# Trust No One: A Decentralized Matching Service for Privacy in Location Based Services

Sharad Jaiswal and Animesh Nandi
Bell Labs Research India, Alcatel-Lucent
Bangalore, Karnataka, India
Sharad.Jaiswal@alcatel-lucent.com, Animesh.Nandi@alcatel-lucent.com

## ABSTRACT

We propose a new approach to ensure privacy in location based services, without requiring any support from a "trusted" entity. We observe that users of location based services are sensitive about their i) location coordinates and ii) their interests and social relationships, as captured in their queries. We also observe there are entities that naturally have access to at least one of these pieces of information. The user and/or their mobile operator has access to their current location, and the LBS provider needs to know of the interests (in businesses, services and acquaintances) of a user. In this paper we consider whether it is possible for these entities to exchange information such that a user's queries to the LBS can be answered without i) any one entity coming to know of all sensitive information ii) a loss in the quality of service of the query, or an inordinate load on the user. Specifically, we outline the design of a decentralized matching service that takes *encoded* information from both the participating entities, and creates triggers when a user, and their objects of interest are in the vicinity of each other. Given that each component of the matching service has access to only a limited amount of encoded information, we argue that it will be impossible to recreate any sensitive user-specific information.

## Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*
; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Commercial services*

## General Terms

Design, Algorithms, Security

## Keywords

mobile user privacy, location based services (LBS), distributed hash tables (DHTs)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*MobiHeld 2010,* August 30, 2010, New Delhi, India.
Copyright 2010 ACM 978-1-4503-0197-8/10/08 ...$10.00.

## 1. INTRODUCTION

As Location Based Services (LBS) [9, 22, 23] become popular, concerns about the privacy of users of such services becomes increasingly important. Location can be easily connected with identity (by correlating with the environmental context [2] e.g. if a query is generated from inside the user's home, or using additional information from reverse white pages lookup [21]), resulting in the LBS provider (already aware of user interests, social network etc.) acquiring an unacceptably intrusive profile of an user.

In the past few years, a considerable body of work [1, 2, 4, 5, 11–15, 18, 20, 25, 26, 28, 29] in the research community has focussed on techniques and systems to allow users to access useful location based services (such as businesses of interest, or whether an acquaintance is in the vicinity etc.) without revealing the user location to the LBS provider.

A substantial body of work in this direction relies on the presence of a trusted intermediary [1, 2, 11, 15, 18, 25, 26] to ensure user privacy. Such an intermediary has knowledge of the user's location and also their requests. The role of the intermediary is to cloak the user location from the LBS provider. This can be achieved either through some form of $K$-anonymity, i.e. the user's location is not distinguishable from $K$ other users [1,11,15,26], or creating and occasionally "mixing" the pseudonyms of users [2], or generating confusion about the user path [18,25]. Much of the recent research focus has been to minimize the loss of the quality of service to the user that emerges because of obscuring user location, or delaying and suppressing the queries to the LBS provider.

However, another fundamental question that emerges over the assumptions of these works is, who would assume the role of the trusted entity? If the LBS provider cannot be trusted with knowledge of both the user location and their interests/relationships, then who would be this entity on which the users could bestow such faith? The mobile operator, who is already required by law to be aware of the user's location[1], could be a candidate to be this trusted entity. However, its not clear if the user will be willing to allow the operator to also know about their personal preferences and relationships.

In this paper we suggest a novel approach to deploy location-based services in which user privacy is guaranteed without any entity having knowledge of both pieces of sensitive user information i) location ii) queries (interests + social relationships). The LBS provider has knowledge about user queries (and has a need for this information to offer sophis-

[1]e.g. in the US, e911 regulations require the operator to be able to place a user within 100meters of their location

ticated recommendations, e.g. in Geolife like applications). A network mobile operator already has access to the user location at a certain granularity (based on cellular triangulation). Given this, it can also be a trustworthy recipient of much more accurate location coordinates (based on GPS) from the end-user. Based on these observations, we pose the following question in this paper: is it possible for the mobile user, the mobile operator and the LBS provider to co-operate, and exchange information in a manner that users can access LBS services without the mobile operator coming to know about user queries, and the LBS provider, the user location?

In this paper, we demonstrate that this can be done by encoding both the identity of users and businesses (encoded by the LBS provider) and their locations (encoded by the mobile operator) into an abstract hash-space, and have a decentralized matching service which has access to only the values in this abstract hash-space, and whose role is only to trigger updates when any two interested parties are in the same location (hash-value). In this paper we outline an architecture for a structured overlay DHT [6] based decentralized implementation of such a service, and argue that with this approach it will be impossible for any participating parties to recreate the crucial information about users.

There exist several works that attempt to ensure location privacy for users, without the need for a trusted entity, e.g. i) *Peer-peer cloaking:* in which a group of $K$-peers combine their queries [5, 13, 14] to provide K-anonymity. Note that [13] also utilizes a DHT service, but as it will become clear later, it is for a completely different purpose - to speed up the look-up for peers that can be utilized for cloaking, and not to offer a distributed matching service; ii) *Cryptographic approaches:* include novel techniques such as Private Information Retrieval (PIR), that allow users to query a database, without revealing the query [12]; iii) *Cloaking at the end-user:* in which the end-user assumes responsibility for cloaking by generating queries with a multiple set of locations or paths [20, 29]. However, each of these approaches have other shortcomings, resulting in either a lack of spatial accuracy or delays in response (peer-peer cloaking), increased communication and computational overhead at the end-users (PIR, end-user cloaking), or a reduction in the functionality of the LBS (e.g. cryptographic approaches can only handle nearest-neighbor like queries, where the user explicitly asks for services based on their current location, and not queries that require triggers to be pushed to the user when they come in the vicinity of a service or an acquaintance.).

Some other related works are as follows. In [28], each user maintains a "Virtual Individual Service" proxy, that maintains (amongst other things, location) state about the end-user in the cloud, and restricts access based on user permissions. While this notion - that the user decides who gets access to their location is reasonable when the entities of interest are from a user's social graph, its not clear how this approach can be extended to entities such as businesses, without explicit participation from each business. SMILE [24], uses a hash-based matching scheme to identify mobile users who had a past 'encounter' without revealing the location of the mobile users to the central server. In SMILE, the matching is done using the hash of an encounter key that is exchanged wirelessly amongst mobile users when in proximity.

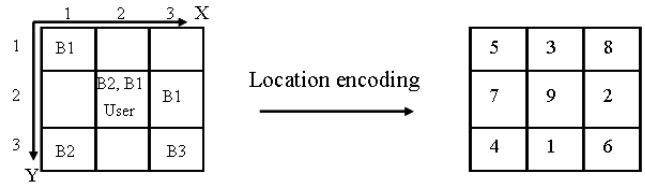


**Figure 1: Pseudonymized locations (PLs) of a 2-D location space**

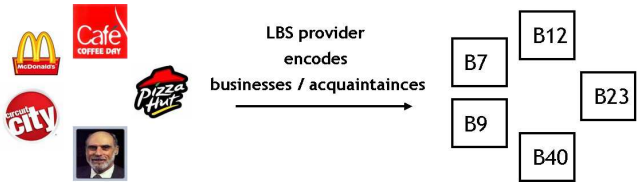


**Figure 2: Pseudonymized identifiers (PIs) of businesses/acquantainces**

# 2. DESIGN: A DECENTRALIZED MATCHING SERVICE

## 2.1 Design Overview

We now describe an approach wherein the matching of an end-user's location and their interests in a LBS is carried out by an entity unaware of the actual value (or identification) of both the location and the interests. This is done by *encoding* the two pieces (location, business-interests / social-acquaintances) of sensitive information in symbol space, and designing a *matching service* that solely operates on the encoded values, and informs the user of nearby services of interest.

The encoding of the two pieces of sensitive information is done as follows:

- **Pseudonymized Locations (PLs)** The location information of the mobile user is either coarse-grained information (via cellular triangulation or cell-ID localization) known to the mobile operator or fine-grained information (via GPS-enabled phones if available) known to the mobile user. Encoding the location information, as shown in Figure 1 is done by partitioning the physical location space into say 2-D *location grids* of size *grid-size* (say 100 m), and assigning a *pseudonymized location (PL)*, a number in the range [0,N] to each grid.

  We require some entity to assign PLs. Given that the mobile operator already has coarse-grained information of every mobile user, the mobile operator is in the best position to serve as this entity assigning PLs. Mobile users equipped with GPS-enabled phones can transfer their coordinates to the mobile operator and get assigned PLs corresponding to their current location.

- **Pseudonymized Identifiers (PIs)** The LBS provider has a list of all businesses and users which are part of the LBS service. To each the LBS provider assigns an

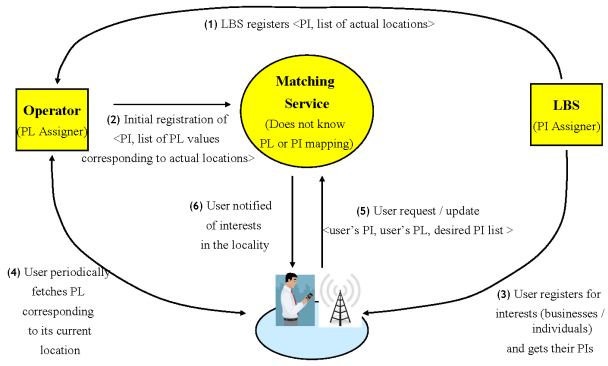


**Figure 3: Architecture of matching service**

*pseudonymized identifier*, say a number in the range [0,N] where N is the total number of entities. Figure 2 shows sample PIs for some businesses. Thus the LBS provider is the entity that serves as the PI assigner.

We now describe the design of the *matching service* that takes *encoded* inputs from the mobile operator and the LBS service, and triggers the user when entities of interest are near its current location.

The operator creates a mapping from actual locations to PLs for the addresses of businesses (and the current locations of users). Similarly, the LBS provider creates PIs from the identities of businesses and users (PI assignment for users are used for locating social acquaintances). Both these mappings are *periodically refreshed* every T minutes (say typically 15 mins) in order to avoid collusion attacks of the form mentioned in Section 3.1.

As shown in Figure 3, in the boot-strap phase, the set of PIs corresponding to businesses, and their encoded location PLs are fed into the matching service. This is done in the following steps - first the LBS notifies the mobile operator of the physical locations for each encoded business PI it caters to (indicated as Step 1 in Figure), followed by the the mobile operator notifying the matching service of each PI with the corresponding PL values for each of the locations of that business (Step 2). The mobile user also relies on an initial registration phase with the LBS to get the PIs corresponding to the businesses and users of interest (Step 3).

Before sending a LBS query, the mobile user contacts the mobile operator for the PL corresponding to its current location (Step 4). It can then initiate a LBS query, the generic form of which is a 3-tuple <user's-PI, user's-PL, desired-PI-list> (Step 5). Note that the 'user's PI' field is required only when the mobile user wants its social-acquaintances to locate them, else this field could be NULL.

The role of the matching service is then only to check if their are any PIs of interest to a user, that are located in a location-grid with the same PL as the mobile user, and if yes, create a trigger to the end-user app notifying of nearby PIs of interest (Step 6). Such a matching can be trivially done, since the matching service knows the set of PLs in which a particular PI is located.

Note that because of initial registration phase of the mobile user with the LBS in Step 3, the mobile user can decode which businesses / acquaintances correspond to the notified PIs. Note the matching service knows of (and interacts with) only the PI and PL values of the various entities - never their actual names or locations.

Through our architecture, we aim to cover the range of services currently supported by popular LBS applications. Functionally, we cover both the *PULL* queries wherein a user specifically queries for points of interest, as well as the *PUSH*-based triggers where services of interest are automatically notified to the user as he moves around based on the user's prior subscriptions.

## 2.2 The need for a matching service

The goal of our paper was to design a system in which no one entity has access to all pieces of sensitive information related to the usage of location based services. Is it possible for entities in a LBS system(mobile user, LBS provider, mobile operator) to encode and exchange information they possess, in a manner that removes the need for a separate matching service? We consider such alternatives, and build a case for why a matching service is essential.

**LBS exchanges PIs directly with the mobile operator:** In this approach, the LBS encodes businesses with PI encodings and notifies the mobile operator of the actual locations corresponding to each PI. The mobile user (as in Step 3 of Figure 3) registers with the LBS for interested services and gets the corresponding PIs of its interests. The user's query with the <desired-PI-list> is sent directly to the mobile operator. The mobile operator, based on the current location of the user, returns the PIs that are located in the vicinity of the user. In this case, the mobile operator, based on the PI/location mappings, and the user locations (and their PIs of interest) carries out the match between an user and their interests.

However, the mobile operator can decode the PI mapping and infer the user's objects of interest. Suppose the operator has access to a yellow page service though which it can find businesses at any particular location. Consider a business with PI = 385, present at locations $L = \{L_1, L_2, L_3, ....L_n\}$. The operators can find the set of businesses $B(L_i)$ corresponding to any location $L_i$. Then, if $\bigcap_{i=1}^{i=n} B(L_i)$ corresponds to a single entity (i.e. cardinality of set is one), it can decode the business/interest corresponding to PI = 385.

**Mobile operator exchanges PLs directly with the LBS:** In this approach, there are no business / acquaintance encodings (i.e. no PIs). The operator however assigns location encodings (i.e. PLs) to each location-grid. In order to match businesses, the LBS provider initially gives the mobile operator a list of businesses and their locations, and the operator returns to the LBS, a set (permuted to ensure it does not trivially reveal the PI to location mapping) of PLs associated with each business. In order to locate nearby businesses, a user sends its PL (that it acquired from the operator) to the LBS, and is notified of businesses whose PL-set contains the user's current PL.

In this approach however, the LBS provider can infer the location of an user, by correlating the PL sets corresponding to different businesses, and since it has knowledge of the physical locations of businesses. As shown in Figure 1, suppose B1 and B2 share a common location (here (x=2, y=2)). Since the only common entry in PL sets $\{9, 2, 5\}$ corresponding to B1 and $\{4, 9\}$ for B2 is 9, the LBS can decipher that the location having PL of 9 is (x=2, y=2). Through mul-

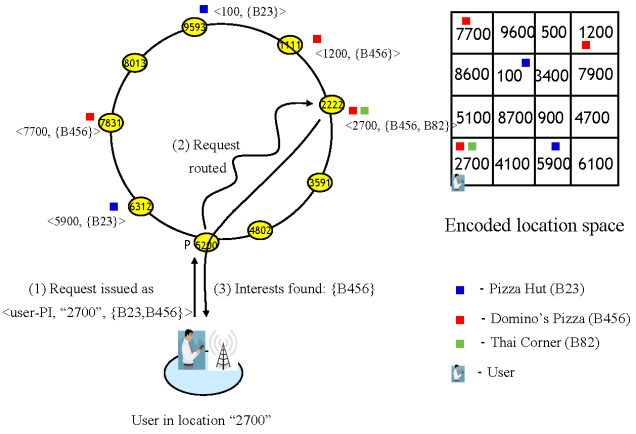


**Figure 4: A decentralized implementation of matching service on DHT abstraction of structured overlays**

tiple such correlations, the location hash mapping function will be compromised, and the users location can be exposed.

**The need for the matching service:** We thus make the observation that if any entity, that already has access to the actual value of the user location or their interests, also gets hold of the other piece of information (irrespective of whether it is encoded), it can mount attacks and acquire both pieces of information. In contrast, with a matching service (that is assumed to not collude with either the LBS provider or the mobile operator), that operates solely on the abstract space of numeric encodings (PLs and PIs), such attacks can be avoided.

## 2.3 The case for decentralization of the matching service

Although we have ensured that the matching service operates only on encoded information, enabling a single entity to have access to all this information leaves room for a sophisticated *map-stitching* attack. As the user moves around, and its PL is continuously updated, the matching service can infer which PLs are adjacent to each other. From the partial location-adjacency information inferred from the movement of multiple users, a layout of the regions (modulo rotation) of the 2-D location space can be formed. With additional information on the density of businesses, it might be possible to completely decipher the location-to-PL mapping. The act can in fact be made easier if the matching service has access to a colluding user, for which it knows the actual locations.

In order to prevent such attacks (and since for previously discussed reasons, the notion of a "trusted entity" is at odds with the goals of this paper) we propose the matching service be *decentralized* over multiple non-colluding entities.

# 3. DECENTRALIZED IMPLEMENTATION OF THE MATCHING SERVICE

We will now describe a *decentralized* implementation of a matching service using the *key-based-routing (KBR)* primitive [6] provided by structured overlays (e.g. [27]). Very briefly, all entities in the structural overlay are assigned a *nodeId* from a large identifier space, and the KBR primitive allows general access (i.e. locating the node, storing and retrieving state from it) to the node that is currently responsible for a given *key* in the identifier space.

As a design principle, we would *not* like to rely on any third-party to host this matching service. Thus the entities running the structured overlay protocol, should ideally be contributed by the end-users themselves. For example, this could be implemented through a network of Virtual Individual Servers (VIS) [4, 28] like proxies. Each VIS would correspond to a LBS user, and would run in a cloud infrastructure such as Amazon's Elastic Compute Cloud [7]. However in contrast to the model in [4, 28], where each VIS is simply a repository of user data, in our architecture, such a proxy will be a node in the DHT overlay and responsible for a portion of the DHT service.

We now describe how the system works. As illustrated in Figure 4, the area of interest is divided into a 2-D location-grid, with the number in each tile denoting the PL for that sub-division. We consider a user (with current location marked in the grid) interested in Pizza. In our example, three business establishments corresponding to the request, Pizza Hut, Dominoes's Pizza and Thai Corner, (with PI values B23, B456, B82 respectively) are shown in the grid. Also we assume a structured overlay with a circular identifier space (here [0,1000]) like Pastry [27], as being the implementation substrate of the matching service.

In the boot-strap phase the operator registers the businesses (i.e. <PL, PI> tuples) in the matching service using the KBR primitive provided by the underlying structured DHT substrate. For example, state for Pizza Hut (PI = B23) located in PLs '100' and '5900', is inserted as tuples, <'100',B23> and <'5900',B23> on the nodes currently responsible for location keys '100' and '5900' (here, nodes '9593' and '6312'). When multiple businesses are situated in the same PL, tuples are stored in the appended form, for example, the node with identifier '2222' stores an appended tuple of the form <'2700', {B456,B82}>.

A mobile user (currently in PL '2700') issues its request for pizzas by specifying the corresponding PI-list of {B23,B456} (this is got from the LBS in registration phase). The request is routed in the matching service via a *proxy* (here *P*), which could be the user's proxy in the Amazon EC2 cloud or could be a randomly chosen proxy when the mobile user is not contributing its own proxy. The proxy *p* routes the request in O(log N) hops (N being the size of the structured overlay) to the node that is currently responsible for the PL of '2700' (here node '2222'). The node '2222' computes from the list of businesses located in that region (i.e. PL of '2700'), a subset of businesses that match the desired services (here B456) the user is interested in, and forwards this result via the initial proxy 'P' to the mobile user.

We can incorporate a few optimizations that are aimed at improving the query load distribution and query latency. For instance, the query load imbalance resulting from nodes serving location-grids corresponding to densely populated regions of the city, can be addressed by non-uniform partitioning of the location space. Similarly, optimizations like one-hop DHT lookups [17] can be incorporated to reduce query latency.

To support the 'push' model of triggers/updates, a list of subscribed services per user is stored/retrieved in the structured overlay using the *get() / put()* interface supported via the DHT [6] abstraction. In the 'push' model, a query

containing only the current PL of the mobile user reaches the DHT node responsible for the user's location-grid. The node first retrieves the user's interested services, before doing the usual computation in the 'pull' model. Note that optimizations like sending a query only when the PL of a user changes, and caching user-subscriptions on overlay nodes can be applied to improve performance.

## 3.1 Security issues

In our model, entities in the matching service are proxies contributed by (some) mobile users. A colluding set of mobile users (and their proxies) can attempt to observe the location trail of a mobile user[2]. By observing a location trail, a "bread-crumb" attack can be mounted (as described in [2, 16]), through which the identity of a mobile user can be revealed (and tracked) if a query originates from a location that can be easily tied to the end-user (e.g. home address).

To launch a successful collusion attack, it would require participating proxies to i) infer the PL to actual physical location mapping in any area, and then ii) be able to track a *continuous* trail of the end-users through the physical locations. To achieve i) requires colluding mobile users to inform their locations and PL values (received in Step 4, Fig. 3) to the proxies. Since the PL map is re-computed periodically - gathering the PL mapping will need to be done in pace with the refresh rate. Even if the PL-map for a region is known, the granularity of the location-grids (say 100m*100 m) will not be sufficient to pinpoint the queries to a particular home/business.

To achieve ii) (i.e. observe a continuous trail), proxies representing *adjacent* physical regions must collude. But since the region a proxy is responsible for, is based on the initial random assignment of DHT-nodeIds and also based of the periodic refresh of the PL-mapping - the chances of colluding proxies seeing a continuous location trail is highly improbable.

## 3.2 Supporting queries with different granularity of vicinity zones

Users of LBS services however might want to be notified of services within *vicinity zones* of varying radius. To support different zone sizes, we need to be able to match businesses that are not only situated in the user's location-grid, but also those that are reasonably close (say grids within radius of 1 km).

One way (referred to as *planar approach*) to accomplish this is by modifying the inserted tuples to be <PL, (d, PI)> denoting that the entity PI is 'd' distance units apart from the location-grid PL. A business can then insert multiple tuples corresponding to the location-grids within an *advertising zone* of desired radius 'k' centered at its location. The overlay node responsible for a location-grid, can trivially sort the businesses based on the 'd' parameter. Given a user request, it can respond with a *sorted-list* (based on distance 'd') of businesses within the requested zone.

Another way to accomplish vicinity zones with varying granularity, is a *hierarchical approach* of partitioning the location space. Every location is associated with $k$ PLs $(PL_0, ..., PL_k)$ of different levels. $PL_0$ corresponds partitions with the smallest grid-size (say 100m). A business then inserts $k$ tuples corresponding to each successive PL level. Given a query, issuing multiple (at most $k$) DHT lookups to retrieve the state corresponding to different $PL_i$, and taking a set difference from returned results at each successive level, enables returning an approximately sorted list of businesses.

[2]i.e. their pseudonym, which could be e.g. an (periodically changing) IP address

## 3.3 Using a local 'mashup' application to display exact address

In the design so far, the matching service can only trigger the user that an entity of interest is "nearby", but not return the exact address of the entity. Once the user has decided on a particular service (from a list of returned matches), the mobile app can contact the LBS provider to get all the locations in a larger region (of say 5 km radius) around the current user's location, effectively cloaking the user's location from the LBS. Comparing this list of locations with the current location, a *mashup* application on the user's phone can highlight the nearest location. Note however, while this results in redundant traffic to the end-user, this occurs only when the user has made his choice of the service it wants to avail of, which is infrequent as compared to notifications of all nearby services of potential interest.

Note that a trivial solution of achieving privacy where the user a priori downloads from the LBS all the locations of all the businesses that the user might be interested in, and relies on a similar Mashup application to display the nearest business does not scale. Neither does this support dynamic LBS services like being notified of friend acquaintances.

# 4. DISCUSSION

**Evaluation Roadmap** - We intend to build the decentralized matching service using the FreePastry [10] framework, that allows applications to be evaluated in a wide range of environments (simulations, emulations and wide-area Internet). We intend to use realistic workloads of businesses from either yellow-pages or an operational LBS. We intend to use tool-based modelling of user movement (e.g. [3, 8]), or use exhaustive GPS traces if available. Will will assume the service is hosted collaboratively by some mobile users contributing proxies in the Amazon EC2 like cloud.

We intend to evaluate the following -

**i**. Is the total amount of state stored in the overlay a concern, how severe is the load imbalance resulting from non-uniform distribution of businesses?

**ii**. How much is communication overheads from user movement updates and trigger updates of nearby services?

**iii**. What is the typical query latency observed by users?

**iv**. Does the throughput (wrt to volume of queries served) scale with the size of the structured overlay?

**v**. Compare the tradeoffs (query latency, state overheads, load imbalance) of the two approaches of supporting queries of different granularity (Section 3.2).

**Business model for service deployment** - We envision the mobile operator as rolling out a Value-Added-Service (VAS) wherein the VAS subscribers can get privacy-enabled access to LBS. Lets say this service is rolled out by Verizon (approx 100M subscribers in USA), and 1% (i.e. 1M) subscribe to this VAS. Assuming a user query rate of 1 query/min, the DHT will see approx 17K DHT gets()/sec. Similarly, assuming 1M registered business locations, and a PL-mapping refresh period of 15 mins, the DHT will see approx only 1K DHT puts()/sec. Accounting for O(logN) mes-

sages per DHT get/put, and assuming each DHT node can easily handle approx 100 messages/sec, a 1000 node DHT matching service will be enough.

The cost of hosting a node in a Amazon EC2 cloud would be atmost $100, which is a total of approx $70/month for compute(24hrs/7days at $0.095/hr) and $30/month for data transfer(200GB at $0.15/GB). Mobile users could be incentivized to host a node, by paying them $1000 in return for an investment of $100. The total cost incurred by operator to host the matching service would then be $1M/month. With 1M VAS subscribers, the mobile operator can easily recover this cost by charging a VAS subscription fee of $5/month (small compared to basic plan of $40/month). Even paying the LBS operator $1M/month, would result in the mobile operator making a profit of $3M/month from this VAS.

**Further research** - Our next goal would be to investigate the feasibility of implementing such a service by direct participation (in the DHT) from user's mobiles. There has been some work [19, 30] on running structured overlays on end-user mobiles. We want to understand better the feasibility of using such a platform for our matching service.

# 5. CONCLUSION

We have presented a novel approach to achieving privacy in LBS services, in which no single entity is trusted with sensitive information (location, business interests, and friend relationships) about the user. The approach revolves around a *decentralized matching service* implemented using structured overlays, that interfaces between the operator and the LBS. Inspite of operating on encoded information got from the operator (user location) and the LBS (user's business interests and friend relationships), it is able to trigger updates to the mobile user whenever the user is in the same location of its interested services (businesses/friends).

# 6. REFERENCES


[1] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *WWW*, 2008.

[2] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. In *PervasiveComp*, 2003.

[3] T. Brinkhoff. A framework for generating network-based moving objects. In *GeoInformatica*, 2002.

[4] R. Caceres, L. Cox, H. Lim, A. Shakimov, and A. Varshavsky. Virtual individual servers as privacy-preserving proxies for mobile devices. In *ACM MobiHeld*, 2009.

[5] C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based services. In *Advances in Geographic Information Systems (GIS)*, 2006.

[6] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for structured peer-to-peer overlays. In *IPTPS*.

[7] Amazon elastic compute cloud (ec2). http://aws.amazon.com/ec2/.

[8] M. Fiore, J. Harri, F. Filali, and C. Bonnet. Vehicular mobility simulation for VANETs. In *"Annual Simulation Symposium"*, 2007.

[9] Foursquare. http://www.foursquare.com.

[10] Freepastry. http://freepastry.rice.edu/.

[11] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS*, 2005.

[12] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: Anonymizers are not necessary. In *ACM SIGMOD*, 2008.

[13] G. Ghinita, P. Kalnis, and S. Skiadopoulos. Mobihide: A mobile peer-to-peer system for anonymous location-based queries. In *Symposium on Spatial and Temporal Databases (SSTD)*, 2007.

[14] G. Ghinita, P. Kalnis, and S. Skiadopoulos. Prive: Anonymous location-based queries in distributed mobile systems. In *WWW*, 2007.

[15] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*, 2003.

[16] M. Gruteser and B. Ho. On the anonymity of periodic location samples. In *Security in Pervasive Computing (SPC)*, 2005.

[17] A. Gupta, B. Liskov, and R. Rodrigues. Efficient routing for peer-to-peer overlays. In *NSDI*, 2004.

[18] B. Ho and M. Gruteser. Location privacy through path confusion. In *Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, 2005.

[19] I. Kelenyi and B. Forstner. Distributed hash tables on mobile phones. In *Consumer Communications and Networking Conference*, 2008.

[20] H. Kido, Y. Yanagisawa, and T. Satoh. Protection of location privacy using dummies for location-based services. In *International Conference on Data Engineering Workshops (ICDEW)*, 2005.

[21] J. Krumm. Inference attacks on location tracks. In *Pervasive*, 2007.

[22] Google's latitude. http://www.google.com/latitude.

[23] Loopt. http://www.loopt.com/.

[24] J. Manweiler, R. Scudellari, and L. Cox. SMILE: Encounter-based trust for mobile social services. In *CCS*, 2009.

[25] J. T. Meyerowitz and R. R. Choudhury. Hiding stars with fireworks: Location privacy through camouflage. In *MobiCom*, 2009.

[26] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *VLDB*, 2006.

[27] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware*, 2001.

[28] A. Shakimov, H. Lim, L. Cox, and R. Caceres. Virtual individual servers as privacy-preserving proxies for mobile devices. In *Duke University, Technical Report TR-2009-01*, 2009.

[29] P. Shankar, V. Ganapathy, and L. Iftode. Privately querying location-based services with sybilquery. In *Ubicomp*, 2009.

[30] S. Zoels, S. Schubert, W. Kellerer, and Z. Despotovic. Hybrid DHT design for mobile environments. In *AP2P Workshop at AAMAS*, 2006.