Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 4, Number 1, pages 54–57, January 2015

Sensor Web 2.0 as a Service for Internet of Things

I-Ching Hsu
Department of Computer Science and Information Engineering , National Formosa University
64, Wenhua Rd., Huwei Township, Yunlin County 632, Taiwan
hsuic@nfu.edu.tw

Yu-Ju Su
Department of Computer Science and Information Engineering , National Formosa University
64, Wenhua Rd., Huwei Township, Yunlin County 632, Taiwan

***Abstract*—The Internet of Things (IoT) is a new paradigm that various objects associated with sensors are connected to Internet. The IoT can enable real-world devices to offer their functionality via Internet, but it has no mechanism to facilitating the interoperability and reusability among heterogeneous IoT applications and various physical devices. This study addresses these issues to propose a novel Sensor Web 2.0 as a Service, called SWaaS, which is a fundamental IoT service model. The SWaaS is developed based on a Multi-layer Sensor Web 2.0 Architecture (MSWA) that is composed of Sensor Web 2.0 Mashup technologies to associate with cloud computing environment to facilitate IoT applications development. To demonstrate the feasibility of this approach, this study implemented a Campus Monitoring System that is a IoT application developed based on SWaaS to provide continuous and context-aware monitoring of campus status.**

Keywords- Sensor Web 2.0; IoT; Cloud Computing

I. INTRODUCTION

In the Internet of Things (IoT) vision, the near future more and more various objects in the real-world will connect to internet through wireless sensor networks to interact with each other dynamically[1]. However, as the number of objects in the future Internet will increase rapidly, we will face a lot of challenges such as interoperability, reusability, and security[2]. The main problem of IoT is the lack of an uniformly access to cope with the heterogeneous effects, including various context presentation, context information, mobile device constraints, and context middleware. To allow interoperability among the various IoT applications and sensor devices, a common standard is needed to uniformly access information provided via Internet. The Sensor Web 2.0 [3, 4] technologies provide a catalytic solution to this problem.

Sensor Web 2.0 can be regarded as a kind of Web 2.0 Mashup, which integrates Web 2.0 and wireless sensor technologies as a basis for supporting pervasive IoT applications. It enables the development of IoT module easy to integrate the various Web 2.0 standards to facilitate the sharing and exchange of IoT information in the Internet. This study argues that Sensor Web 2.0 can be adopted as a common scheme to uniformly integrate IoT applications via a fundamental service model. The fundamental service model is Sensor Web 2.0 as a Service, called SWaaS, which developed based on a Multi-layer Sensor Web 2.0 Architecture (MSWA). The MSWA is composed of Sensor Web 2.0 technologies, including Web feed, Web API, Web presentation, and wireless sensor, to remove the heterogeneous issues of IoT. Additionally, SWaaS can also be combined with existing cloud computing service models, SaaS, PaaS, and IaaS to facilitate IoT applications development. Cloud computing is an emerging business model that has become a popular information technology solution in recent years. The first to introduce the term cloud computing was Google's CEO Eric Schmidt [5]. The term refers to the important and long-term trend in computing over the Internet. Many institutions and companies provide definitions and solutions for cloud computing [6, 7].

The remainder of paper is organized as follows. The next section presents the MSWA based on Sensor Web 2.0 technologies and describes the SWaaS. In Section 3, the study implemented a Campus Monitoring System to demonstrate the feasibility of SWaaS. Finally, summary and concluding remarks are included.

II. SENSOR WEB 2.0

A. *Multi-layer Sensor Web 2.0 Architecture*

This study proposes a Multi-layer Sensor Web 2.0 Architecture (MSWA) for supporting most of the tasks involved in dealing with pervasive IoT SaaS development. This study argues that Sensor Web 2.0 technologies can be adopted as a common scheme to integrate IoT objects uniformly using a fundamental cloud service model. This architecture is depicted in Figure 1, which can be represented in six layers: Cloud Computing Layer, IoT Object Layer, IoT Information Layer, IoT Service Layer, IoT Presentation Layer, and IoT Applications SaaS Layer. The Sensor Web 2.0 technologies consist of Web Feed, Web API, Web Presentation and Wireless Sensor, and can be mapped into IoT Information Layer, IoT Service Layer, IoT Presentation Layer, and IoT Object Layer of MSWA, respectively. The IoT applications SaaS is across the four layers. The MSWA provides a flexible infrastructure that IoT developer can dynamically add, replace, and remove components in each layer. Each layer contains multiple technologies, all of them providing a service suitable to the function of that layer.

IoT information can be acquired from heterogeneous and distributed sources, including sensors, objects, and applications. It contains a structured information source which is written in XML-based standard, such as RSS, Atom, and SensorML, to facilitate the machine-readable. IoT services are developed to support a reusable solution for objects acquisition to simplify the development of IoT applications. Web presentation technologies are mainly to provide a valid markup language for IoT presentation on a specific device. All of the layers are involved when sending request from a client to a IoT application. Therefore the upper layers have to rely on the lower layers to process the IoT objects over the Internet. Sensor Web Feed is used to distribute users regarding changes of contents at some IoT systems. Web API is used to facilitate data exchange between IoT applications and allow the creation of new IoT applications. The Web Presentation provides independence from differences in IoT information representation by translating the format from application format to a valid markup language for a specific client device.

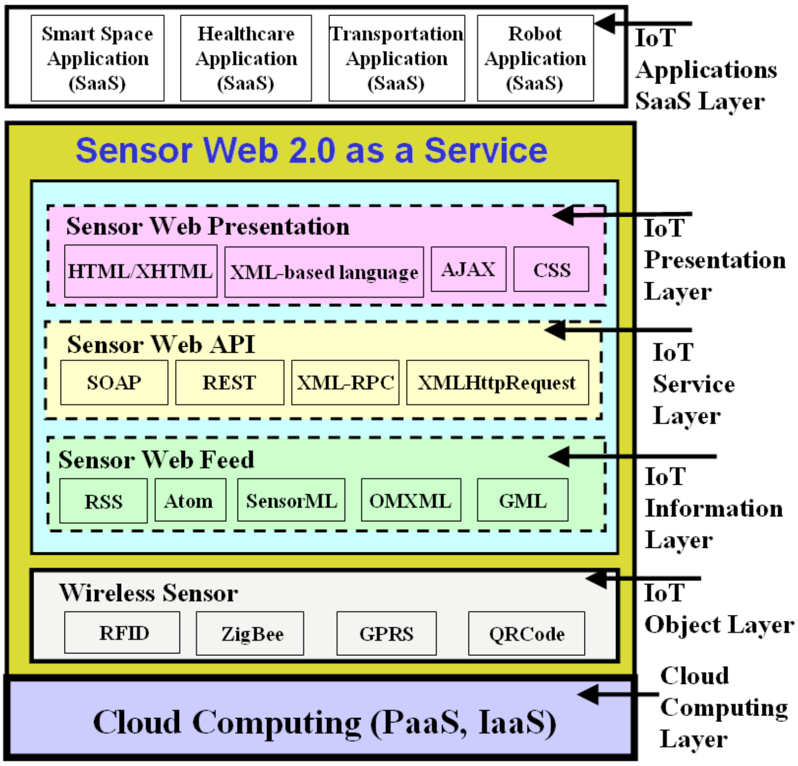

Figure 1. Sensor Web 2.0 technologies mapping to SWaaS

B. *Sensor Web 2.0 as a Service*

The components of SWaaS, shown in Figure 1, are described in the previous section. This section describes how SWaaS can be associated with existing service models of cloud computing. The cloud computing framework is composed of there layers: hardware layer (i.e. IaaS), platform layer (i.e. PaaS), and application layer (i.e. SaaS). IoT application is an SaaS located in the application layer.

Sensor Web feed is a typical data resource such as object information or metadata. In contrast, Sensor Web API is a typical service resource such as a wireless sensor service. The proposed MSWA has published APIs based on web standards so that IoT developers can access their context-aware services, including wireless sensor services (ZigBee, RFID, or QRCode) and message service (SMS).

III. IMPLEMENTATION AND EVALUATION

A. *Campus Monitoring System based on SWaaS*

This section demonstrates the feasibility of SWaaS, we implemented a Campus Monitoring System (CMS) that is a IoT application to provide a continuous and context-aware monitoring for campus. The main components of CMS include: Community Cloud, IoT Cloud, and Client. The Community Cloud consists of the CMS Portal Website that is deployed in Google App Engine to community inquiry for validated members. The IoT Cloud is composed of the RFID-based Sensor Network and IoT Middleware, which is retrieved by specific managers. In the CMS, student attendance information are collected through the SYRD245-1N (RFID Reader) and transferred to the ASUSTS300E5 for storage. Each SYTAG245-2F1 has a unique ID ties to a student (i.e. object) for identification. The dataflow-oriented architecture of CMS is depicted in Figure 2.

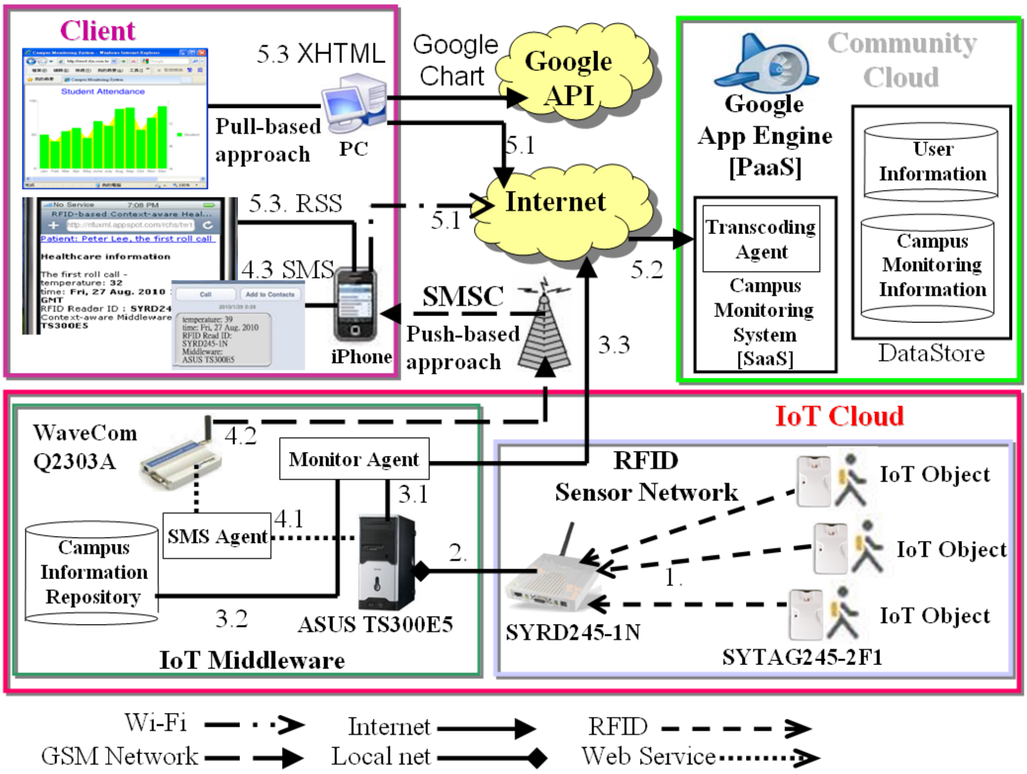

Figure 2. The dataflow-oriented architecture of Campus Monitoring System

The following steps explain the message flow illustrated in Figure 2:

1. Each student has a SYTA245-2F1 tag associated with a unique ID as the student identification. These active RFID tags send attendance context-aware information to the SYRD245-1N.
2. SYRD245-1N receives the context-aware attendance information and then sends them to ASUS TS300E5 by local network.
3. The ASUS TS300E5 invokes Monitor Agent to filter the available attendance information, and then save into the campus information repository.

3.1 It invokes Monitor Agent to filter the available attendance information.

3.2 Monitor Agent save these information into the campus information repository.

3.3 Monitor Agent encodes the student attendance information to an XML-based document, and then imports the XML-based document into Google App Engine.

4. The step is a push-based interaction scheme that accomplishes the following tasks:

4.1 ASUS TS300E5 requests SMS Agent with the real-time attendance information to start WaveCom Q2303A (GSM Modem).

4.2 WaveCom Q2303A serves as a GSM cell phone to transmit SMS message that contains the real-time attendance information. The SMS message is sent to SMSC (SMS Center).

4.3 SMSC is responsible for handling the SMS operations and routes the SMS message to mobile phone, such as iPhone.

5. The step is a pull-based interaction scheme that accomplishes the following tasks:

5.1 The various cloud devices, such as iPhone or iPad, can send a request to CMS with the student ID to browse the student attendance information.

5.2 The CMS invokes Transcoding Agent with student ID to acquire the attendance information form campus information data store, and then converts the attendance information into a XML-based document.

5.3 The XML-based document can be converted to various XML-based documents, such as RSS or XHTML document, to display in iPhone and PC, respectively.

B. *Evaluation*

This section evaluates the Campus Monitoring System (CMS) for Sensor Web 2.0 as a Service (SWaaS) against our requirements. The response time of CMS can be evaluated along two fronts. Firstly, the response time of pull-based approach, which underlies the Web-based information transcoding and Internet transmission, evaluates how the response time of the CMS increases when the number of the request increases. Secondly, the execution time of push-based approach, which underlies SMS Web Service and GSM communication, evaluates the increasing trend of the execution time of the CMS when the number of client device increases.

The pull-based response time contains the Web-based information transcoding time and Internet transmission time. The same experiment will be executed on the local server and GAE cloud platform, respectively. This experiment evaluated the CMS as a IoT broker that processed from 10 to 1000 requests. Figure 3 shows the average values obtained for pull-based response time. Notably, the threshold value for request number was about 500. When request number was lower than the threshold value, average response time was about 0.6 seconds. Conversely, when the number of concurrent requests increased beyond this threshold number, response time increased very rapidly because both I/O consumption and contextual information transcoding performance increased substantially. Additionally, for each dataset, execution time in local server was longer than that in the GAE cloud platform.

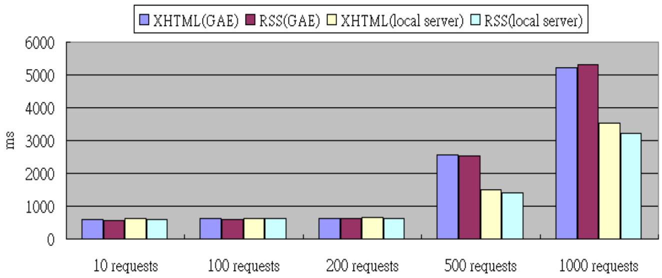

Figure 3. Results of the pull-based approach experiments

The push-based execution time contains the SMS Web service execution time and GSM communication time. Due to the limited number of mobile devices and the cost of sending SMS, the actual test of the experiment can only send SMS to 20 different mobile devices. The experience results of the push-based average execution time are shown in Figure 4. We observed that the average execution time of SMS Web service and the number of mobile device increase in equal proportion. The average time of GSM communication is about 3.1 second. Additionally, this tests show that the execution time of SMS Web Service takes a very limited percentage of the receive SMS time (about 1%). It is worth noting that a significant variation on push-based execution time result form variable GSM communication. The push-based experiment executed only on the local server. This is because the GSM modem can not be installed in the GAE cloud platform.

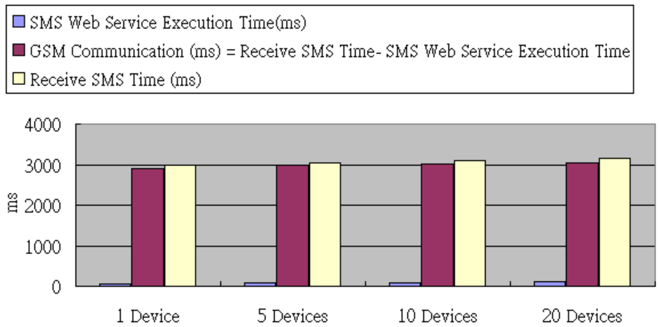

Figure 4. Results of the push-based approach experiments

IV. CONCLUSIONS

The rapid development of the wireless sensor technologies, including wireless sensors, smart objects, and communication protocols, has led to diverse devices of accessing various IoT systems. This study proposes a Multi-layer Sensor Web 2.0 Architecture (MSWA) that consists of contest sensor layer, context information layer, context service layer, context representation layer, cloud computing layer, and context-aware mobile Web 2.0 SaaS layer. The author argues to adopt Sensor Web 2.0 technologies as the backbone of IoT applications to facilitate the sharing and exchange of objects in cloud computing environment. From the Campus Monitoring System carried out in the paper, we demonstrated that Sensor Web 2.0 technologies serve as a core technology for Web 2.0 Mashups to enhance added value of traditional IoT applications. Additionally, this study also realizes how Sensor Web 2.0 technologies can be integrated into IoT applications for cloud computing environment. One future work is to investigate how to integrate Semantic Web technologies [8] into SWaaS to facilitate the development of intelligent Web Sensor 2.0 applications.

REFERENCES

[1] Jiang, Y., L. Zhang, and LingWang, Wireless sensor networks and the internet of things. International Journal of Distributed Sensor Networks, 2013.

[2] Atzori, L., A. Iera, and G. Morabito, The Internet of Things: A survey. Computer Networks, 2011. 54(15): p. 2787-2805

[3] Bröring, A., et al., New Generation Sensor Web Enablement. Sensors, 2011. 11(3): p. 2652-2699.

[4] O'Hare, G.M.P., et al., Sensor web interaction. International Journal on Artificial Intelligence Tools, 2012. 21(2).

[5] Schmidt, E. Conversation with Eric Schmidt hosted by Danny Sullivan, Search Engine Strategies Conference. 2006 March 21, 2011]; Available from: http://www.google.com/press/podium/ses2006.html.

[6] I-Ching Hsu, Multilayer context cloud framework for mobile Web 2.0: A proposed infrastructure. International Journal of Communication Systems, 2013, 26(5), p.610–625.

[7] I-Ching Hsu, Extending UML to model Web 2.0-based context-aware applications, Software: Practice and Experience, 2012, 42(10), p.1211-1227

[8] I-Ching Hsu, Lee Jang Yang, Der-Chen Huang, Kuan-Yang Lai , Integrating Semantic Web technologies with XML Schema using role-mapping annotations, The Electronic Library, 2014, 32(2), p.147-169