Research Article

Path Prediction Method for Effective Sensor Filtering in Sensor Registry System

Sukhoon Lee, 1 Dongwon Jeong, 2 Doo-Kwon Baik, 1 and Dae-Kyoo Kim 3

1 Department of Computer and Radio Communications Engineering, Korea University, 1, Anam-dong 5-ga, Seongbuk-gu, Seoul 136-701, Republic of Korea
2 Department of Statistics and Computer Science, Kunsan National University, 558 Daehangro, Gunsan, Jeollabuk-do 573-701, Republic of Korea
3 Department of Computer Science and Engineering, Oakland University, 2200 N. Squirrel Road, Rochester, MI 48309-4401, USA

Correspondence should be addressed to Dongwon Jeong; djeong@kunsan.ac.kr and Doo-Kwon Baik; baikdk@korea.ac.kr

Received 30 January 2015; Revised 29 May 2015; Accepted 10 June 2015

Academic Editor: Antonino Staiano

Copyright © 2015 Sukhoon Lee et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The Internet of Things (IoT) has emerged and several issues have arisen in the area such as sensor registration and management, semantic interpretation and processing, and sensor searching and filtering in Wireless Sensor Networks (WSNs). Also, as the number of sensors in an IoT environment increases significantly, sensor filtering becomes more important. Many sensor filtering techniques have been researched. However most of them do not consider real-time searching and efficiency of mobile networks.

In this paper, we suggest a path prediction approach for effective sensor filtering in Sensor Registry System (SRS). SRS is a sensor platform to register and manage sensor information for sensor filtering. We also propose a method for learning and predicting user paths based on the Collective Behavior Pattern. To improve prediction accuracy, we consider a time feature to measure weights and predict a path. We implement the method and the implementation and its evaluation confirm the improvement of time and accuracy for processing sensor information.

1. Introduction

The Internet of Things (IoT) has emerged with the advance of networks and embedded software technologies. The IoT is a novel paradigm that is rapidly growing and has a significant influence over application domains such as telecommunications, transportation, and healthcare [1, 2]. Environmental monitoring and context-awareness technologies are required for devices to be connected to each other for communication in the IoT. Particularly, abundant IoT services can be provided in the application domains where various sensors are used. In the IoT paradigm, a sensor network plays an important role as a critical and indispensable infrastructure to provide richer services to users.

In the IoT paradigm, several issues are considered in the Wireless Sensor Networks (WSNs) area. The first issue is registering and managing sensors as the number of sensors increases explosively. To address this, the sensor web has been developed, which enables sharing and browsing sensor data through the web [3]. OGC Sensor Web Enablement (SWE) is a representative specification of platforms that provide sensor information and sensor data on the web [4]. SensorMap [5] and Sensorpedia [6] are significant researches based on SWE. Recently, SensorCloud [7] and OpenIoT [8] have been researched to manage sensors and to provide sensor data using a cloud-computing infrastructure. Through the research, users can receive sensor data wherever needed by mobile device.

The second issue is semantic interpretation and processing. Several works present semantics of sensors [9]. Semantic Sensor Web enhances semantics by adding simply structured sensor metadata to the sensor web [10]. Adding semantic information such as time, space, and theme to sensor metadata enables interpretation of sensor semantics and processing various sensor data by inference [11]. In particular, Semantic Sensor Network Ontology (SSNO) developed by W3C can represent sensor information in various perspectives [12].
The third issue is searching and filtering sensors. Sensor filtering is based on searching specific sensors in the aforementioned platforms for registering sensors such as SensorCloud and OpenIoT. Also, sensor ontologies are used to recognize user contexts and provide user-centric services. Searching specific sensors especially in a large number of entities requires effective sensor filtering technologies. Linked Sensor Middleware (LSM) is a web-based sensor management platform connected to the Semantic Web [13]. LSM uses sensor information such as sensor type and location for searching sensors. Mayer et al. proposed a method for searching sensor information by location using web-based structures of a building [14]. Perera et al. proposed a context-awareness sensor searching technique [15]. Using SSNO, they search relevant sensors for the given context by setting weights of five features including accuracy, reliability, energy, availability, and cost.

The existing sensor searching techniques provide context-aware services including tracking user location. However, these techniques do not consider architectural problems caused by real-time searching or efficiency of mobile networks. Considering applications with location-based services such as iBeacon [16], the existing sensor searching techniques are not sufficient for real-time reaction when the mobile device receives information from a sensor close to the user. Because the mobile device needs to receive sensor information whenever it is requested, the mobile network QoS influences services to be provided successfully to the user.

In this paper, we present a path prediction approach for effective sensor filtering. This approach predicts user paths based on the user location and provides sensor information located nearby the predicted paths to a mobile device beforehand. The path prediction approach can handle real-time situation or mobile network status. The presented method identifies the location of a user using predefined road information and predicts user paths using a simple path prediction algorithm. To implement the method, we use Sensor Registry System (SRS) [17] which registers and shares sensors. For effective sensor searching, the system provides sensor information by tracking the user, predicting user paths, and identifying sensors located near the predicted paths. We also analyze user patterns using a time feature to validate prediction accuracy.

The remainder of the paper is organized as follows. Section 2 describes the problem being addressed in this paper and gives an overview of the presented approach for solving the problem. Section 3 describes the path prediction method for SRS. Section 4 presents the implementation and experiment of the method and Section 5 gives an overview of the existing path prediction algorithms. Section 6 evaluates the method in comparison to the existing work. Section 7 concludes the paper.

2. Problem Definition and Solution Approach

This section discusses several problems in sensor filtering. As a solution for the problems, we present a sensor registering and sharing system, Sensor Registry System (SRS), which is able to search and filter sensors. Then, we extend SRS based on path prediction to resolve the problems.

2.1. Problem Definition. The existing sensor filtering techniques are not suitable for real-time synchronization. Sensor platforms such as SensorCloud and OpenIoT have several problems when they have to support real-time sensor search in the mobile computing environment. One problem is that the performance of personalized sensor search is low because of limited mobile resources. This mandates the sensor filtering process to be worked out on the server side instead of the mobile device. The next problem is that a mobile device using the sensor platforms has to allocate a large amount of resources for requesting and receiving sensor information. This problem makes service process slow when the user wants to provide some services. Therefore, the sensor platforms should send sensor information before request and mobile device immediately uses the information needed. Another problem is that mobile devices are sensitively affected by the mobile network QoS. If a user moves into an unstable network connection area, the mobile device of the user cannot access sensor platforms and, thus, is not able to receive the necessary sensor information. Therefore, the sensor filtering process should recognize the user context, predict needed sensors for the user, and receive sensor information in advance.

To detail the last problem, Figure 1 shows an example of an unstable network connection status situation in the UK network coverage map provided by the EE coverage checker [18]. In Figure 1(a), the green area indicates the 4G mobile network coverage, the purple area indicates the 3G mobile network coverage, and the pink area indicates the 2G mobile network coverage. Figure 1(b) illustrates an example of success and failure in obtaining sensor information when a user moves into an unstable network connection area. Suppose that a user is moving and communicating with the sensor platform in real-time using a 4G mobile device (which is not supported in a 3G mobile communication system). When the user moves along the blue arrow in the 4G coverage area in Figure 1(b), the mobile device successfully connects to the sensor platform and obtains the sensor information. However, when the user moves into the 3G coverage area following the red arrow, the mobile device fails to obtain the sensor information around the user.

This paper proposes a sensor filtering method that can provide necessary sensors when a mobile device requires sensor information to the sensor platform in real-time while predicting the user paths and sending sensor information nearby predicted paths to the mobile device using the sensor platform.

2.2. Path Prediction Approach. This paper aims at improving sensor filtering in Sensor Registry System (SRS) [17], which is a sensor platform for registering and sharing sensors. SRS was proposed for semantic interoperability between sensors and devices in a heterogeneous sensor network environment [19]. Based on ISO/IEC 11799 [20], SRS registers and manages sensor information. SRS also shares and provides metadata and sensor information such as locations, units, types, and
other relevant information (e.g., manufacturer information, installation organization information). SRS enables a mobile device to instantly and directly interpret and process sensor data from heterogeneous sensors.

A primary feature of SRS is that it allows a mobile device to access the system through the Internet and obtain directly sensor data from a sensor network. The mobile device changes its location and communicates with different sensors as the user of the device moves to a different sensor network. However, mobile devices can obtain only raw data from sensors. To address this, SRS provides sensor information for mobile devices to process semantics of raw data.

In this work, we transmit sensor information from multiple sensors simultaneously for rapid services specific to mobile devices. Because SRS receives a request to return the sensor information of one sensor, the mobile device preloads the sensor information of near sensors from the sensor information set created by the proposed approach. Thus, the mobile device can immediately use the sensor information when it is required. This approach is enabled by recognition of user patterns and sensor filtering in advance.

The approach collects user locations, learns movement patterns, predicts user paths, and preloads sensor information of the sensors located near the predicted paths. The mobile device synchronizes with SRS and transmits the current location to SRS. Then, SRS predicts a user path based on the current location and transmits the sensor information set of the sensors located near the predicted path to the mobile device. This enables the mobile device to process services using the preloaded sensor information, even if the user moves into an unstable network connection area.

Figure 2 shows SRS architecture extended by the presented approach. It consists of Sensor Filtering Module, Sensor Information Management, Path Prediction DB, and Sensor DB. The Sensor Filtering Module involves user location monitoring, path prediction, and path and sensor matching. The mobile device of a user constantly accesses SRS, which allows SRS to monitor the user location. If the user changes his/her location, SRS predicts the moving path and collects the identifiers of the sensors located near the user. The Sensor Filtering Module connects to the Path Prediction DB and exchanges related data during the process. In Sensor Information Management, the collected sensor identifiers are used to search for sensor information. The sensor information acquired from Sensor DB is returned to the user after SRS creates a sensor information set. In this work, we focus on path prediction for effective sensor filtering in SRS. We also define a time feature for accurate prediction results and evaluate the prediction accuracy.

3. Path Predication Method

This section presents the path prediction method for sensor filtering in SRS. The method is composed of a path identification method and a path prediction algorithm. We first define a set of variables used in the method and then discuss time-based predication.

3.1. Path Prediction Process. Figure 3 illustrates an overview of the path prediction process. Predication involves a set of
locations for \( u \) is \( u p_1 \rightarrow u p_2 \rightarrow u p_3 \rightarrow u p_4 \rightarrow u p_5 \) and \( U P_u = \{ u p_1, u p_2, u p_3, u p_4, u p_5 \} \). The blue shadow represents roads on which a user can move and the solid lines in the shadow represent roads. A crossroad point (\( cp \)) indicating a connection between the roads is represented as a point. A path fragment (\( f \)) is a unit of paths and it is represented as a line connection between two \( cp s \) (e.g., \( f_k = (cp_1, cp_2) \)). A path fragment also has a direction (\( d \)) based on the user movement history and the direction is decided by selecting a start point between \( cp_1 \) and \( cp_2 \). The path fragment set (\( F \)), which includes a direction for each road, is defined as follows:

\[
F = \{ f_i^d \mid 1 \leq i \leq n, \ d \in \{1, -1\} \},
\]

where \( n \) is the number of path fragments and \( d \) is the direction of a path fragment.

The direction \( d \) of a path fragment \( f \) may be either 1 representing forward or -1 representing backward. Therefore, in Figure 4, \( f_4 \) is expressed as \( f_4^1 = cp_1 \rightarrow cp_2 \) and \( f_4^{-1} = cp_2 \rightarrow cp_1 \). An undirected path fragment (\( f_i \)) implies a pair of \((f_i^1, f_i^{-1})\). In this work, we use predefined road information such as the crossroad points and path fragments for learning and predicting user paths. As the results of prediction, the proposed method returns a predicted path fragment \((f_i^d)\).

Finally, the set of connected fragments (\( CP_i^F \)) defines all the path fragments that are connected to \( f_i^d \). For example, in Figure 4, \( CP_4^1 \) represents all the path fragments connected to \( f_4^1 \). Because of the direction of \( f_4^1 \), \( CP_4^1 \) includes all the path fragments starting from \( cp_2 \), and it is expressed as \( CP_4^1 = \{ f_2^{-1}, f_3^1, f_2^1 \} \). \( CP_i^d \) is defined as follows:

\[
CP_i^d = \{ f_i^d \mid f_i^d = cp_a \rightarrow cp_b, \ f_i^d = cp_b \rightarrow cp_a \rightarrow cp_c, \ \{ f_i^d, f_i^d \} \subseteq F, \ d \in \{1, -1\} \}.
\]

The following list of symbols and their definitions are used in path prediction.

Symbols and Definitions for Path Prediction

\( u \): A user

\( User \): A set of users, \( User = \{ u_1, u_2, \ldots, u_n \} \)

\( p \): A point, \( p = (\text{latitude}, \text{longitude}) \)

\( up \): A user point, \( up = (\text{latitude}, \text{longitude}) \)

\( U P_u \): The set of user points by user

\( U P_u = \{ up_1, up_2, \ldots, up_n \} \)

\( cp \): A crossroad point, \( cp = (\text{latitude}, \text{longitude}) \)

\( CP \): The set of crossroad points

\( CP = \{ cp_1, cp_2, \ldots, cp_n \} \)

\( f_i \): The \( i \)th path fragment, \( f_i = f_i^1, f_i^{-1} = (cp_a, cp_b) \)

\( f_i^d \): The \( i \)th path fragment with direction \( d \)

\( f_i^1 : cp_a \rightarrow cp_b, f_i^{-1} = cp_b \rightarrow cp_a \)

\( F \): The set of all path fragments

\( F = \{ f_1^1, f_1^{-1}, f_2^1, f_2^{-1}, \ldots, f_n^1, f_n^{-1} \} \)

\( CP_i^d \): The set of path fragment connected with \( f_i^d \).
3.3. Path Fragment Identification. In this section, we present a method for identifying the path fragment of a user using the location of the user. That is, a location up is projected on a path fragment by the path fragment identification. The projection implies that the user exists on the path fragment which is expressed as a road.

The process of path fragment identification measures the vertical distance between the user point and a path fragment and defines the user located path fragment with the lowest vertical distance. Figure 5 illustrates a graphical representation of the path fragment identification. To measure the vertical distance between the location up and a path fragment \( f_i \), we use an equation that determines the height of a triangle given by three side lengths. \( f_i \) is a path fragment between \( cp_1 \) and \( cp_2 \) and we calculate a vertical distance \( l_{h1} \) between up and \( f_i \) using three points, \( up, cp_1, \) and \( cp_2. \) \( p_i \) is the projection of \( up \) on \( f_i \) and \( l_{pr} \) is a length between \( cp_1 \) and \( p_i. \) Angles \( \theta_1 \) and \( \theta_2 \) are angles of distance pairs \( l_1, f_i \) and \( l_2, f_i \), respectively. The vertical distance is measured as follows:

\[
dist(up, f_i) = l_{h1} = \sqrt{l_1^2 - l_{pr}^2} = \sqrt{l_1^2 - \left( \frac{l_1^2 - l_2^2 + l_f^2}{2f} \right)^2},
\]

where \( l_1^2 + l_f^2 \geq l_2^2 \) and \( l_2^2 + l_f^2 \geq l_1^2 \).

When a path fragment is identified using \( prevf \), it is possible to identify not only the path fragment located by the user, but also the direction along which the user has moved. Given that, the identified path fragment \( (idf_{up}^d) \) is defined as follows:

\[
idf_{up}^d = \arg \min_{f_i \in F} \{ \text{dist}(f_i, up) \cap CF_{prevf} \},
\]

where \( prevf \) is the previous path fragment \( (f_i^d) \) before reaching the current path fragment \( (idf_{up}^d) \).

3.4. Collective Behavior Pattern and Weight Measurement. The proposed path prediction method aims at enabling SRS to provide effective and stable sensor information. SRS should have an acceptable performance in a mobile environment where resources are limited. Also, the path prediction algorithm for sensor filtering does not need to predict entire-range user paths because the path prediction is used only in an unstable network connection area where the network connection of a mobile device might be intermittently disconnected. Therefore, the algorithm should be able to predict close-range path prediction with fast performance in a mobile environment.

There exist several personalized path prediction algorithms (see Section 6.1), but they are not suitable for the above requirements. To satisfy the requirements, we propose a path prediction algorithm based on Collective Behavior Pattern (CBP) [21]. CBP is a concept that a collective behavior influences personal behavior (e.g., Point of Interest). The CBP-based path prediction algorithm (CBP-PP) has lower accuracy than the personalized path prediction algorithms. However, CBP-PP is able to measure weights and predict paths on the server side and it has high performance in a mobile environment. CBP-PP also supports the case where the user has no history in a specific path.

To predict paths based on CBP, we need to learn and measure weights for path fragments. In the preprocessing of measuring weight, we allocate a weight to each path fragment. In this paper, we have the frequency of using a path fragment to be the weight. The weight \( w_i^d \) of a path fragment \( f_i^d \) is defined as follows:

\[
w_i^d = \text{the number of moves along } f_i^d.
\]

That is, \( w_i^d \) indicates how frequently a user has passed along the \( i \)th path fragment in the direction of \( d. \) Since CBP-PP takes into account all the users, measuring weight involves the move history of all users to be used and the same value is used for \( w_i^d \) for all the users. If an individual weight is measured for each user, it requires additional costs for storing different weights for all the path fragments for each user. Therefore, one path fragment has the same weight determined by the path fragment history of all the users.

Algorithm 1 presents a weight-measuring algorithm. Each user \( (u) \) has a sequential set of user points \( (UP_u) \) and acquires the identified path fragment \( (idf_{up}^d) \) for a user point \( (up). \) If \( idf_{up}^d \) is equal to the previous path fragment \( (prevf) \), it indicates that the user has not moved into the next path.

![Figure 5: Graphical representation of path fragment identification.](image-url)
fragment and the algorithm returns and processes the next up to be identified. If it is not equal, it indicates that the user has moved into the next path fragment and, thus, the weight \((w^d_i)\) of \(idf^d_i\) is increased by 1. The algorithm assigns \(idf^d_i\) to \(prev_f\).

### 3.5. CBP-Based Path Prediction Algorithm

The presented path prediction method produces the next path fragment to which the user moves after the currently located path fragment is evaluated for prediction. The method is based on a greedy algorithm that determines heuristic solutions using empirical knowledge. The finding mechanism for a local solution in the greedy algorithm is suitable for close-range path prediction. The use of empirical knowledge in the greedy algorithm can satisfy the requirement that path prediction must be based on collective behaviors, not personal behaviors.

The presented path prediction algorithm compares path fragments by weight and selects one that has the maximum weight as the predicted path fragment using the greedy algorithm. The compared path fragments are then connected to the currently located path fragment of the user. The predicted path fragment \(pf^d_i\) is defined as follows:

\[
 pf^d_i = \arg \max_{f^d \in CF^d_i} w^d_f, \tag{6}
\]

\(pf^d_i\) represents the path fragment \(f^d_i\) that has the maximum weight \(w^d_f\). The path fragment is selected from the set of path fragments connected to \(f^d_i\) which may be the identified path fragment for the current user point.

Figure 6 shows an example applying the CBP-PP algorithm. In the figure, a user \(u\) has made a sequential move \(UP_u = \{up_1, up_2, up_3\}\) and is currently located at \(up_4\). From the current location, the user may move to \(up_4, up_5\), or \(up_6\).

At all the points of \(up_1, up_2, \) and \(up_3\), the user identifies \(f^1\) as the identified path fragment using \(idf^d_i\). The next path fragment is selected from \(CF^d_i = \{f^{-1}_2, f^1_2, f^1_3\}\) which is the set of path fragments connected to \(f^1_1\). The weights of the path fragments in \(CF^d_i\) are 30, 15, and 10 and the fragment \(f^1_3\) has the highest weight. Thus, \(f^1_3\) is selected as the predicted path fragment \(pf^d_i\).

Algorithm 2 presents the CBP-based path prediction algorithm. This algorithm uses the \(up\) and \(w^d_i\) measured in Algorithm 1 and identifies the path fragment \(idf^d_i\) currently located by \(up\). Then, the algorithm determines a set of connected path fragments \((CF^d_i)\) with respect to \(idf^d_i\) and selects the path prediction that has the maximum weight \(w^d_i\) in \(CF^d_i\) as the predicted path fragment \(pf^d_i\).

The approach predicts one path fragment at a time. The algorithm takes into account mobile computing power and human walking speed for accurate results. The approach is effective for predicting short paths supported by the fragmentation of paths. In the case that the amount of sensor information provided by SRS is overly large, a dynamic path revision is required for correct prediction.

---

**Algorithm 1: Weight-measuring algorithm.**

```plaintext
(1) User [] ← get_all_users ()
(2) for each User :u do
(3) U1P [] ← get_user_points (u)
(4) prevf ← null
(5) for each UP : up do
(6) idf ← get_identified_path_fragment (up)
(7) if prevf # idf then  // when user moves another path fragment
(8) i ← get_path_fragment_number (idf)
(9) d ← get_path_direction (idf)
(10) w[i][d] ← w[i][d] + 1
(11) prevf ← idf
(12) endif
(13) endfor
(14) endfor
(15) return w[]
```

**Figure 6: An example of CBP-PP.**
Algorithm 2: CBP-based path prediction algorithm.

Table 1: Time elements and time duration.

| Time  | Time duration |
|-------|---------------|
| $t_1$ | 06:00–08:59   |
| $t_2$ | 09:00–11:59   |
| $t_3$ | 12:00–12:59   |
| $t_4$ | 13:00–16:59   |
| $t_5$ | 17:00–18:59   |
| $t_6$ | 19:00–21:59   |
| $t_7$ | 22:00–05:59   |

3.6. CBP-PP with a Time Feature. CBP which is used as the base for the path prediction algorithm has a limitation that its accuracy is lower than personalized path prediction. To improve accuracy, we consider time in the algorithm. The improved algorithm is named CBP-PP/t. A user makes moves to different locations on certain patterns throughout a day. For example, a user goes to work in the morning, moves out for lunch during the lunch hour, and comes back to home after work in the evening. A similar behavior is observed in many people. This is a type of collective behavior patterns by time. We analyze such patterns in terms of relevant time duration to improve the accuracy of prediction.

Suppose the time elements and time durations in Table 1. We appropriately divide 24 hours into 7 elements by behavior patterns of users. For time analysis, the expression of the path fragment set, the connected path fragment set, and the weight defined above are modified to take into account time. The expression of the predicted path fragment is also modified. The following redefine $F$, $CF_{i}^d$, $w_i^d$, and $p_{f_i}^d$ in consideration of time:

$$F = \{ f_i^d \mid 1 \leq i \leq n, \ d \in \{1, -1\}, \ t \in T \},$$

$$CF_{i}^d = \{ f_i^d \mid f_i^d = cp_a \rightarrow cp_b, \ f_j^d = cp_b \rightarrow cp_c, \ f_j^d \in F, \ d \in \{1, -1\}, \ t \in T \},$$

$$w_i^d = \text{the number of moves along a path fragment } (f_i^d) \text{ at a time } (t) ,$$

$$p_{f_i}^d = \arg \max_{f_j^d \in CF_{i}^d} w_j^d .$$

Algorithm 3 presents the weight-measuring algorithm with time. The algorithm is similar to the algorithm in Algorithm 1. However, the addition of time $t$ details the weight $w_i^d$, which further elaborates the prediction.

Algorithm 4 describes the CBP-based path prediction algorithm with time. The algorithm also is similar to the algorithm in Algorithm 2, but it uses the time-considered weight ($w_i^d$).

4. Implementation and Experiment

4.1. System Implementation. To implement the proposed path prediction method, we have developed several applications to be run on the server and mobile devices. On the server side, applications are developed for managing path fragments and user locations, predicting path fragments, and returning the prediction results. On the mobile device side, applications are developed for tracking user locations, displaying identified path fragments from user locations, and verifying path prediction. Table 2 specifies the development environment for the implementation.

Figure 7 shows the data model for implementing the path prediction algorithm. The table User is created to identify users and the table UserPoint is created to store and track user locations and times. To represent roads, crossroad points and path fragments are created in the table CrossroadPoint and PathFragment, respectively. The table PathFragmentWeight stores weights for path fragments with a direction and time.

Figure 8 presents screenshots of the implementation on a mobile device. Figure 8(a) displays crossroad points for
Algorithm 3: Weight-measuring algorithm with time.

```plaintext
weight_measuring_with_time
(1) User [] ← get_all_users()
(2) for each User: u do
(3) UP [] ← get_user_points(u)
(4) prevf ← null
(5) for each UP: up do
(6) idf ← get_identified_path_fragment(up)
(7) if prevf ≠ idf then // when user moves another path fragment
(8) i ← get_path_fragment_number(idf)
(9) d ← get_path_fragment_direction(idf)
(10) t ← get_current_time()
(11) w[i][d][t] ← w[i][d][t] + 1
(12) prevf ← idf
(13) endif
(14) endfor
(15) endfor
(16) return w[][][]
```

Algorithm 4: CBP-based path prediction algorithm with time.

```plaintext
path_prediction_with_time(up, w[][][])
(1) t ← get_current_time()
(2) idf ← get_identified_path_fragment(up)
(3) CF[] ← get_connected_fragments(idf)
(4) pf ← null // pf is a predicted path fragment
(5) maxweight ← 0
(6) for each CF: cf do
(7) i ← get_path_fragment_number(cf)
(8) d ← get_path_fragment_direction(cf)
(9) if maxweight < w[i][d][t] then // set pf by maximum weight
(10) maxweight ← w[i][d][t]
(11) pf ← cf
(12) endif
(13) endfor
(14) return pf
```

Table 2: Development environment.

| Feature                  | Details                        |
|--------------------------|--------------------------------|
| OS                       | Windows 7 Professional K (x86) |
| Processor                | Intel(R) Core(TM) i5-2500 3.30 GHz |
| RAM                      | 4 GB                           |
| Development language     | Android, JSP                   |
| Mobile OS                | Android OS                     |
| Android emulator version | 4.1.2                          |
| Web server               | Apache Tomcat 8.0.8             |
| Database                 | MySQL 5.5                      |

a path prediction and path fragments connected to each crossroad point. Figure 8(b) shows the sequence of actual user points. Figure 8(c) shows the projection results for the user points on path fragments. As shown in Figure 8(c), it can be confirmed that each user point is correctly identified along path fragments.

Figure 9 shows the path prediction results. The blue lines in the figure represent the path fragment currently occupied by the user. On the other hand, the black lines represent the actual path fragments taken after the blue line. The red lines represent the predicted path fragment for the current user location. Figure 9(a) shows the path prediction results without considering time and Figure 9(b) shows the results with time considered. In the figure, we can confirm that time consideration obviously influences the prediction results.

4.2. Experiment. For the experiment, we have also developed a mobile application for tracking user locations, collecting actual user GPS points, and predicting user paths. Five users participated in the experiment. They collected user points by moving around a university campus and near areas for ten days. The user points that are outside of the experiment areas are removed from the collection.
Figure 7: Data model for path prediction.

Figure 8: Screenshots of implementation: (a) crossroad points and path fragments; (b) sequenced user points; and (c) identified path fragments and projection points.

Figure 10 presents the screenshots of the user points used in the experiment. We collected 5871 user points and distinguished 117 datasets from the collection as user paths. Figure 10(a) indicates the collected user points within the university area and Figure 10(b) shows user points near to the university area. The collected user points are used for measuring weights and fed into the path fragment prediction algorithm.

5. Evaluation

This section evaluates the effectiveness of the path prediction-based approach by simulation. It also evaluates the implemented system and the proposed algorithm using the experiment results. First, we discuss an advantage of the extended system, Path Prediction-based SRS (PP-SRS) in comparison to the previous version of SRS. Then, we compare the CBP-based path prediction algorithm (CBP-PP) and the CBP-based path prediction algorithm with the time-considered algorithm (CBP-PP/\(t\)) in terms of processing time and accuracy.

5.1. Service Reliability Evaluation. This section describes the comparison SRS and PP-SRS for reliability. A mobile device tries to access SRS or PP-SRS and acquires sensor information in real-time. However, if the device fails to access SRS or PP-SRS due to the low quality of the mobile network, it is impossible for the device to interpret the semantics of sensors, which further disables a user to provide services using sensors. In general, the QoS of the mobile network is
evaluated in terms of coverage, accessibility, and audio quality [22]. Coverage is the signal strength received by a mobile terminal. It indicates the probability of network connection of the mobile device at the user location. Coverage is divided into coverage, bad coverage, and absence of coverage by signal strength. Accessibility is the capacity to successfully establish communication calls between two terminals. It is the probability of connection failure by an interruption when a mobile device attempts to connect to a server. Accessibility is divided into normal calls release representing successful connection and abandoned calls representing connection failure. Audio quality is the status of conversation perception during a successful call. It is the probability of receiving unclear answers from a server concerning requested information after the mobile device accesses the server. Audio quality is divided into poor, fair, and good.

A mobile device might fail to access SRS when a user is located in an unstable network connection area. In such an area, the QoS of the mobile network is low in terms of coverage, accessibility, and audio quality. Table 3 shows examples of the QoS factors for high and low quality. Low quality QoS causes frequent failures of access to SRS. That is, a mobile device receives incomplete sensor information from SRS or PP-SRS when it requests. An access failure may occur when any of the QoS factors is of low quality. A ratio of access failure ($R_{AF}$) is calculated by dividing the number of access failures by the number of access requests.
Service reliability rate ($R_{SR}$) is the probability of successfully providing services to a mobile device when they are requested. To measure $R_{SR}$, we have developed a simulator to generate access failures when services are requested and we count the number of successful services. In the case of SRS, a mobile device is able to receive immediately necessary sensor information according to $R_{AF}$ and provide the requested service to the user. In PP-SRS, the mobile device is also able to receive necessary sensor information according to $R_{AF}$. If the mobile device cannot receive sensor information due to the access failures, it can use preloaded sensor information according to a path prediction accuracy ($R_{PA}$). Therefore, $R_{SR}$ is measured as follows:

$$R_{SR} = \frac{\text{the number of Service Successes}}{\text{the number of Service Requests}} = (1 - R_{AF}) + (R_{AF} \times R_{PA}),$$

where $R_{AF}$ is the access failure rate and $R_{PA}$ is the path prediction accuracy. Since PP-SRS only uses a path prediction method, $R_{PA}$ is set to zero in SRS evaluation.

$R_{SR}$ is the ratio of the number of service successes to the number of service requests. It can be also calculated by the equation about the access successes rate and the prediction success rate after the access failure as shown in (8). If a mobile device successfully accesses PP-SRS, the requested services are provided to the user; on the other hand, if it failed, service providing depends on the rate of path prediction accuracy.

For comparison evaluation, we use a simulator for measuring $R_{SR}$ and counting provided services for a mobile device when services are requested. $10^5$ service requests were used and the simulator stochastically decides by (8) the success or failure of the services.

Figure 11 shows $R_{SR}$ for SRS and PP-SRS when $R_{AF}$ is 10%, 30%, and 60%. We compare SRS with three cases of PP-SRS with different $R_{PA}$ of 50%, 70%, and 90% for each case. As a result, each system has the highest $R_{SR}$ at $R_{AF}$ 10% and all the three cases of the PP-SRS have a higher $R_{SR}$ than SRS. The higher the $R_{PA}$ of the PP-SRS is, the higher the $R_{SR}$ is. If an access failure occurs, the service fails in SRS, whereas PP-SRS is able to successfully provide services using preloaded sensor information through the path prediction. The experiment shows that the proposed PP-SRS is more reliable than SRS.

5.2. Processing Time Evaluation. We evaluate the processing time of CBP-PP and CBP-PP/t with ten path fragments with a direction selected from the collected path fragments. We also compare the results of identifying paths and prediction time of CBP-PP and CBP-PP/t. This also shows the overheads caused by the time consideration in CBP-PP/t. Table 4 shows the processing time of CBP-PP and CBP-PP/t and the time difference for the ten selected path fragments. The results show that CBP-PP is faster than CBP-PP/t in all path fragments except one "f006." The average processing time of CBP-PP is measured as 42.32 ms, while that of CBP-PP/t is measured as 44.41 ms, which results in a 2.09 ms difference. The difference reflects the overhead (4.66% decline) caused by time consideration in CBP-PP/t.

5.3. Accuracy Evaluation. The evaluation of accuracy is concerned with measuring the accuracy of the predicted path fragment using the datasets collected by the five users. Figure 12 presents the accuracy comparison of CBP-PP and CBP-PP/t. The user path for the prediction test is not considered in the evaluation.

Figure 12(a) shows the accuracy of CBP-PP and CBP-PP/t for 50 datasets. CBP-PP shows 24.8% accuracy on average and CBP-PP/t shows 43% accuracy on average. Figure 12(b) indicates the accuracy for 116 datasets. The average accuracy of CBP-PP is 55.6%, while that of CBP-PP/t is 87.4%. In both cases, CBP-PP/t shows a higher accuracy than CBP-PP, which confirms that time consideration improves the accuracy of path prediction. Table 5 shows the accuracy of CBP-PP and CBP-PP/t and the difference rate for 116 datasets. The result confirms that the accuracy of CBP-PP/t is 64.6% on average superior to CBP-PP.

6. Related Work

This section presents related work about path prediction research. We describe personalized pattern-based path prediction research using personal location tracking data and
discuss problems of the existing work in applying them to extending SRS. We also discuss the work on Collective Behavior Pattern- (CBP-) based path prediction using location tracking data of groups.

6.1. Personalized Pattern-Based Prediction. Numerous techniques have been studied for predicting locations or paths using user mobility [23–25]. The majority of the existing research uses probabilistic models along with context-awareness and data mining techniques. They also use personalized path prediction using variable user information.

Samaan and Karmouch [23] proposed an architecture for predicting personal mobility using contextual knowledge and a spatial conceptual map. Given a user context and an area of interest defined on a map, the system predicts a user location using the Dempster-Shafer theory. The system returns a predicted path created by searching a path from the current location of the user to the predicted location. The prediction result is only influenced by user profiles and defined rules. So the prediction result cannot be improved by data collection such as the user mobility data and system experiences.

Chen et al. [24] presented a personal route prediction system that stores user location data from GPS and predicts paths by learning the data. It defines Regions of Interest (ROI) as a criterion, which is the staying time of the user. It creates a basic Markov model based on frequency. The Markov model is then used to predict paths from the current location. They divide a map into cells and provide patterns moving towards the ROI of the users. Unlike our work, they do not predict detailed paths.

Kim et al. [25] described a probabilistic graphical model that acquires user location data from GPS. It uses a prediction approach similar to that in the work by Chen et al. The model includes processes for combining several paths that have high similarity in path learning.

The existing research is based on user data for prediction. If a user moves to a new area (e.g., touring), personalized learning is very hard since there exist no training datasets for the user.

6.2. Collective Behavior Pattern-Based Prediction. There are some works (e.g., [21, 26]) based on CBP for addressing the personal pattern problem in Section 6.1. CBP is based on that collective behaviors influence personal behaviors, which enables predicting user locations and moves. A CBP-based method can predict paths using the information of people that have visited an area even if there is no history for a specific user [21].

Xiong et al. [26] proposed a prediction method based on collective behavioral patterns. This method predicts user locations based on the cell tower id of a phone. They use a hybrid method of CBP and personalized patterns. However, the method cannot provide detailed user paths since it can predict only cell towers.

CBP-based methods have two advantages. Firstly, they can predict a user path using group location data without the user location data. Also their prediction is fast at the group level. However, group-level models often cause low accuracy because it does not analyze the personal pattern. This motivated the hybrid method of the CBP-based method and personalized pattern-based model by Xiong et al.

7. Conclusion

The Internet of Things (IoT) has emerged and systems for registering and managing sensor information have advanced. SRS is developed to dynamically support sensor information and accurately process the semantics of heterogeneous sensors. As the number of sensors in the IoT environment increases explosively, so does the importance of sensor filtering in sensor management systems.
There have been several sensor filtering problems arisen in mobile computing environments such as low performance, low resource, and unstable network status. Searching sensors in real-time requires a rapid connection and process and providing services consistently and immediately regardless user mobility. To address this, we have presented a path prediction method for effective sensor filtering. In the method, we use SRS as the sensor platform for providing sensor information. We have described path representation, identification, and prediction algorithms for path prediction. The presented path prediction algorithm is based on CBP and takes into account time. We evaluated the algorithm by implementing it in SRS and PP-SRS and compared the outputs. We also evaluated the processing time and accuracy of prediction between the CBP-PP algorithm and CBP-PP/t algorithm. The evaluation shows that CBP-PP/t takes a longer processing time on average than CBP-PP, which is attributed to the overhead of time consideration. However, the difference is slight. On the other hand, CBP-PP/t demonstrates significantly higher accuracy in prediction over CBP-PP.

In the future, we plan to implement SRS and evaluate the connection performance with SRS. We also plan to develop a hybrid path prediction algorithm including CBP-based and personalized approaches to improve the accuracy of the prediction.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2014R1A1A2058992).

References

[1] O. Vermeenan and P. Friess, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, River Publishers, 2013.

[2] L. Atzori, A. Iera, and G. Morabito, “The internet of things: a survey,” Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] L. Luo, A. Kansal, S. Nath, and F. Zhao, “SenseWeb: sharing and browsing environmental changes in real time,” in Proceedings of the Microsoft eScience Workshop, Microsoft Research, December 2008.

[4] C. Reed, M. Botts, G. Percivall, and J. Davidson, “OGC sensor web enablement: overview and high level architecture,” OGC White Paper, Open Geospatial Consortium, 2013.

[5] S. Nath, J. Liu, and F. Zhao, “SensorMap for wide-area sensor webs,” Computer, vol. 40, no. 7, pp. 90–93, 2007.

[6] B. L. Gorman, D. R. Ressegue, and C. Tomkins-Tinch, “Sensors-pedia: information sharing across incompatible sensor systems,” in Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS ’09), pp. 448–454, Baltimore, Md, USA, May 2009.

[7] M. Yuriiyama and T. Kushida, “Sensor-cloud infrastructure—physical sensor management with virtualized sensors on cloud computing,” in Proceedings of the 13th International Conference on Network-Based Information Systems (NBiS ’10), pp. 1–8, September 2010.

[8] The European Union’s Seventh Framework Programme, “Open Source cloud solution for the Internet of Things,” http://openi.o.eu/.

[9] M. Compton, C. Henson, L. Lefort, H. Neuhaus, and A. Sheth, “A survey of the semantic specification of sensors,” in Proceeding of the 2nd International Semantic Sensor Networks Workshop, International Workshop on Semantic Sensor Networks 2009, pp. 17–32, Washington DC, USA, October 2009.

[10] A. Sheth, C. Henson, and S. S. Sahoo, “Semantic sensor web,” IEEE Internet Computing, vol. 12, no. 4, pp. 78–83, 2008.

[11] Y. Shi, G. Li, X. Zhou, and X. Zhang, “Sensor ontology building in semantic sensor web,” in Internet of Things, vol. 312 of Communications in Computer and Information Science, pp. 277–284, Springer, Berlin, Germany, 2012.

[12] M. Compton, P. Barnaghi, L. Bermudez et al., “The SSN ontology of the W3C semantic sensor network incubator group,” Web Semantics: Science, Services and Agents on the World Wide Web, vol. 17, pp. 25–32, 2012.

[13] Digital Enterprise Research Institute, Linked Sensor Middleware (LSM), https://code.google.com/p/deri-lsm/.

[14] S. Mayer, D. Guinard, and V. Trifa, “Searching in a web-based infrastructure for smart things,” in Proceedings of the 3rd International Conference on the Internet of Things (IOT ‘12), pp. 119–126, IEEE, Wuxi, China, October 2012.

[15] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, “Sensor search techniques for sensing as a service architecture for the internet of things,” IEEE Sensors Journal, vol. 14, no. 2, pp. 406–420, 2014.

[16] M. Köhne and J. Sieck, “Location-based services with iBeacon technology,” in Proceedings of the 2nd International Conference on Artificial Intelligence, Modeling and Simulation, pp. 315–321, November 2014.

[17] D. Jeong, “Framework for seamless interpretation of semantics in heterogeneous ubiquitous sensor networks,” International Journal of Software Engineering & Its Applications, vol. 6, no. 3, pp. 9–16, 2012.

[18] EE, UK Coverage Checker, http://ee.co.uk/ee-and-me/network/4gee/coverage-checker/.

[19] D. Jeong and J. Ji, “A registration and management system for consistently interpreting semantics of sensor information in heterogeneous sensor network environments,” Journal of KIISE: Database, vol. 38, no. 5, pp. 289–302, 2011.

[20] ISO/IEC JTC 1/SC 32, ISO/IEC 11179-3:2013—Information Technology—Metadata Registries (MDR)—Part 3: Registry Metamodel and Basic Attributes, 2013.

[21] F. Calabrese, G. Di Lorenzo, and C. Ratti, “Human mobility prediction based on individual and collective geographical preferences,” in Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems (ITSC ’10), pp. 312–317, Maderia Island, Portugal, September 2010.

[22] Anacom, “GSM mobile networks—quality of service survey,” Anacom Quality Report, Anacom, 2002.

[23] N. Samaan and A. Karmouch, “A Mobility prediction architecture based on contextual knowledge and spatial conceptual maps,” IEEE Transactions on Mobile Computing, vol. 4, no. 6, pp. 537–551, 2005.
[24] L. Chen, M. Lv, Q. Ye, G. Chen, and J. Woodward, “A personal route prediction system based on trajectory data mining,” *Information Sciences*, vol. 181, no. 7, pp. 1264–1284, 2011.

[25] J.-M. Kim, H. Baek, and Y.-T. Park, “Probabilistic graphical model based personal route prediction in mobile environment,” *Applied Mathematics & Information Sciences*, vol. 6, supplement 2, pp. 651S–659S, 2012.

[26] H. Xiong, D. Zhang, D. Zhang, and V. Gauthier, “Predicting mobile phone user locations by exploiting collective behavioral patterns,” in *Proceedings of the 9th International Conference on Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC ’12)*, pp. 164–171, IEEE, Fukuoka, Japan, September 2012.
