A Study on the IoT Instance Hosting in Edge Cloud and NFV

Jinhong Yang¹, Hyojin Park¹, SungKwan Jung¹
and Hyuncheol Kim²

¹*Dept. of Information and Communications,
Korea Advanced Institute of Science and Technology, South Korea*

²*Dept. of Computer Science, Namseoul University, South Korea
E-mail: {jinhong.yang; gaiaphj,skjung}@kaist.ac.kr; hckim@nsu.ac.kr*

Received 2 May 2016; Accepted 13 June 2016;
Published 8 July 2016

Abstract

Recently, as the number of IoT devices generate tremendous volume and variability of data. By the time data makes its way to the cloud for processing the chance to act on it might be disappeared. This paper presents a novel idea, proposes methods for virtualizing Internet of Things devices and hosting them on the home access point instead of relying on a cloud. For this, we design profile of the IoT gadgets and virtualize into JavaScript-based objects. Then, to execute and control the instances of the virtualized IoT objects on the wireless AP, a novel instance management method and their interfaces are designed. The implementation and performance test demonstrates the proposed system's stability and operability in the real wireless AP running environment. Adopting this platform feature over SDN/NFV infrastructure might be beneficial in to the IoT mashup service quality and its network efficiency.

Keywords: Internet of Things, Virtual Object, Instance Hosting, Edge Cloud.

1 Introduction

Recently, smart gadgets, sensors and appliances for personal healthcare and home automation are rapidly being deployed, which builds up an IoT (Internet

Journal of Software Networking, 37–52.

doi: 10.13052/jsn2445-9739.20161.003

© 2016 River Publishers. All rights reserved.

of Things) environment. On the contrary to the conventional sensor market which is target to the industry or service providers, this emerging sensory device market is target to the individual consumers in personal and home environment. This implies that the number and the kind of sensory devices belong to an individual person are also being increasing.

In these days, there are several kinds of App-Enabled accessories [1] or smart-home gadgets and sensors which are the consumer IoT devices that can be bought and configured by end users themselves. To control and access these gadgets requires their own service applications in forms of mobile applications, web services or native embedded system services.

This dedicated service application based access and control approach requires the end consumer to be burdened down with searching a dedicated service application for each IoT gadget and configuring it properly. And this approach also makes it more difficult the service provider to provide an integrated control and service mashup among several IoT gadgets. Thus, providing a vertical service solution for individual IoT gadget has hindered the development of IoT eco-system [2, 3].

To deal with this problem, there are some approaches, such as a service-based interworking approach and a cloud-based service approach. In a service-based interworking approach, the individual service (mobile application or web service) for a specific IoT device supports the registration of other IoT devices and provides an interworking function among IoT devices within its service. And in a cloud-based service approach, each IoT device connects to the cloud platform, and a service on the cloud platform provide all the necessary IoT service and interworking functions [4]. Using these approaches enables some more enhanced mashup services in user environment, but has a limit that the pre-built IoT service channels can interwork with each other [5]. Thus, users still suffer the limitation that any combination of IoT gadgets cannot be mashed up as his/her needs.

In this paper, a method that enables interworking and collectively controlling various IoT gadgets on a Wi-Fi AP (Access Point) system is suggested to solve these problems. Using a Wi-Fi AP system as an integrated IoT gadget controller has several benefits. First, it doesn't require any additional device but only the existing Wi-Fi AP system. Second, the personal/private data that is created from IoT gadgets can be stored and managed within his home safely. Third, more instantaneous service response is possible on the contrary to the cloud platform based service which might suffer from the service latency or drop due to network conditions. And last, it might support legacy (non-IoT) devices due to the possibility of supporting various physical device interfaces. To achieve these benefits, we propose an optimized IoT gadget

integration system and management scheme for its IoT service interworking feature with regarding the limited resources of Wi-Fi AP's and their embedded programming environment.

For this, a virtualization method to make a real world IoT device as a virtual object (VO) to be used in mashup environment is described, and the interface design and service management system architecture to create, execute and manage the created VOs on the in-Home Wi-Fi AP system is suggested. Also, the performance evaluation of the suggested platform is described to verify its intended operational behavior and its effect on the packet routing performance of Wi-Fi AP system by implementing the suggested platform on the real Wi-Fi AP. At last, the advantages and consideration points of making an IoT edge cloud over SDN/NFV infrastructure by extending the suggested IoT integrated management platform feature are described.

2 Related Works

The early stage IoT researches were mainly focused on how to connect physical things and devices to Internet. After that, the main research trend is how to utilize various connected IoT devices effectively. Thus, these researches are focus on the interworking various gadgets and the integration and analysis of their data. But, they are mainly beneficial to the large scale service providers, not to the end users who want mashup their own IoT gadgets as their needs. For example, Pachub (now, Xively) service only provides a simple triggering feature from IoT device data based on user-specified regular expression and doesn't provide any inter-device message exchange nor any additional service logic to make a mashup service.

The research on the virtualization of physical things could be classified the research on the device profile and interface and the research on the programming model for virtual object handling. First, for the description of device information, most of the researches adopt the standard model such as DPWS (Devices Profile for Web Services), and others develop its own profile model suit for their service model and environments. For the interface model, most of researches on Web of Things (WoT) utilize the SOAP (Simple Object Access Protocol) or HTTP-based REST (Representational State Transfer) structure to achieve message interoperability [6–8]. The early researches on the programming model were mainly conducted on the limited embedded system, such as TinoOS or Contiki. Thus, these models only supported the very limited programming model which required the device logic to be pre-built with the C programming language and to be loaded in the embedded running environments. Meanwhile, most of now-a-days researches adopt the

VM (Virtual Machine) based programming models [9, 10]. This VM based model makes the service providing structure to be more general one, but requires some more high performance device platforms.

The researches on the execution of virtualized objects have followed the similar trend with the development of the programming models: from the C-based object code native execution to VM based indirect execution. In addition, as the service expandability via interworking with existing web services gets main focus, there are several researches on the web based execution model along with the external web services [11, 12]. In this web based execution model, IoT gadgets expose their controls and information in RESTful foams via gateway devices with web servers, and the external web services can receive IoT gadget messages and execute their own service logic under own web environments [13, 14]. This execution model has the advantage that makes the development of virtualized object functionalities more easy due to the supports of several script based languages for web services, such as HTML, JavaScript, PHP, Python and etc. In addition to that, the web based execution model provides the user friend running and visualizing environments via web browser interfaces. But, most of the researches on the web based execution model utilized the gateway device only for the interfacing media and did not utilize it for service logic executions.

Though recent research approaches can provide several advantages, they also have some problems;

Under the script-based VO execution environments, there might be too frequent context switches for loading several instances, which requires somewhat more computing resource.

But, there might be a lack of all-time available computing resource for handling several VOs on network edge realm, especially in the home networks,

Thus, we suggest a novel system model that can describe the IoT gadgets as VOs, load and execute them with a capability of flexible IoT service mashup over the very limited computing resources.

3 IoT Instance Hosting Platform

First of all, these are important to make an IoT device in real world as a virtual object to be used in cyber world, and to equip an operating environment for executing the device in order to implementing and controlling IoT device at wireless internet router. In this Chapter, a methodology of device virtualization, connecting method between objects through an interface definition and structure design of instance manager for operating them are described.

3.1 Schema for Defining VOs

Virtual object is a thing which is defined to an object describing IoT device and functions provided from the device. It contains basic profile information of a physical device, technologies of network interface and operation logic of object for executing in virtual environment. In this paper, objects are defined separately according to a usage of virtual object; thing associated virtual object and mashup service virtual object. Thing associated virtual object is an object made for providing same functions in remote environment based on the message from IoT device. Therefore, basic information about a physical device (name, image information, explanation, etc.) and information of network interface for connecting (interface type, network address, port number, etc.) are described in XML. A logic of the physical device is also described in JavaScript. Table 1 shows a summary of main elements in thing associated virtual object.

Table 1 Example schema for thing associated virtual object

```

<!-- Omit the definition of Schema -->
<!-- "Profile" element is used as a top element -->
<xs:element name="profile">
  <!-- Virtual Object unique id -->
  <xs:element name="profileId">
  <!-- Virtual Object version -->
  <xs:element name="profileVersion">
  <!-- Basic information of virtual object -->
  <xs:element name="instanceDetailURL">
  <xs:element name="instanceDescription">
  <xs:element name="instanceIconURL">
  <xs:element name="instanceRunningPeriod">
  <!-- Connecting information of thing associated virtual object -->
  <xs:element name="associatedThingInformation">
  <xs:element name="configurableValue">
  <!-- Information related to interface between virtual objects -->
  <xs:element name="interfaceInformation">
    <xs:element name="oAuth">
    <xs:element name="feeder">
    <xs:element name="subscriber">
    <xs:element name="receiver">
    <xs:element name="controller">
  </xs:element>
  <!-- Describe the executing logic of virtual object in JavaScript -->
  <xs:element name="instanceScript">
</xs:element>

```

Table 2 Example schema on filter information for mashup service virtual object

```

<xs:element name="subscriber">
  <xs:element name="filter">
    <xs:attribute name="profileId"/>
    <xs:attribute name="feederId"/>
    <xs:attribute name="minVersion"/>
    <xs:attribute name="maxVersion"/>
  </xs:element>
</xs:element>

```

Mashup service virtual object is an object which has an independent service logic regardless of direct connection to a device. This object can provide a mashup service combined thing associated virtual object and service logic. Mashup service virtual object contains an extra filter information for searching connectable virtual object. Table 2 shows that filter information has a unique ID of connectable object for searching other virtual object, ‘profileID’, ‘feederID’ and version information.

3.2 Interface for Message Exchange Between VOs

Virtual object works by ‘instance’ which is the type of executable program of defined object when it is operated in the environment. Thus, thing associated virtual object is executed by ‘thing associated instance’, mashup service virtual object is executed by ‘mashup instance’. In this paper, five interfaces are defined for connecting instances easily; ‘Feeder’, ‘Subscriber’, ‘Controller’, ‘Receiver’, ‘OAuth’. Figure 1 shows interface connecting structure between instances graphically on an instance manager designed for controlling and executing virtual objects at wireless internet router. It also describes a relationship that mashup instance gets a message from the thing associated instance which received data from each device and provides mashup service connecting through OAuth and outside web service.

At first, ‘Feeder’ interface is an interface of exposing information for providing data and message generated from IoT device to instance or outside service. Feeder interface works the form to store information for feeding to Message Queue of instance, not to deliver message directly.

Secondly, ‘Subscriber’ interface provides a function that instance receives information from Feeder interface of other instance or outside thing. Mashup service object can be subscribed about interface information matched to the condition of ‘filter’ element.

Figure 1 Overall interfaces for instances hosting.

Thirdly, ‘Controller’ interface’ provides a function to deliver messages to outside directly. It is used to deliver message which has information to other instance or send a message about controlling order to certain IoT device.

Lastly, ‘OAuth’ interface provides a function of connecting outside 3rd Party service. Thus, it is used to send and receive a message through 3rd Party service.

The reasons of presenting call out structure of indicative interface to instance manager are for designing interface more simply in the view of virtual object developer, for using call out between instances and for blocking malfunction in advance through controlling message call out between instances in the view of instance manager.

3.3 Instance Manager

Instance manager provides a function to execute logic of virtual object in a form of each JavaScript instance. Instance manager consists of main modules like Figure 2 and the main module related to create and run instance based on virtualized object consists of Interface Server (providing a function of communicating between instance manager and outside modules), Message Router (providing a function of routing read and write message between instances), Instance Lifecycle Manager (providing a function of controlling

Figure 2 Architecture of the instance manager.

and scheduling instance lifecycle), Instance Pool (providing a function of pooling contexts of several instances), V8 (providing a function of executing instance).

Another module, IAS, provides web service for controlling instance manager in a web-based form and deploying function setting up a virtual object file. Network Adapter is responsible for managing messages between receiving device through HTTP, ZigBee interface and External Web Services is responsible for connection of 3rd Party web service.

Instance manager adjusts simultaneously handling quantity of instance for executing several virtualized objects effectively in an environment of embedded system and it should have the structure of providing quick response about each instance execution. It can be possible by pooling context information about the instance using instance pool after the first instance execution and by minimizing context creation time when instance is called out.

Since avoiding additional I/O usage except for the case of call out instance at first, performance can be higher.

These structure especially in IoT environment, is suitable to a service pattern which has a lot of message reception and handling between devices frequently. Therefore, it can be possible to use pooled context information to instances which contains frequent call out periodically and minimize creation time of the context by binding JavaScript engine directly.

3.4 Implementation and Performance

To evaluate the performance of the suggested Instance Hosting platform, we used an ASUS RT-N61 Wi-Fi AP which does support the customizable embedded OS, and added a TI ZigBee adapter to the internal serial port to support legacy non-IoT gadgets.

After porting the GNU/Linux based DebWRT embedded OS to the Wi-Fi AP, we built the Instance Manager using C++ language and the Instance Application Server using Apache-Rewrite Module, PHP5, MySQL and Python, and installed them on the AP system. The VO execution handling module was developed using V8 engine.

The execution logic for VO is described in JavaScript language and is processed on the V8 JavaScript engine. At the first time when a VO is to be executed, the actual object instance is created according to the VO description and this object instance is handled in the Instance Pool and used for the specific IoT gadget virtualization.

To evaluate the performance of the Instance Hosting platform, we conducted some experiments in real working environment, and some of the experiment results are shown in Figures 3 and 4. (During the experiment, each IoT gadget changes its state in every 10 seconds, i.e. its instance is invoked in every 10 seconds.)

Figure 3 Memory consumption by the deployed number of instance increases.

Figure 4 CPU consumption by the deployed number of instance increases.

As the number of hosted instance increases, the process usage (shown in Figure 3) and the memory usage (shown in Figure 4) are generally increased, but keep the stable behaviors. The initial memory consumption to execute the Instance Hosting platform is about 11 MB, but additional memory consumption for each instance is only less than 100 KB. This means that our proposed Instance Hosting platform can handle the enough number of IoT gadgets in home environments and the extended number of mashup instance under the limited embedded system environment.

4 Edge Cloud and SDN/NFV

So far, how to create and control the VOs in light-weight embedded system is recognized and it is confirmed that these structure works as expected. However, there can be a limitation to guarantee this performance continuously in light-weight embedded system at the situation of increasing IoT devices continuously at user side. Especially, the longer time user uses the IoT devices, the earlier storage which can afford the accumulation of data of intelligent service and previous data history for advanced intelligent service and process capacity approaches to the limit.

If Instance Hosting platform is implemented in SDN/NFV, a structure of Instance Hosting platform, as suggested before, could be changed as shown in Figure 5.

Figure 5 Instance hosting platform architecture over SDN/NFV platform.

Therefore, it is necessary that Edge Cloud which can provide intelligent mashup service storing and controlling data from several IoT devices and existing at the edge of network. But, it is not a realistic approach to set up the equipment for new Edge Cloud just for the connecting service between IoT devices at the edge of network. Therefore, loading Instance Hosting platform for connecting the IoT devices in approach of SDN/NFV framework which comes into the spotlight these days.

Interface Server for affording each IoT devices in Instance Hosting platform can be replaced to Network interface function module of SDN/NFV. Message Router in an instance manager can apply efficient router function module implementing at SDN/NFV when each VO Instances and Scheduler instances make to become virtual network entity in SDN/NFV. Instance connecting common outside service (virtualized instance which has information of local weather, regional traffic) which should be existed for each instance manager can be made to become general VO which can be used in common between several instance managers.

In addition, Point of Exist (POE) which has previous instance hosting platform existed on the physical edge of outside network and each user network on virtual edge of private/public network on the structure of SDN/NFV network exists on the virtual edge of private/public network in structure of SDN/NFV network when instance hosting platform over SDN/NFV platform is constructed. Therefore, there are additional advantages about connection

which can be new between many instances hosting manager. (It becomes to edge point since virtual edges of each users' network-public network co-exist physically in SDN/NFV platform finally. Therefore, mashup service which can cover all network of various each user can be possible at this edge point.)

5 Conclusion

IoT gadgets especially for individual person and smart home might make our daily lives more convenient and efficient. But, the increased number of personal belongings which needs to be managed also induces another kind of burdens. In this paper, we suggest the IoT management platform that can be located in the light-weight embedded system (like a Wi-Fi AP system) and handles the configuration, control and mashup service creation for several IoT VOs with user friend Web technologies. And by the experiments in real environment, we show that the suggested platform can provide the designed services with stable performance. In near future, this kind of IoT management platform can be deployed on Edge Clouds and SDN/NFV infrastructure.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. B0101-16-0331, Development of Multi-Screen Adaptive Autonomous Smart Media Technology).

References

- [1] "Appcessory Economics: Enabling loosely coupled hardware/software innovation," Cornell University Library, Oct. 28, 2013, from <http://arxiv.org/abs/1209.5901>
- [2] Andreas Nettsträter, "Deliverable D1.3-Updated Reference model for IoT v1.5," IoT-A, Jul. 16, 2012
- [3] Rob Chandhok, "A fast track to the Internet of Everything," Oct. 18, 2013 from <https://www.alljoyn.org/sites/default/files/alljoyn-alliance.pdf>
- [4] Xively, "Public Cloud for Internet of Things," <https://xively.com/>
- [5] IFTTT, "IF This Than That," <http://www.ifttt.com>

- [6] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, “*The Web of Things: Interconnecting Devices with High Usability and Performance.*” In Proc. ICESS, HangZhou, Zhejiang, China, May 2009.
- [7] D. Guinard, V. Trifa, and E. Wilde. “*A Resource Oriented Architecture for the Web of Things.*” In Proc. IoT, Tokyo, Japan, 2010.
- [8] J. Park and N. Kang, “*Entity authentication scheme for secure WEB of Things applications,*” J. KICS, vol. 38B, no. 5, pp. 394–400, May 2013.
- [9] P. Levis and D. Culler, “*Mate: A Tiny Virtual Machine for Sensor Networks,*” In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS-X, October, 2002.
- [10] N. Brouwers, K. Langendoen, and P. Corke. “*Darjeeling, a feature-rich VM for the resource poor,*” In Proc. SenSys, Berkeley, California, 2009.
- [11] Youngjun Kim, Yongkeun Jeon, and Ilyoung Chong, “*Device Objectification and Orchestration Mechanism for IoT Intelligent Service,*” J. KICS, vol. 38C, no. 1, pp. 19–32, January 2013.
- [12] Kovatsch, M, Lanter, M., Duquennoy, S., “*Actinium: A RESTful runtime container for scriptable Internet of Things applications,*” Internet of Things (IOT), 2012 3rd International Conference on the, pp. 135–142, Oct. 24–26, 2012.
- [13] Guinard, D., Trifa, V., Wilde, E., “*A Resource Oriented Architecture for the Web of Things,*” Internet of Things (IoT) 2010, pp. 1–8, Nov. 29, 2010.
- [14] V. Trifa, S. Wieland and D. Guinard, “*Towards the Web of Things: Web Mashups for Embedded Devices,*” Second Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web, Madrid, Apr. 20–24, 2009.

Biographies

J. Yang (S'05) received M.S. in computer science from InJe University in 2005 and HERIT Inc. in 2008 and currently he is Ph.D. candidate student in Korea Advanced Institute of Science and Technology (KAIST). His main research interests include next generation network, multimedia streaming issues, and IoT.

H. Park (S'07) received M.S. in communications engineering from Information and Communications University in 2007 and currently, she is Ph.D. candidate student in Korea Advanced Institute of Science and Technology (KAIST). Her main research interests include IPTV, broadcast networks, multimedia streaming issues, and next generation network.

S. K. Jung received Ph.D. in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2007.

Currently, He is team leader in KAIST KI ITC. His main research interests include Internet of Things, Smart Devices and next generation network.

H. Kim received his Ph.D. degree in the department of information engineering from Sungkyunkwan University, Korea, in 2005. From 1992 to 2002, he was senior engineer in Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He has been a professor in the department of computer science in Namseoul University, Korea. His research interests include cloud computing, next generation network and embedded computing systems.