Design and Implementation of Service Framework for Presence Sensing in Home Network System

Yuki Kashio, Shinsuke Matsumoto, Seiki Tokunaga, Sachio Saiki and Masahide Nakamura
Graduate School of System Informatics, Kobe University
1-1 Rokkodai, Nada, Kobe, Hyogo 657-8501, Japan
Email: {kashio@ws.cs, shinsuke@cs, sachio@carp, masa-n@cs}.kobe-u.ac.jp.

***Abstract*—To achieve efficient presence sensing within home network system (HNS), an inexpensive and elastic system that can be shared by various HNS applications is required. This paper presents *Presence Sensor Service Framework*, which can uniformly manage presence information in various places using various types of sensors. The framework consists of Presence Sensor Device (PSD), Presence Sensor Terminal (PST) and Presence Sensor Aggregator (PSA). A PST monitors PSDs to detect any changes of presence, and notifies a PSA of the changes. Upon the notifications from PSTs, the PSA estimates human presence around each PSD. The estimation process is deployed as Presence Sensor Service (PSS), which is used by various HNS applications. The proposed framework can dynamically add or change sensing places within a HNS by adding PST and PSD as needed. It can also integrate presences in multiple HNSs by deploying PSA on an external cloud. In this paper, we implement a prototype of the proposed framework by using Phidgets sensors and Java Web service. Moreover, we develop a presence visualization application to conduct a preliminary evaluation.**

***Keywords*—presence sensing, home network system, infrared motion sensor, web service, smart system**

I. Introduction

Presence sensing is a technology that automatically detects if a person (or an object) is present at a certain place. It is widely used in the fields of automation and smart services. For example, the presence sensing is used in automatic doors, energy-saving household appliances [1], efficient air-conditioning in office [2], and automatic light controls in buildings [3]. Our research group has been extensively studying the *Home Network System (HNS)* [4]. The HNS integrates various household appliances and equipment via home network to realize value-added services. In this paper, we especially aim to develop an efficient method of presence sensing suited for the HNS within general houses.

To realize indoor presence sensing, many solutions with various types of presence sensors have been proposed. The sensors include infrared sensor [5], magnetic sensor [6], pressure sensor [7], ultrasonic sensor [8], and wireless LAN [9]. However, it is yet challenging to introduce the current presence sensing systems in general households, because they are basically closed systems that require expensive devices or large-scale infrastructure. Moreover, the current systems do not assume HNS services or *smart city services* with multiple HNSs, where sensing places are dynamically changed.

In order to solve these problems, this paper presents a new application framework, called *Presence Sensor Service Framework (PSSF)*. PSSF connects various types of sensors, and uniformly manages the sensors to detect *human presence* in various places within HNS.

PSSF consists of *Presence Sensor Device (PSD)*, *Presence Sensor Terminal (PST)*, and *Presence Sensor Aggregator (PSA)*. PSD refers to any physical sensor device used for the presence sensing. PST works as a local hub managing several adjacent PSDs. PSA is global data store that aggregates sensor information from multiple PSTs. A PST has several adjacent PSDs and monitors values of the PSDs. If value of a PSD exceeds a threshold, the PST notifies a PSA. Upon receiving a notification from the PST, the PSA records id and the value of the PSD, and the date and time of the notification. A PSA implements an algorithm that estimates the likelihood of human presence based on the raw data recorded. The PSA also provides the human presence as a Web service, which is called *Presence Sensor Service (PSS)*. PSS is an abstract service with presence sensing API, which achieves *loose-coupling* between applications and sensor devices.

By using PSSF, various applications can easily get and use presence information in the HNS, without considering heterogeneous physical sensor devices. PSSF also allows service administrators to dynamically add, change, or remove any PST or PSD as needed. Moreover, by deploying the PSA on the cloud, it is possible to implement *smart city services*, which integrate presence information from multiple HNSs.

In this paper, we also implement a prototype system of PSSF. The prototype exploits Phidgets infrared motion sensors as PSDs, and a RaspberryPi (with a Phidgets interface kit) as a PST. A PSA is developed as Java program. Then, the PSA is deployed with Apache Axis2 web service as PSS. To evaluate the proposed method, we install the prototype system in an actual HNS in our lab (CS27-HNS). We then visualize human presence in our lab, by developing a Web application with the proposed PSSF and Google Chart API.

II. Preliminaries

A. Home Network System (HNS)

Home network system (HNS) is a system that achieves value-added services by connecting household appliances, sensors and equipment to home network. Various home appliances such as TVs, DVD recorders, lights, air-conditioners, electric fans, and air-cleaners are integrated to implement various HNS services and applications.

To create HNS environment in our laboratory (CS27-HNS) [4], we have extensively utilized the concept of *service oriented architecture (SOA)* within HNS. In CS27-HNS, every feature of the household appliances can be used as the

ISBN: 978-1-4799-6375-1

Web-API, accessed by standard Web service protocols (i.e., SOAP or REST). Underlying proprietary protocols and device-specific controls are wrapped by Web services. For example, to change a TV channel to 6, a user or an application simply access a URL http://cs27-hns/TVService/setChannel?channel=6.

We have been developing value-added services within CS27-HNS. For example, the appliance control service from outside, the energy consumption visualization service [10], the electric peak shaving service[11], the integrated service of multiple appliances [12], the voice operation service [13], the personalized home control service [14], and so on. We have also developed the sensor service framework (SSF) [15], which facilitates installation of environmental sensors in the HNS. The sensor service binder(SSB) [16] supports non-expert users to develop custom context-aware services with the sensors.

B. Presence Sensing within HNS Environment

The presence sensing is a technology that determines whether a person (or an object) is present at a certain place. It is used in various fields [17]. In this paper, we especially focus on presence sensing of human users within a HNS environment. The presence sensing within HNS environment provides an efficient means to detect user's movements or location in a house. It can be used to check if a user is in a room, to estimate user's position, or to detect user's approach to a specific location or object (e.g., a door, an entrance, or an appliance). These information are useful to implement more user-centric applications and services in the HNS.

When we introduce a presence sensing system in a general household, the system must be installed easily with inexpensive cost. Also, it must be able to be adapted for the future change of the house. Therefore, in this paper, we especially consider the following four requirements:

- **Requirement R1:** The system can use various sensor devices for presence sensing.
- **Requirement R2:** The system can dynamically add, change or remove the places of sensing in the house.
- **Requirement R3:** Various HNS services and applications can obtain and share the presence information.
- **Requirement R4:** The system is not limited in a single house, and can be extended for multiple houses.

The purpose of Requirement R1 is to allow users to select various sensor devices suited for individual circumstances (the size of house, family, budget, etc.). Requirement R2 enables the system to adapt heterogeneous house environments (room layout, equipment, etc). Requirement R3 allows external applications to make full use of the system. Requirement R4 considers the adaption to smart city services in the future.

There have been various types of presence sensing systems so far. However, none of the conventional systems is for general households, and thus they do not meet these requirements. Most existing systems are for business office environment or public facilities, which are basically closed systems for proprietary services or appliances.

Therefore, we are conducting research and development of a new presence sensing system that can be easily adopted for general houses with the HNS environment.

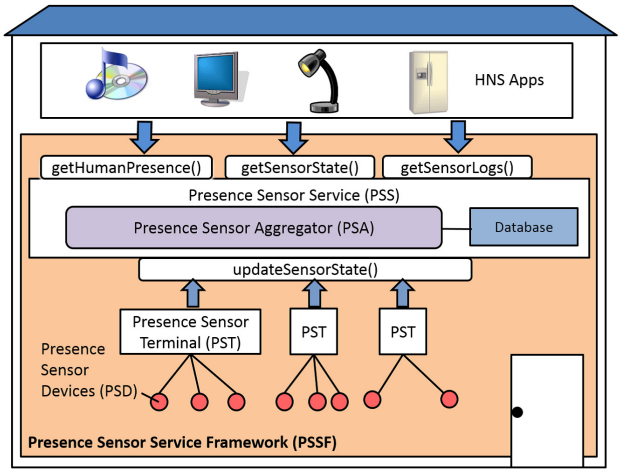

Fig. 1. Architecture of the proposed framework

III. Service Framework for Presence Sensing in HNS environment

A. System architecture

Figure 1 shows the architecture of the proposed framework, called *Presence Sensor Service Framework (PSSF)*. In the framework, a *presence sensor device (PSD)* is a physical sensor device that detects human movements, a *Presence Sensor Terminal (PST)* is a local hub managing several adjacent PSDs. A *Presence Sensor Aggregator (PSA)* is a global data store that aggregates sensor information from all the PSTs in a house.

A PSD is deployed in a place, where a user wants to detect human presence. To improve accuracy and cover wider range, several PSDs can be deployed nearby. The adjacent PSDs are connected to a PST. The PST monitors values of the PSDs. If a value of a PSD exceeds a pre-determined threshold, the PST notifies the PSA of *sensor information*, consisting of id of the PST, id of PSD, the value of PSD, and location information.

Upon receiving a new sensor information, the PSA updates the sensor information, and records the last update date and time. The PSA implements a method that estimates whether a person is present around each PSD. The method is exposed as a Web service, so that various HNS applications can easily play with the human presence information via Web-API. The Web service that wraps PSA is called *Presence Sensor Service (PSS)*. The details are described in the following sub sections.

B. Presence Sensor Device (PSD)

The *Presence Sensor Device (PSD)* plays a roll of detecting human movement within the HNS environment. A PSD is a physical sensor device supposed to be installed in a place where a user wants to perform presence sensing. As for the PSD, the user can use various types of sensors, including infrared, magnetic, pressure or wireless LAN. However, to introduce the PSSF in a general house, the sensor should be inexpensive, and be installed easily with low cost.

Considering the cost, we chose an infrared motion sensor (Phidget 1111_0) in this study. This motion sensor measures a change of infrared light generated by movement of goods

having heat. Thus, it can sense the human movement around the sensor. Software driver of the sensor allows to add an *event listener* that notifies an event when the sensor value changes. Each PSD is associated with an unique id (called, psdId).

The accuracy and coverage of presence sensing can be improved by placing several PSDs nearby. For example, when we detect human movements in a room, using four sensors instead of one can cover larger area with better accuracy. However, improving coverage and accuracy increases the cost of sensors, which is a trade-off relation.

C. Presence Sensor Terminal (PST)

The Presence Sensor Terminal (PST) plays a roll of a local hub connecting several adjacent PSDs, and notifies a PSA when a sensor value changes. A PST has a set of *observers* each of which monitors a connected PSD. Moreover, a PST has an id (pstId) and a URL of the PSA to be notified. The PST sets these information to the observers when every observer is created.

Each observer has a *threshold* of the sensor value, by which a human movement is determined. It also has meta-data explaining location of the sensor (locationInfo). Using the event listener of the PSD, each observer monitors change of the sensor value of the PSD. If the value exceeds the threshold, the observer notifies a PSA (specified in a URL) of a *sensor information*. The sensor information consists of pstId, psdId, locationInfo, and the sensor value.

Note that notifications to the PSA are executed by observers autonomously, based on an *event-driven communication*. Thus, the communciation occurs only when a sensor value exceeds the threshould, and the the direction of communication is always from a PST to a PSA. This achieves the high scalability of the PSA in the number of PSDs and the number of PSTs.

On the other hand, there are many cases where the sensor values are not changed frequently even if a person is present. For such cases, the estimation of human presence is not determined by PST only, and is delegated to the PSA. In other words, we divide the responsibility between the PST and the PSA. That is, the PST detects a major change of sensor values as an event, whereas the PSA estimates the human presence based on the events and their history.

D. Presence Sensor Aggregator (PSA)

The *Presence Sensor Aggregator (PSA)* is a global data store that aggregates sensor information (sensorInfo) from all the PSTs in a house. A PSA is associated with a URL, to which a PST can send the sensorInfo. A PSA first collects sensorInfo from the PSTs, and then estimates human presence around every PSD. The result of estimation is stored to a database to construct log (i.e., history) of human presence.

For each PSD connected to a PST, the PSA manages a *sensor state* to characterize the current status of each sensor. A sensor state consists of pstId, psdId, locationInfo, sensor value, last update date, and last update time. Each PST notifies a PSA of a sensorInfo by using `updateSensorState()` API. When this API is executed, the PSA updates the sensor state using the data in the sensorInfo with the current date and time.

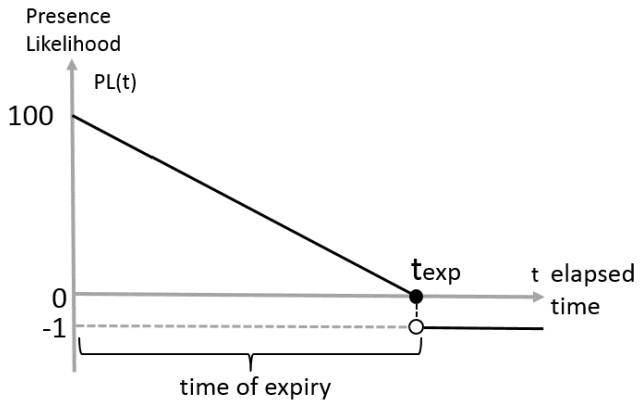

Fig. 2. Graph of the presence likelihood

The PSA provides the latest sensor state for external applications via `getSensorState()` API with a PSD key. The PSD key is a concatenation of pstId and psdId, which uniquely identifies a PSD within a house.

The PSA also has `getHumanPresence()` API, which estimates human presence based on the sensor state. For a given PSD key, the API returns *presence information* with respect to the PSD. Although there are many ways to describe human presence, we here choose to represent the presence as *likelihood*, which is a possibility that a human is present near a certain place. We call it *presence likelihood*. Thus, the presence information consists of pstId, psdId, and presence likelihood.

The presence likelihood takes a value from -1 to 100, where -1 means "nobody is present" and 100 represents "somebody must be present". Let t be the time elapsed from the last update time of the sensor state until the time `getHumanPresence()` methods is invoked. Intuitively, the smaller t yields more presence likelihood, because there is a fact that a human movement was detected within a short time. Thus, if the time is not elapsed enough after a PSD detects a human movement, the possibility that a person is present is high. Conversely, as the value of t becomes larger, the presence likelihood decreases because the person moves out of the range of the sensor. Based on the above principle, we define the presence likelihood as the following function $PL(t)$ with respect to the elapsed time t:

$$PL(t) = \begin{cases} 100\ (1 - t/t_{exp}) & (0 \leq t \leq t_{exp}) \\ -1 & (t > t_{exp}) \end{cases}$$

In the function, t_{exp} represents the *time of expiry*. When t is smaller than t_{exp}, the presence likelihood linearly decreases from 100 to 0. When t exceeds t_{exp}, the presence likelihood becomes -1, indicating that nobody is present. The graph of $PL(t)$ is depicted in Figure 2.

The presence information (i.e., pstid, psdid, and presence likelihood) is stored in a database every second, which forms *presence log*. Table I shows a data table of the presence log. Each record consists of log id, pstId, recorded date, recorded time and presence likelihood. The presence likelihood is represented as key-values of psdID and the value of likelihood of the PSD.

TABLE I. TABLE OF PRESENCE LOG

Log id	date	time	pstId	presence likelihood
ObjectId("539f0697e4b014")	2014-06-13	16:55:50	149180	{1:-1, 2:20, 3:100, 4:50, 5:-1, 6:-1}
ObjectId("539f0697e4b015")	2014-06-13	16:55:51	149180	{1:100, 2:10, 3:90, 4:30, 5:-1, 6:-1}
ObjectId("539f0697e4b016")	2014-06-13	16:55:52	149180	{1:90, 2:0, 3:80, 4:20, 5:-1, 6:10}

The presence log can be retrieved by `collectSensorLog()` API of PSA. This API can be invoked by specifying pstId, psdId, start_date, end_date, start_time, end_time, and term. start_date (or end_date) specifies the date from (or until, respectively) when you retrieve the presence log, which is given in the form of "yyyy-MM-dd". start_time (or end_time) specifies the time from (or until, respectively) when you retrieve the presence log, which is given in the form of "hh:mm:ss". term specifies an interval of date of retrieving the presence log, which is given in the forms of "yyyy-**-**", "****-MM-**", or "****-**-dd". For example, by giving "2014-**-**" as a term, it is possible to retrieve all presence log within 2014. Similarly, giving "****-11-**" as a term retrieves get presence log in November of every year.

In `collectSensorLog()` API, arguments that are not needed for the search can be given as "null". For example, to obtain presence log of all PSTs and PSDs from November 1, 2014 at 00:00:00 until November 23, 2014 at 12:00:00, the user specifies [null, null, 2014-11-01, 2014-11-23, 00:00:00, 12:00:00, null] to the API. Both pstId and term are independent of any other arguments. In contrast, pstId cannot be omitted when psdId is given. Moreover, both start_time and end_time must be given at a time. Similarly, both start_date and end_date must be given as a pair.

E. Presence Sensor Service (PSS)

The *Presence Sensor Service (PSS)* provides features of the PSA as a Web service, so that various HNS applications can easily use the features. The PSS wraps a PSA, and provides various Web services using the API of PSA (such as `getHumanPresence()` and `collectSensorLog()`). For example, the PSS includes Web-API `getAllSensorStates()` which returns the current sensor states of all PSDs, as well as `getAllPresence()` which returns the presence information of all PSDs.

The PSS also provides Web-API `getSensorLogsBypstID()` that returns presence log related to a given PST. In this Web-API, arguments required for `collectSensorLog()` are automatically configured by setting null values to arguments except pstId, which is convenient for the user. When a HNS application obtains presence logs, arguments that are not required for search (e.g., start time, end time, and start date) can be omitted. PSS can be extended easily by adding more API adapting to the requirement of the data search.

Providing the PSS as a Web service allows heterogeneous HNS applications to use the presence sensing information in a standardized manner. Each application invokes the Web-API with the REST or SOAP protocol, and receives the result in the form of XML.

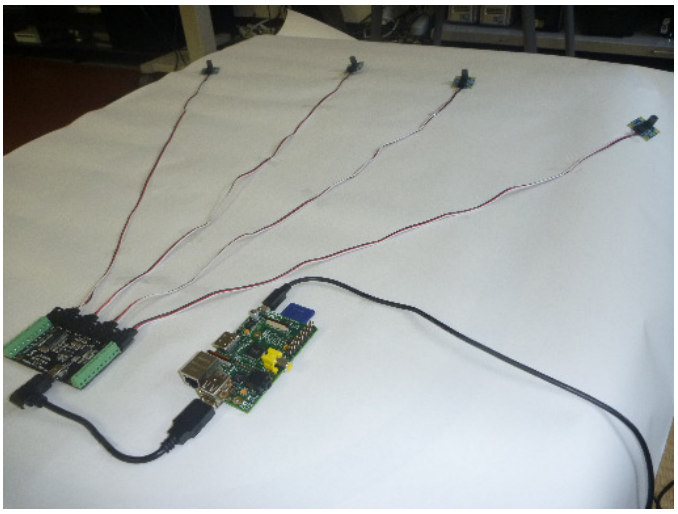

Fig. 3. Presence sensor terminal with four presence sensor devices

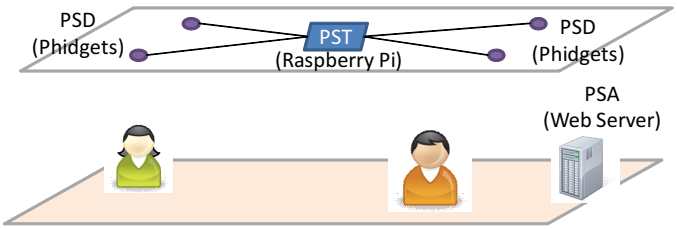

Fig. 4. Installation of the prototype in our laboratory

IV. IMPLEMENTATION

A. Prototype of Presence Sensor Service Framework

Based on the proposed method, we have developed a prototype of the PSSF. We have also installed the prototype in our CS27-HNS to conduct presence sensing in our laboratory.

Figure 3 shows the developed prototype of a presence sensor terminal (PST) with four presence sensor devices (PSDs). In order to develop the prototype, we have used the Phidgets infrared motion sensor (1111_0 - MotionSensor) for the PSDs. We then connected four motion sensors with the Interfacekit (1018_2 - PhidgetInterfaceKit 8/8/8). The Interfacekit was connected to an one-board PC (Raspberry Pi) via USB. The one-board PC was finally connected to CS27-HNS via LAN.

The software of the PST presented in Section III-C was implemented in Java. We integrated the motion sensors in the software using phiget21 library, provided by Phidgets Inc. The developed prototype was installed on the ceiling in our laboratory. Figure Figure III-C shows the image of the installation of the prototype.

Moreover, both the PSA and the PSS were developed in Java. The developed program was deployed on a web server

ISBN: 978-1-4799-6375-1 ©2015 IEEE

TABLE II. TECHNOLOGIES USED IN THE PROTOTYPE

PSSF Layer	Technologies Used
Presence Sensor Device (PSD)	Phidgets 1111_0 – Motion Sensor, Phidgets 1018_1 – InterfaceKit
Presence Sensor Terminal (PST)	Raspberry Pi Model B, Phidget21.jar, Java JDK7
Presence Sensor Aggregator (PSA)	Vine Linux 4.2, Java JDK7, mongo-2.10.1.jar
Presence Sensor Service (PSS)	Java JDK7, Apache Tomcat6.0, Apache Axis2 1.4
Database	mongoDB 2.6.1

```
<ns:getHumanPresenseResponse>
  <return type="jp.kobe_u.cs27.pss.Human
  Presense">
    <locationInfo>
        West Door of S101
    </locationInfo>
    <presence>94</presence>
    <psdID>1</psdID>
    <pstID>149180</pstID>
  </return>
</ns:getHumanPresenseResponse>
```

Fig. 5. Result of `getHumanPresence()` Web-API

as the Web service. For this, we used Apache Axis2 Web service middleware, and Tomcat 6.0 for the Web server. Table II summarizes technologies used for the prototype.

Figure 5 shows a result of invocation of Web-API `getHumanPresence()`, shown in an XML format. In this example, the result shows that the presence likelihood at West Door of the room S101 is estimated to 94.

B. PresenceViewer: Client Application of Framework

We have also implemented *PresenceViewer*, which is a client application of the developed framework. PresenceViewer visualizes the human presence on a room map in a web browser. PresenceViewer was developed by using JavaScript and Google Chart Tools. It internally invokes the Web-API of the PSS using the REST protocol.

Figure 6 shows the screenshot of PresenceViewer. For a given term between the start date/time and end date/time, PresenceViewer retrieves the presence log from the PSS, and visualizes the likelihood with *bubbles* on the room map. Each bubble represents a value of the presence likelihood by a color and the size. When the value of presence likelihood is larger, the bubble becomes darker and bigger. For every second, the bubbles are refreshed to visualize the presence log in the next seconds (here it is set to 5 seconds). Thus, the presence sensing is animated as the time passed. Using PresenceViewer, one can easily review the past human presence within our laboratory.

V. DISCUSSION

A. Satisfaction of Requirements

We here discuss how the proposed framework satisfies the requirements mentioned in Section II-B.

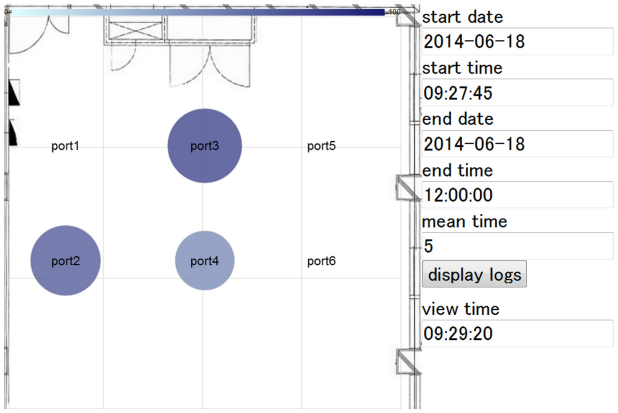

Fig. 6. Screenshot of PresenceViewer

Requirement R1 is satisfied in the proposed framework, since the framework does not rely on specific sensor devices. For any sensor device given, a developer just writes a method of getting sensor value in PSD, defines an appropriate threshold for detecting human movements in PST, and sets an address of a PSA to be notified. Even a smartphone with embedded sensors can be adapted to the framework, by using the sensors as PSDs and the smartphone as a PST.

Requirement R2 is satisfied by the layered architecture of the proposed framework. If a user wants to add a place of presence sensing, the user just installs a new PSD in the place, and connects it to a nearest PST. For this, no modification is required in PSA or PSS. The same thing applies to changing or removing the place of sensing.

Requirement R3 is satisfied by Web service of the PSS. Through the Web-API of the PSS, the presence information can be used by various HNS applications and appliances in a platform-independent manner. Also, as seen in the developed PresenceViewer, it is quite easy to integrate the PSS with other Web services, such as Google Charts, Email, Twitter, SNS.

Finally, Requirement R4 can be satisfied by deploying the PSA and PSS on the external cloud server. The URL of the PSA is not necessary closed within a house. Therefore, by installing PSA and PSS in the external server as the cloud service, the presence information can be collected from multiple houses with HNS environment. However, when gathering human presence of multiple houses within the external server, we must consider security and privacy issues carefully. We would like to leave the security and privacy issues for one of our most important future work.

B. Other Applications and Services

We here consider more other applications and services using the proposed PSSF. Using the presence information and the presence log of the PSSF extensively, the following services can be developed.

1) Daily Life Review Service: Using the PSSF, a user can look back daily life with respect to the human presence. Reviewing the past situation may evoke some idea to improve user's life style, or may detect unexpected intrusions during the absence of the user.

ISBN: 978-1-4799-6375-1 ©2015 IEEE

2) Task Remind Service: This service is used to remind a user of periodic tasks or daily routines using the presence information. For example, when a user forgets to throw out the garbage in the garbage collection day, the service detects the fact from the presence information in the kitchen, and alerts the user to wake up.

3) Time Estimation Service: By analyzing the presence log of the PSSF extensively, we may find the time patterns when people are often present. Based on the log, we can estimate the presence likelihood in the future, which realizes a value-added service. For example, if the service can estimate the time when the family is present a living room, the HNS automatically turns on a light of the living room on the time.

4) Location Estimation Service: Similarly, by analyzing the presence log of the PSSF, we may guess a place where a user moves next. Since a PST usually has multiple PSDs, the order of reactions of the PSDs may characterize user's actions or movements of direction. By analyzing the patterns, the service may derive the next location where the user moves. For example, if the service knows that a user often goes to the second floor when the presence is detected near the stair, then the service automatically turns on the lights of the second floor in advance.

VI. Conclusion

In this paper, we have presented a framework, called Presence Sensor Service Framework (PSSF), for efficient presence sensing within the home network system (HNS). The proposed framework is designed by a layered architecture consisting of the presence sensor devices (PSDs), the presence sensor terminals (PSTs), the presence sensor aggregator (PSA), and the presence sensor service (PSS). A PSD is a physical sensor device for detecting human movements. The PST is a local hub monitoring adjacent PSDs. A PSA collects all sensor information generated in a house. A PSS is a service to provide presence information to HNS applications. Various HNS applications and appliances can obtain presence information easily through Web-API provided by the PSS.

We have also prototyped the proposed framework using Phidgets infrared motion sensors, a Raspberry Pi, and Java, and installed the prototype within the HNS in our laboratory. To confirm the effectiveness, we developed an application, PresenceViewer, which visualizes the human presence in our laboratory. We finally discuss other potential applications and services.

In our future work, we would like to evaluate the performance of the proposed system (e.g., accuracy of human presence and response speed). Also, we investigate the security and privacy issues in presence sensing of multiple houses for emerging smart city services.

Acknowledgement

This research was partially supported by the Japan Ministry of Education, Science, Sports, and Culture [Grant-in-Aid for Scientific Research (C) (No.24500079, No.24500258), Scientific Research (B) (No.26280115), Young Scientists (B) (No.26730155)] and Kawanishi Memorial ShinMaywa Education Foundation.

References

[1] SONY, "Presence sensor," http://docs.esupport.sony.com/referencebook/en/ve5/pages/funfeatures/presencesensor.html, [Online; accessed 11-Dec-2014].

[2] HITACHI, "Heat recovery operation," http://www.hitachi-ap.com/products/business/ac/packaged/recovery/operation.html, [Online; accessed 11-Dec-2014].

[3] GIRA, "Automatic light control," http://www.gira.com/en/gebaeudetechnik/produkte/automatische-lichtsteuerung.html, [Online; accessed 11-Dec-2014].

[4] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. ichi Matsumoto, "Constructing home network systems and integrated services using legacy home appliances and web services," *International Journal of Web Services Research*, vol. 5, no. 1, pp. 82–98, January 2008.

[5] G. T. by NABCO Entrances Inc, "About sensors," http://www.nabcoentrances.com/about_sensors.pdf, [Online; accessed 11-Dec-2014].

[6] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 141–154. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000010

[7] freescale, "Pressure sensors," http://www.freescale.com/webapp/sps/site/overview.jsp?code=DRSNSPRSSR&uc=true&lang_cd=en, [Online; accessed 11-Dec-2014].

[8] M. Hazas and A. Hopper, "Broadband ultrasonic location systems for improved indoor positioning," *IEEE TRANSACTIONS ON MOBILE COMPUTING*, vol. 5, no. 5, pp. 536–547, 2006.

[9] P. Tao, A. Rudys, A. M. Ladd, and D. S. Wallachm, "Wireless lan location-sensing for security applications," http://www.cs.rice.edu/~dwallach/pub/wise2003.pdf, [Online; accessed 11-Dec-2014].

[10] Y. Watanabe, M. Nakamura, and S. Matsumoto, "Implementing personalized energy visualization service in home network system," in *14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD2013)*. IEEE Computer Society, July 2013, pp. 529–536, honolulu, USA.

[11] K. Tokuda, S. Matsumoto, and M. Nakamura, "Implementing a mobile application for spontaneous peak shaving of home electricity," in *Sixth International Workshop on Selected Topics in Mobile and Wireless Computing (STWiMob2013)*. IEEE Computer Society, October 2013, pp. 284–289, lyon, France.

[12] M. Nakamura, H. Igaki, Y. Yoshimura, and K. Ikegami, "Considering online feature interaction detection and resolution for integrated services in home network system," in *10th International Conference on Feature Interactions in Telecommunications and Software Systems (ICFI2009)*. IOS Press, June 2009, pp. 191–206, lisbon, Portugal.

[13] S. Soda, M. Nakamura, S. Matsumoto, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Implementing virtual agent as an interface for smart home voice control," in *Asia-Pacific Software Engineering Conference (APSEC2012)*, December 2012, pp. 342–345, hong Kong.

[14] K. Tokuda, S. Matsumoto, and M. Nakamura, "Implementing personal home controllers on smartphones for service- oriented home network," in *International Conference on Wireless and Mobile Computing, Networking and Communications (Wimob 2012)*, October 2012, pp. 777–784, barcelona, Spain.

[15] M. Nakamura, S. Matsuo, S. Matsumoto, H. Sakamoto, and H. Igaki, "Application framework for efficient development of sensor as a service for home network system," in *the 8th IEEE 2011 International Conference on Services Computing (SCC 2011)*, July 2011, pp. 576–583, washington D.C.

[16] M. Nakamura, S. Matsuo, and S. Matsumoto, "Supporting end-user development of context-aware services in home network system," in *Studies in Computational Intelligence*, R. Lee, Ed. Springer, November 2012, pp. 159–170.

[17] JAXA, "Wireless lan location-sensing for security applications," http://global.jaxa.jp/article/special/michibiki/yoshitomi_e.html, [Online; accessed 11-Dec-2014].