This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

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

Han-Chuan Hsieh, Kai-Di Chang, *Student 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 contributes to large-scale logistic network applications result from offers both polling and an event-driven mechanism for delegating IoT applications to the agent with the reporting event of the specified device. 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*—Internet-of-Things (IoT), Script Framework, Event-Driven Mechanism.**

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 of prominent effort from the industry enable comprehensive framework of IoT technology implementation in the environment. Since IoT has to deal for object identification; unique identity (UID) management becomes an important role for guaranteed the efficiency of IoT system. A proposed scheme using distributed hast table (DHT) to improve the look-up scheme for improving UID management system [3]. The 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 causing a bottleneck due to the protocol translation between ZigBee and UPnP. Therefore a solution using constrained application protocol (CoAP) is proposed and reduces the congestion at the gateway with satisfactory performance [4]. A challenge of IoT service composition is the difficulties in extending SOA. A special approach has to be address for huge number of services including user-centric and situation aware process. Dealing with such problem, the paper proposes a new methodology to enable web service in very large scale (VLS) IoT system. The VLS system elaborates the design of proposed composition that separately defines the choreography and orchestration modules (COM) [5].

A large number of sensors that are used in the IoT field used exclusively form 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; (a) it must have a unified API must be used in the development of applications; (b) it must be easily ported among platforms; (c) IoT systems must be easy to configure [7]. Recently, a growing research topic on mobile agents to access data in IoT environment [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.

Manuscript received April 16, 2015; Revised Jun 22, 2015

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

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

The rest of this paper is organized as follows. Section 1 introduces to brief the proposed framework, the previous solutions, the highlighted problems, and a summary of the contributions. Section 2 then describes the background of IoT and script language for the proposed framework. Next, Section 3 presents the proposed script framework that copes with the elements of IoT architecture. Section 4 presents a simulation of the proposed ScriptIoT and verifies the feasibility and performance. Moreover, Section 5 proposes the framework for performance analysis in processing time and the C code program. Conclusions are finally drawn in Section 6, along with recommendations for future research.

II. BACKGROUND

The industry initiative in IoT research would expect a realization of commercial product that instantly impact humans life. As 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. Internet of Things (IoT)

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

Pachube: A platform that bridging 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 create a connection to a particular application through web service [9].

Mi:ror: 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 by RFID, Wireless Sensor Networks and Global Positioning Systems (GPS). IoT architecture is broadly divided into three categories - sensor networks architecture, middleware architecture, and 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]. With Internet Protocol (IP) is proposed to Sensor Networks for an All-IP world (SNAIL) protocol to approach to IoT architecture on the realization by combining the heterogeneous networks [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. Service-Oriented Architecture (SOA), established by OSGI, is based on Java VM [18]. IoT Cloud architecture, based on a 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 plugin-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 Service-Oriented Architecture (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 sub-systems, 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 Service-Oriented Architecture (e-SOA) mechanism, have involved dynamic sensing and the event response times of various connection proposed 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 frequently among Unix-like systems, has been widely implemented in various platforms such as GNU/Linux, Mac OS, MS-DOS, Windows and etc., most of which are downward-compatible with the older Bourne Shell.

III. PROPOSED SCRIPT FRAMEWORK

Figure 1 presents the proposed IoT service architecture, which is composed of four parts - Devices, Agent Servers, Agent Clients and 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 pre-determined actions and responses. The Agent Server also provides another interface that is connected to the Agent Client,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

and can be regarded as an API that can be accessed by the Host.

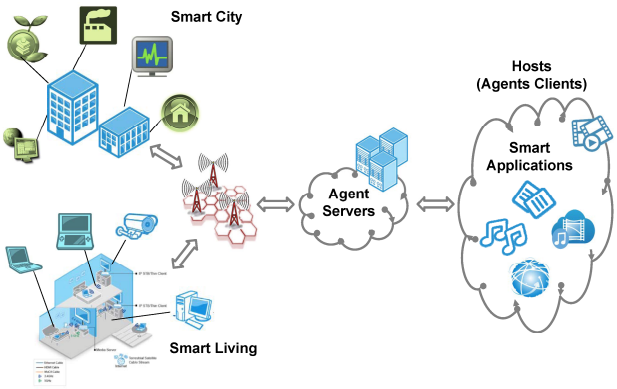


Fig. 1. IoT Service Architecture

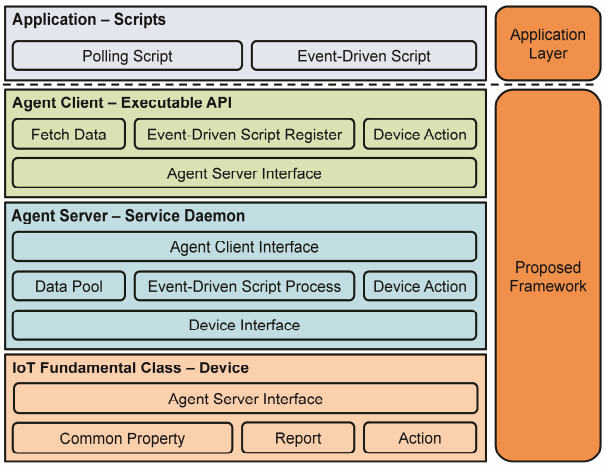


Fig. 2. IoT System Architecture

Device refers to various IoT devices, such as environmental sensors, GPS receivers, RFID readers and others, which are connected to the Agent Server through the network, which is the proxy Agent Server of Host group, is in charge of managing all data feedback from devices in the group 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. Figure 2 presents the proposed system architecture, whose four parts are as follows.

A. Device

To integrate all IoT devices, a so-called IoT fundamental class is presented in Figure 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.

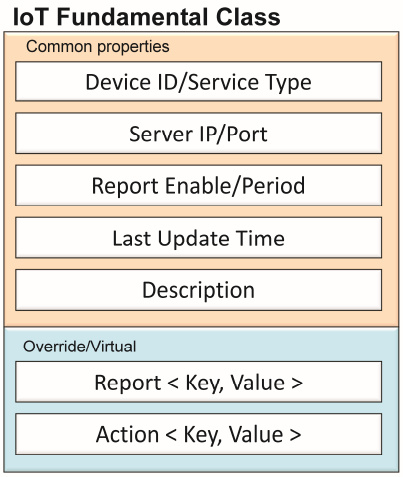


Fig. 3. IoT Fundamental Class

This class can be divided into two major parts - Common and 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 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 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, the IoT device interface and 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

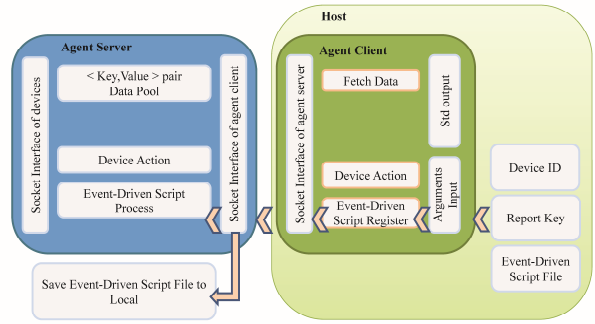


Fig. 4. Event-Driven Script Register Flow Chart

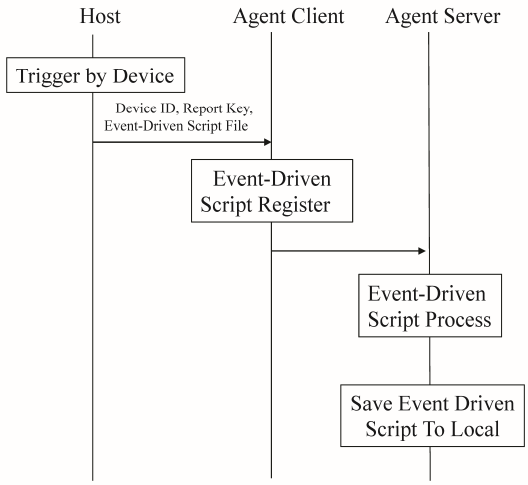


Fig. 5. Signaling flow for Event-Driven Script Register

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. Figure 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 signaling flow for Event-Driven script register presents in Figure 5.

2) Fetch Data

Figure 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 presents in Figure 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 below 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

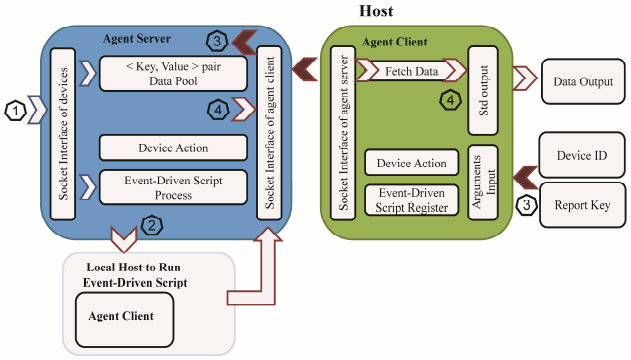


Fig. 6. Fetch Data Flow Chart

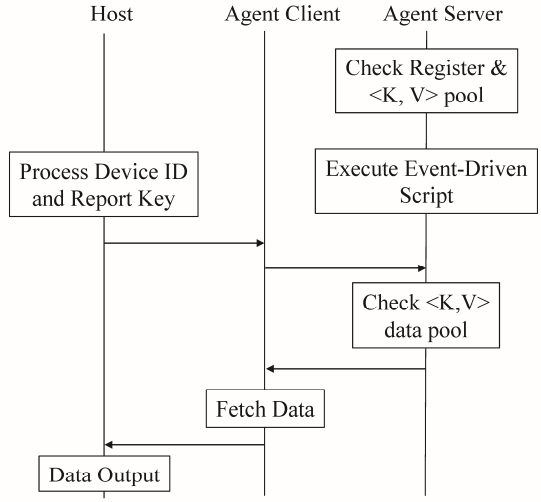


Fig. 7. Signaling flow for Fetching Data

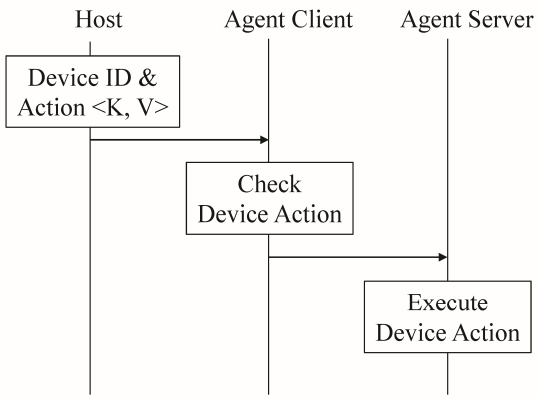


Fig. 8. Signaling flow for Device Action

Server. Various groups can be crossed by coordinating several Agent Servers by exploiting such flexibility.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

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.

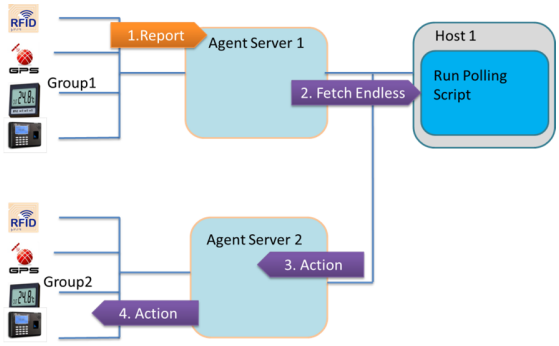


Fig. 9. Polling Script Block Diagram Flow Chart

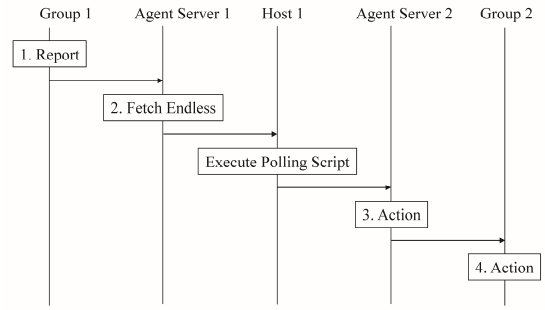


Fig. 10. Flow for Polling Script

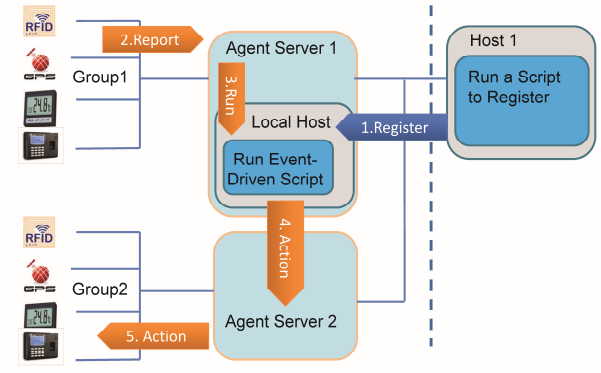


Fig. 11. Event Script Block Diagram

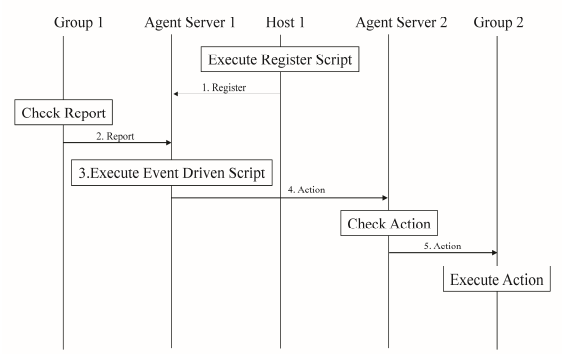


Fig. 12. Flow for Event Script

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 Figure 8. Table 1 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 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 Figure 9, and the signaling flow for Fetching Data presented in Figure 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. A simple change to the Agent Server IP easily supports cross-group control, as presented in Figure 11 and flow for Event Script presented in Figure 12.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

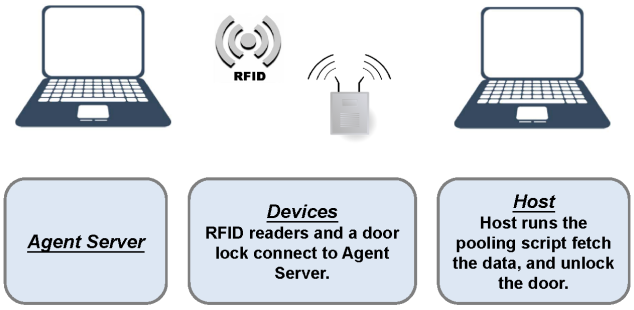


Fig. 13. Event Script Block Diagram

```
#!/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

IV. System Applications

The infrastructure and scripts for proposed simulations will complete in this section. The registration of the script on Host-side, and assign two ports of the interfaces on Agent Server-side to verify the RFID TAG values and then trigger Event-Driven script to issue door events. In this work, RFID TAG access control is difficulty on implementation to simulate the proposed ScriptIoT and verify the feasibility.

A. Infrastructure

Two laptops are used for simulation, as presented in Figure 14. Laptop A on the left is used to simulate the RFID reader and 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.

```
#!/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

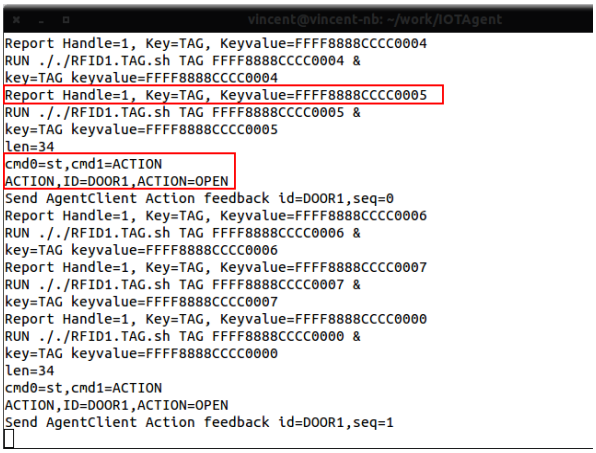


Fig. 17. Event-Driven Script--Agent Server Running Screen Capture

1) Host-side

Host-side refers to the script template on the Host, presented in Figure 14.

2) Device-side

a) The RFID is used to simulate as report function, allowing the Reader continuously to report TAG values between "FFFF8888CCCC0000" and "FFFF8888CCCC0007".

b) Simulate unlocking of the door. The action "OPEN" is supported, and a beep sound indicates that the door has been unlocked.

c) Simulate the script of device, as presented in Figure 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, Port-5000 for all devices and Port-5001 to serve the Agent Client. The Agent 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 pre-set values "FFFF8888CCCC0000" and "FFFF8888CCCC0005", the Event-Driven script is triggered to lock the door. As presented in Figure 16, the <Key, Value> are proposed by the Agent Server to trigger the event. Figure 17, once the TAG that is reported by the device is a pre-set value FFFF8888CCCC0000 or FFFF8888CCCC0005, the Agent Server will trigger the Event-Driven Script, which will send out the command to the Agent Server through the Agent Client to unlock the door.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

TABLE II
COST PARAMETERS

Parameter	Note
Ps	Agent Server/Local processing time
Pc	Agent Client/Host processing time
α	Transmission Delay

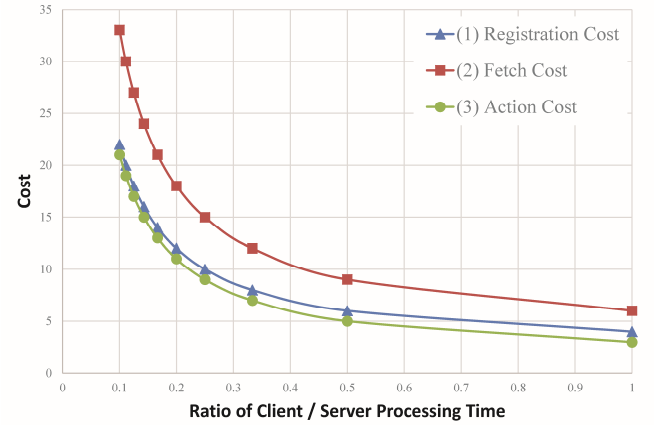


Fig. 18. Cost analysis result of ScriptIoT

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

In order to analysis the cost of each Event-Driven script register, Fetch Data, and Device Action in the proposed ScriptIoT framework. The detailed signaling flow of are shown in previous section. Then, the following parameters are used to analysis the cost:

By applying the parameter to the, the cost can are shown as the following equations: The cost of Register Event-Driven is

$$2(Ps+Pc)+2\alpha. \quad (1)$$

The cost of Fetch Data is

$$3(Ps+Pc)+4\alpha. \quad (2)$$

The cost of Device Action is

$$(2Pc+Ps)+2\alpha. \quad (3)$$

The computing power or resource of Agent Server and Agent Client are different. From the cost analysis in Figure 18, which shows different ratio of computing power of client and server. Table 2 presents the cost parameters. The Ps is assumed with value 1, which means maximum computing power. The parameter α is neglected is this analysis due to the 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 result shows that with higher Ps and Ps ratio, which means the Agent Client is the same computing power with Agent Server, would get loser 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.

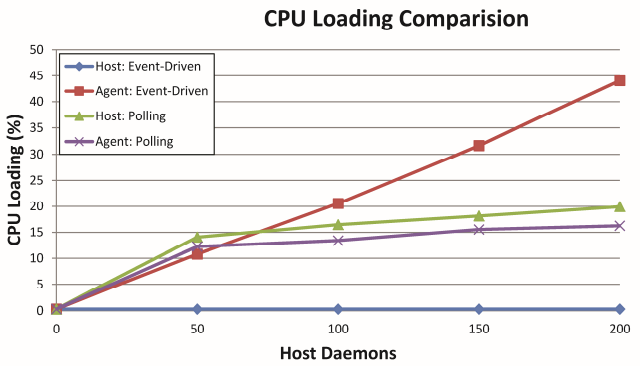


Fig. 19. CPU Loading Comparison

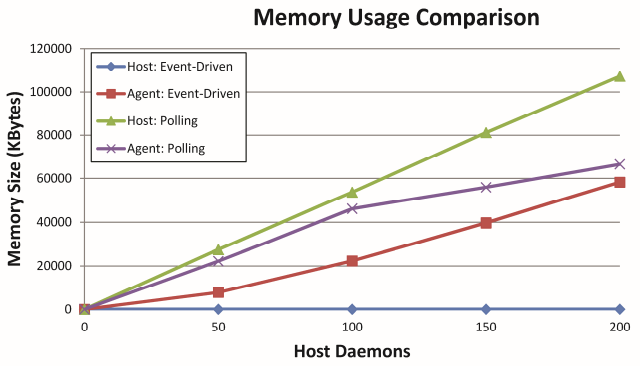


Fig. 20. Memory Usage Comparison

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.

3) The performance and resources of the systems are monitored ten times, once every 1s.

4) When the number activations have reached 5000 times, the time difference between the C code program and script is calculated.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

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 **	576000	1460000	572000	1460000	576000	1460000	560000	1460000
Memory Usage -2 **	568000	1542000	576000	1460000	564000	1460000	576000	1460000
Memory Usage -3 **	572000	1460000	576000	1460000	576000	1460000	568000	1460000
Average Time Elpase(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	572000	1487333	574667	1460000	572000	1460000	568000	1460000

The performance statistics are obtained as described in Table 3 with mean time consumption, average CPU loading, and memory consumption. And the comparison with Host and Agent Server, as presented in Figure 19 and Figure 20. The Agent Client uses approximately 388 Kbytes, while the environment of the Bash shell will require approximately 1184 Kbytes.

The average research statistics have indicated that:

1) The memory consumption by ScriptIoT, which is the memory used total capacity of its calling upon Agent Client is 1854833.
2) The memory capacity used by ScriptIoT can be obtained from the basic overhead of the Bash shell, as 670833.
3) Memory usage rate is 670833/571667=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 work proposes 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 is 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 is 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 proved 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 maintain its expandability and functionality. This framework contributes to large-scale logistic network applications.

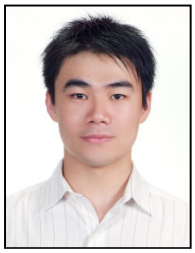
REFERENCES

[1] J. Gubbia, R. Buyyab, S. Marusica and M. Palaniswamia, “Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions,” Future Generation Computer Systems, Vol.29, No.7, pp.1645-1660, September 2013.
[2] http://www.ibm.com/smarterplanet/
[3] Qing Shen, Yu Liu, Zhijun Zhao, Song Ci, and Hui Tang, “Distributed hash table based ID management optimization for internet of things,” In Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC '10), pp.686-690, Caen, France, July 2010.
[4] Jin Mitsugi, Shigeru Yonemura, Hisakazu Hada, and Tatsuya Inaba, “Bridging UPnP and ZigBee with CoAP,” Proceedings of Conference on Emerging Networking Experiments and Technology, Tokyo, Japan, December 2011.
[5] Kashif Dar, Amirhosein Taherkordi, Romain Rouvoy, and Frank Eliassen, “Adaptable service composition for very-large-scale internet of things systems. In Proceedings of the 8th Middleware Doctoral Symposium (MDS '11), Lisbon, Portugal, December 2011.
[6] Jing He, Yanchun Zhang, Guangyan Huang, and Jinli Cao, “A smart web service based on the context of things,” ACM Transaction of Internet Technology, 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,” Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops, pp.385-388, San Diego, USA, March 2013.
[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,” Computer Networks, 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] Luigi Atzori, Antonio Iera and Giacomo Morabito: The Internet of Things: A survey. Computer Networks 54 (2010) 2787–2805
[12] H. Hada and J. Mitsugi, “EPC-based Internet of Things Architecture,” Proceedings of the IEEE International Conference on RFID-Technologies and Applications, pp.527-532, Sitges, Spain, September 2011.
[13] S.M. Hong, D.Y. Kim, M.K. Ha, S.H. Bae, S. J. Park, W.Y. Jung and J.E. Kim, “SNAIL: an IP-based Wireless Sensor Network Approach to the Internet of Things,” Proceedings of the IEEE Wireless Communications, Vol.17, No.6, pp.34-42, December 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 Recent Trends in Network Security and Applications, Vol.89, pp.430-439, Chennai, India, July 2010.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2015.2483023, IEEE Internet of Things Journal

[15] T. Li and C. Liping, "Internet of things: Principle, framework and application," Future Computing, Communication, Control and Management, Vol.144, pp.477-482, 2012.
[16] S. Bandyopadhyay, M. Sengupta, S. Maiti and S. Dutta, "Role of middleware for internet of things: A study," International Journal of Computer Science and Engineering, Vol.2, No.3, pp.94-105, Aug. 2011.
[17] Conzon, T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi and M.A. Spirito, "The VIRTUS Middleware: An XMPP Based Architecture for Secure IoT Communications," Proceedings of the IEEE International Conference on Computer Communications and Networks, pp.1-6, Munich, Germany, July 2012.
[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," Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp.1954-1959, Liverpool, UK, June 2012.
[19] G.C. Fox, S. Kamburugamuve and R.D. Hartman, "Architecture and Measured Characteristics of a Cloud based Internet of Things," Proceedings of the International Conference on Collaboration Technologies and Systems, pp.6-12, Denver, USA, May 2012.
[20] P. Castellani, M. Dissegna, N. Bui and M. Zorzi, "WebIoT: A web application framework for the internet of things," In Wireless Communications and Networking Conference Workshops (WCNCW), pp.202-207, Paris, French, April 2012.
[21] Sehgal, V. Perelman, S. Kuryla and J. Schonwalder, "Management of resource constrained devices in the internet of things," Communications Magazine, IEEE, 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," Proceedings of the IEEE International Conference on Networked Embedded Systems for Enterprise Applications, pp.1-6, Suzhou, China, November 2010.
[23] S. Babar, A. Stango, N. Prasad, J. Sen and R. Prasad, "Proposed Embedded Security Framework for Internet of Things (IoT)," Proceedings of the International Conference on Wireless Communication Vehicular Technology Information Theory and Aerospace & Electronic Systems Technology, pp.1-5, Chennai, India, February 2011.

Han-Chuan Hsieh was received a B.S. degree in Electrical Engineering from National Taipei University of Technology (NTUT), in 1998, and an M.S. degree in Communication Engineering from Tatung Institute of Technology, Taipei, Taiwan, in 2008. He is currently a PhD candidate in Department of Electrical Engineering of National Taiwan University of Science and Technology (NTUST). His major interests are in Long Term Evolution-Advanced, Internet of Things, Software Defined Networking and Network Functions Virtualization in 5G.



Kai-Di Chang received his B.S. degree in electrical engineering from National Dong Hwa University, Taiwan in 2007. He received his Master's degree in institute of computer science and information engineering at National I-Lan University, Taiwan. He is currently pursuing his Ph.D. degree in electrical engineering at National Taiwan University of Science and Technology. He is a student member of IEEE. His research interests include Mobile Communications, Cloud Computing, IP Multimedia Subsystem, Internet of Things and Network Security. Also, he is a researcher, works at United Daily News Group Co., Ltd, Taipei.

Ling-Feng Wang was received a B.S. Degree in Electrical Engineer form National Yunlin University of Science and Technology (NYUST) in 1997 and an M.S degree in Electrical Engineering and Computer Science from Nation Taiwan University of Science and Technology in 2013. He interests in IoT and embedded system applications.

Jann-Liang Chen 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 of National Dong Hwa University, where he is a professor and Vice Dean of Science and Engineering College. Prof. Chen joins the Department of Electrical Engineering, National Taiwan University of Science and Technology, as a full professor now. His current research interests are directed at cellular mobility management, digital home network, telematics applications, cloud computing and RFID middleware design. Prof. Chen is an IEEE Senior Member and UK BCS Fellow. He has published more than 150 papers in journals and conferences, and also holds several patents.

Han-Chieh Chao is a joint appointed Full Professor with the Department of Computer Science and Information Engineering and Electronic Engineering, National Ilan University, I-Lan, Taiwan (NIU). He is serving as the President since August 2010 for NIU as well. He was the Director of the Computer Center for Ministry of Education Taiwan from September 2008 to July 2010. His research interests include High Speed Networks, Wireless Networks, IPv6 based Networks, Digital Creative Arts, e-Government and Digital Divide. He received the M.S. and Ph.D. degrees in electrical engineering from Purdue University in 1989 and 1993, respectively. He has authored or co-authored 4 books and has published about 400 refereed professional research papers. He has completed more than 100 M.S.E.E. thesis students and 4 Ph.D. students. Dr. Chao has been invited frequently to give talks at national and international conferences and research organizations. Dr. Chao 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. Dr. Chao has served as the guest editors for Mobile Networking and Applications (ACM MONET), IEEE JSAC, IEEE Communications Magazine, IEEE Systems Journal, Computer Communications, IEE Proceedings Communication, the Computer Journal, Telecommunication Systems, Wireless Personal Communications, and Wireless Communications & Mobile Computing. Dr. Chao is an IEEE senior member and a Fellow of IET (IEE).