Journal of Web Engineering, Vol. 13, No.1&2 (2014) 001-023

ENHANCED LBS DISCOVERY IN A DECENTRALIZED REGISTRY BASED WEB SERVICES ENVIRONMENT

MELWYN D'SOUZA V.S. ANANTHANARAYANA

National Institute of Technology Karnataka, Surathkal, India
melwyn_in@yahoo.com anvs@nitk.ac.in

Received August 30, 2012
Revised July 25, 2013

Location Based Services (LBS) is the most happening thing in the mobile industry today. Everybody is trying to generate revenue from location based services. Mobile phone manufacturers are developing new smart phones every day and network providers are offering high speed data connections. Several LBS providers and applications are available in the market but the major problem is service provider dependency. This paper gives an overview of a decentralized registry based architecture using web services technology which facilitates dynamic discovery, interoperability and provider independence. The web services technology uses UDDI registry service for publishing and discovering services but the discovery results obtained are not reliable as the service discovery considers only static service description. This paper contributes to enhancing LBS discovery by considering service dynamics and expanding LBS discovery process to neighboring locations.

Key words: location based services, web services, SOA, service discovery, query propagation
Communicated by: M. Gaedke & S. Murugesan

1 Introduction

Mobile phones have now become a basic necessity of life as they help us connect with each other ubiquitously. As a result, new mobile phone manufacturers and mobile network operators are emerging in the market everyday and thus the technology is improving tremendously. On the other hand, mobile users are constantly in search of enhanced features in their mobile phones and improved services from mobile operators. To satisfy the end user requirements mobile phone manufacturers today are in the market with smart phones which are like small computers. On the other hand, mobile network operators are working on improved infrastructure to provide reliable and high speed networks at affordable cost but their profit share is decreasing due to stiff competition between them. Mobile operators are now trying to generate revenues from various add-on services in addition to those from voice services. They began delivering value added services (VAS) like short messaging service (SMS), multimedia messaging service (MMS) and general packet radio service (GPRS) to attract more customers and increase revenue per user (RPU). Due to advanced technology and infrastructure, mobile operators are now able to provide high speed internet connection over mobile networks which

have opened several avenues for them to generate additional revenue. Location based services (LBS) is one such area.

Services provided to mobile users according to their geographic location are known as location based services [1]. Requesting the location of nearest ATM, petrol pump, restaurant, hospital and ambulance services are some examples of location based services in a mobile environment. Location based services exploit the position of mobile terminals to provide customized services that are relevant to the current location of mobile users. Thus, location is an essential parameter for all location based services.

LBS is a fast-growing research area. LBS has very high potential to generate revenue to service providers, mobile operators, mobile phone manufacturers and application developers. According to ABI Research forecast [2], mobile LBS revenue is expected to reach an annual global total of $13.3 billion by 2013, up from an estimated $515 million during 2007. According to eMarketer forecast [3], the number of worldwide users of mobile location based services is expected to jump to 486 million in 2012 from 18.9 million in 2007. Navigation services, social networking and local search services are believed to emerge as widely used applications by the mobile users.

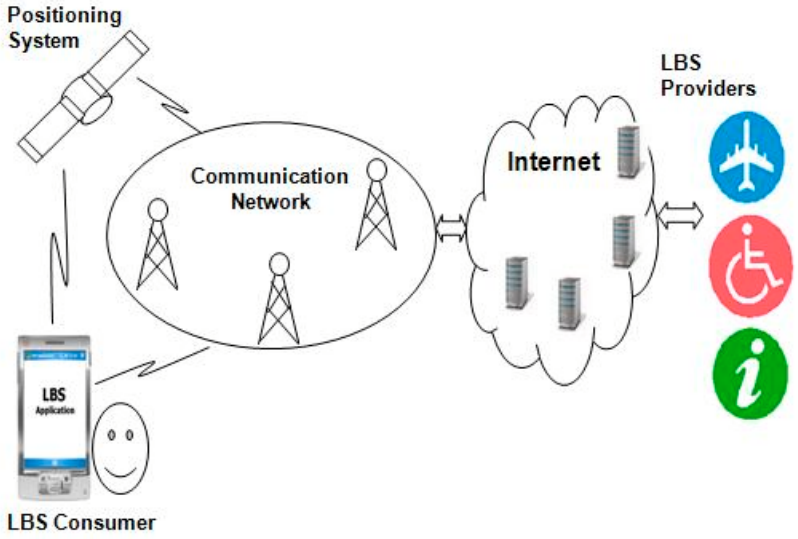

Figure 1. Mobile LBS Components

LBS system basically consists of five components [4]: Mobile device, LBS Application, Positioning System, Service Provider and the Communication network as shown in figure 1. Mobile device is a tool for the end user to use LBS application and to consume services. LBS application is the one through which end user consumes location based services. Positioning system is responsible for providing the end user's position information. The mobile user position can be obtained by either using the global positioning system (GPS) or the mobile communication network. The Service provider is responsible for processing the service requests and providing various services. Communication network is the one which is responsible for communication between the mobile device and LBS providers.

At present, mobile LBS applications provided by the service providers are tightly coupled with their own systems. This tightly coupled architecture helps service providers in security and billing but results in service provider dependency to service consumers. The best solution here is to use service oriented architecture which supports dynamic discovery of services to avoid service provider dependency. Service oriented architecture (SOA) [5] is the latest paradigm in the evolution of software development aimed at facilitating the design and development of dynamic, flexible and loosely coupled platform independent applications. SOA uses the publish-find-bind and execute paradigm to facilitate dynamic discovery of services. Service providers publish their services by registering them in a registry. This registry is used by the service consumers to find required services. The registry provides consumers with a contract and an endpoint address for those services which match the search criteria. Web Services technology is the preferred standards-based solution to realize SOA. Web services [6] are loosely coupled components and use open standards to provide interoperability between various applications. Web Services support openness and heterogeneity in addition to the dynamic discovery of services by using XML, SOAP, WSDL and UDDI open standards. The data is tagged with XML [7] and transferred using SOAP [8]. Services are described using WSDL [9] and published in UDDI [10]. The block diagram of service oriented architecture using web services technology is shown in figure 2. Applying Web services technology to the mobile environment is known as Mobile Web Services (MWS) [11]. User mobility with their terminals is a distinguishing characteristic of Mobile Web services compared to conventional Web services.

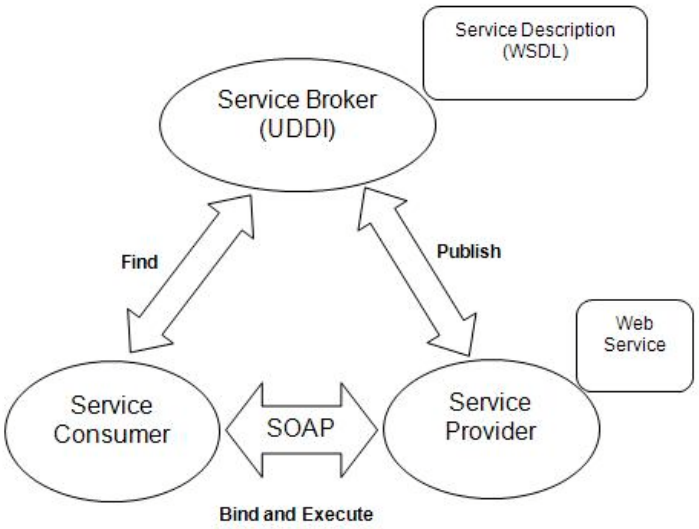

Figure 2. SOA using Web Services technology

One of the most important sub tasks in service oriented architecture is service discovery. Discovery corresponds to the activity of locating a resource that meets certain functional criteria. Web service discovery is the process of locating information that describes a particular web service using the web services description language (WSDL). It is through the discovery process that web service clients learn that a web service exists and where to find the web service's description document. The Inquiry API is used to locate and obtain details about web service entries in a UDDI registry.

In SOA, service broker, usually a registry is the one which facilitates dynamic discovery of services. For LBS, a centralized registry model is probably the most efficient in terms of design, configuration, control and maintenance. In this case, mobile clients only have to maintain the location information of one registry. The main drawbacks as is in any centralized system lie in the bottleneck of the central registry and the inactivation of entire system when the central registry shuts down or crashes. Centralized registry models results in lower performance if there are too many services to be registered or queried. Furthermore, storage may be a constraint when the number of service registrations grows very large, since one computer node must host all service registrations. Also, service providers must include location scope of each service in the service description.

Registry replication attempts to overcome the disadvantages of the centralized approach for LBS by replicating the entire information and putting them on different sites. However, replication may temporarily improve the performance if the number of publishers and subscribers is limited. Also, data becomes less consistent when the system has more number of replication sites. The UDDI Business Registry (UBR) is an example for registry replication model.

Decentralization is a natural solution when a system grows large and becomes complex. There are several reasons why a decentralized approach is attractive in case of location-based services. It improves query processing as services are distributed on multiple servers and hence the response time. Decentralized systems have no weak point that can bring the entire system down if individual registries crash or go down and hence are robust against failure and attacks. A decentralized registry based architecture for LBS using web services technology is explained in [12].

Though it has become possible to discover and consume web services on mobile devices, reliable service discovery is the most challenging and an essential requirement in the mobile environment. Cellular phone is the most widely used mobile device but it still has many limitations like low storage, low processing power, small screen and difficult to use keyboard. Cellular network's low bandwidth and intermittent connectivity are also major concerns.

Web services architecture uses UDDI, a registry that employs a search service based on keyword matching. Pure keyword based search fails to retrieve services as syntactically different query keywords may be semantically equivalent and UDDI does not have any functionality to infer from keywords. Also, keyword based search fails to understand similarities and differences between the capabilities provided by web services. One of the solutions to overcome the drawbacks of keyword based matching is semantic search where concepts are used instead of words for matching. OWL-S (Web Ontology Language for Web Services) [13], WSMO (Web Service Modelling Ontology) [14] and WSDL-S (Web Service Semantics) [15] are some of the well known standards developed in this regard. Though the semantic matching improves the search mechanism, there is no guarantee that web services included in the result set are actually available over the internet.

UDDIe [16] is an extended registry for web services, which includes finite and infinite lease time concept to avoid missing or out of date services information in registries. However, the lease time does not ensure the real availability of web services as the relationship is established in advance between service providers and the registry. Also, there is no functionality for checking the real availability of web services. Ad-UDDI [17] introduces active monitoring of web services by checking the real time status of services and collecting the services information periodically. By active monitoring, even if

the availability of web services is ensured, there is no guarantee that the requested real world (abstract) service is available at the moment. For example, movie ticket reservation web services might be available but there is no guarantee that tickets for a particular show are available at the time of service invocation. Contribution of our work is to enhance the web services discovery process by using web services availability and abstract service availability checks. This paper also explains service availability time concept which avoids invoking web services for availability check whenever real world services are not available and thus improves response time.

In a mobile environment, mobile users are interested in consuming services which are available at the current location. If the required services are not available at the current location then they would prefer to consume services available at immediate neighboring locations. In this case, mobile users need an automated system which helps them to search for services available at the current location and expand this search to immediate neighboring locations if the required services are not available at the current location. Another important aspect of mobile environment is user mobility with their terminals which leads to change in user location. Change in user location is very likely when a mobile user is travelling in a vehicle or roams around the border of a location. The automated service search system should consider this change of user location during the search process so that results returned are corresponding to the current location of the mobile user. The term handover or handoff is used in cellular telecommunication system to refer the process of transferring an ongoing call or data session from one cell connected to another. This process of handover indicates the change in user location.

A distributed registry based architecture which explains how to move a web service execution from one service execution server to another is proposed in [18]. During the service execution, if mobile client moves from one location to another then the proposed system finds an execution server at the new location and transfers the service execution to that server. The basic idea here is to have minimum distance between the service consumer device and service execution server so that the data transfer path between them is very short. In [19], authors have explained how to move service registrations from one registry to another and maintain caching while a service provider moves from one location to another. This work uses peer-to-peer architecture in which service providers host web services on their mobile terminals. When mobile service providers move from one location to another, it is required to unregister their services from the source location registry and register at the destination. Also, authors have used flooding concept to forward queries but change in user location is not considered. Contribution of our work is the search query propagation when user changes his location or required services are not available at the current location using one-hop and two-hop neighbor information.

Most of the recent literature contributes significantly on location privacy. Location attribute is an inherent property of all location based services. Many researchers are working on location privacy issue as sharing location information with unknown people is very risky. L2P2 [20], VERDICT [21], 3PLUS [22] and PShare[23] are some of the latest privacy preserving schemes. With respect to user specific service delivery, a data mining based LBS delivery mechanism is explained in [24]. In this data mining based method, various usage patterns are utilized to deliver user specific services. A semantic web based LBS delivery system is explained in [25]. Semantic web technologies help in developing more intelligent location-based applications that deliver more accurate and relevant information to the user.

The main objective of our work is to explain a decentralized registry based web services discovery system that allows service consumers to find and consume services available within proximity. This paper starts with the overview of decentralized registry based architecture for LBS using web services technology and explains how to search local services in such an environment in section 2. It then proceeds with the classification of LBS based on various characteristics of abstract services in section 3 and contributes in enhancing LBS discovery by using web services availability and abstract service availability check in section 4. Other contribution of our work is the search query propagation to immediate neighboring locations in case of required services not being available at the current location or in case of mobile user changing his location after submitting the query but before receiving the results is explained in section 5. The proposed LBS discovery process is intended to improve reliable and reduce query execution total cost and wireless bandwidth consumption. We conclude general observation of our work in section 6.

2 The overview of proposed LBS System

2.1 Registry Organization

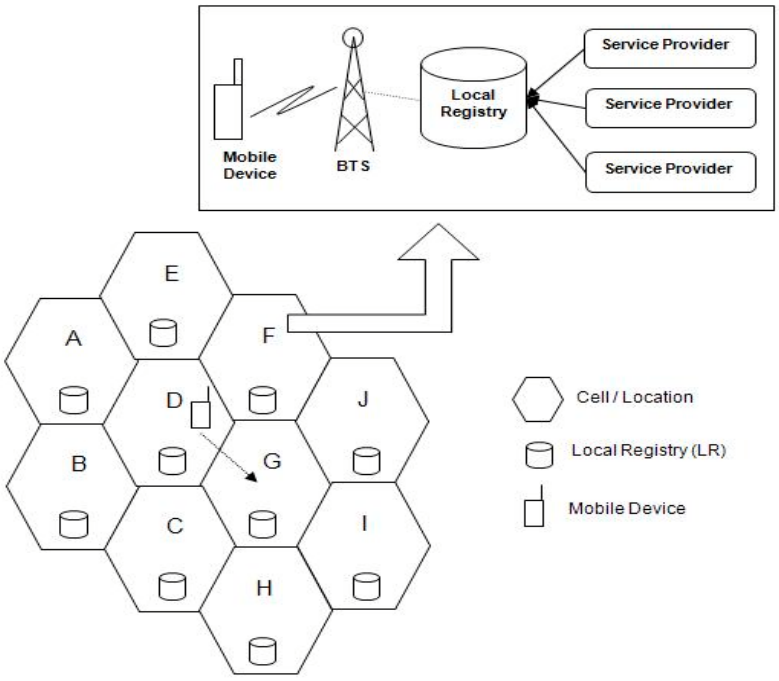

Figure. 3. Decentralized registry organization for LBS.

In the proposed system, cellular network system is considered in order to divide the entire geographical area into locations. In a cellular network system [26], a cellular service area is divided into smaller areas called cells. A cell or group of cells is considered as a location in the proposed system. Whenever a mobile user is roaming in cell X, the user is considered to be in location X. Service registry is decentralized and registries are placed in each of these locations. Registry located in

each location or cell is called as local registry (LR). The local registry is the one which allows service providers to register their services and service consumers to search for required services. Service providers publish interface and binding information of their services to the local registry. Service consumers use this information to bind and execute provider services. Each local registry contains service registrations belonging only to its own corresponding location. Search response time improves when compared to centralized registry based system due to less number of service registrations in each LR and location based service search accuracy increases as LR contains only local service registrations. Decentralized registry organization in a cellular network is shown in figure 3 along with the main components of LBS system at each location.

2.2 Registry Discovery

In the proposed system, by-broadcast mode has been used to discover the local registry. In the by-broadcast mode, a broker in every location periodically broadcasts the local registry address over the wireless channel. Mobile clients listen to the wireless channel and download the local registry address to discover local services. In the cellular network system, each cell is served by a base transceiver station (BTS), also known as a base station (BS). Each BTS broadcasts both the location area identity (LAI) and the Cell-ID on the broadcast control channel to its cell. A mobile terminal always knows its Cell-ID and LAI as it always receives these broadcast messages. In the proposed method, the cellular network base station is configured to broadcast the local registry address along with other existing parameters.

Following is the J2ME source code which is used to get the Cell-ID and LAI of user's current location.

String cellid = System.getProperty("CellID");

Similarly, in the proposed system, usage of the below given code is suggested to get the local registry address.

String lraddress = System.getProperty("LRAddress");

The other option is to use the cell global identity (CGI) as part of local registry address. The cell global identity (CGI) is a major network identity parameter. CGI consists of location area identity (LAI) and cell identity (CI). LAI includes mobile country code (MCC), mobile network code (MNC), and location area code (LAC). For example, for a given cell, if MCC=111 and MNC=22 and LAC=33333 and CI=12345 then the address of local registry could be represented in terms of web URL as www.111223333312345.reg. In case of multiple cells grouped as a single location, one could configure multiple URLs to point to the same LR which is associated with that location. Once the local registry address is obtained, users can proceed with the service discovery process.

2.3 Functional Details

The location based application functionality mainly consists of three steps. They are local registry discovery, local service discovery and local service consumption as shown in figure 4. Whenever a mobile user is interested in searching for local services, the location based application listens to the base stations broadcast channel and downloads the local registry address (step 1). Once the local registry address is discovered, it proceeds with the web services discovery process (step 2). It then

binds and executes a web service based on the user's choice (step 3). An algorithm is given below with the steps which are required to be executed to search and consume local services by the location sensitive applications.

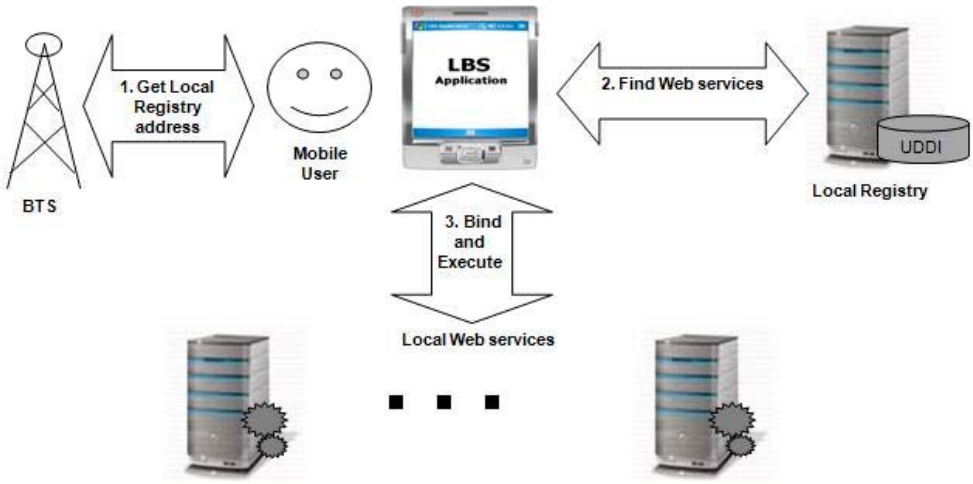

Figure 4. Functional flow diagram of a decentralized registry based web services environment.

Algorithm: Search&ExecuteLocalServices

/* Executed by Location based applications whenever mobile users need to consume local services */.

Begin

1. Get location sensitive application input from the mobile user.
2. Get Local Registry address by listening to the BTS's Broadcast Channel.
3. Find Web services in the local registry by using the address obtained in step 2 and passing the input parameters obtained in step 1.
4. Download binding information of a Web Service based on the user's choice.
5. Bind and execute the selected Web service.
6. Display results to the user.

End

3 LBS Classification

Location based services could be broadly classified into several categories based on the various characteristics of abstract services. It is very important to understand the difference between web services and abstract services to classify LBS.

Abstract services are the real world services offered by service providers. For example, in case of a hotel offering twenty five double bed rooms, each double bed room is an abstract service. The availability of these abstract services is dynamic in nature and hence changes over time. For example, a hotel will not be able to book a room with double bed on a specific date if all such rooms in the hotel are already booked on that date.

Web services are computational entities using standardized interfaces that allow service consumer applications to interact with a provider to consume abstract services. Basically, it is a tool to automate abstract service consumption. More about abstract services & web services can be found in [27].

Classification of Location Based Services is given below:

1. Based on the abstract service provider
    a. **Owner Services** are the services provided by the abstract service providers themselves. In this case, web service provider and abstract service provider is the same entity. For example, a hotel room reservation web service published by the hotel owner.
    b. **Agent Services** are the services provided by the providers who are not the abstract service providers. In this case, web service provider and abstract service provider are different entities. For example, an agent providing room reservation services at different hotels using his own web service.
2. Based on nature of the abstract service
    a. **Services of Information** are the services which provide requested information to consumers. These are like read-only data accessing services. For example, weather information, nearest ATM details, nearest gas station information, etc.
    b. **Services of Transaction** are the services which allow consumers to carry out some transactions. These are like updatable data accessing services. For example, room reservation, ticket reservation, ordering food, etc.
3. Based on the number of types of abstract services involved
    a. **Atomic Service** is the one which provides a single type of abstract service. For example, hotel room reservation.
    b. **Composite Service** is the one which provides multiple types of abstract services. For example, a local tour package service which includes hotel room reservation service and taxi booking service.
4. Based on the time of abstract service availability
    a. **Time dependent Services** are the ones which are available only during particular time of the day. For example, a restaurant service available from morning 6:00AM to evening 9:00PM.
    b. **Anytime Services** are the ones which are available round the clock. For example, weather information service.
5. Based on abstract service provider mobility
    a. **Fixed Location Services** are services that are provided at a fixed location. Most of the location based services fall under this category. For example, hotel rooms, restaurants, etc.

   b. **Mobile Services** are the ones that are not provided at a fixed location. For example, taxi service, ambulance service, etc.

6. Based on the abstract service provider's proximity
   a. **Local Services** are the ones whose provider is available within the current locality. It means that the service provider exists within some approachable distance from the service consumer. For example, restaurants, ambulances, taxi, hospital, etc.
   b. **Remote Services** are the ones whose provider is not available within the current locality. In this case, the scope of the service matters more than the distance. For example, a weather service provider located in USA could provide weather services to any part of the world based on the name of location or the PIN code.

4 Enhancing LBS Discovery

As discussed in the introduction section, service discovery is one of the most important sub tasks in service oriented architecture. In the proposed architecture, UDDI service is used as a broker to facilitate dynamic search of location based services. UDDI is a search service based on keyword matching which considers static information provided in the web service description and does not consider dynamics like web service availability and abstract service availability. Web service description available in UDDI registry does not guarantee the availability of web service and abstract service availability. Thus, in order to improve the LBS discovery, service availability check module is included in the local registry and the details are given below.

4.1. Service Availability

In a decentralized registry based web services environment for location based services, web service discovery and execution occurs dynamically. In such a dynamic environment, successful execution of web services depends mainly on the web service discovery. The execution of a web service may fail due to any of the following reasons:

a) Web service is not available
b) Abstract service is not available
c) Runtime error

Web service availability and abstract service availability could be traced before the execution of a service, whereas, runtime errors could be traced only during the execution of web services. The existing UDDI is a keyword based matching service which is not considering web service availability and abstract service availability as part of service discovery and hence the results obtained by the UDDI Inquiry API are not completely reliable. This paper considers the web service availability and abstract service availability as part of service discovery to improve the reliability of results obtained during the service discovery.

The web service availability check could be done by using the following pseudo-code. This pseudo-code is written by assuming that each web service has a standard function called isAvailable() which returns a value True upon execution.

```
Try
{
        isAvailable = webService.isAvailable();
}
Catch(Exception)
{
        isAvailable = False;
}
```

The Abstract service availability check could be done by using the following pseudo-code. This pseudo-code is written by assuming that each web service has a standard function called isServiceAvailable() which accepts required parameters and returns relevant results.

```
Try
{
        isAvailable = webService.isServiceAvailable(parameter list);
}
Catch(Exception)
{
        isAvailable = False;
}
```

The parameter list differs from service to service. In case of a hotel room booking service, the parameters could be check-in date, check-out date, room type and number of people to reserve a room whereas in case of an airline ticket reservation, it could be source, destination, date of travel, number of adults, number of children and number of infants, etc.

To find web service availability and abstract service availability, it is not necessary to invoke the same web service twice. The network system will return a service not found error, “404: Not Found”, during the abstract service availability check if the web service is not available. But, during the experiment it was observed that the abstract service availability check takes a minimum of about 20 seconds even if there exists a single unavailable web service. This duration is the default timeout period when a web service is not available or the web server is not reachable. This increases web services query response time and hence it was decided to have a separate process to check web service availability. This process is placed under a scheduler and executes periodically.

Local registry architecture for the availability check process is shown in figure 5. As shown in the figure, services in UDDI are extended with three properties, Status, CTime and Count. The Status property could have three possible values. Value “Yes” indicates that the WS is available, “No” indicates that the WS is not available and “NULL” indicates that the WS is obsolete. The CTime property contains WS last checked date and time. The Count property contains a value which indicates how many times WS availability check returned False value from the last obtained True value.

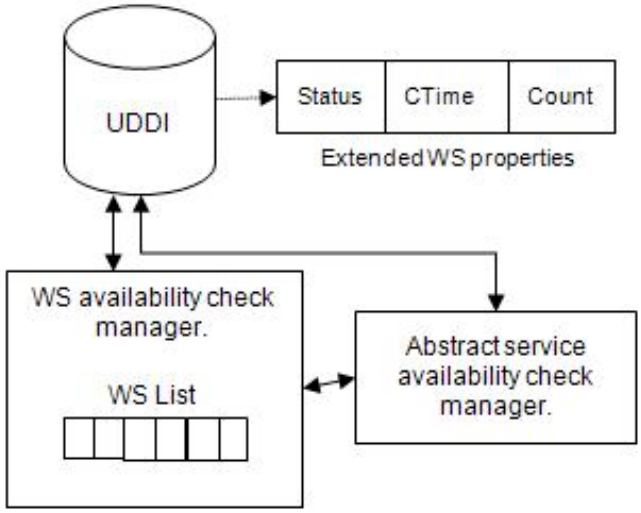

Figure 5. Local registry architecture for Service Availability check

During WS availability check or an abstract service availability check, if a WS is found available then the Status property would be set to "Yes" and Count to Zero. If the WS is not available then the Status property would be set to "No" and Count incremented by one. If the Count property exceeds some threshold value then the Status property would be set to "NULL" and an email sent to the corresponding service provider stating the WS has become obsolete due to WS unavailability. In the experiment, threshold value was set to ten. Thus, if a WS was found to be not available consecutively for ten times then the WS becomes obsolete and an email is sent to the provider. Note that, the CTime property would be set to current DateTime in case of an abstract service check process and current scheduled time in case of WS availability check.

Abstract service availability check is part of service discovery process. This process fetches only those web services from the UDDI where Status = "Yes" to avoid unavailable and obsolete web services. The WS availability check manager is a separate process, placed under the scheduler to execute periodically. It maintains a WS List which contains services that are to be invoked to check their availability. Whenever the scheduler starts this process, the WS List is filled with only those web services where (1) Status = "No" and (2) Status = "Yes" AND CTime <= WS availability check last executed time to avoid obsolete services and services which are already invoked by the abstract service check process after the previous execution of WS availability check process.

The abstract service availability increases the reliability of service discovery results but the drawback is that it requires additional time to check abstract service availability by invoking the web services. For example, if the UDDI returns description for hundred web services then all those hundred web services are required to be invoked to check the abstract service availability. This increases service discovery time which is not appropriate in mobile environment. Thus, the next challenge here is to reduce the service discovery time.

In the proposed system, it was observed that most of the location based services are local services and are time dependent services. For example, a restaurant service would not able to book a table for 10:00AM if the restaurant opens at 11:00AM, whereas a hotel room reservation service might be available twenty four hours a day. Time of abstract service availability depends upon individual service providers. Restaurant A might open at 8:00AM and close at 10:00PM, whereas Restaurant B might open at 6:00AM and close at 11:00PM. Almost all the restaurant services remain closed between 1:00AM and 5:00AM, whereas, most of them remain open between 7:00AM and 11:00PM. If the time of abstract service availability is made to be stored in the local registry as part of web service description then abstract service availability check time could be reduced significantly. Thus, service availability time property was also included in the local registry to reduce number of WS invokes during abstract service check.

4.2. Experimental Results

The proposed system was developed using various Microsoft tools. The local registry web service interface was developed by using Microsoft .NET web services which internally uses Microsoft UDDI as a service registry. BTS functionality which provides local registry address to the location sensitive applications was made available by using a web service. The Restaurant Finder client program which is a location sensitive application and runs on the mobile device was developed in Microsoft .NET CF 3.5. The client application was successfully tested on Windows Mobile 5.0 Pocket PC R2 emulator.

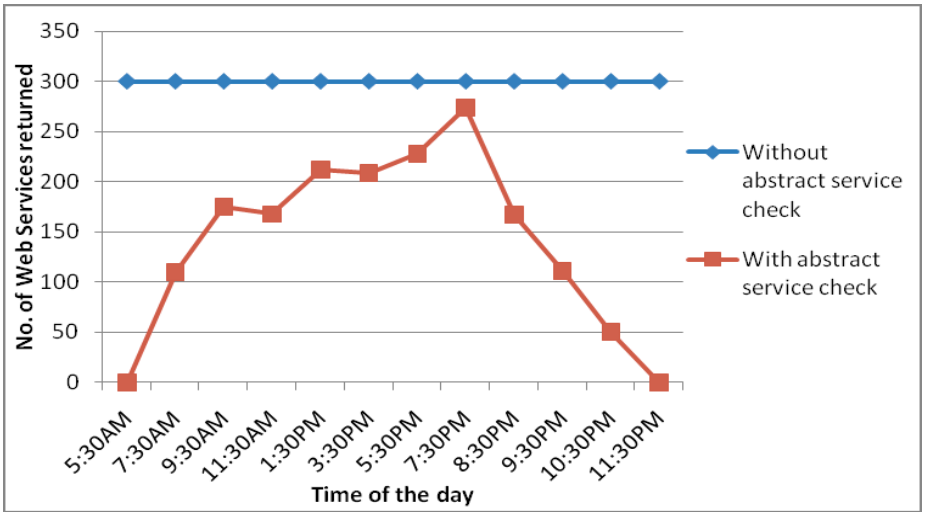

Figure 6. Number of web services descriptions returned by a local registry comparison: with abstract service check versus without abstract service check.

A comparison between the number of web services returned by the local registry with abstract service check and without abstract service check is shown in figure 6. It was observed that the local registry always returned 300 matching restaurant web services without the abstract service check. Whereas, when the abstract service check was applied, the results obtained were very promising. The

local registry returned zero web services between 11:30PM and 5:30AM as there was no restaurant open during that duration. It returned less than 200 web services during 5:30AM to 11:30AM and also 8:30PM to 11:30PM. It returned more than 200 web services only during 11:30AM to 8:30PM as most of the restaurants remain open during that time.

The availability check not only increases the reliability of the service discovery results but also reduces bandwidth consumption. For example, during the experiment, the local registry always returned 300 web service descriptions to the mobile device without the abstract service check and returned zero web service descriptions between 11:30PM and 5:30AM with the abstract service check. The average web services descriptions returned between 5:30AM and 11:30PM is 142 with the abstract service check against 300 web services descriptions without the abstract service check.

As it was mentioned earlier, the major drawback is the time consumed while invoking web services to check the abstract service availability. In the case of above mentioned experiment, local registry always invoked 300 web services to check the abstract service availability. In the experimental setup it was taking about 9 seconds time to invoke matching 300 restaurant web services to check abstract service availability but the results obtained were highly reliable. Thus, it's a trade-off between response time and reliable results.

An experiment was carried out to reduce web service invokes for the abstract service check by introducing a time filter in the local registry as most of the LBSs are time dependent services. A comparison between number of web services invoked for the abstract service check with time filter versus without time filter is shown in figure 7. None of the web services were invoked between 11:30PM and 5:30AM as none of the restaurant services were open during that duration but the time filter has no effect when all restaurants are open. For example, as shown in figure 7, all three hundred web services were invoked at 7:30PM as all the restaurants were open.

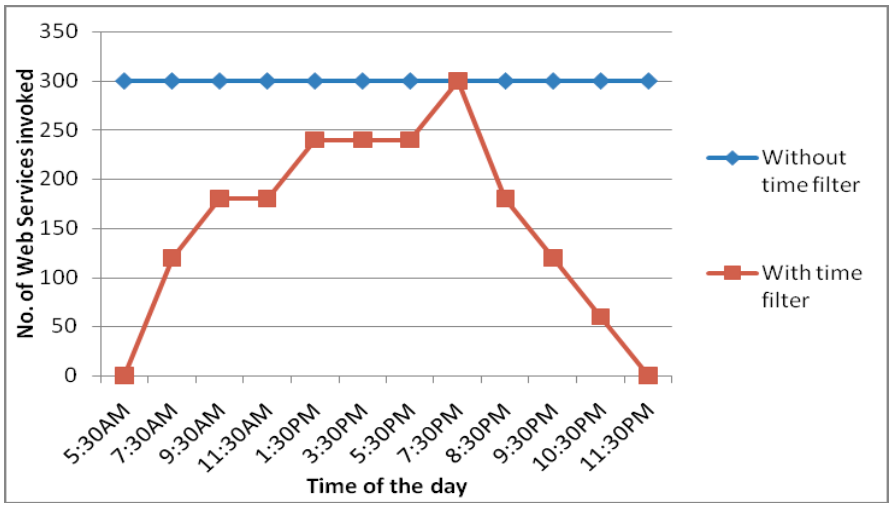

Figure 7. Number of web services invoked by local registry comparison: with time filter versus without time filter

5 Expanding LBS Discovery

It is natural that mobile users are interested in searching for services available at the current location but if the required services are not available at the current location then it is also obvious that they would prefer to search for services available at the immediate neighboring locations. For example, during some emergency, an ambulance service available at the current location is of priority but ambulance service available at the immediate neighboring location is the best choice if ambulance service is not available at the current location. Similarly, for a mobile user who is searching for near-by restaurants, if none of the restaurants are available at the current location then he would prefer to search for restaurants available at the nearest neighboring locations. Thus, the requirement is to search for services available at the current location and then expand the search to immediate neighboring locations if no such services are available at the current location.

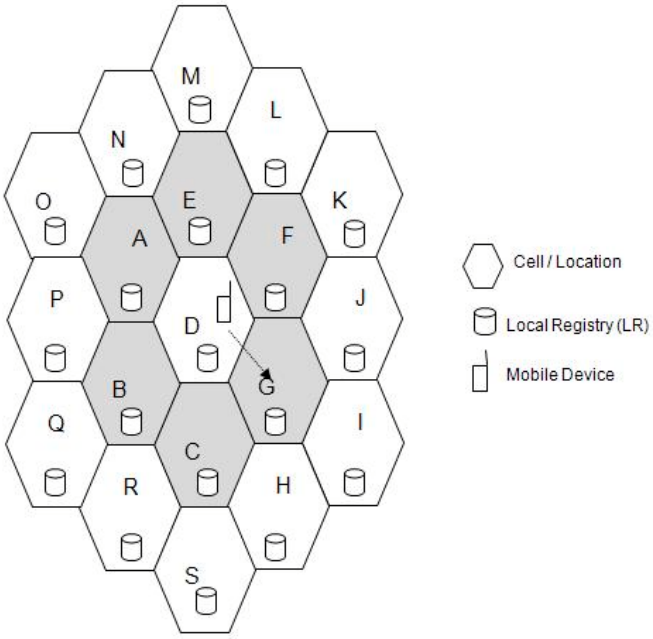

Figure 8. One-hop and two-hop neighbor registries of local registry D.

Consider another situation where a mobile user submits a search query when he was at location D, as shown in figure 8, and would reach location G before he had obtained the search results. In this case, the search results obtained are not relevant to the location G. Change of location is very likely in a mobile environment. Thus, the requirement is to get search results relevant to the current location of a mobile user, even if the user is in mobility.

The above mentioned requirements motivate us to expand the LBS discovery by implementing query propagation in a decentralized registry based architecture in case of (1) services which are requested by the end users are not available at the current location, (2) mobile users change their location after the search query submission and before obtaining the results back. In order to implement

these two requirements, our proposed system requires mobile location information and neighboring local registry information.

In a cellular network, position of a mobile phone could be obtained either from the mobile device itself or the network provider. The most common positioning system available in a mobile device is global positioning system (GPS). The GPS works with the help of 32 satellites orbiting in space and a receiver fixed in mobile devices. A low-cost mobile GPS tracking solution is explained in [28]. The main drawbacks of GPS are high power consumption and failure in indoor environment. In the case of network provider based tracking, Cell-ID is the simple and most economical mobile tracking system. Cellular base stations periodically broadcast this Cell-Id, a unique id for each cell. Cell-ID based tracking system is not highly accurate as reception area of base stations is large in rural areas but sufficient for many location based applications. According to the Federal Communications Commission (FCC) wireless 911 rules [29], every wireless service provider must provide more precise location information to Public Safety Answering Points (PSAPs). Accuracy of this location information must be within 50 to 300 meters depending upon the type of location technology used [fcc.gov]. Researchers are working on different tracking methods like time difference of arrival (TDOA) and angle of arrival (AOA) for higher accuracy of mobile device position [30].

In the proposed system, a cell or group of cells is considered as a location and hence it is sufficient to know the Cell-Id of a mobile phone to find the current location. A location system (LCS) is included in the architecture for this purpose as shown in figure 9. This location system is part of network provider which provides Cell-Id, latitude and longitude of mobile phones under their coverage. Local registry finds the Cell-Id information of a mobile phone from this LCS and maps to corresponding local registry address. Mobile terminals can get their own current position by using either GPS receiver attached to them or from the location system of their network provider.

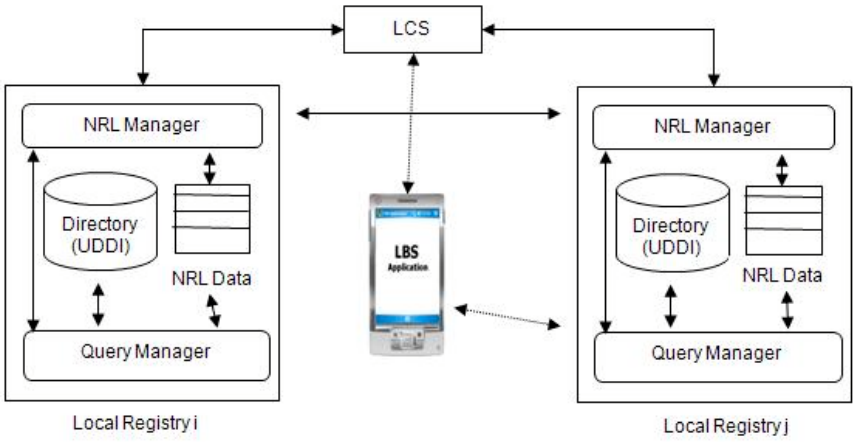

Figure 9. Functional diagram of LBS query propagation.

In order to find neighboring local registry information, the proposed system follows neighbor cell list (NCL) concept of cellular network [31]. In cellular networks, each cell is configured with a neighbor cell list, a list of neighbor cells as handover candidates. Handover is one of the most critical

features in cellular networks that allow transferring a call from one cell to another in order to avoid call termination when the mobile phone gets outside the range of the first cell. The BTS broadcasts NCL to all mobile phones connected to the serving cell. Mobile phones measure the channel signal quality of cells included in the NCL of the serving cell and submit the measurement results to the network. The network initiates a handover if the signal quality from a neighbor cell is better than that of the serving cell by a handover margin. In recent practice, the NCL is manually configured at the beginning of the network deployment by predicting static information such as base station locations, antenna patterns, and received signal maps. The NCL is manually updated when new cells are installed. Many researchers are working on automated NCL configuration systems.

Based on NCL concept, each local registry in the proposed system is configured with a list of neighboring local registries. This list is called as neighbor local registry list or simply neighbor registry list (NRL). For example, neighbor registry list of local registry D is {A, E, F, G, C, B} and neighbor registry list of local registry G is {D, F, J, I, H, C}. Note that if registry A contains B in its NRL then registry B contains A in its NRL. Each registry exchanges its NRL information with the members of its own NRL by communicating with each other. The list which contains registry information of each of its neighbor's NRL is called as neighbor's neighbor registry list (NNRL). Using NRL and NNRL information, each local registry, LR, creates two-hop neighbor registry list (TNRL) by finding distinct local registries of each of its neighbor's NRL and removing LRs of its own NRL and itself. That is TNRL = Distinct (NNRL) – NRL – {LR}. For example, two-hop neighbor registry list of local registry D is {H,I,J,K,L,M,N,O,P,Q,R,S}. Basically, NRL contains one-hop neighbor registry information and TNRL contains two-hop neighbor registry information. Thus, each registry can directly communicate with its one-hop neighbors, one-hop neighbor's neighbors and two-hop neighbors.

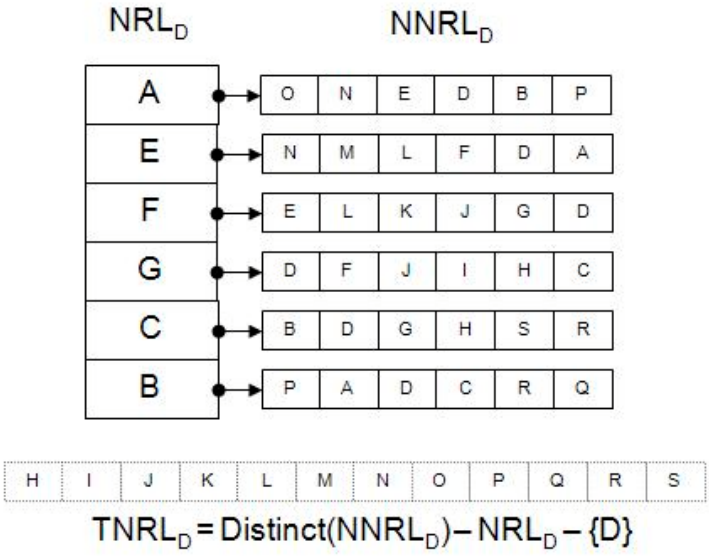

Figure 10. NRL, NNRL and TNRL of local registry D.

The NRL Manager and NRL Data components are included in local registries as shown in figure 9. The NRL manager is responsible for updating NRL data. The NRL manager periodically communicates with its neighboring registries and updates the NRL, NNRL and TNRL data. The details of NRL data of local registry D in figure 8 is shown in figure 10.

There are several advantages of maintaining NRL and NNRL information in a local registry. First and foremost is that a local registry can directly communicate with its one-hop neighbors, neighbors of one-hop neighbors and two-hop neighbors. This ensures that the query propagation would not result in infinite loops and hence hop_count field as in flooding concept is not required. Also, for a given query, query_id field as in flooding concept is not required as each registry is reached only once.

When a search query request is received by a local registry, it would first search in its own repository for the required services and then it propagates the query to immediate neighboring locations only if the required services are not available at the current location. Assume that the mobile client is at location D and wants to search for some restaurant services. Initially, the LBS application would connect to the local registry D and request for restaurant services available at that location by executing GetServices() method. The local registry D first searches in its own repository (UDDI) and retrieves web services list, say result R= $\{r_D\}$. If there exists no service which would match client's requirement then the local registry D will communicate with its all one-hop neighbor local registries which are configured in its NRL. In this case, the retrieved result is R = { r_A, r_E, r_F, r_G, r_C, r_B}. This functionality is shown below in line 1 to line 6 of GetServices() function pseudo-code.

Before returning results to the mobile client, local registry checks for client's current location. In this process, mobile phone's current location is obtained from the cellular network provider. Whenever a mobile phone moves from one cell to another, it registers itself in the new cell and the network system updates its database accordingly. If the location has not changed then the result obtained so far is valid and returned to the client. If the client location has changed then the local registry should return the results corresponding to the new location. Assume that the mobile client has now moved to location G from D. Now, instead of propagating the query directly to local registry G, the system checks whether it has already communicated with G and has a result. That is, the system checks whether G is in its NRL. If yes then it checks whether r_G is empty. If r_G is not empty then the resultset, R={r_G}. If r_G is empty then the local registry checks whether it has already communicated with any one-hop neighbors of local registry G and then contacts remaining one-hop neighbors of local registry G. In this case, resultset R={ r_C, r_D, r_F, r_J, r_I, r_H }. This functionality is shown below from line 7 to line 25 of GetServices function pseudo-code.

In conventional systems, where there is no search query propagation on location change of a mobile user, mobile users are required to re-initiate the search process every time user changes his location. Let us consider total cost of LBS search query processing which includes two way communications cost and local registry search processing cost as x units per local registry. In the case of above mentioned example, if r_G is not empty then the total cost reduction is 1*x units as r_G is already available and it is not necessary to contact local registry G again. If r_G is empty then the total cost reduction is 4*x units as results from four local registries C, D, G and F is already available and the

system needs to contact only remaining three registries H, I and J. Thus, for n users, best case cost reduction is n*4*x units.

As discussed earlier, if a mobile user is roaming around the border of a location then there is a high chance of user returning to the original location due to handover process. Also, if the location coverage area is small and mobile user is travelling in a vehicle with high speed then there is a possibility of further change in user's location. Thus, the local registry again checks for the change in location. If user has moved back to the original location D then the original resultset R= $\{r_D\}$ or R = $\{r_A, r_E, r_F, r_G, r_C, r_B\}$ is returned. In this case, reduction in total cost is either 1*x units in case of r_D is not empty or 7*x units in case of r_D is empty as results from all seven registries is already available. If user has moved to a different location from where the non-empty resultset is already obtained then that resultset is returned. For example, user has moved to location I and the local registry D already has non-empty r_I then it returns resultset r_I. In this case, reduction in total cost is 1*x units. Otherwise, the local registry throws "Location Changed" exception so that LBS application can proceed with a fresh search request connecting to the current local registry. This functionality is shown below from line 26 to line 39 of GetServices function pseudo-code.

```
//  LR => Object representing the current local registry
//  nrl => neighbor registry list
//  wsList => Object representing collection of web services retrieved  from registries

Function GetServices(params)
1.   wsLsit[LR] = LR.GetLocalServices(params);
2.   If (wsLsit[LR] == EMPTY) Then
3.       ForEach (reg in LR.nrl)
4.           wsList[reg] = reg.GetLocalServices(params); // Parallel execution
5.       End For
6.   End If

7.   LR1 = GetClientLR();
8.   If (LR1 == LR) Then
9.       Return wsList;
10.  End If
11.  If (LR1 memberOf LR.nrl) Then
12.      If (wsList[LR1] != EMPTY) Then
13.          wsList1[LR1] = wsList[LR1];
14.      Else
15.          ForEach (reg in LR1.nrl)
16.              If ( reg memberOf wsList) Then
17.                  wsList1[reg] = wsList[reg];
18.              Else
19.                  wsList1[reg] = reg.GetLocalServices(); // Parallel execution
20.              End If
```

```
        End For
    End If
Else
    Throw Exception("Location Changed");
End If

LR2 = GetClientLR();
If (LR2 == LR1) Then
    Return wsList1;
End If
If (LR2 == LR) Then
    Return wsList;
End If
If (LR2 memberOf  LR1.nrl AND wsList1[LR2] != EMPTY ) Then
    Return wsList1[LR2];
End If
If (LR2 memberOf  LR.nrl AND wsList[LR2] != EMPTY ) Then
    Return wsList[LR2];
End If
Throw Exception("Location Changed");
End Function
```

The above mentioned pseudo-code is the default search functionality of the proposed local registry architecture. The drawback of the above mentioned pseudo-code is that mobile user does not have any control on search area. It always starts from the current local area and then proceeds to neighboring locations only if required services are not available at the current location. Thus, proposed system also supports very flexible additional search functionality which allows user to specify search area. The search area could be the current location, one-hop neighbors, two-hop neighbors, current location and one-hop neighbors, current location and one-hop neighbors and two-hop neighbors or one-hop neighbors and two-hop neighbors. Pseudo-code for the GetServicesByArea function which supports flexible search area functionality is given below. In this case, FromLevel and ToLevel indicate the search area range and possible values are 0 for current location, 1 for one-hop neighbors and 2 for two-hop neighbors.

```
// LR => Object representing the current local registry
// nrl => neighbor registry list
// wsList => Object representing collection of web services retrieved  from registries

Function GetServicesByArea(params, FromLevel, ToLevel)
1.  If (FromLevel == 0) Then
2.      wsList[LR] = LR.GetLocalServices(params);
3.  End If
```

```
4.  If (fromLevel < 2 AND ToLevel > 0) Then
5.      ForEach (reg in LR.nrl)
6.          wsList[reg] = reg.GetLocalServices(params);  // Parallel execution
7.      End For
8.  End If
9.  If (ToLevel == 2) Then.
10.     ForEach (reg in LR.nnrl)
11.         wsList[reg] = reg.GetLocalServices(params); // Parallel execution
12.     End For
13. End If
End Function
```

6 Conclusions

This paper presented decentralized registry based architecture for location based services discovery in a mobile environment. This architecture allows mobile users to search for services available at the current location. This architecture provides enhanced LBS discovery by including dynamics like web services availability and abstract services availability. This functionality takes more time than conventional UDDI search but ensures reliable results and reduced wireless bandwidth consumption. Also, LBS discovery is extended to one-hop and two-hop neighboring registries when either required services are not available at the current location or mobile user changes his location during the search process.

Using this proposed architecture, cellular network providers can generate additional revenue by providing location information and charging service providers and consumers to access their local registries. Software companies can develop LBS applications using any technology as Web services technology offers interoperability and mobile users can enjoy service provider independence due to the usage of service oriented architecture.

The proposed system can be further improved by introducing caching of abstract service availability results at each local registry. Caching is an important mechanism to reduce LBS discovery time and hence we would like to consider caching in future studies.

References

1. Beaubrun, R., Moulin B. and Jabeur, N., An architecture for delivering location-based services, International Journal of Computer Science and Network Security (IJCSNS), Vol.7 No.7, 2007, 160 - 166.
2. X Pan, J Xu and X Meng, Protecting Location Privacy against Location-Dependent Attacks in Mobile Services, IEEE Transactions on Knowledge and Data Engineering, Vol.24, Issue: 8, 2012, 1506 - 1519.

3. The Mobile Location-Based Services Report, Available at http://www.businesswire.com/ news/home/20081014006008/en/Mobile-Location-Based-Services-Report-Estimates-63-Million.
4. Steiniger, S., Neun M. and Edwardes, A. 'Foundations of Location Based Services', Lecture Notes on LBS, V. 1.0.
5. Agustinus B.W., David Taniar, Wenny R, and Bala S., Mobile service oriented architectures for NN-queries, Journal of Network and Computer Applications, 2009, 434-447.
6. Kreger, H., Web services conceptual architecture (WSCA 1.0), IBM Software Group. 2001
7. Extensible Markup Language, Available at http://www.w3.org/TR/ 2008/REC-xml-20081126/
8. Simple Object Access Protocol, Available at http://www.w3.org/ TR/soap12-part1/
9. Web Services Description Language, Available at http://www.w3.org/ TR/wsdl20/
10. Universal Description, Discovery and Integration, Available at http://www.oasis-open.org/ specs/index.php#uddi
11. Farley, P and Capp, M., Mobile Web Services, BT Technology Journal, Vol. 23 No. 2, 2005, 202 - 213.
12. D'Souza, M. and Ananthanarayana V. S., Decentralized registry based architecture for location-based services, Sixth International Conference on Industrial and Information Systems , ICIIS 2011, (2011), 136 - 139.
13. Web Ontology Language for Web Services, Available at http://www.daml.org/ services/owl-s.
14. Web Service Modeling Ontology, Available at http://www.wsmo.org/.
15. Web Service Semantics, Available at http://www.w3.org/Submission/WSDL-S/
16. ShaikhAli, A., Rana, O.F., Al-Ali, R.J. and Walker, D.W., UDDIe: An Extended Registry for Web Services, Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops). 2003, 85 – 89.
17. Du, Z., Huai, J. and Liu, Y., Ad-UDDI: An Active and Distributed Service Registry, Proceedings of the 6th international conference on Technologies for E-Services, Trondheim, Norway, 2005, 58–71.
18. Linwa, A.C.B., Pierre, S., Discovering the architecture of geo-located web services for next generation mobile networks, Mobile Computing, IEEE Transactions on, Volume: 5, Issue: 7, 2006, 784 – 798.
19. Doulkeridis, C., Zafeiris, V., Norvag, K., Vazirgiannis, M. and Giakoumakis, M., Context-based caching and routing for P2P web service discovery, Distributed and Parallel Databases Journal, Volume 21 Issue 1, (February 2007), 59 - 84.
20. Yu Wang, Dingbang Xu, Xiao He, Chao Zhang, Fan Li and Bin Xu, L2P2: Location-aware location privacy protection for location-based services, INFOCOM, 2012 Proceedings IEEE, 2012, 1996 – 2004.
21. Haibo Hu, Qian Chen and Jianliang Xu, VERDICT: Privacy-preserving authentication of range queries in location-based services, 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013, 1312 – 1315.
22. Ben Niu, Xiaoyan Zhu, Haotian Chi and Hui Li, 3PLUS: Privacy-preserving pseudo-location updating system in location-based services, 2013 IEEE Wireless Communications and Networking Conference (WCNC), 2013, 4564 – 4569.
23. Marius W, Frank D and Kurt R, PShare: Ensuring location privacy in non-trusted systems through multi-secret sharing, Pervasive and Mobile Computing, Volume 9, Issue 3, (June 2013), 339-352.
24. Shang-Pin Ma and Duan-Yi Yan, Location-Based Web Service Delivery: A Data-Mining-Based Approach, 2012 International Symposium on Computer, Consumer and Control (IS3C), 2012, 666 – 669.
25. Mahmood F.M. and Bin Abdul Salam Z.A., A conceptual framework for personalized location-based Services (LBS) tourism mobile application leveraging semantic web to enhance tourism

experience, 2013 IEEE 3rd International Advance Computing Conference (IACC), (2013), 287 - 291.
26. Patel, U.R. and Gohil, B.N., Cell Identity Assignment Techniques in Cellular Network: A Review, 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), 2010, 594-596.
27. Keller, U., Lara, R., Lausen, H. and Fensel, D. 'Semantic Web Service Discovery in the WSMO Framework', Semantic Web Services: Theory, Tools and Applications, 2007, 281 - 316.
28. Moloo, R.K. and Digumber, V.K., Low-Cost Mobile GPS Tracking Solution, 2011 International Conference on Business Computing and Global Informatization, 2011, 516 – 519.
29. Wireless 911 Services, available at http://www.fcc.gov/guides/wireless-911-services/
30. Goetz, A., Fischer, G., Weigel, R., GSM Mobile Phone Localization using Time Difference of Arrival and Angle of Arrival Estimation, 9th International Multi-Conference on Systems, Signals and Devices, 2012, 1 - 7.
31. Van Minh Nguyen and Claussen, H., Efficient self-optimization of neighbour cell lists in macrocellular networks, IEEE 21st International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), 2010, 1923 - 1928.