Jurnal Teknologi

Full Paper

ARCHITECTURE AND FORWARDING MECHANISM FOR GEO-FENCING APPLICATIONS WITH MULTIPLE INFORMATION PROVIDERS

Teduh Dirgahayu

Department of Informatics, Universitas Islam Indonesia, Yogyakarta, Indonesia

*Corresponding author
teduh.dirgahayu@uii.ac.id

Article history
Received
26 November 2015
Received in revised form
14 January 2016
Accepted
10 October 2016

Abstract

Geo-fencing application is a class of location-based service that provide mobile users with services, i.e. information or functionality, when the users are within certain geographical areas. In this paper, we present an architecture for geo-fencing applications that allow information provisioning from multiple providers based on users' locations. The architecture includes a central component called *service router* whose main task is to forward information requests from the users' mobile applications to targeted information providers. The architecture assumes that information is stored in specific content management systems (CMSs). We also present a location-based request forwarding mechanism for the service router. Every request from the applications must include the users' location coordinates. These coordinates are used to determine to which information provider the request should be forwarded. In addition, the forwarding mechanism includes a caching mechanism to make efficient the forwarding process. The architecture and forwarding mechanism are implemented in RESTful Web Services. This architecture offers three main benefits, i.e.: (i) natural fit to real-world situation, in which each area is administered by an authority, (ii) scalability by delegating the routing tasks to a composition of service routers in a hierarchical architecture, and (iii) consistent presentation by allowing the mobile applications to restructure and reformat information from the providers.

Keywords: Architecture; location-based service; geo-fencing applications; forwarding mechanism; service router; multiple providers

1.0 INTRODUCTION

Nowadays services are targeted to specific users by considering the users' contexts, e.g. preferences, transactions history, or locations. Location-based service (LBS) refers to services, i.e. information or functionality, which are provided by taking into consideration the users' geographical locations [1]. The vast development of LBS is primarily supported by the advancement of global positioning system (GPS), mobile devices and networks that are able to identify users' locations [2]. LBS are mainly provided as mobile applications [3].

Geo-fencing application is a class of LBS in which the service is provided only to users that are within a certain virtual perimeter on a geographical area. Currently, many geo-fencing applications are developed to provide services to moving objects, e.g. transportation, logistics, and security [4].

Using a geo-fencing application for information service, mobile users can get information that is specific to their locations. For example, an airport can be defined as a geo-fencing area. When a passenger travels from one airport to another, it means that the passenger moves from one geo-fencing area to another. A geo-fencing application can be developed to display information about facilities of those airports.

Suppose that a passenger carrying a mobile device travels by plane from Yogyakarta to Makassar Airport. Before departure, he may check information about airport facilities using a geo-fencing application. His mobile application will display information about facilities in Yogyakarta Airport. This could be done without requiring the passenger to select which airport information to access. After landing, he may check again the information about airport facilities. Now, his mobile application will display information about facilities in Makassar Airport [5].

Current geo-fencing applications, and also LBS in general, provide services to users in different areas from a single service provider, e.g., in [4][6]. In the

example above, a single provider provides information from both airports.

A more-advanced service provisioning can make use of multiple service providers; each of which is dedicated to serve users in different areas [5]. Consider that each airport has its own information server that provides information about its facilities, as illustrated in Figure 1. When a passenger is in Yogyakarta Airport, his access request is forwarded to and is served by the Yogyakarta provider. When the user arrives at Makassar Airport, his access request is forwarded to and is served by the Makassar provider. Since the geo-fencing application knows its geographical location, the passenger does not have to choose which information provider to access. The application will automatically select the correct one. To enable this, access requests must include the geographical coordinates of the mobile device in which the application runs.

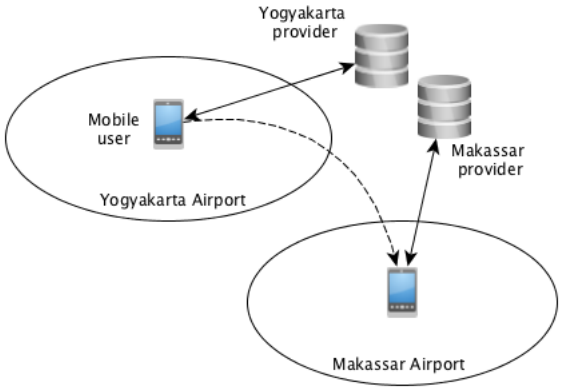

Figure 1 Geo-fencing application with multiple information providers

The use of multiple service providers in a geo-fencing application, i.e. one provider for each geographical area, would be naturally fit with real-world situation. In real world, an authority administers service provided in an area, which may range from an airport and university campus to a restaurant and from a country and district to a town.

1.1 Research Questions

This paper addresses two research questions.

1. What architecture is needed to allow a geo-fencing application access different information providers transparently?
2. What mechanism is needed to forward request from the geo-fencing application to a target information provider?

1.2 Objectives

The objective of this paper is twofold. First, this paper is to present an architecture for a geo-fencing application with multiple service providers. In this paper, we focus on information service provisioning. The architecture allows users to access information from different providers in a consistent way. The architecture includes a central component namely *service router*, whose task is to forward information access request to correct information service providers.

Second, this paper is to present a mechanism for forwarding access requests by consider the users' current locations. A service router in a geo-fencing application with multiple providers should employ this mechanism. This paper also illustrates the implementation of the mechanism in a service router.

1.3 Related Work

LBS have been widely used to support people traveling around, e.g. navigation and tracking, thanks to free availability of the global positioning system (GPS). LBS can be either push or pull services. Push services give users information without requiring them to actively request for it. Pull services require users to actively request for information. An application may integrate both push and pull services [7].

Literatures on LBS have proposed different architectures based on different service goals, requirements, and types, e.g. recommender systems [8][9][10], social networks [11][12][13], and marketing [14][15]. In addition, those architectures differ on their scopes and abstraction levels. Those architectures assume that information is provided from a central provider.

Geo-fencing [16] refers to the concept and technology that creates a virtual perimeter on a geographical area. It allows LBS to be provided only to users on a certain area. While it was originally to confine Wi-Fi coverage for privacy concerns [17], developments have been done to extend it with GPS technology for tracking and providing services to moving objects, e.g. delivery and fleet management [4][19][21], tourism [18], healthcare [20][22][23], and disaster management [24]. Those applications specify different architecture since they have different goals and requirements. None considers service provisioning with multiple providers.

Basically, the architecture of a geo-fencing application consists of a mobile application and service provider as depicted in Figure 2. The application should run in a mobile device with a positioning system, e.g. GPS or Wi-Fi. The provider includes a database to contain information contents. Interaction between the mobile application and service provider is done via the provider's service interface. This interface is represented with a grey rectangle attached to the provider. The request should include the location coordinates of the mobile device.

It should be noted that architectures presented in the literatures might differ on scopes and abstraction levels. In some applications, the architectures also

include a control center with which a supervising user tracks the locations of mobile devices.

The limitation of this architecture becomes visible when information contents from different sources need to be provided to the mobile application. Those sources have to put their contents to the provider's database. In some situation, this database sharing cannot be accepted since it might compromise data privacy and security.

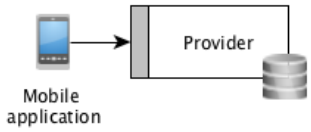

Figure 2 Basic architecture of geo-fencing applications

As a class of LBS, the service provider of a geo-fencing application can be a push or pull service. In a push service, the service provider gives users information, e.g. notifications, when the users are within, entering, or leaving a geo-fencing area. In a pull service, users have to request information to a service provider. The provider will respond only if the users are within a geo-fencing area. A geo-fencing application may integrate both push and pull services [7].

The architecture in Figure 2 assumes that services are provided from a single provider. In order to allow different providers serve different geo-fencing areas, a specification proposes an architecture of LBS with multiple service providers [5]. It includes a central component called *service router* to forward requests from mobile applications to correct service providers. The forwarding is done by taking into account the geographical location of the mobile device in which the application runs. This architecture will be described in more detail in Section 3.

Location-based routing or forwarding is investigated in the field of ad hoc networks, e.g. [25][26][27][28]. These routing protocols are developed to address the problems of scalability and mobility in ad hoc networks. By knowing the physical (geographical) locations of devices in a network, routing can be established via the closest neighboring devices. Thus packets transmitted from one device to another will not flood the network. It is a different problem from what a service router has to do in geo-fencing applications.

Furthermore, those routing protocols are of network protocols (OSI layer 3). Interactions between a mobile application and service providers are at application-level. In a geo-fencing application, locations are application-level information. Service router hence should be at the application layer (OSI layer 7) as well. A mobile application and service provider establish an end-to-end association via a service router.

1.4 Structure

This paper is further structured as follows: Section 2 presents our research method. Section 3 presents and discusses the research results that include architecture for a geo-gencing application with multiple information providers, a forwarding mechanism to use in the architecture, and the implementation of the forwarding mechanism. Section 4 concludes this paper and identifies future work.

2.0 METHODOLOGY

The main steps of our research were (i) specifying system requirements that are needed to allow a geo-fencing application to access multiple service providers, (ii) defining an architecture that satisfies those requirements, (iii) developing a location-based request forwarding mechanism needed by the architecture, and (iv) implementing those architecture and mechanism. Those steps are explained as follow.

Firstly, we specified a list of system requirements. The most importants were

- Information contents must be provided by taking into consideration the users' current geographical locations.
- Users must be transparent from the selection of service providers.
- Information contents in the service providers must be accessible from Web browsers.
- The architecture should be scalable to accommodate a large number of service providers.
- The architecture should allow consistent presentation of information contents from different service providers.

Secondly, we defined a system architecture to satisfies those requirements. The architecture is presented in Section 3.1. We introduced a service router as a component to forward request from a geo-fencing application to a target service provider [5].
Thirdly, we developed a forwarding mechanim to be implemented in the service router [29]. The mechanism requires that every request includes the user's location coordinate. The mechanism is presented in Section 3.2,

Finally, we implemented those architecture and mechanism using RESTful Web Services. The implementation is presented in Section 3.3.

3.0 RESULTS AND DISCUSSION

3.1 Architecture

Figure 3 depicts an architecture for a geo-fencing application with multiple service providers [5]. It consists of a mobile application, a service router, and a number of service providers. The service router

includes a database that contains a list of registered geo-fencing areas and their service providers. This database is not to contain information contents. Contents are stored in the service providers' database. In principle, the service can be information or functional service. In this paper, we focus on information service provisioning.

We assume that the providers store their information in specific content management systems (CMSs) built for this purpose [4]. For this purpose, the CMSs must be equipped with a programmable interface to allow information access from a mobile application. In our previous example (Figure 1), each airport provides its information on its own CMS.

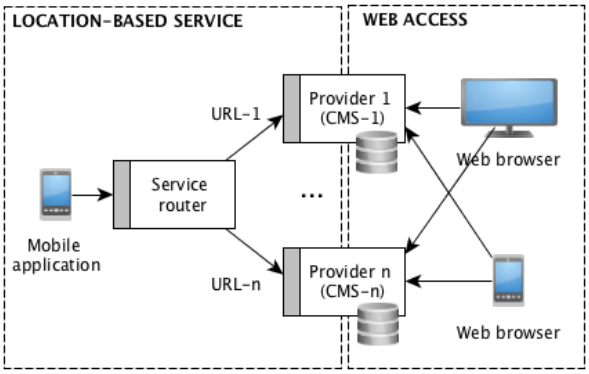

Figure 3 Architecture for geo-fencing application with multiple providers

On the right side, the figure shows traditional Web access to different information providers from computers or mobile devices using Web browsers. A user must type in the URL of a targeted information provider. The browser then displays the information in a presentation format as specified by the provider's CMS.

On the left side, the figure shows how a geo-fencing application accesses multiple information providers. User access is facilitated with a mobile application that is connected to a service router. The application runs on a mobile device with GPS.

This architecture specifies that every request from the mobile application to the service router should include the location coordinate of the mobile device on which the application is running. Upon receiving a request, the service router calculates those location coordinates to determine a target information provider that should serve the request. This is done by calculating whether the location coordinate is within a registered geo-fencing area. If so, the provider of that area is the target provider.

The service router then forwards the request to the targeted service provider. After processing the request, the provider sends a response back to the service router. Finally, the service router sends the response further to the mobile application.

The service router should maintain an association between a user's mobile application and a targeted service provider to make efficient the request forwarding process. This association is established when the user sends first request from within a registered geo-fencing area. When the user leaves that area and enter a new area, the service router should recalculate the user's new location coordinate to determine which provider should now serve requests from that user. This association operates as a caching mechanism.

All requests from a user are forwarded to the same information provider as long as the user does not leave the area served by that provider. When the user enters into a new area, the service router calculates the user's new location coordinates to determine a target information provider that should now serve the user in the new area.

3.1.1 Hierarchical Architecture

We specify that service router and information providers should have the same service interface. Therefore, the architecture can be extended to a hierarchical architecture as depicted in Figure 4. Service interfaces are indicated with grey bars attached to service routers or information providers. Upon receiving a request from a user, a service router can forward the request to another service router or an information provider.

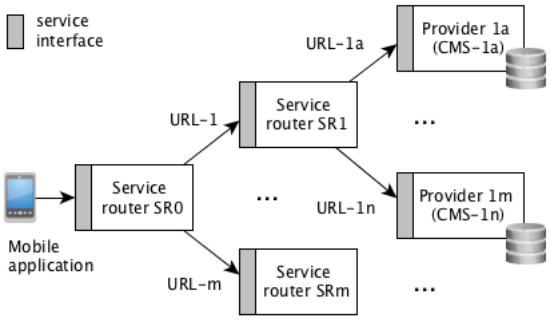

Figure 4 Hierarchical architecture

With this hierarchical architecture, a very large region can be divided into several sub-regions; each of which is handled with a dedicated service router. A sub-region contains a number of areas. For example, service router *SR0* handles all requests from the whole region. Instead of determining a target provider, it determines a service router that is dedicated to the sub-region in which the user is located, e.g. service router *SR1*. Service router *SR0* then forwards the request to *SR1* that will calculate the user's location to determine a target provider.

This hierarchical architecture delegates the calculation for determining target information providers from a service router to another service router. In turn, this delegation makes the architecture scalable.

3.1.2 Content Restructuring and Reformating

Accessing information via a service interface would also allow a mobile application to restructure and reformat the information presentation. By restructure, we mean that the application defines the navigation structure of the information items. By reformat, we mean that the application defines the presentation format (theme) of the information items. This restructuring and reformatting would support the mobile application in providing consistent presentation of same types of information from different providers. Such consistency would increase the usability [30][31][32].

3.2 Forwarding Mechanism

A service router consists of four main components as depicted in Figure 5.

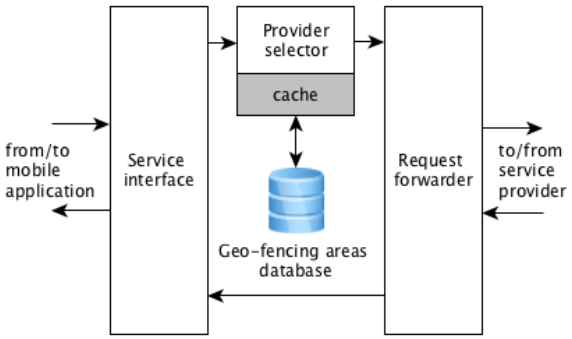

Figure 5 Service router component

The components are as follows.

- *Service interface*. This component provides a programmable interface to allow service invocation from mobile applications.
- *Provider selector*. This component handles requests from mobile applications and determines service providers to which the requests should be forwarded. This component includes a cache unit to make efficient its process.
- *Geo-fencing areas database*. This component stores a list of geo-fencing areas and the URLs of the corresponding service providers.
- *Request forwarder*. This component modifies and forwards request from mobile applications to a target service provider determined by the provider selector.

The forwarding mechanism [29] is depicted in an activity diagram in Figure 6. Two possible situations are shown in a sequence diagram in Figure 7. The lifelines refer to the components of a service router as indicated in Figure 5. Interactions with external components (i.e. mobile applications and service providers) are not shown.

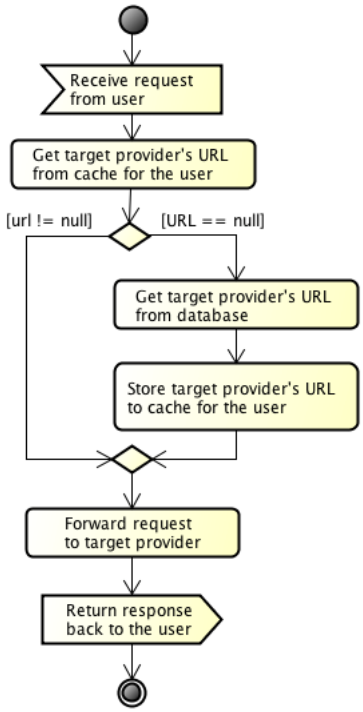

Figure 6 Forwarding mechanism in a service router

The first situation happens when the request is the first request that a user sends from within a geo-fencing area. In Figure 7, messages of this interaction are depicted in the upper part and prefixed with 1. The service router does not yet know which provider should serve that request, i.e. message `getURL()` returns null. In other words, an association between the mobile application and a service provider is not yet established and stored in the cache unit. Hence the cache unit should consult with the geo-fencing area database to determine a target service provider, i.e. message `getURLfromDB()`. When a provider is found, the cache unit stores the association between the mobile application and the target service provider, i.e. message `store()`.

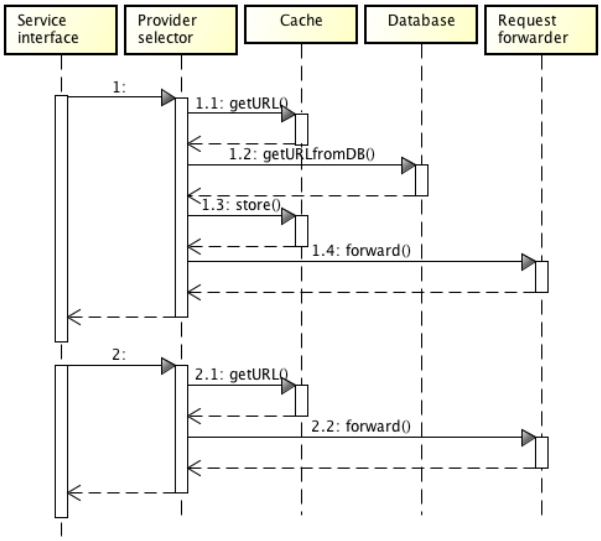

Figure 7 Forwarding mechanism in a service router

The second situation happens when a request is not the first message that a user sends from within a geo-fencing area. Messages of this interaction are depicted in the lower part and prefixed with 2. In this situation, an association between the mobile application and a service provider has already been established and stored in the cache unit. Thus the cache unit does not need to consult with the geo-fencing area database.

The use of a caching mechanism is necessary to make efficient the process in the provider selector. When a mobile device stays in a geo-fencing area, it is likely that the device would not be in the same location coordinate during its stay. The mobile device may move around in that area. Without a caching mechanism, every request from a mobile device staying in a geo-fencing area must be consulted with the geo-fencing areas database. The caching mechanism reduces the number of consultation needed.

Consultation with the geo-fencing area database may consume significant amount of computing resource. It depends, among others, on the number of registered geo-fencing areas, the shape of each area, overlaps between areas, and the precision level for area representations. For example, consider two neighboring geo-fencing areas A and B as in Figure 8. Each area is represented with one or more circles such that the circles cover the whole area. Area A is a rectangle that is represented with two circles; area B is a square that is represented with a circle.

These representations however are not precise enough. When a user *u1* resides between area A and B, but is within the circles representing area A and B, the geo-fencing area database may inform that user *u1* is located in area A or B, depending on the detection method. To increase the precision level, an area should be represented with many smaller circles. In another case, a user may reside in area B, but is within the circles representing area A and B. Consequently, in either case, the database must do more calculation and comparison to determine the area within which a user is located.

In addition, the precision also depends on the precision of the location coordinates given by GPS.

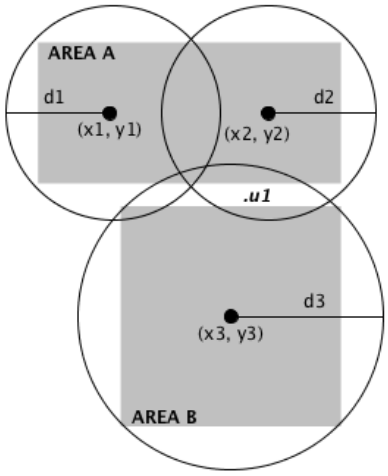

Figure 8 Geo-fencing area representation

3.3 Implementation

We specify that this router is to be implemented as a Web service. In our research, we specify this router as a RESTful Web service [33] [34]. In our research, we take the provisioning of airport information [35] as a case study.

3.3.1 Service Interface

In RESTful Web services, resources are identified using URI (uniform resource identifier). A resource can be a single information item or a set of information items. Filters can be applied to select a subset of information items from a larger set of information items.

In the case study, a client (i.e. a mobile application running on a mobile device) reads resources in a target service provider via a service router. We use HTTP GET methods to make requests. The location coordinate of the device is included as parameters of GET methods. A location coordinate is a pair of longitude and latitude values.

To enable a caching mechanism, the service router must be able to relate requests from the same user. We indicate the relation by user identifier. This identifier is also included as a parameter of GET methods.

Therefore, we define the following format to make requests to service interface:

```
http://url/resource/[filter]?long=x&lang=y&
id=z
```

where **url** is the service router's URL; **resource** is the information item to read; **filter** is a keyword to narrow down the resource resulted (filter is optional); **x** and **y** are the longitude and latitude value of the device's location coordinate, respectively; and **z** is the user identifier. Users are identified by their email addresses. We assume that a user actively uses the application on one device only.

For example, a user Johan wants to read information about the arrival of flight GA204 when he is at Adisucipto Airport. His location coordinate is (-7.7866503, 110.4293701). He has logged into the application using identifier `johan@situ.com`. The request is thus as the following:

```
http://servrouter.com/arrivals/GA204?
long=7.7866503&lat=110.4293701&
id=johan@situ.com
```

where **servrouter.com** is the service router's URL; **arrivals** is the information item to read that return a list of flight arrivals; and **GA204** is a filter so that this request returns only the information about the arrival of flight GA204.

Since the information provider is a CMS, the responses would be in HTML (hypertext markup language). It is the task of the mobile application to reformat the information presentation.

3.3.2 Provider Resolver

Requests received by the service interface are given to this component. The main task of this component is to identify target providers that must serve those requests. The identification process is done by calculating the location coordinates passed as the parameters of the requests. For doing that, this component has to consult with the area-URL database. The identification process might need an efficient algorithm since there can be large numbers of providers and simultaneous requests.

This component provides the following methods to determine the URL of a target service provider.

```
String getURL (long, lat, id)
String getURLfromDB (long, lat)
```

The first method is to retrieve the URL of a targeted service provider from the cache unit. The second method is to retrieve the URL from the database.

Internally, this component has a method for storing the URL of a found targeted provider for a user in a cache unit.

```
void store(id, url)
```

When the location of a mobile user is identified, this component returns the URL of the provider to which the requests from that user should be forwarded. An example is illustrated in Figure 9. *Area1* is an area that is registered in the service router's database. *provider1* is a provider that serves requests from users in *area1*. When a request is received from *user1* that is within *area1*, this component should return *url1*, i.e. the URL of *provider1*. Meanwhile, when a request is received from *user2* that is outside of *area1*, this component should create a message for notifying *user2* about the user's situation, instead of returning *url1*.

3.3.3 Geo-fencing Areas Database

This component stores information about registered areas and providers that must serve mobile users at those areas. In most cases, an area might not be simply represented by a circle area. An area could be in any shape. More representative approaches can therefore be developed, e.g. using polygons or multiple circle areas covering that area (as illustrated in Figure 8).

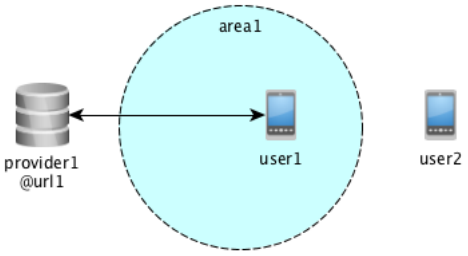

Figure 9 Users in and out of location

We represent a geo-fencing area (i.e. an airport in our case study) with one or more circles. In this database, we define the following structure to store information of a circle that (partially) represents an area:

```
<long, lat, radius, url>
```

where **long** and **lat** are the longitude and latitude value of the circle's centroid, respectively; **radius** is the circle's radius; and **url** is a service provider's URL that is assigned to serve requests from that area.

For example, information about geo-fencing areas in Figure 8 are stored as follows.

```
<x1, y1, d1, urlA>
<x2, y2, d2, urlA>
<x3, y3, d3, urlB>
```

In our case study, we do not deal with overlapping circles that represent different geo-fencing areas since neighboring airports usually have very far distances.

3.3.4 Cache Unit

Once a user's location is identified, an association between a mobile application and a service provider is stored in the cache unit. The association is stored for

a finite duration of time by considering how long a typical user stays in a geo-fencing area. E.g., in an airport, users might stay about half to one hour. Thus, storing an association for five or ten minutes can be expected to reduce the number of interactions to the geo-fencing area database. When a user sends requests from the same airport after the time expires, a new association is established.

We define the following structure to store association between a mobile application and service provider:

<**id, url, timestamp**>

where **id** is the application's unique identifier; **url** is the service provider's URL that is associated to the application; and **timestamp** is the time when the association is established. The time duration is specified in a separated configuration file, since it applies to all associations within the service router. For now, we assume that a service router supports one application only.

A garbage-collection mechanism might be necessary to ensure that the cache unit stores active associations only. It is implemented as an active process that periodically scans the cache and removes expired associations.

3.3.5 Request Forwarder

This component forwards requests from mobile users to a targeted provider. This is implemented as request calls to a targeted service provider. Since the service provider have the same interface as of the service router, we use the same format to make requests to service providers:

```
http://url/resource/[filter]?long=x&lang=y&
id=z
```

where **url** is now the service provider's URL. Other components are the same as ones in the original request to the service router. For example, the previous original request is forwarded as

```
http://airportJOG.com/arrivals/GA204?
long=7.7866503&lat=110.4293701&
id=johan@situ.com
```

where **airportJOG.com** is the targeted provider's URL address. Users' location coordinates (i.e. longitude and latitude) should also be included in the forwarded requests. Actually these parameters are not needed by the service provider. They are necessary to allow a *hierarchical* architecture (as in Figure 4) to work.

4.0 CONCLUSION

In this paper, we have specified an architecture for geo-fencing applications with multiple information providers. The architecture allows users in different geo-fencing areas be served by different service providers. As long as a user stays in the same geo-fencing area, the same service provider serves that user. This architecture is naturally fit with real-world situation. It supports consistent information presentation by allowing a mobile application to restructure and reformat information from multiple providers when users are traveling across several areas. Also, it can be extended into a hierarchical architecture to make the architecture scalable.

We have also presented in more detail the central component of the architecture, namely service router. The service router receives requests from mobile users, identifies target information providers, and forwards the requests to the targeted providers. When responses are received from the providers, the service router sends those responses back to the originating users.

Furthermore, we have presented a location-based request forwarding mechanism to be employed in the service router. The mechanism considers the users' current geographical locations in determining target service provider. The mechanism includes a caching mechanism to make efficient the request forwarding process. We have also described an implementation of that mechanism with a case study of airport information service provisioning.

Many literatures have presented geo-fencing application and LBS architectures for different service goals, requirements, and types. All those architectures assume that information is provided from a single provider. Our proposed architecture differs from them in that it accommodates multiple service providers; each is for different area. Hence, our architecture needs a service router to identify target service providers and forward request to them.

We foresee that service providers would be implemented using a specific content management system (CMS) [35]. In the future, we will investigate how to facilitate the development of service interface. Also, we will further investigate alternative algorithms for provider identification in a service router.

Acknowledgement

This work is part of a research project funded by the Directorate of Research and Community Service, the General Directorate of Higher Education, the Ministry of Education and Culture, Republic of Indonesia, contract no. 001/HB-LIT/III/2015.

This paper is an extended version of papers presented in 2015 IEEE Conference on Open Systems, Melaka, Malaysia and 3rd International Conference on Technology, Informatics, Management, Engineering & Environment 2015.

References

[1] Küpper, A. 2005. *Location Based Service: Fundamental and Operation*. Chichester: John Wiley & Sons.

[2] Virrantaus, K., J. Markulla, A. Garmash, V. Terziyan, J. Veijalainen, A. Katanosov, and H. Tirri. 2001. Developing GIS-Supported Location-Based Services. *2nd International Conference on Web Information System Engineering (WISE)*. Kyoto, Japan. 3-6 December 2001. Vol. 2: 66-75.

[3] Open GIS Consortium Inc. 2003. *OpenGIS Location Service (OpenLS): Core Services*. http://www.opengeospatial.org/standards/ols

[4] Reclus F. and K. Drouard. 2009. Geofencing for Fleet & Freight Management. *9th International Conference on Intelligent Transport Systems Telecommunications (ITST)*. Lille, France. 20-22 October 2009. 353-356.

[5] Dirgahayu, T., N. Setiani, and F. Wijayanto. 2015. An Architecture for Location-Based Service with Multiple Information Providers. *2015 IEEE Conference on Open Systems*. Melaka, Malaysia. 24-26 August 2015. 119-123.

[6] Oliveira R.R., I.M.G. Cardoso, J.L.V. Barbosa, C.A. da Costa, M.P. Prado. 2015. An Intelligent Model for Logistics Management Based on Geofencing Algorithms and RFID Technology. *Expert Systems with Applications*. 42(15–16): 6082-6097.

[7] Spiekermann, S. 2004. General Aspects of Location-Based Services. In J. Schiller and A. Voisard (eds). *Location-Based Services*. San Fransisco: Elsevier.

[8] Kuo, M.H., L.C. Chen, and C.W. Liang. 2009. Building and Evaluating a Location-Based Service Recommendation System with a Preference Adjustment Mechanism. *Expert systems and Applications*. 36: 3543-3554.

[9] Tumas, G. and F. Ricci. 2009. Personalized Mobile City Transport Advisory System. *Information and Communication Technologies in Tourism*. Springer: 173-183.

[10] García-Crespo, A., J.L. López-Cuadrado, R. Colomo-Palacios, I. González-Carrasco, and B. Ruiz-Mezcua. 2011. Sem-Fit: A Semantic Based Expert System to Provide Recommendations in the Tourism Domain. *Expert Systems with Applications*. 38(10): 13310-13319.

[11] Zheng, Y., Y. Chen, X. Xie, and W.Y. Ma. 2009. GeoLife2.0: A Location-Based Social Networking Service. *10th International Conference on Mobile Data Management (MDM): Systems, Services and Middleware*. Taipei, Taiwan. 18-21 May 2009. 357-358.

[12] Shankar, P., Y.W. Huang, P. Castro, B. Nath, and L. Iftode. 2012. Crowds Replace Experts: Building Better Location-Based Services using Mobile Social Network Interactions. *IEEE International Conference on Pervasive Computing and Communication (PerCom)*. Lugano, Switzerland. 19-23 March 2012. 20-29.

[13] Zhenyu, W., Z. Chunhong, J. Yang, and W. Hao. 2010. Towards Cloud and Terminal Collaborative Mobile Social Network Service. *IEEE 2nd International Conference on Social Computing (SocialCom)*. Minneapolis, USA. 20-22 August 2010. 623-629.

[14] Aalto, L., N. Göthlin, J. Korhonen, and T. Ojala. 2004. Bluetooth and WAP Push Based Location-Aware Mobile Advertising System. *2nd International Conference on Mobile Systems, Applications, and Services (MobiSys)*. Boston, USA. 6-9 June 2004. 49-58.

[15] Ververidis, C. and G. Polyzos. 2002. Mobile Marketing using a Location Based Service. *1st International Conference on Mobile Business (M-Business)*. Athens, Greece. 8-9 July 2002.

[16] Sheth, A., S. Seshan, and D. Wetherall. 2009. Geo-fencing: Confining Wi-Fi Coverage to Physical Boundaries. In H. Tokuda, M. Beigl, A. Friday, A.J. Bernheim Brush, and Y. Tobe (eds). *Pervasive Computing*. LNCS 5538. Heidelberg: Springer-Verlag.

[17] Klasnja, P., S. Consolvo, J. Jung, B.M. Greenstein, L. LeGrand, P. Powledge, and D. Wetherall. 2009. When I am on Wi-Fi, I am Fearless: Privacy Concerns & Practices in Everyday Wi-Fi Use. *ACM SIGCHI Conference on Human Factors in Computing Systems*. Boston, USA. 4-9 April 2009. 1993-2002.

[18] Martin, D., A. Alzua, and C. Lamsfus. 2011. A Contextual Geofencing Mobile Tourism Service. In R. Law, M. Fuchs, and F. Ricci (eds). *Information and Communication Technologies in Tourism 2011*. Mörlenbach: Springer-Verlag/Wien. 191-202.

[19] Schneider, G., B. Dreher, and O. Seidel. 2008. Using Geofencing as a Means to Support Flexible Real Time Applications for Delivery Services. *5th International Workshop on Ubiquitous Computing (IWUC)*. Barcelona, Spain. 12-16 June 2008. 22-27.

[20] Wong, A.K.S., T.K. Woo, A.T.L. Lee, X. Xiao, V.W.H. Luk, and K.W. Cheng. 2009. An AGPS-Based Elderly Tracking System. *1st International Conference Ubiquitous and Future Networks (ICUFN)*. Hong Kong. 7-9 June 2009. 100-105.

[21] Wang, Y. and A. Potter. 2007. The Application of Real Time Tracking Technologies in Freight Transport. *3rd International IEEE Conference on Signal-Image Technologies and Internet-Based System (SITIS)*. Shanghai, China. 16-18 December 2007. 298-304.

[22] Carr, N. and P. McCullagh. 2014. Geofencing on a Mobile Platform with Alert Escalation. In L. Pecchia, L. Chen, C. Nugent, and J. Bravo (eds). *Ambient Assisted Living and Daily Activities*. LNCS 8868. Springer. 261-265.

[23] Pongpaichet, S., V.K. Singh, R. Jain, and A.S. Pentland. 2013. Situation Fencing: Making Geo-fencing Personal and Dynamic. *1st ACM International Workshop on Personal Data Meets Distributed Multimedia*. Barcelona, Spain. 21-25 October 2013. 3-10.

[24] Szczytowski, P. 2015. Geo-fencing Based Disaster Management Service. In F. Koch, F. Meneguzzi, and K. Lakkaraju (eds). *Agent Technology for Intelligent Mobile Services and Smart Societies*. Springer. 11-21.

[25] Blazevic, L., J.Y. Le Boudec, and S. Giordano. 2005. A Location-Based Routing Method for Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing*. 4(2): 97-110.

[26] Füßler, H., M. Mauve, H. Hartenstein, M. Käsemann, M., and D. Vollmer. 2003. Location-Based Routing for Vehicular Ad-Hoc Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*. 7(1): 47-49.

[27] Liao, W.H., J.P. Sheu, and Y.C. Tseng. 2001. GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks, *Telecommunication Systems*. 18(1-3): 37-60.

[28] Camp, T., J. Boleng, B. Williams, L. Wilcox, W. Navidi. 2002. Performance Comparison of Two Location Based Routing Protocols for Ad Hoc Networks. *INFOCOM 2002*. New York, USA. 23-27 June 2002. Vol. 3: 1678-1687.

[29] Dirgahayu, T., and F. Wijayanto. 2015. Location-Based Request Forwarding in A Geo-fencing Application with Multiple Providers. *3rd International Conference on Technology, Informatics, Management, Engineering & Environment (TIME-E) 2015*. Samosir Island, Indonesia. 7-9 September 2015. 93-98.

[30] Christine Roy, M., O. Dewit, and B.A. Aubert. 2001. The Impact of Interface Usability on Trust in Web Retailers. *Internet Research*. 11(5): 388-398.

[31] Folmer, E., J. van Gurp, and J. Bosch. 2003. A Framework for Capturing the Relationship between Usability and Software Architecture. *Software Process: Improvement and Practice*. 8(2): 67-87.

[32] George, C.A. 2009. Usability Testing and Design of a Library Website: An Iterative Approach. *OCLC Systems & Services: International Digital Library Perspectives*. 21(3): 167-180.

[33] Fielding R.T. and R.N. Taylor. 2002. Principled Design of The Modern Web Architecture. *ACM Transactions on Internet Technology*. 2(2): 115-150.

[34] Christensen, J.H. 2009. Using RESTful Web-Services and Cloud Computing to Create Next Generation Mobile Applications. *24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*. Orlando, USA. 25-29 October 2009. 627-634.

[35] Dirgahayu,T., N. Setiani, and Z. Zukhri. 2014. Information Requirement Engineering for Specific Content Management Systems. *2014 IEEE Conference on Open Systems*. Subang Jaya, Malaysia. 26-28 October 2014. 54-59.