Service Discovery in Location Based Services for Low-Powered Mobile Clients

Yen-Wen Lin and Hsin-Jung Chang
Department of Computer Science and Information Engineering,
National Chiayi University, Chiayi, Taiwan, R.O.C.

Abstract

Location based services are services that exploit knowledge about where user is located. Due to the mobility of mobile clients, the location information becomes highly varying. The efficiency of finding the geographical location of the requesting users and locating proper service providers for the mobile clients profoundly dominates the performance of the system. In this paper, we propose a service discovery protocol supporting location based services for mobile clients with the aspects of efficient mobility management, lightweight client, dynamic subject-oriented service binding, and effective service handoff.

Keywords: Service Discovery, LBS (Location Based Services), Mobile Clients, Service Binding, Service Handoff

1. Introduction

Location Based Service (LBS) demands the ability of finding the geographical location of the calling device and provide services based on this location information. Emerging application of location enabling services, such as the mobile user inquiries about travel information, routes guides, and local traffic/weather information, are optimistically anticipated. The progress in the field of communications, positioning systems, and GIS (Geographic Information System), further accelerated the development of the LBS. In the foreseeable future, the LBS will be benefiting both the users and network operators. The users will have better personal safety and security and more personalized services, while the network operators will collect, extract, and address market sections based on the different service portfolios.

Due to the mobility of mobile clients, the location information becomes highly varying. The efficiency of tracking the geographical location of the requesting users and locating proper service providers profoundly dominates the performance of a LBS system. In this paper, keep in the mind the particular characteristics of mobile devices (mobility, limited powered, limited computation and communication capability) and wireless communications (limited and changing bandwidth, high error rate) we propose a protocol addressing the service discovery supporting LBS for the mobile clients. Associated data structures are designed to efficiently achieve the service discovery task. Furthermore, the proposed system is designed with the aspects of efficient mobility management, lightweight client, dynamic/subject-oriented service binding, and effective service handoff. Making use of the proposed protocol, proper service provider supporting LBS can be discovered and allocated with effectiveness and efficiency.

The organization of this paper is as follows. In Section 2, background and related researches are present. We introduce the system model, architecture and protocol in Section 3. Considered issues are discussed in Section 4. In Section 5, performance evaluation and analysis are presented. Concluding remarks are present in Section 6.

2. Background and Related Research

Location Based Services (LBS) are services that exploit knowledge about where user is located. Emerging applications of location enabling services are promisingly expected. Researches [1-4] on LBS bloom these years. *MapInfo* [1] provides location services solutions for business applications. It also provides "*just-in-time and location*" services that can vastly improve the understanding of the users and the levels of service to the users. That is, the devices and services need to become aware of their location, both in space and time, especially if they are mobile. *Sun Microsystems* [2] provide a wide-range of workstations that are well suited for location-based application processing. *Oracle8i* has built-in java enabled wireless and location technologies. With *Oracle8i* and its spatial data server, *Oracle Spatial*, *Oracle* [3] brings the same scalability, security, multi-user integrity and recoverability to location-based data management as it has to non-location-based data management. *Java Location Service* platform [4] that combines network technologies with robust database management, server supporting, and location based application services enables users to incorporate *location-awareness* and *location-sensitivity* into applications.

Services Discovery Protocols (SDP) find the way software and network resources are configured, deployed, and advertised, all in favor of the mobile user. Researches [5-15] on the emerging technologies of services discovery in the context of wireless and mobile computing are increasingly important. Some of these emerging SDPs include *SLP*, *Jini*, and *UpnP*. Brief descriptions are present as follows.

The service location protocol is a protocol or method of configuring and locating the resources (such as printers, disk drives, and databases) in a network. *Service Location Discovery (SLP)* [5], an IETF version of SDP, is decentralized, lightweight, scale and extensible protocol for service discovery within a site [6]. SLP defines related *Service URL* describing service type and address for the service. A user can, according to the URL, find services available in its site and uses needed services. Besides, wireless local connectivity technologies, such as Bluetooth and Infrared, enable the mobile user to discover and use proximity services transparently with easiness. Bluetooth SDP [7] is designed for Bluetooth environments with limited functionality. Bluetooth SDP supports search by class, search by service attributes, and service browsing. Besides, its service discovery application profile [8] defines protocols and procedures to locate services in other devices.

Jini [9-12], as another representative SDP, federates groups of devices and software components into a single, dynamic distributed system. It also provides mechanisms for service construction, lookup, communication, and use in a distributed environment. The core components include *discovery*, *join*, and *lookup*. Where, *discovery* looks for a lookup service with which to register. *Join* handles the process when a service has located a lookup service and associates with it. *Lookup* handles the process when a user wants to locate and use a service.

Furthermore, *UpnP (Universal Plug and Play)* [13-15] proposes the architecture for pervasive peer-to-peer network connectivity of PCs, IAs (Intelligent Appliance), and wireless handheld devices. In UpnP, a device can dynamically join a network, get an IP address, convey its capabilities upon request, and learn about the presence and capabilities of other devices. UpnP uses SSDP (Simple Service Discovery Protocol) [15] for service discovery for announcing a device's presence and capabilities to others as well as discovering other devices or services. Where, a device sends out an *advertisement* message to control points when it joins the network. When a new control point is added to the network a *search* message is multicast to discover related services. Due to the advent of broadband mobile data networks and powerful handheld devices equipped with new location technologies has invented chances for LBS applications, that revives interest in LBS researches [16-18].

Due to the mobility of mobile clients, the location information becomes highly varying. The efficiency of tracking the geographical location of the requesting users and locating proper service providers will profoundly dominate the performance of a LBS system. SDPs described above put less effort to address the particular characteristics of mobile devices and wireless communications. In this paper, we propose a protocol addressing the service discovery supporting LBS for the mobile clients. There are four modules involved in the proposed protocol including the *Mobile Client*, the *Agent*, the *Service Binder*, and the *Service Provider*. The proposed system is designed to obtain efficient mobility management, lightweight load for low-powered clients, dynamically subject-oriented service binding, and effective and transparent service handoff. Detailed descriptions of this protocol and considerations in the design are present in later sections.

3. System Overview

3.1 System Model

The system model is depicted in Figure 1, where:

- Mobile Host (MH): An MH is a device that can move while remaining its network connections through wireless communication. An MH can be a mobile phone, PDA, or some kind of handheld devices with the capabilities of wireless communications.
- Access Point (AP): An AP is a fixed host augmented with a wireless interface of communicating with MHs and, is connected to the fixed network.
- Wireless Cell: A wireless cell is a geographical coverage area serviced by an AP. An MH can directly communicate with the wireless cell serviced by the AP via a wireless medium.
- Handoff: Handoff is the process of context switch between previous and current APs or servers when an MH enters a new cell.
- Fixed Network: All fixed hosts and the communication paths between them constitute the fixed network.
- Server: Server is the software that runs at a fixed host and provides services to the MH.
- Service Area: A service area is the geographical coverage of a specific service. It is likely that a service area covers several cells.
- Database (DB): A DB is an abstraction of all needed resources maintained by a certain server for a specific service.
- Service Discovery: Service discovery is the procedure to discover and allocate proper service provider for client issuing the service request.

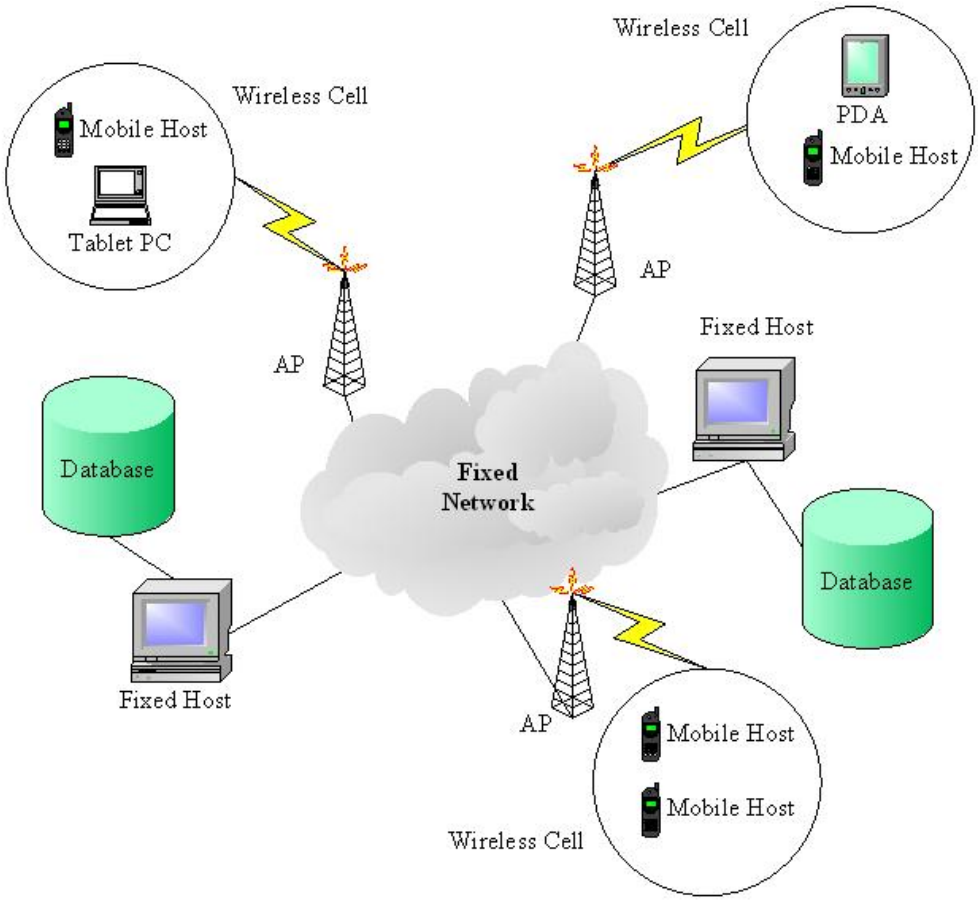

Figure 1: System Model

3.2 System Architecture

There are four modules, as depicted in Figure 2, involved in the proposed system, including the *Mobile Client*, the *Agent*, the *Service Binder,* and the *Service Provider*.

- Service Binder: *Service Binder* is the module collecting, providing and managing related information of announced services in the system. To make its presence well known to the system, the *Service Binder* needs to periodically broadcast *Advertisement* messages.
- Service Provider: *Service Provider* is the module providing specific service in the system. In the proposed system, the *Service Provider* needs to register itself with the *Service Binder* in advance to make its presence and capabilities available to the rest of the system.
- Mobile Client: *Mobile Client* is the module issuing service requests to the system and/or on the move.
- Agent: The AP first receiving the service request automatically becomes the *Agent* and acts on behalf of the requesting client during the whole service session.

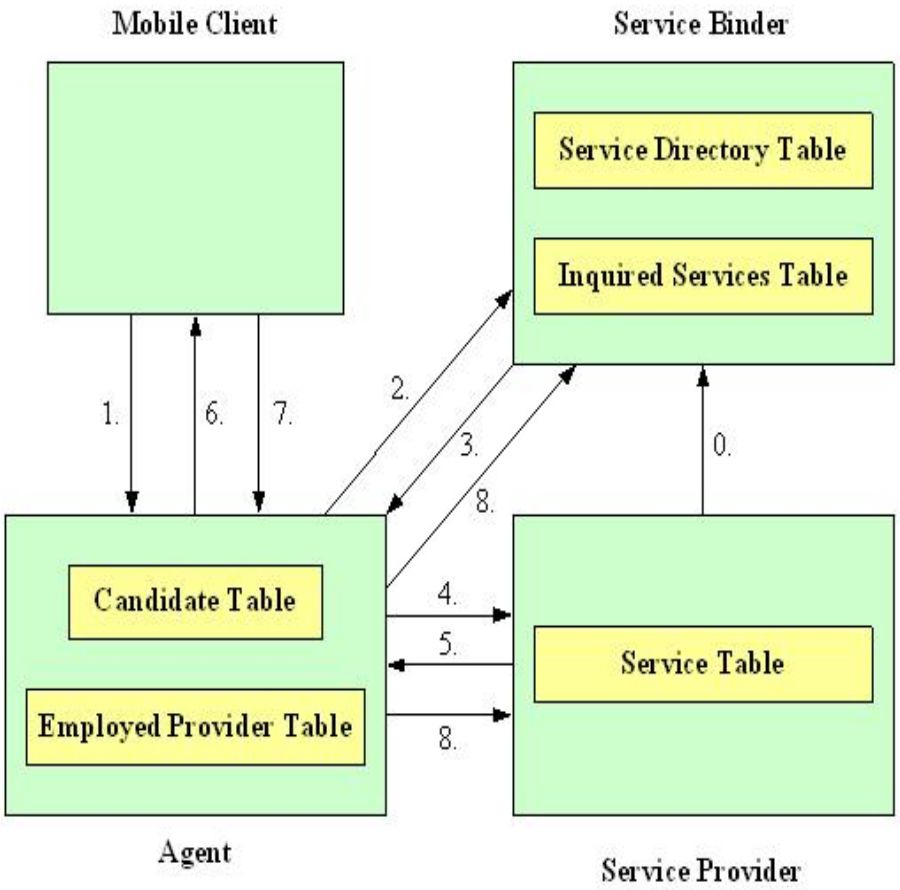

Figure 2：System Architecture

Each *Service Provider* needs to register itself with the *Service Binder* in advance to make its presence and services provided publicly available to the rest of the system. A *Mobile Client* requests a service by sending out a service request (to the *Agent)*. After receiving the service request from the *Mobile Client*, the *Agent* takes over everything about this service request and acts on behalf of the *Mobile Client* during the whole service session. The *Agent* then inquires the *Service Binder* the information about needed service and requests the chosen *Service Provider* to supply the service. After getting the result of the requested service from the S*ervice Provider*, the *Agent* locates the requesting *Mobile Client* and forwards it the result. The *Mobile Client* will send the *Agent* an acknowledgement (ACK) that, then, is respectively forwarded to the *Service Binder* and the *Service Provider* to close this service session.

3.3 Data Structures

To effectively achieve service discovery tasks for *Mobile Client*s, a set of data structures and mutual interactions are maintained to obtain the correctness and consistency among related data.

- Service Table: Each *Service Provider* locally keeps a *Service Table* (as depicted in Figure 3) to maintain information about offered services, geographical coverage of each service, load status, *Agents* being served. As mentioned earlier, to make itself known to the rest of the system, a *Service Provider* needs to register with the *Service Binder* by providing related information about services provided.
- Service Directory Table: The *Service Binder* manages related information about announced services in the *Service Directory Table* (as depicted in Figure 4). Maintained

information includes service names, qualified *Service Providers* of each service, service area and load status of each *Service Providers*. The content maintained in this table is actively provided by the *Service Providers* in advance.

- Inquired Service Table: The *Service Binder* also maintains another table named *Inquired Service Table*, as depicted in Figure 5, to record services ever inquired and the *Agents* issuing these inquiries. The purpose of this table is that the *Service Binder* can actively provide updated information to the inquiring *Agents* at once when related information is changed. Data consistency between the *Agents* and the *Service Binder* can be effectively achieved.
- Candidate Table and Employed Provider Table: After inquiring the *Service Binder* related information about a specific service, the *Agent* locally keeps the information in the *Candidate Table* and the *Employed Provider Table*. The content of these two tables are local cache of the results of inquiring the *Service Binder*. However, the *Candidate Table* (as depicted in Figure 6) keeps related information about all qualified *Service Providers* while only the information of currently employed *Service Provider* is kept in the *Employed Provider Table* (as depicted in Figure 7). The *Agent* can access local cache to save frequent remotely inquiring the *Service Binder*. As mentioned above, the *Service Binder* keeps related information of inquiring *Agents*. In case that associated information of related *Service Provider* is changed, the *Service Binder* can immediately deliver the updated information to associated *Agents*.

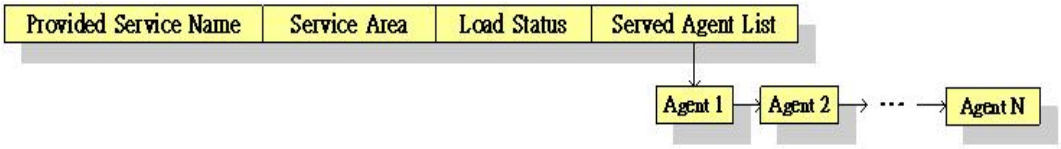

Figure 3：Service Table

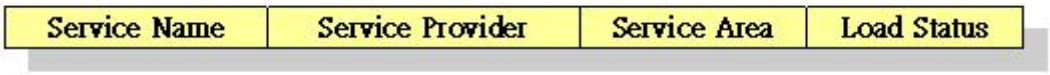

Figure 4：Service Directory Table

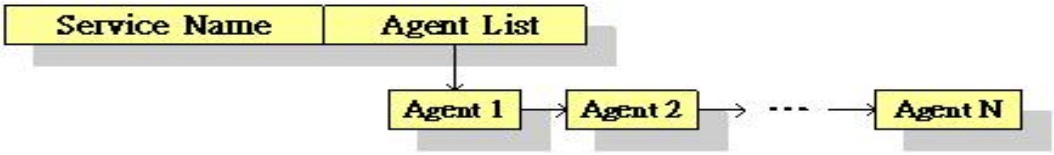

Figure 5：Inquired Services Table

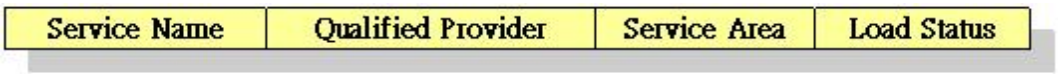

Figure 6：Candidate Table

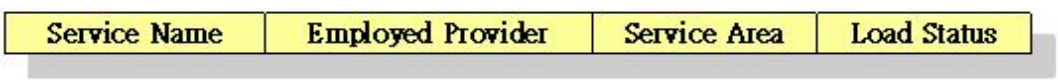

Figure 7：Employed Provider Table

3.4 Proposed Service Discovery Protocol

The proposed service directory protocol is step-wisely present, as depicted in Figure 8, as follows.

Step 0: The *Service Provider* needs to register itself with the well-known *Service Binder* in the system and becomes publicly available to the rest of the system after this registration in priori.

Step 1: The *Mobile Client* sends out service request to the *Agent*. It is worthy mentioning that the concept of *subject-oriented service binding* is considered. That is, in the proposed protocol, clients request a service by specifying the name of the needed service without detailed knowledge about the service such as where is the *Service Provider*, how to access the service etc.

Step 2: After receiving service request from the *Mobile Client*, the *Agent* forwards this request to the *Service Binder*. Meanwhile, the *Agent* acts on behalf of the *Mobile Client* during the whole session.

Step 3: The *Service Binder* checks if any proper *Service Provider* for requested service by looking up the *Service Directory Table* after receiving the service discovery request from the inquiring *Agent*. The *Service Binder* records related information of the *Agent* in the *Inquired Services Table* and, then, sends back the result of the inquiry to the responsible *Agent*.

Step 4: After getting the result of service inquiry, the *Agent* keeps related information of all qualified *Service Providers* in the *Candidate Table* and records the one currently selected in the *Employed Provider Table*. The *Agent*, then, sends out service request to the selected *Service Provider*.

Step 5: After receiving the service request, the *Service Provider* records related information of this *Agent*, and finishes requested service and sends back the result of the requested service.

Step 6: After obtaining the result, the *Agent* finds out current location of the requesting *Mobile Client* and delivers it the result.

Step 7: After receiving the result, the *Mobile Client* sends back ACK to the *Agent*.

Step 8: The *Agent* sends ACK to both the *Service Binder* and the *Service Provider* respectively to formally close the service session. Associated resources and information in the tables are deleted and released to the rest of the system.

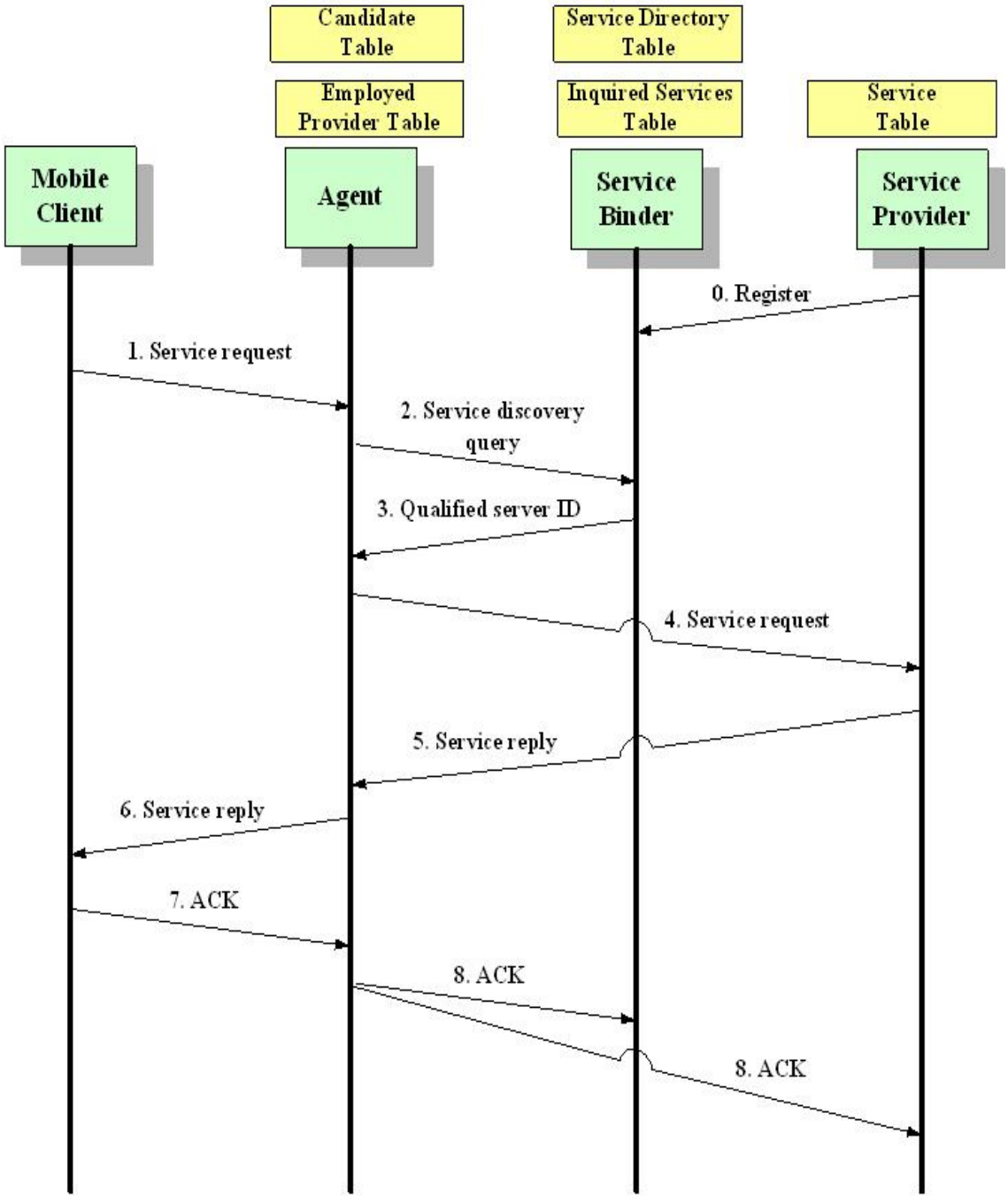

Figure 8：Service Discovery Protocol

4 Considered Issues

Some issues including mobility management, lightweight client, subject oriented service binding, service handoff, data consistency, and dynamic binding, are taken into consideration in our design of the proposed service discovery protocol.

- **Mobility Management**: In our design, the first AP receiving specific service request from an MH becomes the *Agent* and acts on behalf of this MH during the whole service session. Besides, the *Agent* tracks current location of the MH. When an MH moves across the boundary and enters a new cell, it registers itself to the local AP who will report current location of the MH to the HA. The HA informs the *Agent* current location of the MN. Two informing strategies are proposed in this paper.

- Always Push Scheme: In *Always Push Scheme*, each time the MN changes its location, the MN registers itself with the HA who immediately informs current location information to the *Agent* accordingly. It thus generates higher update message cost.
- On Demand Scheme: In *On Demand Scheme*, when the MN changes its location, it registers with the HA. However, the HA informs the *Agent* current location information of the MN only when the *Agent* cannot correctly find the MN (when the *Service Provider* later sends the results of requested services and the *Agent* cannot forward the result to the MN). The HA will inform the *Agent* after receiving the demanding message from the *Agent*. This scheme, as compared to *Always Push Scheme*, yields lower update message cost and higher data routing cost.

- **Lightweight Client**: Though great advances of the portable devices and wireless networks, as contrast to the fixed peers, the limited computing and communications capacity still existing as an unsolved problem. In the proposed system, the *Agent* takes over for the MH during the whole service session after receiving the service request from the MH. Consequently, it is not necessary that the MH keeps in the state of power-on and idles for the result of the service. Instead, the MH should be free to move around or to disconnect from the system and expects to get its service with little efforts. What it needs to do is just sending out service request and getting its service later. Both load of computation/communication on the MH can be effectively shifted to the *Agent*.
- **Subject Oriented Service Binding**: To request the service, the client needs not to learn detailed information about this service. All the client needs to do is requesting the service by specifying the service name not any specific *Service Provider*. The *Agent* and the *Service Binder* will select (according to some factor such as load balance, fault tolerance, and service area etc) a proper *Service Provider* for the client. Subject oriented service binding can be transparently achieved.
- **Service Handoff**: Each time an MH moves out of a physical cell boundary to another, a handoff is needed. That is, the physical connection and related information of this MH are switched from original AP to current one. Handoff can result in heavy overhead and seriously impact the performance of the system. As defined earlier, a service area is the geographical coverage for a specific service. Service handoff is used to depict the virtual connection transfer between original and current *Service Providers* for a requested service. It is likely that a service area covers several physical cells. In our design, service handoff is needed only when an MH moves out of the service area of a *Service Provider* to another. A service area may involve many physical cells. Thus, the frequency of service handoffs is obviously less than the one of handoffs. Ill-influence caused by service handoffs is significantly reduced.
- **Data Consistency**: As introduced earlier, a set of data structures is maintained for transparent service binding between the *Mobile Client* and the *Service Provider*. Data consistency between these geographically distributed data structures is essential to correctly achieve service discovery and binding. Possible conditions resulting in tables updates includes when a *Service Provider* starts/ends the service contract with a *Mobile Client*, or when a service handoff happens, etc. The *Service Provider* sends updating messages to the *Service Binder*, then, the *Service Binder* forwards these messages to related *Agent* who ever inquired this service.
- **Dynamic Binding**: The system may dynamically rebind a *Mobile Client* to another qualified *Service Provider* if one of the following conditions happens.
  - Service Handoff: When a *Mobile Client* moves from one service area to another, the

system will rebind the *Mobile Client* to a new *Service Provider* whose service area covers current location of the *Mobile Client*.

- Load Balancing: To obtain better QoS (Quality of Service), the system may rebind a *Mobile Client* to another qualified *Service Provider* whose load is less heavy then original one.
- Fault Tolerance: When a *Service Provider* cannot continue its service due to kinds of faults, the system will select another qualified *Service Provider* and transparently rebind the *Mobile Client* to the new *Service Provider*.

5 Simulation and Analysis

The aim of the following simulations is to evaluate, compare, and analyze the performance of informing schemes proposed in our system.

5.1 Simulation Model

A grid configuration is used as the logical wireless network which is composed of non-overlapping rectangular cells with equal size. Each cell represents a subnet which is managed by one routing agent (either a HA or a FA). The distance between nodes residing at two different cells *p* and *q* with coordinates (x_p, y_p) and (x_q, y_q) is $| x_p - x_q | + | y_p - y_q |$. Besides, the initial location of a MN in the grid is randomly selected. The MN stays in the same cell for the duration *StayTime* as an *Exponential Distribution* with mean given by *MeanStayTime*. On expiry of the *StayTime*, the MN randomly moves to one of the four neighboring cells. That is, the probability of the four neighboring cells being chosen by the MN to move is equal. Furthermore, the location of an *Agent* is also generated at random. The behavior of an *Agent* communicates with the MN is approximated by a *Poisson Process*. The *SendDurationTime* between two successive datagram sent from an *Agent* to the MN is given by an *Exponential Distribution* with mean of *MeanSendDurationTime*.

5.2 Data Routing Cost Evaluation

In *Always Push Scheme*, as shown in Figure 9, the data routing cost increases with increased LCMR (Local Call to Mobility Ratio) and increased distance between HA and *Agent*. Similarly, In *On Demand Scheme*, as shown in Figure 10, the data routing cost increases with increased LCMR and increased distance between MN and *Agent*. It is because that, higher LCMR implies that more datagram are delivered from *Agent* to MN during the observed duration. Thus, increased LCMR generates higher data routing cost. Since the datagram is sent from *Agent* to MN, the data routing cost is not directly dependent on the distance between HA and *Agent* ($3 \geq 9 \geq 6$).

5.3 Update Message Cost Evaluation

In *Always Push Scheme*, as shown in Figure 11, the update message cost does not change with various LCMR. However, in *On Demand Scheme*, as shown in Figure 12, the update message cost increases with increased LCMR. It is because that, in *Always Push Scheme*, each time MN moves, the HA immediately informs *Agent* current location of MN. The value of LCMR does not directly affect the update message cost. Comparatively, in *On Demand Scheme*, HA informs *Agent* only when the location information kept at the *Agent* is found out-of-dated. Thus, higher LCMR suggests higher probability that the location information kept at the *Agent* is found to be stale, and yields more update message cost. However, the update message cost increases with increased distance between MN and *Agent* in both schemes ($9 > 6 > 3$). Since the update message is sent from HA to *Agent* that is directly

dependent on the distance between HA and *Agent*.

5.4 Total Cost Evaluation

In *Always Push Scheme*, as shown in Figure 13, the Total Cost increases with increased LCMR and increased distance between MN and *Agent*. Similarly, in *On Demand Scheme*, as shown in Figure 14, the Total Cost increases with increased LCMR and increased distance between MN and *Agent*. Remind that, Total Cost is the sum of Data Routing Cost and Update Message Cost. The results shown in Figure 13 and Figure 14 are the effects of Figure 9-Figure 12.

5.5 Comparing *Always Push Scheme* and *On Demand Scheme*

As shown in Figure 15, when the distance between HA and *Agent* is small, the effect of Update Message Cost is trivial. Also, the effect of Data Routing Cost increases with increased LCMR. Thus, *Always Push Scheme* performs better than *On Demand Scheme* when the distance between HA and *Agent* is small. As opposite, when the distance between HA and *Agent* becomes large, as depicted in Figure 16, the effect of Update Message Cost cannot be ignored. Besides, the effect of Data Routing Cost increases with increased LCMR. When the LCMR is small, *Always Push Scheme* outperforms *On Demand Scheme*. However, when the LCMR becomes large, *On Demand Scheme* presents better performance than that of *Always Push Scheme*.

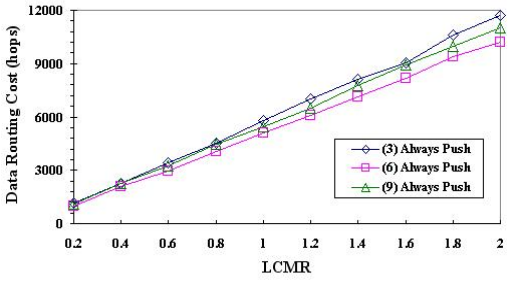

Figure 9：Data Routing Cost of *Always Push Scheme*

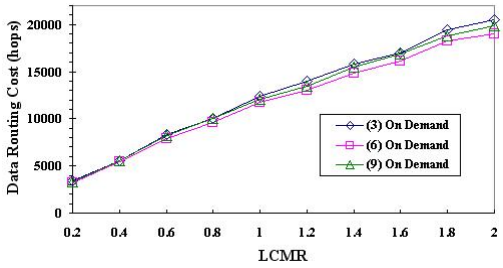

Figure 10：Data Routing Cost of *On Demand Scheme*

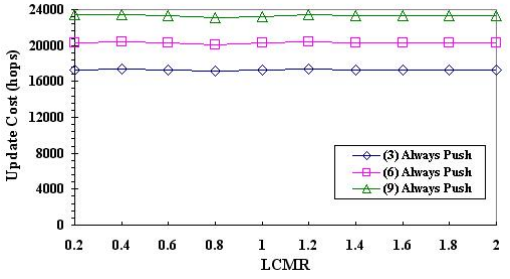

Figure 11： Update Message Cost of *Always Push Scheme*

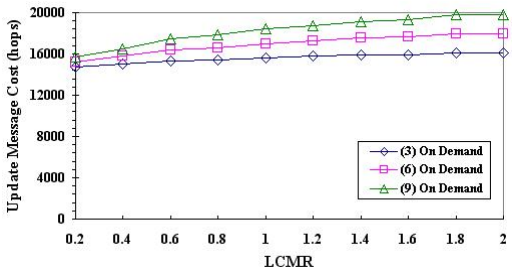

Figure 12： Update Message Cost of *On Demand Scheme*

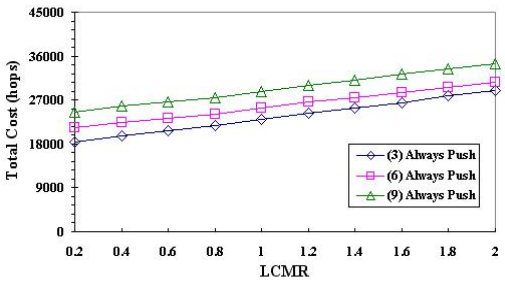

Figure 13：Total Cost of *Always Push Scheme*

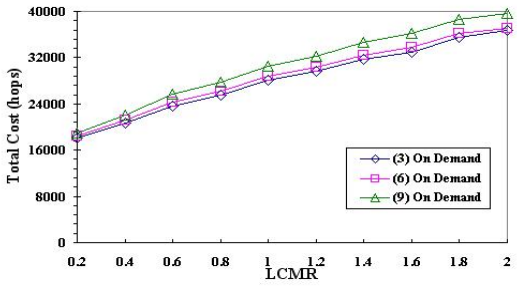

Figure 14：Total Cost of *On Demand Scheme*

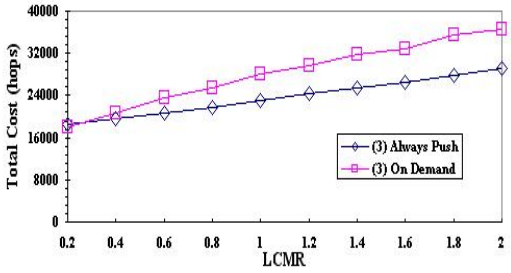

Figure 15：Comparing *Always Push Scheme* and *On Demand Scheme* when the distance between HA and *Agent* is small.

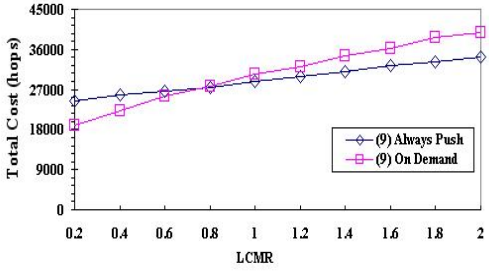

Figure 16：Comparing *Always Push Scheme* and *On Demand Scheme* when the distance between HA and *Agent* is large.

6 Conclusions

Location based service is the ability to find the geographical location of the requesting device and provide services based on this location information. In this paper, we propose a new protocol addressing the service discovery supporting LBS for the mobile clients with the aspects of efficient mobility management, lightweight load for low-powered mobile clients, dynamically subject-oriented service binding, and transparent and effective service handoff. By using the proposed protocol, proper *Service Provider* supporting LBS can be discovered and allocated with effectiveness and efficiency.

Acknowledgement

This research is supported in part by the National Science Council, Taiwan, R.O.C., under grant NSC-92-2213-E-415-003-

References

[1] MapInfo, available at: www.mapinfo.com

[2] Sun Microsystems, available at: www.sun.com

[3] Oracle, available at: www.oracle.com

[4] Java Location Services, available at: www.jlocationservices.com

[5] Guttman E., Perkins C., Veizades J., and Day M., “Service Location Protocol, Version 2”, IETF

RFC 2608, June 1999.

[6] IETF SVRLOC Working Group, “Service Location Protocol Homepage”, available at: www.srvloc.org

[7] Bluetooth Consortium, “Specification of the Bluetooth System Core Version 1.0 B: Part E, Service Discovery Protocol (SDP)”, Nov 29, 1999, available at: www.bluetooth.com/developer/specification/specification.asp

[8] Bluetooth Consortium, “Specification of the Bluetooth System Profiles Version 1.0 B: Part K: 2, Service Discovery Application Profile”, Dec 1, 1999, available at: www.bluetooth.com/developer/specification/specification.asp

[9] Sun Microsystems, “Jini Connection Technology”, available at: www.sun.com/jini/

[10] Sun Microsystems, “Jini Communicty Resources: Jini Technology Architectural Overview”, Jan 1999, available at: www.wun.com/jini/whitepapers/architecture.html

[11] Sun Microsystems, “Jini Communicty Resources: Jini Specification v1.0.1”, available at: www.wun.com/jini/specs

[12] Edwards W. K., “Core JINI”, The Sun Microsystems Press, Java Series, Prentice Hall, 1999.

[13] Microsoft Corporation, “Universal Plug and Play: Background”, available at: www.upnp.org/resources/UpnPbkgnd.htm

[14] Microsoft Corporation, “Universal Plug and Play Device Architecture Version 1.0”, June 8, 2000, available at: www.UpnP.org/UpnPDevice_Architecture_1.0.htm

[15] Goland Y. Y., Cai T., Leach P., Gu Y., and Albright S., “Simple Service Discovery Protocol”, IETF Draft draft-cai-ssdp-v1-03.txt, Oct 28, 1999.

[16] Lei H., Sow D. M., Davis J. S., Banavar G., and Ebling M. R., “The Design and Applications of a Context Service,” Mobile Computing and Communications Review, vol. 6, pp. 45-55, 2002.

[17] Beigl M., Zimmer T., and Decker C., “A location model for communicating and processing of context,” Personal and Ubiquitous Computing, pp. 341-357, 2002.

[18] Tilson D., Lyytinen K., and Baxter R., “A Framework for selecting a Location Based Service (LBS) Strategy and Service Portfolio,” Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.