2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)

PION: Human Mobility-based Service Provisioning Framework for Smartphone Users

Chanmin Yoon, Yohan Chon and Hojung Cha
Department of Computer Science
Yonsei University, Seoul Korea
Email: {cmyoon, yohan, hjcha}@cs.yonsei.ac.kr

***Abstract*—Context-aware service provisioning for mobile phones is challenging because of diverse user contexts and mobile applications. The context recognition process generally reduces the device performance due to the competitive use of limited resources in the mobile phone. While extensive attempts have been made to provide appropriate services based on user context, previous work is limited to supporting diverse user contexts and various services. In this paper, we introduce PION, a framework for personalized service provisioning to manage diverse user contexts and provide appropriate mobile services in daily life. PION comprises the Service Hub and Pioneer. The Service Hub is a service agent server between a smartphone user and a service provider that defines the properties of mobile services. The Pioneer collects cognitive context data, classifies user contexts and their relations, and predicts essential mobile services based on a user's mobility data. We have implemented PION on the Android framework, and our evaluation demonstrates its efficiency in managing diverse user contexts and providing mobile services in real deployments. We believe that the PION framework is a viable context-aware system for smartphone users.**

***Index Terms*—Context-aware service, Middleware, Service-oriented architecture, Smartphone.**

I. Introduction

The recent widespread dissemination of smartphones is a key feature that has led a wide variety of mobile services to be used in daily life; these include entertainment, weather forecasts, traffic status updates, information about public transportation, and social networks. The rapid development of mobile services has been accelerated by (1) advanced mobile devices with rich-sensors and (2) Internet-of-Things (IoT) infrastructures. Todays mobile phones are equipped with various sensors that are able to determine contextual information related to human life such as location, movement, posture, activities, and behavior patterns. This information represents a fundamental source whereby service providers can infer personal demands and provide appropriate services in a timely manner. In addition, IoT infrastructures (e.g., Wireless Sensor Networks (WSNs) or Machine-to-Machine (M2M) systems) provide a ubiquitous computing environment by sharing information between things and humans [1]–[3]. Consequently, the state-of-the-art technique in mobile computing enables information about humans and environments in daily life to be collected/inferred.

Several studies have considered user context recognition techniques in mobile devices [4]–[9]. We argue that previous works have been limited in terms of practical deployment, since they did not focus on the resource shortage problem that occurs when several applications recognize user context simultaneously. Indeed, smartphone users employ various context-aware applications such as traffic information services, health monitoring, exercise guiding [10], life logging, location tracking [11]–[13], and so on. These applications simultaneously use sensors to recognize a common context (i.e., human mobility) for different purposes. Such processes involve the redundant usage of hardware, causing the quality of the collected information to deteriorate.

The various mobile platforms are burdens for service providers due to the redundant implementation required for different platforms. For example, to provide traffic information to smartphone users, a service provider needs to implement similar applications for various mobile devices. Consequently, users need a new framework to use mobile services on demand without installing an application. Service providers also need a new framework to provide mobile service regardless of the platform of the users device. To meet the requirements for both mobile users and service providers, we propose PION, a service-provisioning framework for smartphone users. The basic concept of PION is on-demand service that provides appropriate functionalities without the installation of superfluous applications. PION is a conceptual architecture that manages user-context information and predicts essential services for smartphone users. We designed the PION framework to achieve the two objectives of user-oriented context awareness and flexibility:

- User-oriented context awareness: The PION framework is a middleware system that manages user-context information to provide user context to various applications. The proposed system detects changes in user context to reduce the redundant processes of context recognition required by multiple applications.
- Flexibility: PION manages various mobile services to provide optimal service in a given situation. We have designed a lightweight service descriptor and matching process between user context and mobile services in order to guarantee the flexibility of service usage.

This paper is organized as follows. Section II describes the motivation for this project and related work, emphasizing the need for our approach. We describe the PION architecture in Section III and explain the user-oriented context-awareness

978-1-4673-1905-8/12/$31.00 ©2012 IEEE

scheme in Section IV. We evaluate PION with a case study in Section V and conclude the paper in Section VI.

II. Related Work

Active research has been conducted on context-aware systems and applications in diverse domains [14]. Although many context-aware systems have been proposed, we argue that previous work has basically two limitations in real deployment: (1) The lack of a general framework for a wide variety of domains; and (2) the strict dependency on specific infrastructures designed for specialized applications.

References [15] and [16] provide living assistance and residential monitoring services based on the users activities and status. The system uses a WSN infrastructure to capture user context. AlarmNet [15] provides a separate role-based application for different users: patients, doctors, and nurses. The system defines a restricted context model according to the user's role. Thus, the system is not flexible in terms of supporting additional services. In our work, we have focused on the flexibility and adaptability of mobile services and user context.

Most context-aware systems focus on the physical context: location, sound, temperature, humidity, and so on. However, personalized mobile services can be greatly improved by utilizing cognitive user context. The cognitive user context includes user intention and high-level context. For example, a users current location with latitude and longitude is a physical context, while the user is on his way home from work is a cognitive context. In other words, cognitive context includes the semantics of human behavior. UPCASE [7], [8] introduced a context-awareness system through a decision-tree-based learning method. The system supports personalized high-level context recognition. In PION, we are concerned with cognitive user context rather than physical context to provide personalized service.

The challenge of a context-aware system is the simultaneous support of multiple context-aware applications. The greedy requests of context recognition by multiple applications may significantly aggravate the accuracy of context information. Such a scheme also reduces system performance due to the redundant resource usage. To solve this problem, the middleware should manage user context at the system level to share the common context among multiple applications. Orchestrator [17] proposed a resource management at the system level to recognize changes in user context and provide appropriate services under the personal-area-network environment. In PION, we continuously monitor user context without additional infrastructures to provide on-demand services based on user context and preference.

III. The PION Architecture

We designed the PION architecture to provide personalized service to mobile users. Fig. 1 describes the overall concept of the PION framework, which consists of two parts: Service Hub and Pioneer. The Service Hub is an agent server between a mobile user and smart spaces (M2M infrastructures). The roles of the Service Hub are (1) to define various services in smart spaces, and (2) to search proper services by matching service requirements and context information from the users devices. Pioneer is a middleware group comprising a two-layered structure: the Service Agent Layer (SAL) and the User-context Awareness Layer (UAL). SAL analyzes the user-context information from the UAL to request service provisioning to the Service Hub.

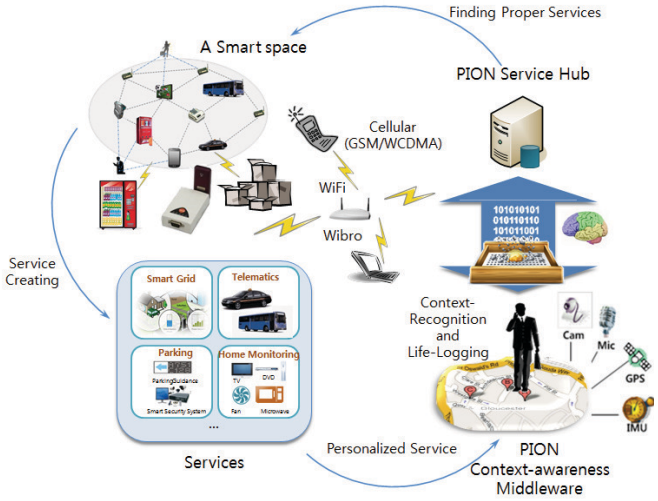


Fig. 1. The overall concept of the PION framework architecture

A. The Service Hub

Fig. 2 illustrates the overview of the Service Hub. The Service Hub is a service agent server between a smartphone user and service providers. It collects and manages the information about services from the service provider through its gateways (e.g., global Internet or intra-net of the smart space).

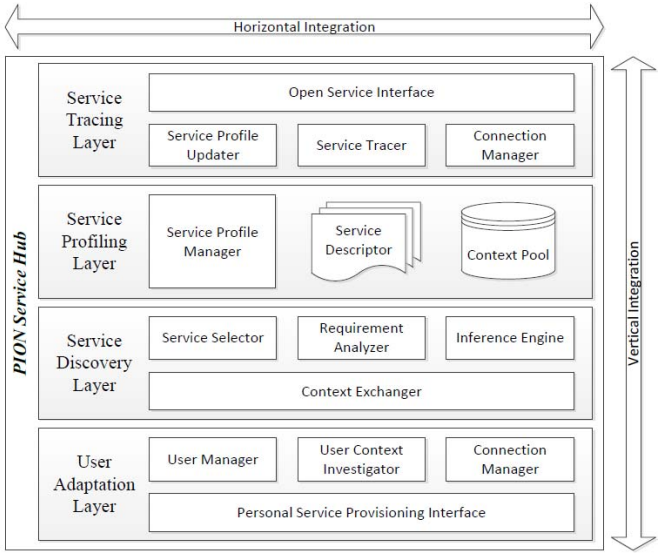


Fig. 2. The overview of PION's Service Hub

The Service Hub defines service profiles using predefined Service Descriptors (see Fig. 6). It searches a list of the most appropriate services when it receives a service-provisioning request. The Service Hub has a four-layered architecture; each layer is designed without functional overlap. In the following, we explain the details of each layer.

1) Service Tracing Layer: The Service Tracing Layer manages the connection between the Service Hub and the smart spaces. In this layer, the PION framework supports Open Service Interface as API to service providers. The Connection Manager is designed (1) to keep the network connection to the gateway of the smart space and (2) to confirm the connection status of each service. The role of the Service Tracer is to check the availability of each service via the Connection Manager. The Service Tracer reports the change of availability of each service to the Profile Updater if it detects any change in a certain service status. Then, the Profile Updater renews the service profile.

2) Service Profiling Layer: The Service Profiling Layer records the properties of the individual service in the smart space with the Service Descriptor. The recorded Service Descriptors are delivered to the Service Dispatcher. The Service Descriptor is the predefined model for the individual service. Each Service Descriptor defines one service and refers to the Context Pool. The Context Pool is a set of collected user contexts, and it is used to search for the relevant service. We describe the Context Pool in Section IV.

3) Service Discovery Layer: The key role of the Service Discovery Layer is to construct a personalized service list by comparing the contextual requirements of the service with user-context information. The Service Selector requests the service descriptor list from the Service Profiler when a user asks for service provision. The Requirement Analyzer examines the contextual requirement of each Service Descriptor and compares the analysis result with the received user-context information. The Requirement Analyzer commands the Context Requester to request additional contextual information if the user-context information is not sufficient to analyze the contextual requirement. Otherwise, the Requirement Analyzer entrusts the Inference Engine to determine the conformance of service for the current user-context. According to the judgement of the conformance, the service list is prepared according to a priority. The module generates the priority by using frequency and history of service usage.

4) User Adaptation Layer: The User Adaptation Layer manages the information of the connected user and maintains the connection until the end of service provisioning. The User Context Investigator validates all of contextual information. Then, the User Context Investigator sends this information to the Service Dispatcher when all required information is valid.

B. The Pioneer Middleware Group

Pioneer is a middleware group on the user smartphone of the PION framework. Fig. 3 shows the overview of Pioneer. It comprises two layers: the Service Agent Layer (SAL), called ServOn, and the User-context Awareness Layer (UAL), called LifeMap. ServOn collects Fragmentary Contextual Data (FCD) provided by LifeMap and requests essential contextual data from LifeMap, if additional information is required to search a certain service. ServOn requests service provision to the Service Hub server by examining the FCD. We describe the FCD in Section IV. If the Service Hub succeeds in finding proper service, ServOn provides an appropriate service with a different interface depending on each service type. LifeMap is a user context-aware middleware that can assist in human-centric mobile sensing [11]. The major role of LifeMap is to automatically collect a users mobility data, specifically a time-resolved path and places with room-level accuracy in real-time.

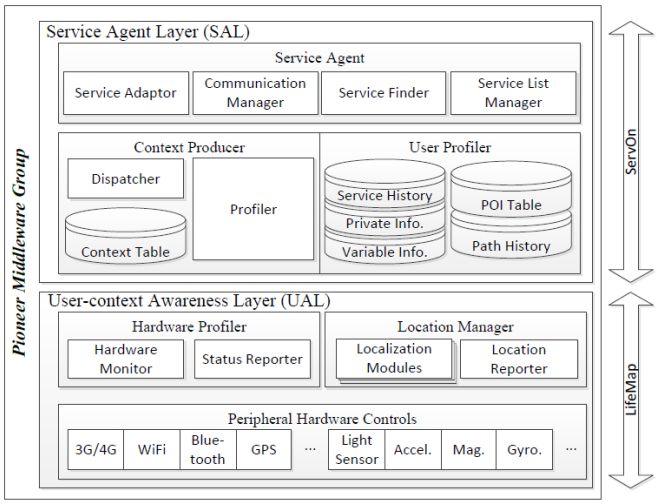


Fig. 3. The overview of user-context awareness middleware, Pioneer

1) The ServOn middleware in the Service Agent Layer: As the top layer of the Pioneer, ServOn has two major roles. The first is context recognition by collecting the simplest user-context information provided by the UAL. For example, ServOn retrieves the FCD, such as the user is located outdoors or indoors or the user is at the workplace or at home by using the availability of the GPS signal or the sum of residence time, respectively. In short, the system obtains the FCD by using the composite context information.

The second role of SAL is to detect the need of the service request. In Pioneer, SAL supports two types of service request: the Direct Service Request (DSR) and the Indirect Service Request (ISR). A DSR is an intentional service request by a user. When a DSR is triggered by user interaction, ServOn tries to recognize the current user context by compounding collected FCDs, and sends the service request to the Service Hub. ISR is an automatic service request by ServOn. ServOn tries to determine the need of ISR if it detects a change in user context. The system uses an event-driven method between UAL and SAL to determine whether the ISR is necessary or not. In order to handle DSRs and ISRs, ServOn organizes the collected FCDs in User Profiler. TABLE I describes five types of datum in User Profiler.

The Context Producer uses the FCD to predict current user context. The Profiler in Context Producer assembles

TABLE I
DESCRIPTION AND EXAMPLE OF FIVE TYPES OF PROFILE DATUM STORED IN USER PROFILER

Type	Description	Example
Service History	Information related to service used by user	SERVICE: Seoulbus(13110) LATEST POI: Workplace PATH ID: 143 TIMESTAMP: 2011041182556FRI
Personal Information	Personal information recorded by user	BIRTHDATE: 19820422 GENDER: Male MARRIED: Y
Variable Information	Personal information generated by Pioneer	MOST VISITED POI: Home MOST PLAYED SONG: Telephone(Lady Gaga)
Point-of-Interest(POI) Table	Recognized POI list by LifeMap	POI NODE ID: 53 NAME: Home ADDRESS: N/A ACCURACY: $<$ 50 m
Path History	User mobility information from a POI to another POI	PATH ID: 143 PATH: Workspace to Home TRIP DISTANCE: 6.3 Km DEPARTURE: 20110401102021FRI

the recent profile data in the User Profiler to produce a compounded high-level user context. For example, if UAL detects a departure event from a Point-of-interest (POI), the Context Producer assembles the latest POI information and the user's private information to predict the current user context. If it is determined by the prediction result that the user may need a service, the Dispatcher commands the Service Agent to send a service provisioning request message.

The Service Agent requests and receives the service from Service Hub. Service Agent has the following four components: Communication Manager, Service Finder, Service List and Service Adaptor. The Communication Manager manages the physical network connection and communication protocol. The Service Finder processes ISRs and DSRs. The Service List processes and stores the service provisioning results provided by the Service Hub. The Service List shows the result to user and waits for the user selection if the Service Hub provides several candidate services. The selected service is sent to the Service Adaptor. The core functionality of the Service Adaptor is to provide a proper interface for the selected service. The user interface type is defined in the Service Descriptor. PION supports three types of user interface: web based, list view based, and application based. The Service Adaptor uses different interfaces to provide services: a web browser for web based a list view screen for list view based, and application execution for application based.

2) The LifeMap middleware in the User-context Awareness Layer: LifeMap [11], [18] is a personalized location provider that collects a users mobility data in daily life. The major role of LifeMap is to trace a users location in everyday life and automatically identify POIs with room-level accuracy. The system uses a Wi-Fi fingerprinting method to aggregate identical POIs. To minimize energy consumption for continuous monitoring, LifeMap uses an activity-based sensor selection scheme to activate power-intensive sensors, such as GPS, only if the user is moving or the estimated location includes a considerable amount of uncertainty. In other words, LifeMap activates a minimum set of sensors to generate location information within the required accuracy. The detailed technical description of the locating scheme is available in [18]. In Pioneer, LifeMap directly controls the hardware to generate and provide user mobility data. In other words, LifeMap provides contextual data related to mobility of three types: entrance to a POI, departure from a POI, and path information between POIs. Then, ServOn processes such data to manage a POI table and a path history in the User Profiler.

IV. USER-ORIENTED CONTEXT AWARENESS

Pioneer detects user-context change as an event through continuous monitoring. Based on the detected event, Pioneer predicts cognitive user context, then sends the ISR message to the Service Hub. Fig. 4 shows an example process of user-context prediction in Pioneer. PION provides personalized services according to the result of user-context monitoring. We define this method as user-oriented context awareness. The main challenge of personalized service provisioning is how to find the best match between user context and service requirements. To solve this challenge, we have designed the Context Pool, a specialized structure in the Service Hub (see Fig. 2).

A. Fragmentary Contextual Data

Pioneer collects hardware context to infer current user context in the context prediction process (see Fig. 4). For example, if a GPS signal is not available, Pioneer may infer that the user is located in an indoor environment. FCD is defined as any semantic of the current user context, including the hardware context. The Service Hub defines the required

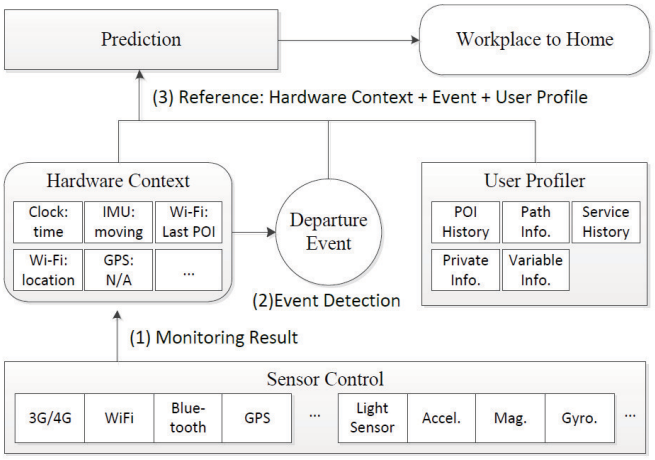


Fig. 4. An example process of user-context prediction in Pioneer

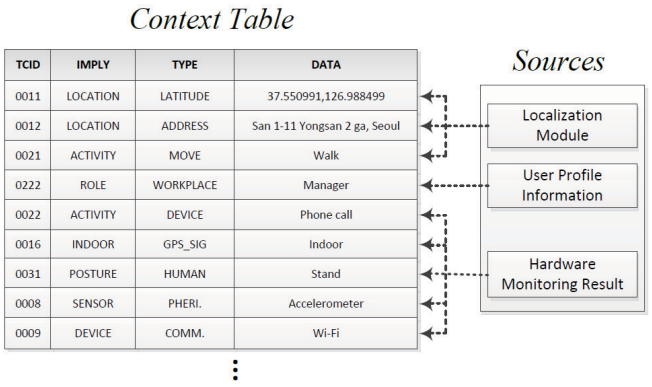


Fig. 5. An example of a Context Table snapshot

FCD in the Context Table and Pioneer collects all FCDs in the Context Table. Fig. 5 gives an example of a Context Table snapshot and the source of each FCD. As shown as Fig. 5, every FCD has a Type of Context ID (TCID) that is allocated identically by a combination of IMPLY and TYPE. TCID implies what kind of information it contains. TCID should be equal in the Context Pool.

B. Context Pool

The Context Pool is a context model to define FCDs and their relations. The Context Pool is classified into the Components Group and the Relations Group. We designed TCID to define context type in the Components Group. The Relations Group is the definition of relationship between user contexts. We use the Relationship between Contexts ID (RCID) to define relations. The Service Descriptor has a list of service requirements related to essential user context. We denote this requirements list as ‘‘CONDITION_LIST’’. Fig. 6 shows an example of the Service Descriptor and its service requirement definition in the Context Pool. This Service Descriptor, for example, describes an indoor Air Conditioner Control service. The CONDITION_LIST in this descriptor states several relations for the relevant service.

C. User-context Classification in ServOn

The Context Producer updates user-context information in the Context Table when ServOn detects an update in the User Profiler. In this case, the FCD provided by UAL is stored in the Context Table with TCID. The Context Producer updates its context information by changing the User Profiler. The Service Hub, however, could request the context collection if required information is missing in the Context Table. For example, in Fig. 6, if the Requirement Analyzer failed to acquire information about “C3) Is the user indoors?” the Service Hub would send a request about C3 with the related TCID. Then, the Service Agent would collect relevant context data in the Context Table and transmit them to the Service Hub. This process reduces the overhead of inquiring into the user-context information.

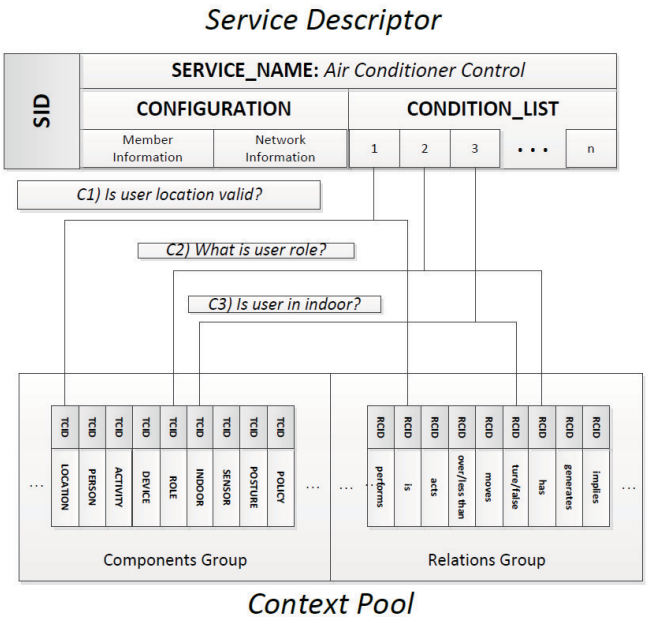


Fig. 6. An example of Service Descriptor and Context Pool

V. Evaluation

We implemented the PION framework using Java and Android SDKs. In the case of the Service Hub, we use the Java SDK in the Ubuntu Desktop environment. The Pioneer middleware group is implemented on the Android platform. We tested Pioneer on Android phones equipped with several built-in sensors such as an accelerometer, GPS, Wi-Fi, and GSM/CDMA.

A. Efficiency of the Service Hub

We evaluate the feasibility of the Service Hub by demonstrating three types of services: real-time bus information service, subway timetable, and campus cafeteria menu service. We registered these services to the Service hub and each service is provided by the ISR based on context prediction in Pioneer. The core process of context-aware applications is similar. For example, in the case of the transportation information service, a user needs to install several applications according to the transportation type and the service area. Such a mechanism is a burden to both users and application developers, since the core functions are the same: that is, sending a request about the location and transportation type, and receiving arrival time information. The PION framework provides the core service functions to match the user context and required services without a costly application development process. In the case of the bus information service, the proposed framework needs only 7,465 bytes of Java code to define the service descriptor and implement the core function.

B. Efficiency of the Pioneer Middleware Group

Pioneer reduces the development cost for context recognition. In the aforementioned cases, separate application installation is unnecessary in Pioneer since it recognizes the required

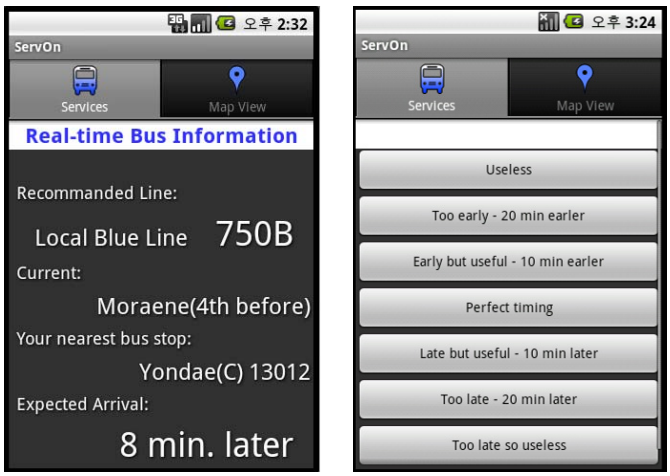
context and shares it with three services. Additionally, Pioneer reduces the intentional manipulation for the use of service. For example, without the Pioneer framework, a user needs to install three applications (e.g., SeoulBus [19], Subway Arrival Information [3], and YonseiApp [20]) to use three services. Such applications require 6.7 MB space in total, while Pioneer requires 3.3 MB (ServOn and LifeMap), making it 50% more efficient in terms of space. Thus, the proposed approach is efficient, as well as being an expandable method, considering that Pioneer is easily applicable to other services. To provide appropriate services based on user context, we have defined the high-level context derived from the User Profiler, as described in Section IV-C.

C. Case Study: PION with Real-time Bus Information Service

The results of the previous section indicate that the proposed framework is cost-effective to define diverse user context and mobile services. In this section, we demonstrate the end-user scenario with PION to evaluate the overall performance in real deployment. Among the various mobile services, we chose a public transportation application that provides the arrival time of buses at a selected stop. We focus on a commute between ''home'' and ''workplace,'' which is the most frequent transition in daily life. PION provides the arrival time of the bus when a user departs from home/workplace to go to the workplace/home. To provide such service in a timely manner, PION needs to generate additional high-level contexts: (1) inferring home/workplace among many POIs, and (2) predicting the destination of departure behavior. ServOn first chooses home and workplace among the generated POIs in the User Profiler. In human life, home is the most visited place and workplace is probably second. Thus, ServOn easily recognizes the home and workplace by using the sum of residence time in each POI. Such a method correctly selected the home and workplace of all participants in the collected data. Here, l_{T1} denotes the most visited location (home) and l_{T2} denotes the second most visited location (workplace). ServOn predicts the commute probability P_t that indicates the possibility of transition between l_{T1} and l_{T2} at time t. To compute P_t, we use an order-1 Markov predictor that estimates the next location from a current location. Accordingly, we only consider a direct transition between l_{T1} and l_{T2} without intermediate locations. Let s_i be the i_{th} staying-behavior that comprises arrival time a_i, departure time d_i, and location l_i. Then, $\{s_1, s_2, \ldots, s_{t-1}, s_t\}$ is the observed squence of staying behavior, where s_t is the current staying behavior. If the current location l_t is equal to l_{T1} or l_{T2} and the LifeMap detects the departure event at time d_t in l_t, P_t is computed as:

$$P_t = \frac{\sum_{i=1}^{t-1} f_{com}(s_i, s_{i+1})}{\sum_{i=1}^{t-1} f_{all}(s_i, s_{i+1})}, \text{ where}$$

$$f_{all}(s_i, s_{i+1}) = \begin{cases} 1 & \text{if } l_i = l_t \text{ and } |d_i - d_t| < 10\text{min} \\ 0 & \text{else} \end{cases},$$



(a) Service screen (b) Evaluation interface

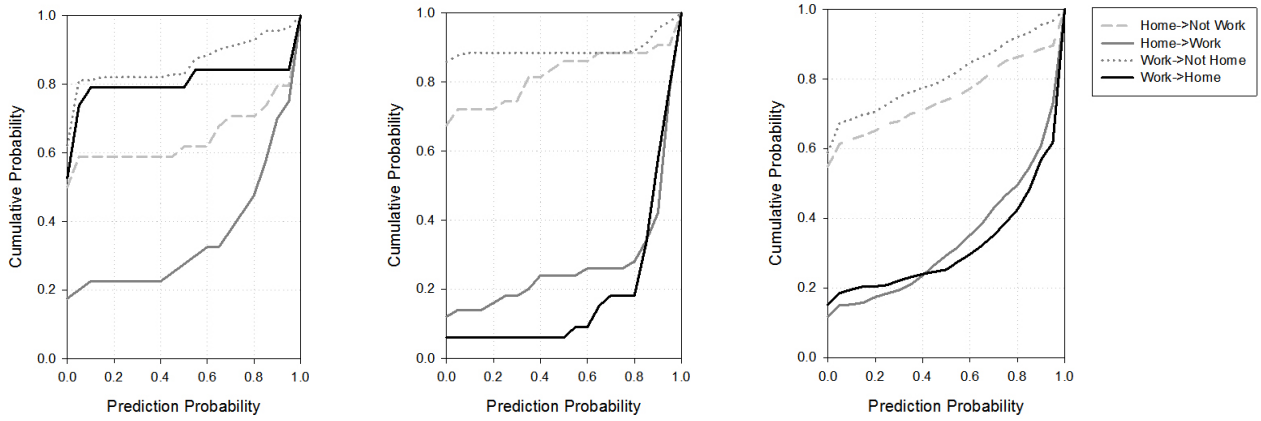
Fig. 7. Screenshots of ServOn: (a) is an example of a list view-based service screen for real-time bus arrival information service and (b) is the evaluation interface.

$$f_{com}(s_i, s_{i+1}) = \begin{cases} 1 & \text{if } l_i = l_t \text{ and } (l_{i+1} = l_{T1} \text{or } l_{T2}) \\ & \quad \text{and } |d_i - d_t| < 10\text{min} \\ 0 & \text{else} \end{cases}$$

In short $f_{all}(\cdot)$ is the sum of departure events and $f_{com}(\cdot)$ is the sum of commute events within 10 minutes time difference at the current departure time. A 10-minute duration is empirically defined to reflect the variance of human behavior. Thus, P_t indicates the ratio of commute events to departure events. For example, if a user departed from home 10 times at 7 a.m. and he/she went to a workplace 7 times and other places 3 times, then P_t at 7 a.m. is 0.7 in the home.

The experiment was performed with six volunteers over 30 days using an HTC Desire and Samsung Galaxy S. Pioneer was running as a background service in participants' primary phones. Pioneer sends an ISR message to the Service Hub when the system detects commute events. Then, the Service Hub makes a candidate service list and returns it to ServOn, which provides appropriate services or shows the candidate services list as a recommendation. In the collected data, there were 501 transitions from home to workplace among 1,111 departure events from home, and 476 transitions from workplace to home among 1,993 departure events from the workplace. To evaluate the service provisioning result, we implemented the evaluation screen after every service provisioning. Fig. 7 shows the interface screens. Each volunteer evaluated the suitability of the provided service using seven predefined levels, as illustrated in Fig. 7b. ''Useless'' includes a wrong service provision and wrong departure detection. The evaluation scores are divided into 10-minute intervals. The scoring could be different according to the volunteers subjective point of view.

The accuracy of the commute prediction is shown in Fig. 8. The result indicates that the system estimates a high probability of commute events (home to workplace or vice versa) at



(a) The worst case (b) The best case (c) All of users

Fig. 8. Result of user-context prediction based on path history in Pioneer (see Fig. 3): (a) show the worst case, (b) shows the best case, and (c) shows prediction probability of all users

67±36%, while the low probability is shown as 20±32% in the rest of the transition events. The large variance is derived from the difference in regularity of each participants life pattern. Fig. 8a and 8b show the worst case and best case results, respectively. In Fig. 8a, the ''workplace to home'' transition shows a slightly higher probability than ''workplace to not home'' transitions, since the quitting time of a participant is irregular. Thus, we used a personalized threshold to cover diverse life patterns.

Fig. 9 shows the histograms of the evaluation results. The red bars in Fig. 9 show the result of all service provisioning cases. A total of 226 service provisioning cases were graded as ''Proper Timing.'' This result corresponds to about 56% among all 405 ISRs. The ''Useless'' case is the second largest case (i.e., 93 cases). This result is reasonable because it includes false-positive cases, where not only wrong service provision but also wrong departure detection occurred. The false-positive case could be serious, since PION would waste system resources and bother the user. The yellow bars in Fig. 9 show the result of providing only the ''real-time bus information service.'' This service was provided a total of 242 times by the prediction results of ''home to workplace'' and ''workplace to home.'' In this case, 60% of cases (i.e., 146 cases) were graded as ''Proper Timing.'' The service provisioning result received a better evaluation than other services, since the prediction probability was relatively high.

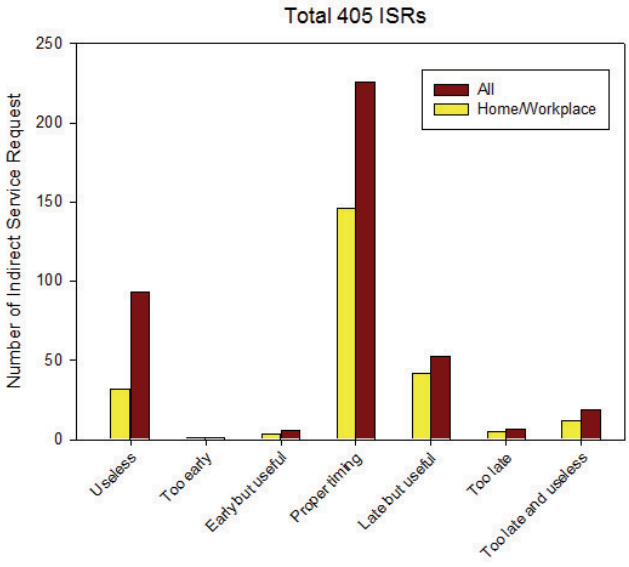


Fig. 9. Evaluation result of six users. The red bar shows result of service provisioning by all 405 ISRs processing during a 30-day evaluation period. The yellow bar shows the result by detecting user movement from home to the workplace and the workplace to the home (242 ISRs).

VI. Conclusion

PION is a service provisioning framework for smartphone users. The system supports service provisioning based on human mobility by archiving user POI information. We designed the PION architecture as a layered structure that is expandable for other context-aware services. We proposed an efficient service descriptor and context model to define diverse mobile services and user context. The conducted evaluation indicates that PION reduces the overhead of context recognition and application development cost in practical deployment. In our case study, a total of 60% of the service provided was evaluated as ''proper.'' We believe that the PION framework frees smartphone users from the redundant hardware usage required by multiple context-aware applications. Further research into energy-efficient context-monitoring schemes, improvement of prediction algorithms, and sensor-control techniques could make PION a viable context-aware system for smartphone users.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government, Ministry of Education, Science and Technology under Grant (No.2011-0015332).

REFERENCES

[1] W. Tobiska, G. Crowley, S. Oh, and M. Guhathakurta, "Space weather gets real-on smartphones," *Space Weather*, vol. 8, no. 10, 2010.

[2] "Android market:nextbus," http://www.nextbus.com/homepage, 10 Accessed 15 July 2011.

[3] "Android market:subway arrival information," https://market.android.com/details?id=com.vtron.subway, 10 Accessed 15 July 2011.

[4] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*, ser. MobiSys '09. New York, NY, USA: ACM, 2009, pp. 165–178. [Online]. Available: http://doi.acm.org/10.1145/1555816.1555834

[5] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th annual international conference on Mobile computing and networking*, ser. MobiCom '09. New York, NY, USA: ACM, 2009, pp. 261–272. [Online]. Available: http://doi.acm.org/10.1145/1614320.1614350

[6] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen, "Contextphone: A prototyping platform for context-aware mobile applications," *IEEE Pervasive Computing*, vol. 4, pp. 51–59, 2005.

[7] A. Santos, L. Tarrataca, J. Cardoso, D. Ferreira, P. Diniz, and P. Chainho, "Context inference for mobile applications in the upcase project," *MobileWireless Middleware, Operating Systems, and Applications*, pp. 352–365, 2009.

[8] A. Santos, J. Cardoso, D. Ferreira, P. Diniz, and P. Chaínho, "Providing user context for mobile and social networking applications," *Pervasive and Mobile Computing*, vol. 6, no. 3, pp. 324–341, 2010.

[9] D. Figo, P. Diniz, D. Ferreira, and J. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 645–662, 2010.

[10] N. Project, "Nike+ project," http://www.nikerunning.nike.com/nikeplus, 10 Accessed 15 July 2011.

[11] J. Chon and H. Cha, "Lifemap: A smartphone-based context provider for location-based services," *IEEE Pervasive Computing*, pp. 58–67, 2011.

[12] M. Lee, A. Khan, and T. Kim, "A single tri-axial accelerometer-based real-time personal life log system capable of human activity recognition and exercise information generation," *Personal and Ubiquitous Computing*, pp. 1–12.

[13] K. Hwang and S. Cho, "Landmark detection from mobile life log using a modular bayesian network model," *Expert Systems with Applications*, vol. 36, no. 10, pp. 12 065–12 076, 2009.

[14] J. Hong, E. Suh, and S. Kim, "Context-aware systems: A literature review and classification," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.

[15] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, and J. Stankovic, "Alarm-net: Wireless sensor networks for assisted-living and residential monitoring," *University of Virginia Computer Science Department Technical Report*, 2006.

[16] A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru, "Context-aware wireless sensor networks for assisted living and residential monitoring," *Network, IEEE*, vol. 22, no. 4, pp. 26–33, 2008.

[17] S. Kang, Y. Lee, C. Min, Y. Ju, T. Park, J. Lee, Y. Rhee, and J. Song, "Orchestrator: An active resource orchestration framework for mobile context monitoring in sensor-rich mobile environments," in *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*. Ieee, 2010, pp. 135–144.

[18] Y. Chon, E. Talipov, and H. Cha, "Autonomous management of everyday places for a personalized location provider," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, no. 99, pp. 1–14.

[19] "Android market:seoulbus," https://www.android.com/details?id=com.astroframe.seoulbus, 10 Accessed 15 July 2011.

[20] "Android market:yonseiapp," https//market.android.com/details?id=net.yutar.ysapp, 10 Accessed 15 July 2011.