Abstract—In this paper, we propose a three-tier framework for Location-Aware Service Provision and Discovery (LASPD), with special emphasis on mobile service provisioning. In the first tier, the world is geographically divided into autonomous areas to facilitate local service administration and management. In the second tier, service providers of an area with adequate computing capability are organized into a structured peer-to-peer network. The third tier supports mobile service providers equipped with less capable devices such as smart phones or PDAs. These providers are allowed to delegate their services to one of the designated peer nodes of the second tier, known as “proxy”. In LASPD, the location-awareness is achieved during a service discovery by exploiting the locality-preserving property of the Hilbert space filling curve. Besides, we leverage on our previously proposed rewiring mechanism to make the network navigable and self-organized. To have a better understanding of the framework characteristics, we conduct simulations to study the routing behaviors of peer nodes in different tiers as well as measuring the performance of the prototype in real-world environments.

Keywords—location-aware framework; service provision; organization; discovery; mobile service provider; proxy;

I. INTRODUCTION

Service oriented computing is becoming the de facto method for efficient development and integration of software in different application domains. The resulting interoperability, due to the adoption of open standards such as Web services, enables us to move closer to the vision of accessing services anywhere and anytime. One domain that requires further proofing of this concept is mobile computing. The widely available of mobile devices, along with the rapid advancements in their computation and communication capabilities, have made service provisioning through mobile devices possible. With these mobile services, future devices will play a more active role by interacting with each other in the ubiquitous environments. However, we anticipate the efficiency and accuracy of mobile service discovery will be the main concern of usage, if these services are not deployed in a well coordinated fashion. The issue is further complicated by the intermittent availability of mobile service providers, due to their mobility, connectivity and power constraints on mobile devices. Moreover, practical issues such as security, privacy, and service administration should also be considered in the provisioning of services.

Existing approaches for service provision and discovery could be broadly divided into two directions: centralized and decentralized. The centralized approach (e.g. UDDI [1] and Jini [2]) was popular in the early days due to the simplicity in implementation and ease of administration; however, it does not scale well when dealing with a large number of services. In the decentralized approach, scalable Peer-to-Peer (P2P) techniques such as Chord [3] and CAN [4] are often used. Nevertheless, portable devices such as smart phones and PDAs can barely maintain P2P connections, due to constraints of their resource and communication capability. Besides, data locality (e.g. geographical location) is often not maintained in most of these approaches, which makes them unsuitable for range queries.

In this paper, we present LASPD — a framework for location-aware service provision and discovery in mobile environments. The features of our framework are: (i) It distinguishes administrative areas and represents each of them with a superpeer. The service providers in each area are organized according to their capabilities. More specifically, relatively powerful and static service providers can take the role of a proxy to host services delegated by the resource-constrained mobile devices; (ii) Location-aware service discovery is supported by considering the location of service providers and by exploiting the locality-preserving property of the Hilbert space filling curve; (iii) It leverages on our previously proposed rewiring method of rewiring long-range links [5] that makes the underlying network navigable and self-organized; (iv) An indexing scheme for the long-range links is developed to reduce the routing effort required on superpeers, thus making the framework more resilient.

The rest of the paper is organized as follows: Section II presents an overview of the related work done in this area. Section III describes the detailed design of our location-aware service provision and discovery framework. Further design considerations for LASPD are also discussed. Section IV shows results of simulation studies and preliminary experimental results obtained from our prototype. Finally,
Section V concludes the paper and highlights our future research direction.

II. RELATED WORK

A detailed survey of many work in service discovery is reported in [6]. Most industry standards such as UDDI [1] and Jini [2] have adopted a centralized approach (registry) to store all publicly available service information. Such schemes are easy to administer and allocate resources. But, due to their potential scalability problem and single point of failure they are better suited for localized discovery of static services. In research arena, P2P overlay networks have been proposed as more scalable service discovery platforms. For instance, WSPDS [7] and Meteor-S [8] utilize unstructured P2P overlays such as Gnutella [9] for the underlying network protocol. However, unstructured P2P overlays do not guarantee any performance bound and may result in unsolvable queries. Therefore frameworks based on structured P2P overlays, such as INS/TWINE [10], GloServ [11] and VIRGO [12], are preferred in principle. They typically rely on the Distributed Hash Table (DHT) concept and assign key ownerships in a predetermined manner. Nevertheless, these approaches do not consider the resource constraints of service providers and thus are not optimized for the resource-strapped mobile service providers. In addition, the DHT technique obliterates the locality of the hashed data which makes range searches difficult.

The feasibility and performance of deploying Web services on mobile devices have been recently investigated in [13], [14], and [15]. Various optimization techniques are also discussed in [13], [15] and [16]. In the industry, Nokia is proposing the “mobile web server” concept to make service provision from the traditional server-centric model to a people-centric one ([17]). Most of these works examine hosting Web services on mobile devices, but the detailed mechanisms for service publishing and discovery are barely disclosed and seldom discussed. One notable approach is proposed in [18], which utilizes the JXTA P2P framework for mobile Web service discovery. However, the results of their scalability study are not significant due to the small number of JXTA peers involved; contexts of mobile service providers are also not considered in their approach.

In this paper, we propose a new approach to service provision and discovery. In contrast to most existing work, we take the resource constraints of mobile devices into the design consideration of our framework. Besides, its location-awareness is realized by exploiting the locality-preserving Hilbert space filling curve. This enables both area-based and distance-based service discoveries. Furthermore, we leverage on the Source Sampling technique [5] to autonomically create and maintain P2P shortcuts, i.e. long-range links. In this way, the network is self-organized and efficient routing of service queries becomes viable.

III. PROPOSED APPROACH

A. Three-tier Service Provision Architecture

1) Motivation: In LASPD, the world is geographically divided into a set of areas, each encompassing a manageable number of service providers. Every area can exercise its own service administration in terms of access control and policy making. When organizing the actual service providers in each area, we distinguish the limitations of mobile devices used for the service provision. The reason is that, they have limited memory, processing power and battery life, and their availability may be intermittent. The latter factor also affects the availability of the services they intend to provide. Besides, by managing these devices separately we allow for flexible security and privacy protection plans which are specifically tailored for them.

2) Architecture Design: Our service provision framework has a three-tier architecture (Figure 1). In the first tier, administrative areas are organized based on a predefined geographical tree. It specifies the semantic name and the geographical boundary of each area (e.g. Walmart Shopping Center). The tree also specifies if an area is contained by another one (i.e. area hierarchy). All the service providers are assumed to reside in one of the specified areas. We are currently using rectangle to model each area and an irregular area could be approximated by multiple rectangles in practice. The complete specification of each area is maintained in a special peer node known as a superpeer. It represents the geographical center of the area, although it may be physically located anywhere. Each superpeer performs some fundamental operations: it helps new service providers to join the area and facilitates cross-area queries by maintaining links with superpeers of neighboring areas. The role of a superpeer can be played by a dedicated server or by any of the service providers’ servers of the area. While the former may have full administrative and management controls, the latter may only perform the mentioned fundamental operations. In any case, we find our proposed indexing

![Figure 1. The three-tier service provision architecture in LASPD.](image-url)
mechanism (Section III-B2) exerts insignificant workload to the computing devices of a superpeer.

After dealing with the organization of areas, the actual service providers within a specific area are managed in the second and the third tier based on their capabilities. Service providers in the second tier tend to be resource rich and highly available, such as the server of a shopping mall. In this tier, P2P concepts are applied to achieve scalability and to mitigate the negative effects of joining and leaving of service providers. We refer to all service providers in this tier as service peers that share common computation tasks such as service indexing and query routing in the area. Note that the superpeers are also a kind of service peers.

The third tier is for mobile service providers via less capable mobile devices. We assume they can connect to the Internet in one or more ways, such as via GPRS, WiFi, or 3G. Each device in this tier shares its service through a proxy in the second tier, which is a service peer with supposedly higher availability and resources. The detailed modeling of the service peer to support mobile service provision is presented Section III-A4 and further design considerations are given in Section III-C.

3) Location-Aware Identifier Allocation: To enable location-aware service discovery, we derive the identifier for each service peer based on its geographical location. The identifier is represented in a binary form and is composed of two parts: areaID peerID. As an example, the identifier for service peer c in Figure 1 is 1110 1001. The areaID is to differentiate peers in different areas so to enable cross-area routing. Its length is \( d \cdot \lceil \log_2(bf_{\text{max}}) \rceil \) where \( d \) is the depth of the geographical tree in the first tier and \( bf_{\text{max}} \) is the maximum branching factor of the tree. Figure 2 illustrates the allocation of areaID for the areas in the first tier of Figure 1. Once the areaID is determined, the connections among superpeers that represent these areas are settled. Each superpeer is connected to the first superpeer whose areaID is greater than that of its own, i.e. in terms of decimal value. The way maximizes the flexibility of area organization; most importantly, it ensures a fixed number of connections is maintained by a superpeer irrespective to changes of area definitions.

While the areaID reflects the coarse-grained location information of a service peer, the peerID is supposed to contain the fine-grained location information. For this purpose, we deploy the Hilbert Space Filling Curve (Hilbert SFC) for each local area. The Hilbert SFC is a continuous fractal curve (self-similar) that can cover a 2-dimensional space through several iterations [19]. We assign peerID for each service peer based on its position on the Hilbert SFC. For instance, consider Figure 3 (top) for an area consisting of seven service peers. Initially, the curve consists of lines which lay over few coarse-grained regions; then it is recursively refined until only one service peer remains in each cell (in two iterations for this case). The peerID of a service peer will then be the ID of the cell it resides in. The set of cell IDs generated in different iterations can be represented in a hierarchy which we call Hilbert construction tree (Figure 3 bottom). The peerID length is \( 2r \), where \( r \) is the depth of the Hilbert construction tree for the area. In practice, we target a predefined cell size of \( 1m^2 \) to determine the number of iterations required, as we assume a density of one service peer per square meter is deemed acceptable for most applications. After assigning identifiers, the connections among service peers in each local area are based on their positions on the Hilbert SFC. That is, if two peers follow each other on the curve, they maintain a connection to each other.

4) Service Peer Modeling and Mobile Service Provision: Each service peer in LASPD can perform two basic tasks: service provision and service discovery. The service provision relates to publication and indexing of services; while the service discovery deals with lookup and invocation of the desired services. Figure 4 shows the core functional components of a service peer and how it relates to services. As a service peer may host multiple services, local service management is essential: it controls the start/termination of
a service and may support service migration as will be discussed in Section III-C2. The service registration component handles (mobile) service registration and publication. In the case there are services delegated by mobile service providers, the service peer is acting as a proxy. To enable efficient service discovery, registered service information such as name and description are indexed using the DHT technique over the respective P2P network for each local area. The range link indexing component is to relieve the workload of a superpeer and also improve the network resilience. The peer link maintenance component is for link monitoring among service peers. In case there is a peer link failure, the corresponding link repair function will be triggered. The rest of components perform tasks related to query generation, routing and processing which are necessary for service discovery and they will be discussed in Section III-B. Once the desired service is looked up, the requester may invoke the service directly through LASPD.

For service provision via less capable mobile devices such as smart phones or PDAs, the mobile service provider has to find a proxy first. We have designated a port for each service peer capable of being a proxy to listen on, so that if the mobile service provider and the proxy are in close proximity, the proxy can be discovered via WiFi broadcast. (In this case, the mobile service provider does not necessarily possess a public IP for its service provision.) Alternatively, a list identifying the address of potential proxies can be retrieved from our dedicated Web server. In addition, we have devised a service mediator component in the service peer model (SM component in Figure 4) to maximize the flexibility of the framework. For each service registered, a corresponding service mediator is created on the service peer. The service mediator is responsible to interact with the underlying functions of a service peer such as service publication and query routing; it also directs requests (e.g. SOAP messages) to the service for invocation. With the help of service mediator, the service registration does not need to differentiate between local service and mobile service provision, which simplifies the process.

5) Keyword-based Service Indexing: When doing service indexing, we map the set of service keywords (e.g. derived from their WSDL descriptions) to the Hilbert curve of the local area by using SHA-1. Each keyword is hashed to a 2r-bit key and is assigned to the first service peer whose peerID is equal to or greater than the key value. We also incorporate the service scopes in the process of service indexing. Consider, for instance, a weather report Web service which is hosted in US but returns weather information of Singapore. To make this service locally discoverable to Singaporeans, this service should be indexed in the Hilbert curve of Singapore but not that of US. Such a service is then labeled as “remote service”.

B. Location-aware Service Discovery

1) Small-world Network: Due to area classifications, a service consumer and a service provider may not be in the same area. This results in differentiating between local-area and cross-area queries. The first two tiers provide the base graph and methods for resolving queries. More specifically, the local-area queries could be routed along the Hilbert curve of that area until they reach the destination service peer. For cross-area queries, the respective query could first be directed to the superpeer of the source area where the query originated, and then further routed to the superpeer of the target area along the curve of the first tier. Unfortunately, such a model is neither efficient nor fault tolerant. Moreover, the cross-area queries may stress the processing capability of superpeers and the failure of a superpeer would fail cross-area query routing. Therefore, we introduce long-range links (shortcuts) to achieve network navigability [20] and fault tolerance. The detailed mechanism is named as Source Sampling and presented in our previous paper [5]. Here, we briefly discuss its properties.

The basic idea behind Source Sampling is to embed the rewiring process (i.e. for long-range links) into the process of query routing, so as to create a small-world network automatically. To do so, each service peer is augmented with k long-range links. During a query routing, every service peer that receives the query will perform a sampling process for the creation of long-range link, i.e. from the service peer where the query originated to itself. The sampling is based on our modified Kleinberg’s hierarchical model and the probability is derived from the identifiers of the two service peers. After sampling, a greedy routing mechanism is applied; that is, the service peer with the closest areaID (or peerID) to the target area (or peer) is chosen as the next hop for cross-area (or local-area) routing. In our previous work, we have demonstrated the navigability of the resulted network model, and also justified the modifications done to the default Kleinberg’s model. For more details, interested readers may refer to [5].

2) Area-based Service Discovery: The greedy routing algorithm mentioned above is based on a target key contained

Figure 4. Service peer functional model. (SM stands for service mediator)
in the query header. For local-area queries, the keywords specified by the requester are first hashed to the respective keys according to the hash function used in Section III-A5. Each key is then looked up on the local Hilbert SFC. During the query generation, the areaID of the target service peer in the query header is set to $-1$; while the peerID is set to the integer value of the key. The greedy routing mechanism is then applied to find the service peer storing the key.

For cross-area queries, the target area is specified by the user and indicated by areaID in the query header. The query is first routed to the target area first, and then to the target peer based on the hashed keys in that area. However, we do not use the classic hierarchical routing algorithm, i.e., routing all the queries to the local superpeer. Instead, we make use of the long-range links created through Source Sampling by service peers in a region. In this way, the superpeer is relieved from the workload for cross-area routing; moreover, the system robustness is enhanced in case the superpeer fails. To let other service peers within the area know the existence of long-range links of the current service peer, an index for the long-range link should be created. The index maps the target areaID of the long-range links to the local service peer IP address. The distribution and maintenance of the index are similar to that of service keyword indexing when using target areaID as the key. Therefore, using the modified cross-area routing mechanism, the query is first routed to the local service peer that stores the index of the target areaID. If there are already long-range links created for the target area, a random service peer with such a link is contacted for the routing of the query; but, if the index is empty, the local superpeer is contacted and used for routing. Such an approach introduces some overhead in the routing of cross-area queries, especially in the initial stage of Source Sampling. Nevertheless, after some rounds we observed that our approach requires significantly less routing effort on the superpeers while the routing performance is not degraded compared to the conventional approach. The observations are demonstrated through simulation results in Section IV-A.

3) Distance-based Range Search: Consider queries like “discover services of type $t$ within $x$ meters” or “browse all services within $x$ meters”. These are examples of distance-based range queries that are extremely useful in real life. It is hard to provide such features via conventional P2P frameworks unless the query is flooded to all peers of the area. In our framework, such queries can be easily handled due to the usage of locality-preserving Hilbert SFC. The query is only matched with those relevant service peers in the range but not to all local peers. In the proposed approach, we first discover segments of the Hilbert curve involved in the range search (Figure 5). Segments are identified with $<\text{startID}, \text{endID}>$ and are stored in the query in ascending order based on startID. The query is then sent to discover the service peers falling within those segments. A temporal segment $<\text{currentStartID}, \text{currentEndID}>$ is kept to trace the current segment to be discovered, initially assigned with the value of the first segment. Once the query reaches the target peer which has the least greater peerID compared to currentStartID, the query is passed through and matched with all service peers along the curve until the current peer has a peerID exceeding currentEndID. The discovery on this segment is then considered finished and the temporal segment is assigned with the value of the next segment. The process continues until all segments are finished.

For the detailed derivation of segments, we have developed an R-tree style search algorithm. Algorithm 1 represents the procedure to get segments. It requires parameters of the area boundary, the number of bits for the peerID and the search bound. After creating and filling the segment list, it will assemble those continuous ones to reduce later process overhead. The actual segments are found by calling the procedure findSegments() in Algorithm 2, which specifies a recursive procedure to find all the relevant segments that are contained in the search bound. It utilizes the process for the Hilbert SFC construction. Each sub-area ($r1,r2,r3,r4$) is checked with the bound in the ascending order of their startIDs (line 10 – 21) so that they can be added into the list orderly. Besides, if the sub-area is already contained by bound, a segment can be directly constructed and added to the list. Due to space limit, only the processing of one area is illustrated in Algorithm 2.

**Algorithm 1** List getSegments(Rectangle areaBound, int areaIDLength, Rectangle bound)

1: Create an empty List segs;
2: $x = areaBound.x$; $y = areaBound.y$;
3: $w = areaBound.width$;
4: leftBits = areaIDLength - 2;
5: findSegments($x, y, w, 0, 0, leftBits, bound, segs)$;
6: for $i = 0$ to segments.size() – 1 do
7: $\text{seg} = \text{segs}.\text{get}(i)$;
8: nextSeg = $\text{seg}$.get($i + 1$);
9: if $\text{seg}.\text{endID} = \text{nextSeg}.\text{startID} - 1$ then
10: newSeg = Segment(seg.startID, nextSeg.endID);
11: segs.addElementAt(newSeg)$;
12: segs.removeElementAt($i + 1$);
13: $i = i - 1$;
Another option is to do service migration as in [16]; that is, to deploy the Web service to the proxy and letting it temporarily host the Web service.

C. Further Discussions

1) Location Determination: Location determination of service providers and requesters is essential for location-aware service discovery. In fact, providing the location information of mobile devices can itself be considered as a service. In LASPD, the physical location of the service provider could be determined by using either the Global Positioning System (GPS), the WiFi networks, or other suitable localization technologies. The coordinates are then transformed to the semantic location (area name) identified by areaID and peerID during the registration of a service peer. For mobile devices that do not support GPS or with GPS function disabled (e.g. due to large power consumption), they could rely on their proxy to determine their semantic location, and use its physical location to approximate their coordinates, assuming they are using WiFi to connect the proxy.

2) Data Replication, Caching and Service Migration: Data replication is extremely important in mobile and dynamic environments since nodes may join or leave at anytime. For the first and second tiers, failure or leaving of a peer without notice may result in key lost. Thus, we replicate the keys by setting the length of a key less than 2r bits as derived by the order of local Hilbert curve. For instance, with key length equal to 2r−3, the last 3 bits of the peerID are not considered when distributing keys, and the same set of keys may be stored by up to 2^3 service peers. For third-tier peers, caching of WSDL files and SOAP responses on the proxy can lessen the workload on the mobile device. Another option is to do service migration as in [16]; that is, to deploy the Web service to the proxy and letting it temporarily host the Web service.

3) Security and Privacy Protection: Security is always a concern in the public domain, especially when not all the entities are trusted. To provide secure service provision, the standard WS-Security [21] can be adopted in LASPD. In addition, we allow the mobile service provider to explicitly select his trusted proxy. For instance, he can select a home computer or an enterprise server as the proxy, even for caching or migrating his service data. Aside from the security considerations, the location of a service provider can be revealed during service provision, while he may only want to be known in his surrounding or a specific area. To address this issue, we utilize the concept of service scope (Section III-A5). The service provider may choose his scope of service provision.

4) Service Mobility: If the mobile service provider is moving, the services provided by him may be interrupted when he switches proxies, e.g. due to the change of access point. Stateful services such as a video streaming service which keeps the frame position for each end user are one possible solution. In LASPD, we consider two mechanisms to handle the mobility issue: (i) if the expected interruption time is short, the original proxy may cache the requests, and once the service provider is attached to a new proxy, the original proxy is informed so that those cached requests can be forward to the service provider; (ii) if the expected interruption time is long, e.g. moving from one city to another, the service could be migrated to the original proxy and after a predefined time, the service is closed.

IV. SIMULATION AND PROTOTYPE IMPLEMENTATION

A. Simulation Study

In [5], we have demonstrated the network navigability and framework resilience by using Source Sampling and the long-range link indexing mechanism. The simulation is carried out on a 1024m x 1024m map with the geographical tree generated randomly. In a trail run, the depth of the tree is 6 and the number of tree nodes (i.e. areas) is 364. Service peers are randomly distributed on the map (certain areas may be empty if having a low-density distribution of service peers). For each round of query routing, two service peers are randomly selected as the source and the target of the query. In this paper, we further study the effect of long-range link indexing mechanism in affecting the network routing behavior. More specifically, the routing efforts imposed on superpeers and service peers are compared. The routing effort is defined as the average number of peers (superpeers/service peers) involved in routing of a single query, and it reflects the required processing workload. Figure 6 presents the results. We observe that in our approach (Figure 6(a) and 6(c)), the routing effort required on superpeers decreases as the number of queries routed increases. This is because the more the number of queries, the more the long-range link indexes are created. It justifies that the role of superpeer in LASPD can be performed by any non-dedicated
server and the failure of a superpeer will not affect cross-area routing severely ([5]). We also note that with larger $k$ values (i.e. number of long-range links augmented for each service peer), less number of queries is required to achieve the same effect in reducing the routing effort. The reason is simply due to more long-range links allowed to be created with the same number of queries routed. However, the performance gain becomes smaller when $k$ increases. This is also observed in the navigability test in our previous paper, where $k = 3$ is the optimal value. For the conventional hierarchical routing approach (Figure 6(b) and 6(d)), the routing effort required on superpeers does not change considerably. Figure 6(b) shows that for the case of low-density distribution of service peers (212 service peers), superpeers play a more critical role in cross-area routing, e.g. the routing effort imposed on superpeers is almost twice as that on service peers when $k = 1$. Note that overall our approach imposes a higher routing effort on service peers; for instance, when $k = 1$ an average of 26 service peers after $10^6$ queries (Figure 6(c)) as compared to 15 in the conventional approach (Figure 6(d)). Nevertheless, the routing path-length does not differ much, i.e. $1 + 26$ versus $13 + 15$ after $10^5$ queries.

B. Prototype Implementation

We have developed a prototype of LASPD in our laboratory. For mobile service provisioning and discovery, the Google Android platform [22] is used. A mobile device can share his services or consume those provided by others anywhere and anytime. In Figure 7(a), the mobile user decides which of his available services should be registered with LASPD. Figure 7(b), shows the result of a range search issued by a mobile user. The available services are annotated by stars. Upon a click on a service (star), the user is prompted for the inputs as required by that service.

Preliminary experiments have been carried out to measure the processing overhead on proxies and mobile devices. The results are plotted in Figure 8. We used an Intel Dual-Core E8400 machine as the proxy and the mobile device is the HTC Hero. The experiments are based on a sample Web service which accepts a number of parameters (words) as input and echoes them to the client. For generation and processing of SOAP messages, we used kSOAP2 library [23].
The overhead on the proxy is mainly due to the direction of a SOAP message to the corresponding mobile device; while for mobile devices the SOAP deserialization/serialization process is the major concern. The overhead for directing SOAP messages on the proxy remains almost constant, even though the number of Web service parameters increases. The processing overhead of the HTC phone remains in an expectable range. Note that we ignored the processing time of services, as this overhead is affected by a number of uncontrollable parameters such as data size and implementation efficiency.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a three-tier service provision and discovery framework (LASPD). While the first tier provides classifications of geographical areas, the second tier considers service providers which are not resource constrained in a structured peer-to-peer network. In the third tier, services hosted by mobile devices are allowed to delegate to a proxy in the second tier. In LASPD, the Hilbert space filling curve is deployed to preserve the physical localities of service providers and to support area-based or distance-based service discovery. Besides, we leverage on our previously proposed Source Sampling mechanism to enable network navigability and adaptability. Simulations are carried out to study the routing behaviors of peer nodes in different tiers and the effect of long-range link indexing mechanism. As a proof of concept, a prototype application for LASPD has been developed and tested. We are currently in the progress of integrating LASPD into our context-aware middleware CAMPH [24], to further study the development of context-aware applications and service collaboration in pervasive environments.

ACKNOWLEDGMENT

This work is partially support by project grant NRF2007IDM-IDM002-069 on “Life Spaces" from the IDM Project Office, Media Development Authority of Singapore.

REFERENCES