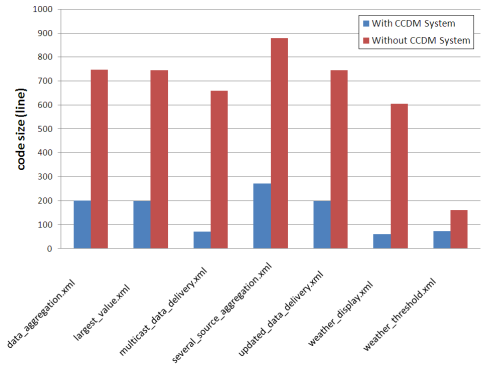

Figure 6. Comparison of Code Size

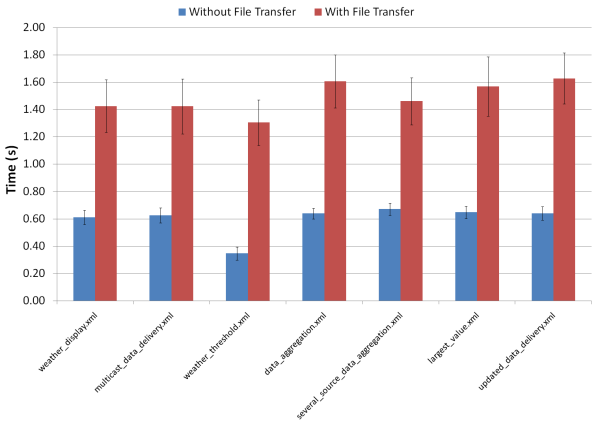

Figure 7. Time of Compilation and Deployment of Service Scripts

Table 1. continuity of the service in disruption

	①	②	③
With CCDM System	○	×	○
Without CCDM System	×	×	○

if gateways lost their connectivity to the controller, they remain working. It is because the control plane and the data plane is separated at CCDM system.

3.4. Time of Compilation and Deployment of Service Scripts

Figure 7 shows time to generate a execution script from the compilation of service scripts and resource scripts. This result shows that the controller compiles scripts and send them to gateways within 2 seconds. We think that 2 seconds waiting time is not so long for new application deployment.

4. Conclusion

We implemented the CCDM prototype and evaluated it at the experimental network. We confirmed users can program it on a single script with generalized device access interface. Programming with CCDM reduced the code size by 76.3 percent in our test case applications. It is because users program applications without knowing device specific protocols and underlying network information; they are stored at resource scripts. Contrary to the programming with a centralized manner, CCDM enables the distributed execution of applications. Our prototype deployment of CCDM showed that the centralized controller is not the single point of failures because of the separation of the control plane and the data plane. We believe that CCDM's controller-based architecture and the script languages are suitable for the device management.

References

[1] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking Control of the Enterprise. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 1–12, Kyoto, Japan, 2007. ACM.

[2] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A Clean Slate 4D Approach to Network Control and Management. *ACM SIGCOMM Computer Communication Review*, 35(5):41–54, 2005.

[3] D. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput Commun. Rev*, 38(2):69–74, 2008.

[5] H. Ochiai and H. Esaki. Networked GPIO Control by High-Level Languages and Protocol Translator. *IPSJ Journal*, 49(10):3451–3461, 10 2008.

[6] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A Survey of Active Network Research. *Communications Magazine IEEE*, 35(1):80–86, 1997.

[7] D. Wetherall and D. Tennenhouse. Towards an Active Network Architecture. *Computer Communication Review*, 26(2):5–18, 1996.