ScriptIoT: A Script Framework for and Internet-of-Things Applications

Han-Chuan Hsieh, Kai-Di Chang, *Member, IEEE*, Ling-Feng Wang,
Jiann-Liang Chen, *Senior Member, IEEE*, and Han-Chieh Chao, *Senior Member, IEEE*

***Abstract*—Following recent advances in sensing and wireless technologies, Internet-of-Things (IoT) applications are being exploited in various fields. The scale of IoT systems and the number of devices that they include has become huge, and the construction of IoT applications is, therefore, becoming increasingly challenging. This work proposes a script framework as a convenient development interface for service-oriented architecture (SOA) scheduling of web-based information of IoT applications, called ScriptIoT, which is composed of the IoT fundamental in case of all type of devices integration and a scriptable agent. Based on the IoT fundamental class, various IoT devices may be developed and the scriptable agent enables IoT applications to be configured using scripts. The proposed ScriptIoT framework, which offers both polling an event-driven mechanism for delegating IoT applications to the agent and reporting event of the specified device, contributes to large-scale applications. Experiments herein reveal that in the proposed ScriptIoT framework, the access time and CPU loading are slightly greater than those achieved using traditional C programming by 3% and 13%, respectively, but the proposed framework exhibits improved flexibility and scalability.**

***Index Terms*—Event-driven mechanism, Internet of Things (IoT), script framework.**

I. Introduction

In RECENT years, substantial progress has been made in sensing and wireless technologies, and smart mobile devices have become increasingly popular; these trends driven the flourishing of the Internet of Things (IoT) industry [1]. Research concept of IoT in providing ubiquitous framework in sensing the environment basically has to be integrated seamlessly into human daily life. Therefore, the IoT R&D cannot be distinguished from the perspective of space and spatial area of living environment. City becomes a major target for IoT R&D implementation; the smart city projects have been initiated sporadically and involved many sectors including the industry. IBM smarter planet [2] is the one prominent effort by industry to enable comprehensive framework for IoT technologies. Since IoT has to deal for object identification, unique identity (UID) management becomes an important role for guaranteeing the efficiency of IoT system. A proposed scheme uses distributed hash table (DHT) to improve the lookup scheme for improving UID management system [3]. This paper deals the problem with incorporating wireless sensor and actuator network on the IoT web-based architecture. The basic solution for this architecture enables UPnP protocol at the gateway; however, the method will cause a bottleneck due to the protocol translation between ZigBee and UPnP. Therefore, a solution using constrained application protocol (CoAP) is proposed, which reduces the congestion at the gateway with satisfactory performance [4]. A challenge of IoT service composition is the difficulty in extending SOA. A special approach has to be addressed for huge number of services including user-centric and situation-aware process. Dealing with such problem, this paper proposes a new methodology to enable web service in very large-scale (VLS) IoT system. The VLS system elaborates the design of the proposed composition that separately defines the choreography and orchestration modules (COMs) [5].

A large number of sensors in the IoT field are used exclusively for a special function network. Regardless of the applications, the challenge that must be met by an IoT system concerns the diversity of physical objects/devices on which the development and maintenance of the system depend, and it is difficult to unify context of the things. Moreover, implementing a smart web interface for that case is also not a trivial problem [6]. Hence, an IoT must satisfy the following requirements: 1) it must have a unified API that must be used in the development of applications; 2) it must be easily ported among platforms; and 3) IoT systems must be easy to configure [7]. Recently, a growing research topic on mobile agent-based for data collection in IoT networks [8], based on the above requirements, this work designs a script-based framework, called ScriptIoT, which is referred to herein as a form of IoT middleware. This middleware allows users with little or no programming expertise to develop IoT applications with minimum effort. Following improvements in hardware performance over the last few years, the scripting language that is utilized herein enables complex tasks to be carried out in relatively few steps. Herein, the framework that is based on script leads to a 13% greater CPU loading, consumes 3% more time, and consumes 17% more memory than that which uses the C code program, and it could simplify the development and maintenance process. This framework proposes the IoT application and contributes to large-scale logistic network applications.

This paper is organized as follows. Section I introduces to brief the proposed framework, the previous solutions, the highlighted problems, and a summary of the contributions.

Manuscript received April 15, 2015; revised July 12, 2015; accepted September 10, 2015. Date of publication September 28, 2015; date of current version July 27, 2016.

H.-C. Hsieh, K.-D. Chang, L.-F. Wang, and J.-L. Chen are with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: lchen@mail.ntust.edu.tw).

H.-C. Chao is with the Department of Computer Science and Information Engineering, National Ilan University, I-Lan 260, Taiwan (e-mail: hcc@niu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JIOT.2015.2483023

Section II then describes the background of IoT and script language for the proposed framework. Section III presents the proposed script framework that copes with the elements of IoT architecture. Section IV presents a simulation of the proposed ScriptIoT and verifies the feasibility and performance. Moreover, Section V proposes the framework for performance analysis in processing time and the C code program. Conclusion is finally drawn in Section VI, along with recommendations for future research.

II. Background

The industry initiative in IoT research would expect a realization of commercial product that instantly impact human life. As a commercialized product, IoT is expected to deliver a new paradigm for consumer electronic. IoT applications definition can be assumed as unique objects that connected through internet to perform information exchanging, object identification, location updating, and security monitoring.

A. IoT

IoT technology is being implemented broadly for information technology and industry applications. Recently, top 10 IoT commercial products are announced to give an insight on how the consumer electronics trend is now approached into the new era. Some of the featured IoT products are as follows.

Pachube: A platform that bridges the application and data to be worked together to convey useful information to the users. User can use real-time sensing data provided by Pachube to create a connection to a particular application through web service [9].

Mirror: It is a commercial product from Violet, a French company, providing a smart detection of a particular object. The technology is merely developed from radio frequency identification (RFID) tag that collaborated with smart web application [10].

Those functions are enabled by the integration of management systems that essentially comprised RFID, wireless sensor networks, and global positioning systems (GPSs). IoT architecture is broadly divided into three categories: 1) sensor networks architecture; 2) middleware architecture; and 3) application-based service-oriented architecture (SOA) [11]. These categories are described below with reference to corresponding script language.

1) Sensor Networks Architecture: This category of architecture is focused on the integration of perception and network layers. For example, electronic product code (EPC) is based on RFID as an integrated ZigBee network architecture [12]. The proposed All-IP world (SNAIL) sensor networks in combination of the heterogeneous networks approach has been realized for IoT architecture [13].

2) Middleware Architecture: Much research concerning middleware architecture is based on widely popular technology [14]–[16], such as the VIRTUS, which is based on XMPP technology [17], whose applications can be expanded using Google-developed tools and API. SOA, established by OSGI, is based on Java VM [18]. IoT cloud architecture, based on the combination of Java and some frequently used open source software, has also been proposed for running cloud applications [19]. A web application framework for IoT that relies on the Google Web Toolkit is proposed [20]. This plug-in-based framework is visualized and controlled using an extensible user interface, but has a high development threshold, and its performance, including code size, has not been analyzed. The interesting perspective on the management of method for managing resource-constrained IoT devices management is proposed [21]; it involves the use of SNMP and NETCONF to manage a specific hardware platform, saving RAM and ROM but at the cost of lost flexibility and scalability.

3) Application-Based SOA: Web architecture is based mainly on passive requests and cannot handle responses from logistic networks in real time. Service-oriented research focuses on optimizing message scheduling or the realization of an event-driven mechanism [22]. The IoT network system can be divided into various subsystems, forming a hierarchical system structure. The concept of SOA can be applied to the scheduling of web-based information to calculate the shortest processing time and provide effective and stable real-time responses. Some investigations based on event-driven SOA (e-SOA) mechanism have involved dynamic sensing and the event response times of various connections proposed in framework monitoring [23].

B. Script Language

Script language is a computer language whose main purpose is to shorten the check of composition, compilation, connection, and execution. Command codes are generally directly executed instead of compiled. The programming languages are used to compose programs for computers. The important purpose of a descriptive language is to accomplish certain complicated tasks simply and rapidly. Accordingly, a description language is usually simpler than a conventional programming language, such as C, C++, and Java language. For example, the Bash Shell, which is the most frequent among Unix-like systems, has been widely implemented in various platforms such as GNU/Linux, Mac OS, MS-DOS, and Windows, most of which are downward-compatible with the older Bourne Shell.

III. Proposed Script Framework

Fig. 1 presents the proposed IoT service architecture, which is composed of four parts: 1) devices; 2) agent servers; 3) agent clients; and 4) hosts as applications. The hosts can use the agent clients to directly access the agent servers or use the script to completely define all interactive behaviors of the entire group for smart applications. The hosts can communicate with the agent servers through the agent clients to receive feedback from devices or to drive devices to take predetermined actions and responses. The agent server also provides another interface that is connected to the agent client, and can be regarded as an API that can be accessed by the host.

Refers to various IoT devices, such as environmental sensors, GPS receivers, RFID readers, and others, which are connected

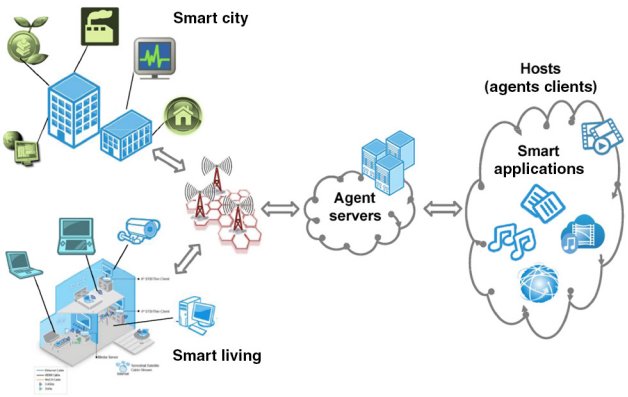


Fig. 1. IoT service architecture.

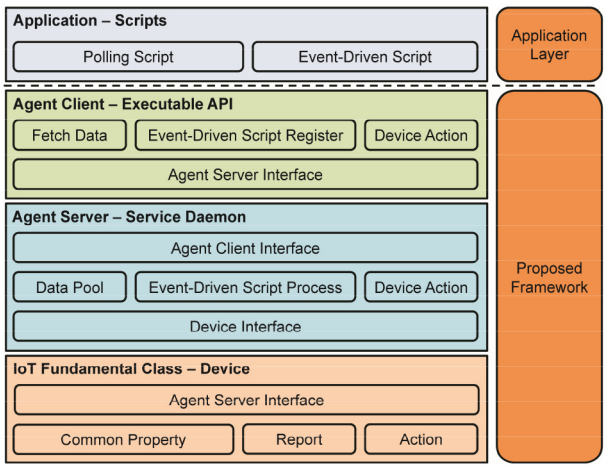


Fig. 2. IoT system architecture.

to Agent Server through the network, the proxy Agent Server of Host group, managing all data feedback from devices and controls mutual correlation behaviors. The script can be easily used to describe the check behaviors of the devices in the entire group. For example, when the RFID reader recognizes a TAG ID, it will send out the command to open a door, or when a particular device moves to a particular position, a particular action will be triggered. Fig. 2 presents the proposed system architecture, whose four parts are as follows.

A. Device

To integrate all IoT devices, the so-called IoT fundamental class is presented in Fig. 3. This class specifies the minimum necessary support function. Any IoT device can be expanded for requirement. All IoT devices that are supported by the architecture in this work will be implemented in compliance with this IoT fundamental class, such that they can be registered and report to the agent server.

This class can be divided into two major parts: 1) common and 2) override/virtual. The common part defines the properties and configuration of the corresponding IoT devices, and these basic properties must be set for all IoT device categories. The override part defines the active and passive types, which

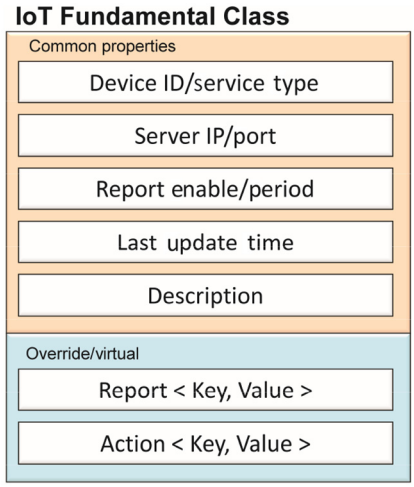


Fig. 3. IoT fundamental class.

must be implemented by various IoT devices. In this work, the simplest <Key, Value> pair is utilized to process all feedback and saved data. Therefore, data processing with the <Key, Value> pair is implemented using the report and the action function. The consistent use of <Key, Value> pair significantly simplifies the processing mechanism during actual implementation, maximizing the expansion flexibility. Access of the <Key, Value> pair by hash mapping during the implementation of data pool also leads to rather high efficiency.

Definition of data structure: Two active and passive types function pointers are defined for the override of the two device categories. A unique function pointer is assigned to each device category, and devices are extended based on the principle of objective orientation for IoT device fundamental class/function override.

B. Agent Server and Agent Client

As the proxy of host in the group, the agent server must implement two interfaces: 1) the IoT device interface and 2) the agent client interface, which are the communication interfaces for device and host, respectively. A sufficiently large data pool in the agent server to save the report <Key, Value> pair constantly feedback from the group is required. The implementation of this data pool is based on asynchronous access to distinguish between the device report write-in and agent client fetch read-out, resulting in more efficient agent client application without the need to wait for the report write-in. Since the report data are <Key, Value> pairs, the hash table method can be used to accelerate the access. The following functions are completed by coordination between the agent server and the agent client.

1) Register Event-Driven Script: The purpose of event-driven script registration module is to enable the host to activate a specified script by agent client registration once the designated <Key, Value> has been sent out by the specific device with agent client registration. Fig. 4 presents the saving of the event-driven script by the agent server and its execution when the conditions associated with the report command are met, and

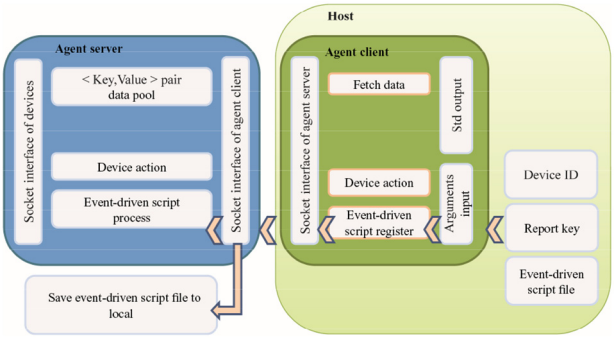


Fig. 4. Event-driven script register flowchart.

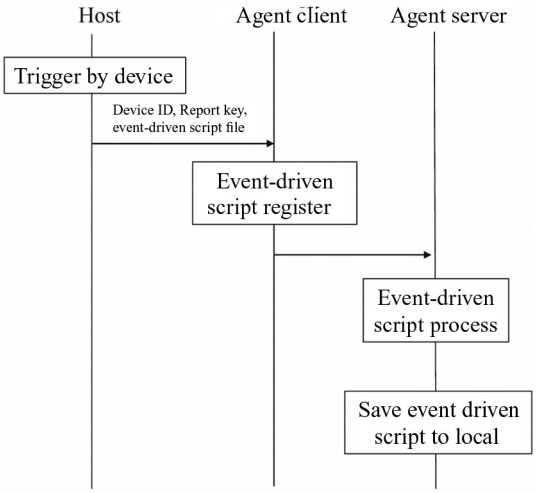


Fig. 5. Signaling flow for event-driven script register.

signaling flow for event-driven script register are presented in Fig. 5.

2) Fetch Data: Fig. 6 presents the arrival at the agent server of the <Key, Value> pair that is reported by the device and signaling flow for fetching data (Fig. 7). The pair is saved to the data pool, and whether the <Key, Value> of this device matches the registered report command, as indicated by arrow ① in the figure, will be determined. If it is registered on agent server, it will call upon and execute the designated script, as indicated by arrow ② in the figure. The executed script, just like the host, is composed of the API of the agent client. It differs from the host only in that the host IP address, introduced by the agent client of the script, generally refers to the IP address of its agent server. Hence, the agent server must support the API of the agent client in addition to its own software function. The agent server IP address that is designated in the event-driven script is not necessarily its actual IP address: it may be the IP address of another agent server. Various groups can be crossed by coordinating several agent servers by exploiting such flexibility.

The host of Agent Server IP, communication port, and a device ID in the group using its Agent Client API can be set to obtain the <Key, Value> that is reported by that devices, and obtaining the reported value that corresponds to the <Key, Value> using the Fetch Command. In Fig. 6, arrows ③ and ④ indicate the data path.

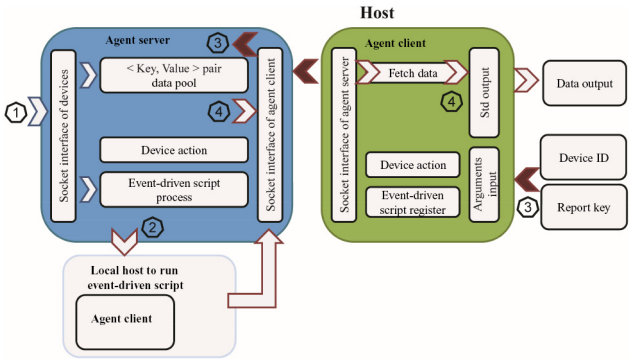


Fig. 6. Fetch data flowchart.

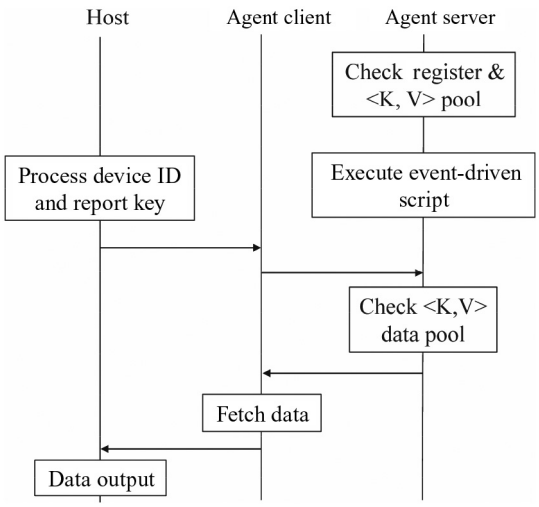


Fig. 7. Signaling flow for fetching data.

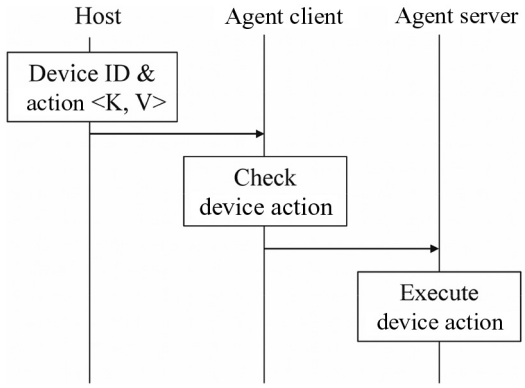


Fig. 8. Signaling flow for device action.

3) Device Action: The host can use the agent client to drive directly the agent server to perform the designated action <Key, Value>, and signaling flow for device action presented in Fig. 8. Table I presents the API of agent client. The host can communicate and cooperate with the agent server using the parameters.

C. Host

The application program on the host is in the form of script and the agent client must be used as an API to access the

TABLE I
AGENT CLIENT API

Agent client	Parameter	Note
Argument 1	Agent Server IP	IP address of the agent server.
Argument 2	Port	Port number of the agent server; default is 5001.
Argument 3	Command	FETCH: Get specific data from agent server. REGISTER: Register the event-driven script. ACTION: Directly drive the device by specified action.
Argument 4	ID	ID = IDValue: Define ID for the device.
Argument 5	REPORT	REPORT = KEY: Command = FETCH: Return the value that corresponds to the key. Command = REGISTER: Trigger the corresponding script when the agent server receives the <Key, Value> from device. ACTION = deviceAction: Command = Action: This parameter specifies the contents of the drive.
Argument 6	ACTION	SCRIPT = ScriptFilePath: Command = REGISTER: This parameter specifies the path of the event-driven script. CLEAR = [True | False]: Command = FETCH: This parameter indicates whether the <Key, Value> record in the agent server should be cleaned when received.

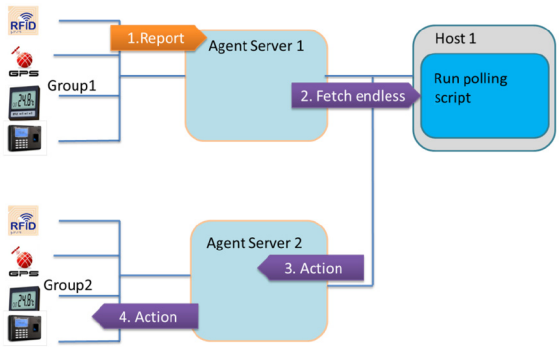


Fig. 9. Polling script block diagram flowchart.

agent server to compose the script. Scripts of the host may be polling scripts or event-driven scripts on immediacy and initiative. When the host is not directly connected to the device, it relies on polling of the agent client to determine all statuses in the group, as presented in Fig. 9, and the signaling flow for fetching data is presented in Fig. 10. This characteristic greatly reduces the immediacy and initiative of the device in the group. This work proposes another approach, called "event-driven script register." The host can register with the agent service to cause it to execute particular scripts upon receiving particular <Key, Value> that are reported by devices. The agent client must also be installed in the agent server hardware to operate as the API to execute scripts on the agent server.

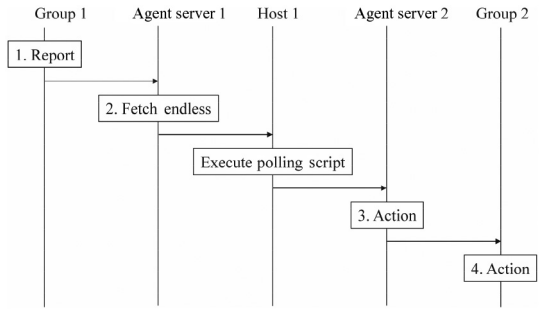


Fig. 10. Flow for polling script.

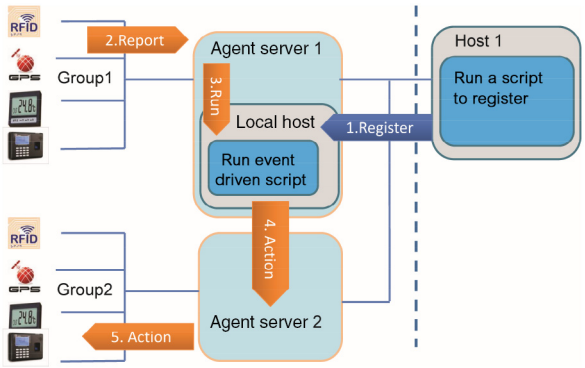


Fig. 11. Event script block diagram.

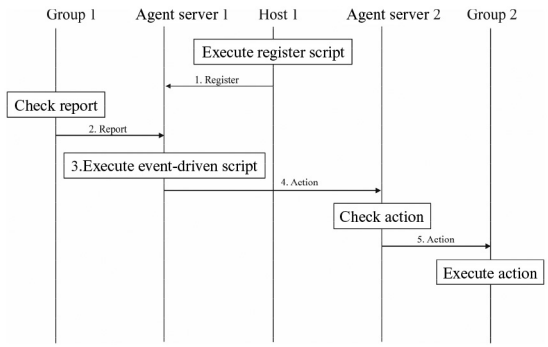


Fig. 12. Flow for event script.

A simple change to the agent server IP easily supports cross-group control, as presented in Fig. 11, and flow for Event Script is presented in Fig. 12.

IV. SYSTEM APPLICATIONS

The infrastructure and scripts for proposed simulations will complete in this section. The registration on Host is the script that runs pooling, and assigns two ports on Agent server to verify the RFID TAG values and then trigger Event-Driven script to issue door events. It is difficult for RFID TAG access control to simulate the proposed ScriptIoT and verify the feasibility.

A. Infrastructure

Two laptops are used for simulation, as presented in Fig. 13. Laptop A on the left is used to simulate the RFID reader and

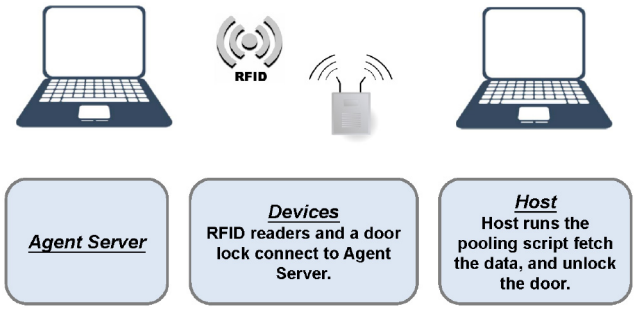


Fig. 13. Event script block diagram.

```
#!/bin/bash

AGENT_SERVER_IP=192.168.0.100
AGENT_PORT=5001

while [ REG_RESULT != "true" ]
    do
        TAG=`./AgentClient $AGENT_SERVER_IP  $AGENT_PORT FETCH ID=RFID1
              REPORT=TAG CLEAR=TRUE`

        if [ $TAG == FFFF8888CCCC0000 ] || [ $TAG == FFFF8888CCCC0005 ]; then
            OPEN_RESULT=`./AgentClient $AGENT_SERVER_IP $AGENT_PORT ACTION
                         ID=DOOR1 ACTION=OPEN`
        fi
        sleep 1
        echo "polling"
    done
```

Fig. 14. Example of host polling script.

```
#!/bin/bash
AGENT_SERVER_IP="127.0.0.1"
AGENT_PORT="5000"

./Device $AGENT_SERVER_IP  $AGENT_PORT RFID1 RFID 2 'Front Door RFID' &
./Device $AGENT_SERVER_IP  $AGENT_PORT DOOR1 DOOR 0 'Front Door' &
```

Fig. 15. Device simulation script.

the lock on a door, which is connecting to the agent service on that laptop. Laptop B on the right is used to simulate the host.

B. Polling Script

This case is an actual access simulation system. The host continuously polls the TAG values that are scanned by the RFID reader and makes judgments. If the received expected, then the code unlock the door through the agent client.

1) Host-Side: Host-side refers to the script template on the host, presented in Fig. 14.

2) Device-Side:

1) The RFID is used to simulate as report function, allowing the reader continuously to report TAG values between "FFFF8888CCCC0000" and "FFFF8888CCCC0007."
2) Simulate unlocking of the door. The action "OPEN" is supported, and a beep sound indicates that the door has been unlocked.

```
#!/bin/bash

AGENT_SERVER_IP=127.0.0.1
AGENT_PORT=5001
TAG=$2
echo "key=$1 keyvalue=$TAG"
if [ $TAG == FFFF8888CCCC0000 ] || [ $TAG == FFFF8888CCCC0005 ]; then
            OPEN_RESULT=`./AgentClient $AGENT_SERVER_IP $AGENT_PORT ACTION
                         ID=DOOR1 ACTION=OPEN`
fi
```

Fig. 16. Example of event-driven script.

```
vincent@vincent-nb: ~/work/IOTAgent
Report Handle=1, Key=TAG, Keyvalue=FFFF8888CCCC0004
RUN ././RFID1.TAG.sh TAG FFFF8888CCCC0004 &
key=TAG keyvalue=FFFF8888CCCC0004
Report Handle=1, Key=TAG, Keyvalue=FFFF8888CCCC0005
RUN ././RFID1.TAG.sh TAG FFFF8888CCCC0005 &
key=TAG keyvalue=FFFF8888CCCC0005
len=34
cmd0=st,cmd1=ACTION
ACTION,ID=DOOR1,ACTION=OPEN
Send AgentClient Action feedback id=DOOR1,seq=0
Report Handle=1, Key=TAG, Keyvalue=FFFF8888CCCC0006
RUN ././RFID1.TAG.sh TAG FFFF8888CCCC0006 &
key=TAG keyvalue=FFFF8888CCCC0006
Report Handle=1, Key=TAG, Keyvalue=FFFF8888CCCC0007
RUN ././RFID1.TAG.sh TAG FFFF8888CCCC0007 &
key=TAG keyvalue=FFFF8888CCCC0007
Report Handle=1, Key=TAG, Keyvalue=FFFF8888CCCC0000
RUN ././RFID1.TAG.sh TAG FFFF8888CCCC0000 &
key=TAG keyvalue=FFFF8888CCCC0000
len=34
cmd0=st,cmd1=ACTION
ACTION,ID=DOOR1,ACTION=OPEN
Send AgentClient Action feedback id=DOOR1,seq=1
```

Fig. 17. Event-driven script–agent server running screen capture.

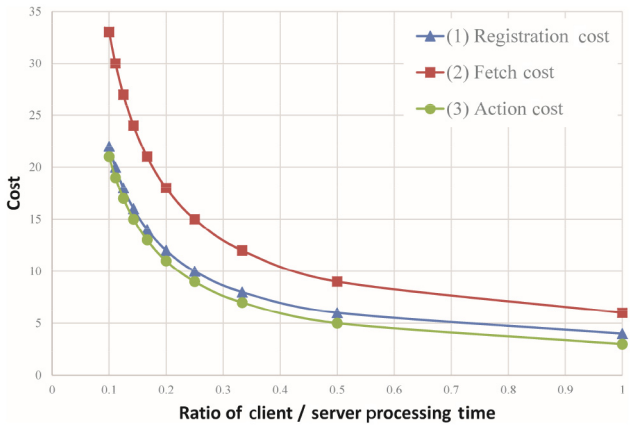


Fig. 18. Cost analysis result of ScriptIoT.

TABLE II
COST PARAMETERS

Parameter	Note
Ps	Agent server/local processing time
Pc	Agent client/host processing time
α	Transmission delay

3) Simulate the script of device, as presented in Fig. 15. RFID reader and door device are simulated, and the devices and some literal descriptions can be set according to the parameters below.

3) Agent Server-Side: The agent server can be activated by simply assigning to two ports of the interface: 1) Port-5000 for all devices and 2) Port-5001 to serve the agent client. The agent

TABLE III
CASE 1 SUMMARY TABLE

	50 RFID readers Host fetch 5000 times -1		50 RFID readers Host fetch 5000 times -2		50 RFID readers Host fetch 5000 times -3		50 RFID readers Host Fetch 5000 times -4	
Ccode /Script	C Code	Script	C Code	Script	C Code	Script	C Code	Script
Time elapse -1 (s)	50	52	49	51	53	54	49	50
Time elapse -2 (s)	49	51	50	51	48	49	50	52
Time elapse -3 (s)	48	49	49	51	52	53	49	50
CPU loading -1 *	1.31%	1.47%	1.39%	1.59%	1.48%	1.69%	1.50%	1.68%
CPU loading -2 *	1.39%	1.60%	1.24%	1.50%	1.49%	1.65%	1.48%	1.66%
CPU loading -3 *	1.50%	1.69%	1.43%	1.62%	1.38%	1.54%	1.46%	1.65%
Memory usage -1 **	576 000	1 460 000	572 000	1 460 000	576 000	1 460 000	560 000	1 460 000
Memory usage -2 **	568 000	1 542 000	576 000	1 460 000	564 000	1 460 000	576 000	1 460 000
Memory usage -3 **	572 000	1 460 000	576 000	1 460 000	576 000	1 460 000	568 000	1 460 000
Average time elapse(s)	49.00	50.67	49.33	51.00	51.00	52.00	49.33	50.67
Avearge CPU loading	1.40%	1.59%	1.35%	1.57%	1.45%	1.63%	1.48%	1.66%
Average memory usage	572 000	1 487 333	574 667	1 460 000	572 000	1 460 000	568 000	1 460 000

server indicates that two devices are connected and that it has begun receiving TAG values.

C. Event-Driven Script

This case is identical to except that the polling script of the host is replaced with the event-driven script. The task of the host is just completed following the registration of the script, such that it does not consume anymore operating resources of the host. When the received TAG is one of the preset values "FFFF8888CCCC0000" and "FFFF8888CCCC0005," the event-driven script is triggered to lock the door. As presented in Fig. 16, the <Key, Value> are proposed by the agent server to trigger the event. In Fig. 17, once the devices have alarmed TAG preset value is FFFF8888CCCC0000 or FFFF8888CCCC0005, the agent server will trigger the event-driven script, which will send commands to the agent server through the agent clients to unlock the door.

V. PERFORMANCE ANALYSIS

The architecture that is proposed in this work makes some sacrifices to preserve considerable flexibility and expandability. This section assesses the performance in processing time as cost of the ScriptIoT Framework and the C code program that are herein.

A. Cost Analysis

The cost analysis of the proposed ScriptIoT framework is composed of each event-driven script register, fetch data, and the device action. For detailed signaling flow has been shown in the previous section. The parameters used in cost analysis are the following equations:

$$2(Ps + Pc) + 2\alpha. \tag{1}$$

The cost of fetch data is

$$3(Ps + Pc) + 4\alpha. \tag{2}$$

The cost of device action is

$$(2Pc + Ps) + 2\alpha. \tag{3}$$

The computing power or resource of agent server and agent client are different from the cost analysis in Fig. 18, which shows different ratios of computing power of client and server. Table II presents the cost parameters. The Ps is assumed with value 1, which means maximum computing power. The parameter α is neglected in this analysis due to very little time. Also, the range of Pc is changing from 1 to 10, which means the computing power are from the 100% same with Ps to lowest 10% Ps.

The results show that the higher Ps and Pc ratio that implies the agent clients computing power is the same as the agent server, which would lower the cost and vice versa. Moreover, the cost overhead exists when the computing power of client is almost the same with server. The impact of overhead presents in the implementation section by comparing different codes.

B. Code Performance Analysis

To be assessed, the program code must meet at least the following three criteria.

1) Bulk access to the agent server is required to determine the difference between access by script through the agent client and direct access by the C code program.
2) The script and the C code must be used to simulate overhead associated with the same functions, other than accessing the agent server. In this case, the overhead of calculating the time difference is considered, and C code and script are used to implement this function.
3) The C code program is obtained by slightly modifying the source code of the Agent Client to reduce the difference between both sides in implementation.

Then, an experiment with the following steps is performed:

1) the agent server is activated to simulate 50 RFID readers, each reporting one set of TAG values per second;
2) the C code program and script fetch agent server are activated simultaneously 5000 times;

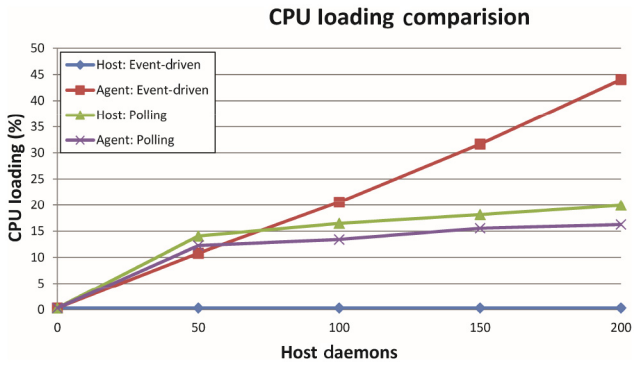


Fig. 19. CPU loading comparison.

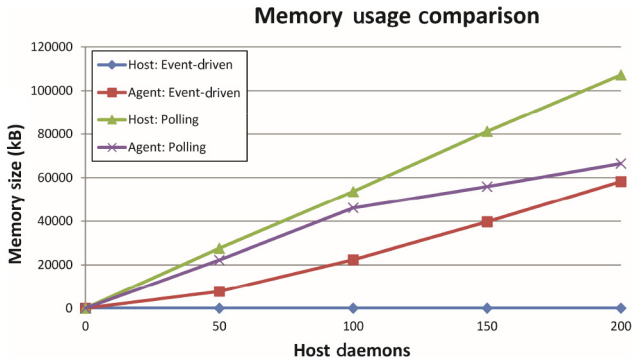


Fig. 20. Memory usage comparison.

3) the performance and resources of the systems are monitored ten times, once every 1 s;
4) when the number of activations have reached 5000 times, the time difference between the C code program and script is calculated.

The performance statistics are obtained as described in Table III with mean time consumption, average CPU loading, and memory consumption, and the comparison with host and agent server, as presented in Figs. 19 and 20. The agent client uses approximately 388 kB, while the environment of the Bash shell will require approximately 1184 kB.

The average research statistics have indicated the following.

1) The memory consumption by ScriptIoT, which is the memory used for the total capacity of its calling upon agent client, is 1 854 833.
2) The memory capacity used by ScriptIoT can be obtained from the basic overhead of the Bash shell, as 670 833.
3) Memory usage rate is $670\ 833/571\ 667 = 1.17$.

Herein, the framework that is based on script leads to a 13% greater CPU loading, consumes 3% more time, and consumes 17% more memory than that which uses the C code program.

VI. Conclusion

This paper has proposed an agent that can use descriptive language to establish logistic network application architecture for diverse logistic network devices on a huge scale. The universal definition of IoT fundamental class has been used to achieve the group-based delegation of IoT devices to the agent server. The host side can use the well-known shell script to access the event-driven agent client of the agent server to control the configuration settings for collaboration within the entire logistic network. The script mechanism has been proposed to resolve the issue of host polling. The script that corresponds to the event of a certain device will be registered, such that the agent can process the event with immediate response. The host is not involved following the registration, so that it does not use host operating resources. This work has proven that, even though the use of script results in the consumption of 3% more time, 13% more CPU resources, and 17% more memory than C code program, it simplifies the development and maintenance process while maintaining its expandability and functionality. This framework contributes to large-scale logistic network applications.

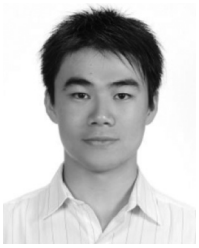
References

[1] J. Gubbia, R. Buyyab, S. Marusica, and M. Palaniswamia, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.
[2] "The Internet of Things," [Online]. Available: http://www.ibm.com/smarterplanet/us/en/overview/article/iot_video.html
[3] Q. Shen, Y. Liu, Z. Zhao, S. Ci, and H. Tang, "Distributed hash table based ID management optimization for Internet of Things," in *Proc. 6th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC '10)*, Caen, France, Jul. 2010, pp. 686–690.
[4] J. Mitsugi, S. Yonemura, H. Hada, and T. Inaba, "Bridging UPnP and ZigBee with CoAP," in *Proc. Conf. Emerg. Netw. Exp. Technol.*, Tokyo, Japan, Dec. 2011, pp. 1–8.
[5] K. Dar, A. Taherkordi, R. Rouvoy, and F. Eliassen, "Adaptable service composition for very-large-scale Internet of Things systems," in *Proc. 8th Middleware Doctoral Symp. (MDS '11)*, Lisbon, Portugal, Dec. 2011, pp. 1–2.
[6] J. He, Y. Zhang, G. Huang, and J. Cao, "A smart web service based on the Context of Things," *ACM Trans. Internet Technol.*, vol. 11, no. 3, pp. 13–22, 2012.
[7] S. Bendel, T. Springer, D. Schuste, A. Schill, R. Ackermann, and M. Ameling, "A service infrastructure for the Internet of Things based on XMPP," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, San Diego, CA, USA, Mar. 2013, pp. 385–388.
[8] M. X. Dong, K. Ota, L. T. Yang, S. Chang, H. Zhu, and Z. Y. Zhou, "Mobile agent-based energy-aware and user-centric data collection in wireless sensor networks," *Comput. Netw.*, vol. 74, pp. 58–70, 2014.
[9] "Pachube opens the Internet of Things to end users," [Online]. Available: http://www.information-age.com/industry/start-ups/1678543/pachube-opens-the-internet-of-things-to-end-users
[10] "Violet's Mirr:or: Internet of Things via RFID," [Online]. Available: http://radar.oreilly.com/2008/09/violets-mirror-internet-of-thi.html
[11] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, pp. 2787–2805, 2010.
[12] H. Hada and J. Mitsugi, "EPC-based Internet of Things architecture," in *Proc. IEEE Int. Conf. RFID-Technol. Appl.*, Sitges, Spain, Sep. 2011, pp. 527–532.
[13] S. M. Hong *et al.*, "SNAIL: An IP-based wireless sensor network approach to the Internet of Things," *IEEE Wireless Commun.*, vol. 17, no. 6, pp. 34–42, Dec. 2010.
[14] P. Mahalle, S. Babar, N. R. Prasad, and R. Prasad, "Identity management framework towards Internet of Things (IoT): Roadmap and key challenges," in *Proc. Recent Trends Netw. Secur. Appl.*, Chennai, India, Jul. 2010, vol. 89, pp. 430–439.
[15] T. Li and C. Liping, "Internet of Things: Principle, framework and application," in *Proc. Future Wireless Netw. Inform. Syst.*, 2012, vol. 144, pp. 477–482.
[16] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta, "Role of middleware for Internet of Things: A study," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 3, pp. 94–105, Aug. 2011.
[17] D. Conzon *et al.*, "The VIRTUS middleware: An XMPP based architecture for secure IoT communications," in *Proc. IEEE Int. Conf. Comput. Commun. Netw.*, Munich, Germany, Jul. 2012, pp. 1–6.

[18] M. Bazzani, D. Conzon, A. Scalera, M. A. Spirito, and C. I. Trainito, "Enabling the IoT paradigm in E-health solutions through the VIRTUS middleware," in *Proc. IEEE Int. Conf. Trust Secur. Privacy Comput. Commun.*, Liverpool, U.K., Jun. 2012, pp. 1954–1959.
[19] G. C. Fox, S. Kamburugamuve, and R. D. Hartman, "Architecture and measured characteristics of a cloud based Internet of Things," in *Proc. Int. Conf. Collab. Technol. Syst.*, Denver, CO, USA, May 2012, pp. 6–12.
[20] P. Castellani, M. Dissegna, N. Bui, and M. Zorzi, "WebIoT: A web application framework for the Internet of Things," in *Proc. Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Paris, French, Apr. 2012, pp. 202–207.
[21] A. Sehgal, V. Perelman, S. Kuryla, and J. Schonwalder, "Management of resource constrained devices in the Internet of Things," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 144–149, Dec. 2012.
[22] S. Alam, M. M. R. Chowdhury, and J. Noll, "SenaaS: An event-driven sensor virtualization approach for Internet of Things cloud," in *Proc. IEEE Int. Conf. Netw. Embedded Syst. Enterp. Appl.*, Suzhou, China, Nov. 2010, pp. 1–6.
[23] S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad, "Proposed embedded security framework for Internet of Things (IoT)," in *Proc. Int. Conf. Wireless Commun. Veh. Technol. Inf. Theory Aerosp. Electron. Syst. Technol.*, Chennai, India, Feb. 2011, pp. 1–5.

Han-Chuan Hsieh received the B.S. degree in electrical engineering from National Taipei University of Technology (NTUT), Taipei, Taiwan, in 1998, the M.S. degree in communication engineering from Tatung Institute of Technology, Taipei, Taiwan, in 2008, and is currently working toward the Ph.D. degree in electrical engineering at the National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan.

His research interests include long-term evolution-advanced, Internet of Things, software-defined networking, and network functions virtualization in 5G.



Kai-Di Chang (S'11–GSM'14–M'15) received the B.S. degree in electrical engineering from National Dong Hwa University, Hualien, Taiwan, in 2007, the Master's degree from the Institute of Computer Science and Information Engineering, National I-Lan University, I-Lan, Taiwan, and is currently working toward the Ph.D. degree in electrical engineering at the National Taiwan University of Science and Technology, Taipei, Taiwan.

He is a Researcher with United Daily News Group Co., Ltd, Taipei, Taiwan. His research interests include mobile communications, cloud computing, IP multimedia subsystem, Internet of Things, and network security.

Ling-Feng Wang received the B.S. degree in electrical engineer from the National Yunlin University of Science and Technology (NYUST), Douliu, Taiwan, in 1997, and the M.S. degree in electrical engineering and computer science from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 2013.

His research interests include IoT and embedded system applications.

Jiann-Liang Chen (SM'10) received the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1989.

Since August 1997, he has been with the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan, where he is a Professor and Vice Dean of Science and Engineering College. He is currently a Full Professor of electrical engineering with the National Taiwan University of Science and Technology. He has authored more than 150 papers in journals and conferences. He holds several patents. His research interests include cellular mobility management, digital home network, telematics applications, cloud computing, and RFID middleware design.

Prof. Chen is a U.K. BCS Fellow.

Han-Chieh Chao (S'88–M'92–SM'04) received the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989 and 1993, respectively.

He is a joint appointed Full Professor of computer science and information engineering and electronic engineering with National Ilan University, I-Lan, Taiwan (NIU). He has been serving as the President since August 2010 for NIU as well. He was the Director of the Computer Center for Ministry of Education Taiwan, Taipei, Taiwan, from September 2008 to July 2010. He has authored or coauthored 4 books and has authored about 400 refereed professional research papers. His research interests include high-speed networks, wireless networks, IPv6-based networks, digital creative arts, e-government, and digital divide. He has completed more than 100 M.S.E.E. thesis students and 4 Ph.D. students.

Dr. Chao is a Fellow of the IET (IEE). He has been invited frequently to give talks at national and international conferences and research organizations. He is the Editor-in-Chief for *IET Networks*, the *Journal of Internet Technology*, the *International Journal of Internet Protocol Technology*, and the *International Journal of Ad Hoc and Ubiquitous Computing*. He has served as the Guest Editor for *Mobile Networking and Applications* (ACM MONET), the IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS, the *IEEE Communications Magazine*, the IEEE SYSTEMS JOURNAL, *Computer Communications*, *IEE Proceedings Communication*, the *Computer Journal, Telecommunication Systems, Wireless Personal Communications*, and *Wireless Communications and Mobile Computing*.