KnockAround: Location Based Service via Social Knowledge

Rizwana Rizia, Mohammad Tanviruzzaman, and Sheikh Iqbal Ahamed
Department of Mathematics, Statistics, and Computer Science
Marquette University, Milwaukee, WI 53233, USA
Emails: {rizia, mtanviru, iq}@mscs.mu.edu

Abstract—With the proliferation of location awareness in smartphones, location-based services (LBSs), such as finding nearby Sushi restaurants, have become popular. In traditional LBS applications, the user’s location is sent from her smartphone to an LBS server over the Internet and the LBS server then serves the information from its database. We observe that people who share space-time context may have overlapping needs for location-based information. Based on this observation, in this paper, we propose a pure P2P-based, pull-type LBS application for smartphones, which exploits the location information already uncovered by the surrounding people (including strangers) during their day-to-day visits, to serve the need for location information of a user. Our application is free from a single point of failure. The database of location information in our system is distributed over people’s smartphones and the users themselves update the database dynamically without the need of centralized maintenance. Also our application can serve location information that is volatile and highly idiosyncratic to a particular locality, and thus a central location server is unlikely to contain it. We present the detailed architecture, algorithms, implementation, and survey-based evaluation of our LBS application along with a discussion on the potential caveats and their possible solutions in using our system. We also propose an incentive mechanism to fight against free-riders and evaluate the mechanism using real-life data and simulation.

Index Terms—bloom filter; free-rider; incentive mechanism; location-based service; partial matching; pull service; p2p; smartphone; social caching

I. INTRODUCTION

Location awareness has become a common feature in today’s smartphones. With the increased popularity of smartphones, location-based services (LBS) have naturally become widely popular, e.g., according to a study by Mobile Marketing Association in April 2010, 63% of the iPhone owners use location-based services once a week [27]. LBSs utilize the location of a smartphone (also of the owner’s) to provide location-specific information. LBSs can be categorized as pull (e.g., finding nearby Chinese restaurants) and push services (e.g., local weather alerts) [28]. In traditional LBSs for smartphones (Fig. 1), a client-server architecture is used, where a user’s location (obtained from the phone’s built-in GPS, Wi-Fi, or cell-tower triangulation) along with her search key (e.g., shopping mall) is sent over the Internet to an LBS server, and the LBS server then replies with the location-specific information (e.g., the shopping malls nearby the user) from its database or the databases of other data-providing servers [1].

We make the following observations, which are crucial to our LBS system: first, location-based information (information about places of interest in particular) needed by the people, who are at the same place and time, may significantly overlap [3], e.g., on a university campus at the lunch time, people may need information about nearby food places. Second, the set of places explored by the people of a particular locality during their day-to-day life may be regarded as a rich source of location-based information (social capital [29]). It is likely that most of the places of interest of that locality are already included in this set. Third, people want to share their experiences with others, when the information being shared might be useful to other people, provided that the information is not regarded as sensitive by the sharer, and her privacy is not breached from the sharing [30]. Fourth, people seek information from other people even though alternative information sources, such as Internet search engines, are available [2]. And fifth, smartphones are able to communicate with other nearby smartphones via Bluetooth or Wi-Fi to form an ad-hoc P2P network.

Fig. 1: Operation of a traditional LBS application

The second observation can be viewed as “social caching,” whereby from the universe of places of interest, those specific to a particular locality have been collectively explored (cached) by the people of that locality through their day-to-day visits (Fig. 2). In other words, around us we have a rich source of location-based information (information about places of interest in particular) aggregated through real experiences of the surrounding people in their day-to-day life. The first and the second observations make it probable that if a user asked the surrounding people about a place of her (the user’s) interest, she would have gotten a satisfactory answer, because some of them (the surrounding people) might already have visited the place. Our P2P-based LBS application harnesses all
five of the above-mentioned observations to provide pull-type location-based services (information about places of interest in particular) to its users. Each user of our application is simultaneously a service provider and a service requester. As a service provider, a user (U_p) saves or updates information (location, user’s comments, whether an event, etc.) about the places she visits during her day-to-day life, which she is willing to share with others, in the on-phone database. The database also serves as a diary of the places a user visits [26]. When some other user (U_q) connects with her (U_p’s) phone over Bluetooth or Wi-Fi and submits a search key (e.g., Sushi restaurant) along with her (U_q’s) location, U_p’s phone then searches through its database and replies with the information (name of place, distance, address, comment, etc.) relevant to the search key. As a service requester, U_p connects to the smartphones of the nearby users, submits her search key and location, and gets results.

Ours is a pure P2P-based system, because there is no central LBS or data-providing server, the location information is collected by the users themselves, the database is distributed over the smartphones of the users, and the peers do not need to maintain information about the network. Thus our application is highly robust and free from a single point of failure. The “social caching” concept makes our application effective, because the location information is already tailored for the particular locality, the information is based on real experiences of the users, and the comments of the users provide further relevant information alongside the location of a place of interest. Our application is able to provide information about places, that are of interest specifically to the people of a certain locality, and thus the information might be absent in a central database of an LBS server (e.g., Scenario 2 of Motivation section). As the users of our application update the database through their day-to-day experiences, our application is able to provide dynamically updated information and thus the application is able to capture ephemeral interests about a place (e.g., an event in the Art museum), without needing any centralized maintenance.

Our contributions:

- We have proposed an incentive mechanism to fight against the free-riders and evaluated the mechanism using real-life data and simulation.
- We have performed a survey on people’s opinion regarding the idea of our application and included the result. In the incentive mechanism against free-riders, we have shown that discarding the requests from free-riders with a small probability can sustain a healthy level of contribution in our pure P2P system, given that the service provided by our application proves fruitful to the users. The survey result reveals the enthusiasm for using our application among the users, which in turn buttresses the potential utility of our application.
- We have used Bloom filters to save memory and facilitate searching through partial keyword matching in the on-phone database.
- We have implemented the application on iPod Touch 4.0.

The rest of the paper is organized as follows: Section II discusses the related works. Section III gives the motivation behind our work, Section IV describes the system architecture, Section V gives the algorithms, Section VI discusses the caveats and their remedies, Section VII presents the implementation and evaluation of the application, finally we conclude the paper with a description of the future works in Section VIII.

II. RELATED WORK

A. Client-Server based LBS systems

An agent based system for flexible, adaptable, and context aware distributed location-based services is presented in [8]. In [9] a hierarchical database scheme is proposed, where a proxy server is created per user under changing user-mobility to maintain the quality of location-based service and to improve response time. Wu et al. [10] propose to use a local server at each cell of the client and to perform judicious caching, proactive server pushing, and neighborhood replication to reduce service cost and to improve response time under changing user-mobility and access patterns. In [11] a new type of web service architecture called Person Wide Web (PWW) is proposed. PWW automatically recognizes geographically effective web resources for the user and periodically notifies the user. In [12] a platform independent markup language is presented, which lets the users create and exchange personalized services based on location. Zhu et al. [13] discuss the importance of considering user-preferences for location-based service: not necessarily, the ‘closest’ option will always be the best one. In [14] a Context and Preference-Aware Location-based Database Server (CareDB) is introduced, which delivers personalized services to its customers based on the surrounding context. In [15] the proposed framework is composed
of service provider, service requester, and service broker. A cache management scheme is proposed in [16], where a proxy server for geographically partial matching queries is used. El-Nahas et al. [17] propose a peer selection based service look-up technique for areas with sudden increase in population density.

B. P2P based LBS systems

In order to reduce the load on a centralized server, in [18] a structured P2P system is introduced, where peers can be any network terminal. The P2P network contains a super peer (R-peer) that all peers need to know about in advance and the R-peer processes the join requests. In the system, peers who are willing to participate, need to actively join the network and maintain links with the network. In [19] the computation is distributed among several agent-peers in a hybrid P2P system to reduce the load on the server. Another hybrid P2P system is proposed in [20], where a peer can share the already fetched information with other peers. Kotilainen et al. [21] propose a P2P system for location-based media sharing. In [22] a hierarchical tree-based P2P overlay network is introduced to provide dynamically updated location-based information. Srirama [23] proposes establishing web services on mobile P2P network to provide context aware services. A hybrid P2P system is proposed in [24], where a query issued from a hand-held device is served by a static peer, e.g., a robust computer.

C. LBS Applications

Popular applications that provide location-based information, like Foursquare, Gowalla, Google Places, etc., rely upon a client-server based architecture, and they consider the inputs provided by the friends only. Our application includes the inputs from nearby strangers in addition to friends, and it essentially emulates “asking surrounding people,” a natural way of information searching. In contrast to those applications, ours detects new places automatically and through the user’s inputs turns her smartphone into a database of her knowledge about the locality to be shared with other people who need information about the locality.

People gather information about a locality from their day-to-day visits to places and this information remains in their memory. Our application emulates this information-gathering process by automatically detecting a place and then saving the location along with the user-provided information about the detected place in her smartphone. When someone asks a nearby person about the location of a place of interest, the information is served to her from the later’s memory. Our pure P2P based application emulates this natural process of information-sharing through asking nearby people (including strangers) by connecting one’s smartphone with other phones around it via Bluetooth or Wi-Fi and then searching through the databases of those surrounding phones. To the best of our knowledge, no previous work on location-based services emulates this natural and effective information-sharing process via asking nearby people as well as our application does.

III. Motivation

Here we attempt to illustrate the motivation behind our application through a few mock scenarios.

A. Scenario 1

A relative of Bob’s has been admitted to a clinic for the treatment of a rare disease. The physician has prescribed a medicine, which is not available at the pharmacies adjacent to the clinic. Bob is a user of our LBS application. While sitting in the waiting room of the clinic, he searches with the name of the medicine through our application and a nearby user, who previously had to find a local pharmacy containing that particular medicine, replies (her phone does) with the required location information of the pharmacy, relieving Bob from his distress.

B. Scenario 2

Muna is a new graduate student of Marquette University and she has been granted a graduate assistantship. She needs to know where she could submit a tax return form. As a user of our LBS application, while on campus, she searches with the key “Tax return” through our application. A nearby graduate student, who previously had to find the same place replies (her phone does) with the necessary location information.

IV. System Architecture

In our application each user simultaneously plays the roles of a service provider and a service requester. The architecture is shown below:

Fig. 3: $P_1$ is a requester and provider of service

In Fig. 3, peer $P_1$ submits a search request to peers $P_3$ and $P_4$, so here $P_1$ is a service requester; at the same time $P_1$ responds to the search request submitted to her by peer $P_2$, thus $P_1$ is also a service provider. Our system has two principal components: Service Provider Component (SPC) and Service Requestor Component (SRC).

A. Service Provider Component (SPC)

SPC has two subcomponents: Database Populator (DP) and Information Retriever (IR).

DP runs continuously in the background as a daemon process. DP periodically checks if the user is currently in a place, which is not yet registered in the on-phone database.
If DP finds such an unregistered place, it will prompt the user asking if she wishes to share the location; if she does, DP requests that she provide information about the location, such as, the name of the place, address, keywords, whether any event is going on, and her comments. When she provides the requested information about the location, DP saves the information about the new place in the database.

Information Retriever (IR) also runs as a daemon process in the background continuously. When a search request is submitted from a nearby smartphone, IR searches through the phone’s (on which IR resides) database and tries to partially match the search key against the place-name, keywords or comments in the database entries, which fall within the acceptable range of distance. It then returns the search result, i.e., the name of the places of interest along with the addresses and additional information such as: distance, comments, etc. to the service requester phone.

B. Service Requester Component (SRC)

When the user needs location-based information, she opens SRC (within the application) and in its UI, inputs keywords (she can set an acceptable range of distance) and initiates searching. SRC then connects the user’s phone to other surrounding phones via Bluetooth or Wi-Fi, and submits the search request to the IRs of those phones. It also shows the search results returned by the IRs of the surrounding phones to the user.

V. ALGORITHMS

In this section we describe the related algorithms in detail and provide the corresponding pseudocodes.

A. Database Populator (DP)

A user’s smartphone has a database that contains information about the places she visits in her day-to-day life, which she is willing to share with others. The database has 11 fields: id, latitude, longitude, name of place, address, keyword, comment, date, shareable (boolean), event (boolean), and bloom filter. DP continuously keeps track of the user’s location. In order to decrease the load of periodic detection of location, nearby users share the location detection cost via Bluetooth or Wi-Fi, as has been proposed in [4]. After acquiring a location update every \( \delta t \) time period, DP checks if the location changes within a certain time interval \( \Delta t > \delta t \); if it does, DP assumes that the user is on the move and it is not a place she is visiting. If the location remains unchanged over a time interval of \( \Delta t \), DP then assumes the location to be a place the user is visiting and it then searches if the location is already registered in the database. If DP finds the location in the database, it does nothing further and resumes periodic location-checking. If DP does not find the location in the database, it assumes that it is a new place and prompts (related HCI issue is discussed in the “Others” subsection in the “Caveats” section) the user asking if she wishes to share the location information with others; if she does, DP then requests her to provide information regarding the location, such as the name of the place, address, keywords, whether an event is going on, and her comments. When the user provides the requested information, DP computes a Bloom Filter (BF) for the trigrams of the concatenated string (the name of the place, keywords, comments) in order to facilitate searching through “partial keyword matching” by IR (we describe the process in the following subsection), and saves all information in the on-phone database. The pseudocode is given in the listing Algorithm 1.

\begin{verbatim}
input : Location update interval \( \delta t \) and place checking interval \( \Delta t \)
1 foreach \( \Delta t \) time interval do
2    \( \text{isPlace} \) ← true;
3    \( \text{prevLoc} \) ← GetLocation();
4     foreach \( \delta t \) time interval do
5         \( \text{currentLoc} \) ← GetLocation();
6         if \( \text{prevLoc} \neq \text{currentLoc} \) then
7             \( \text{isPlace} \) ← false;
8     end
9     if \( \text{isPlace} = \text{true} \) then
10        \( \text{isNewLoc} \) ← CheckDB(\( \text{prevLoc} \));
11       if \( \text{isNewLoc} = \text{true} \) then
12          \( \text{isShareable} \) ← GetPermission();
13         if \( \text{isShareable} = \text{true} \) then
14             \( \text{info} \) ← RequestForInfo();
15             \( \text{bf} \) ← Bloom(\( \text{name}, \text{key}, \text{comment}, \text{event} \));
16             \( \text{info} \) ← \( \text{info} \cup \{ \text{bf} \} \);
17             InsertIntoDB(\( \text{prevLoc}, \text{info} \));
18       end
19   end
20 end
\end{verbatim}

Algorithm 1: Database Populator

B. Information Retriever (IR)

Upon receiving a search request from a nearby phone, IR dissects the search key into its trigrams and sends each trigram through the same set of hash functions, which DP uses to build the Bloom filter (BF) [5], in order to check if it is in a BF of the database whose corresponding location falls within the acceptable range of distance and whose “shareable” field is set to “true,” meaning the user wishes to share the information with others. The “shareable” field lets the user turn off the sharing of the information about a particular place in her phone’s database, which she was willing to share previously, but now she does not want to share it anymore. If the first trigram is found in \( m_1 \) database-BFs, the second trigram is matched only against those \( m_1 \) database-BFs and if the second trigram is found in \( m_2 \leq m_1 \) database-BFs, the third trigram is matched only against those \( m_2 \) database-BFs, and so on. If all the trigrams of the search key match with \( m \)
database-BFs, IR sends back the corresponding place names, addresses, latitudes, longitudes, and other information of those $m$ database entries to the requesting phone. The pseudocode is given in the listing Algorithm 2.

**Algorithm 2: Information Retriever**

```plaintext
input : Search key $s$, location $loc$, and acceptable distance $dist$
output: Information info[]
1 trigrams[] ← BreakIntoTrigrams(s);
2 BF[] ← FetchBloomFiltersFromDB(loc, dist);
3 m ← Length(BF[]);
4 indexSet[] ← {1, 2, ..., m};
5 for $i$ ← 1 to Length(trigrams[]) do
6     newIndexSet[] ← \emptyset;
7     for $j$ ← 1 to m do
8         ifHasKey(BF[indexSet[j]].trigrams[i]) = true then
9             newIndexSet[] ← newIndexSet[] \cup \{j\};
10        end
11    end
12    m ← Length(newIndexSet[]);
13    if m = 0 then
14        return;
15    end
16    indexSet[] ← newIndexSet[];
17    for $i$ ← 1 to m do
18        info[] ← info[] \cup FetchInfoFromDB(indexSet[i]);
19    end
20 return info[];
```

C. Service Requester Component (SRC)

When the user types the search key and initiates a search, SRC randomly chooses $r$ peers from the $n$ available nearby users and sends the key (with the requester’s location and acceptable range of distance) to those $r$ peers. If the user finds the information she needs among the search results (sorted in ascending order of distance) returned by those peers, the searching ends; otherwise the search key is sent again to another set of randomly chosen $r$ peers from the remaining $(n-r)$ available nearby users, and so on; until all $n$ available users have been requested. The pseudocode is given in the listing Algorithm 3.

In order to search for events, the requesting user sends a range of dates (e.g., 1 week) and a special key “event” to the nearby peers along with her location and acceptable range of distance and gets the information of events within that date and distance range from the peers.

VI. CAVEATS

Here we describe the caveats and their possible remedies in using our P2P-based LBS application.

**Algorithm 3: Service Requester**

```plaintext
input : Search key $s$, location $loc$, and acceptable distance $dist$
output: Search result $R[]$
1 users[] ← DiscoverPeers();
2 n ← Length(users[]);
3 peers[] ← \emptyset;
4 unsatisfied ← true;
5 while $n > 0$ and unsatisfied = true do
6     peers[] ←
7         RandomlyChoose(users[], Min(n, r));
8     ConnectToPeers(peers[]);
9     users[] ← users[] \peers[];
10    n ← Length(users[]);
11   for $i$ ← 1 to Length(peers[]) do
12       result ←
13           SendRequest(peers[i], s, loc, dist);
14       $R[] ← R[] \cup \{result\};$
15     end
16     DisplayResult(Sort($R[]$));
17     unsatisfied ← AskUser();
18 end
```

A. Free-riders

A free-rider in a P2P-based system is a peer who consumes P2P network resources without contributing to the network or other peers at an acceptable level [6]. Like all P2P-based systems, ours thrive on the voluntary contribution of the peers; as a consequence, free-riding is a significant threat to our system.

In our system, a contributor’s major costs are due to Algorithm 1 and Algorithm 2, which drain the battery-charge of her phone. The principal cost for Algorithm 1 is due to periodic location-detection. Both of these algorithms become less costly, when there are many contributors in the system, for then location-detection cost is shared among nearby contributors and Algorithm 3 gives satisfactory results quickly (since more contributors make the aggregate database information-rich) resulting in reduced average load (invoking Algorithm 1 less frequently) on a contributor. If peer $i$ has a type $\pi_i$, denoting the maximum cost she is willing to bear in her contribution (her decency level), and $c$ fraction of the users in the system are contributors; then as a rational being she may wish to contribute when $\pi_i > \frac{1}{c}$, otherwise she may prefer to free-ride. The contribution-level ($c$) of the system is then the solution of the following fixpoint equation [7]:

$$c = \text{Probability} \left( \pi_i \geq \frac{1}{c} \right)$$

If we assume the decency level of the users to be uniformly distributed between 0 and $\pi_m$, then the maximum of the two solutions $\frac{\pi_m \pm \sqrt{\pi_m^2 - 4 \pi_m}}{2 \pi_m}$, gives rise to a stable equilibrium.

In Fig. 4 (on the next page), we observe that to sustain a
fair amount of contributors, the system’s maximum decency level has to be high. In order to reduce the dependency on the decency level (as free-riding is substantial in P2P-based systems [6]), we introduce an incentive mechanism, whereby the quality of service (QoS) received by a user increases monotonically with her contribution. We define a user’s contribution as the sum of the amount of time she lets the application run on her phone and the number of entries in her phone’s database. If \( t_j \) denotes the amount of time (e.g., number of hours) she lets the application to run during \( j^{th} \) day and \( b \) is a boost to her contribution for each new “shareable” database entry; over a window of \( D \) days, we define her contribution \( c_i \) as,

\[
c_i = \left[ \alpha \sum_{j=2}^{D} (1 - \alpha)^{D-j} t_j + (1 - \alpha)^{D-1} t_1 \right] + b \sum_{k=1}^{M} \lambda^\tau_k
\]

In Eq. 2, on the RHS, the term in brackets is an exponentially weighted average of the amount of time she lets the application run during \( D \) days and we choose \( 0.5 < \alpha < 1 \), so that recent days have more weight. The term outside of brackets on the RHS denotes that each new database entry gives a boost to her contribution, but the boost decays over time with a decaying rate of \( 0 < \lambda < 1 \), \( M \) is the number of “shareable” database entries, and \( \tau_k \) denotes the time elapsed (e.g., number of days) since the date of insertion (or latest update of the entry) into the database of the \( k^{th} \) entry. Our incentive mechanism is to accept \( i^{th} \) peer’s search request with a probability \( Pr(c_i) \), which is a monotonically increasing function of \( c_i \) such that \( \lim_{c_i \to 0} Pr(c_i) = \epsilon \), where \( 0 \leq \epsilon < 1 \) represents the bootstrapped QoS for the new comers and \( \lim_{c_i \to \infty} Pr(c_i) = 1 \). We choose Gompertz function [32] so that the newly arrived contributors start getting acceptable QoS soon:

\[
Pr(c_i) = e^{\mu e^{\psi c_i}}
\]

(3)

Where \( \mu, \psi < 0 \). The parameter \( \mu \) can be used to control the value of the bootstrapped QoS for the new comers and the parameter \( \psi \) can be used to control the rate at which the QoS increases with \( c_i \) (Fig. 5).

Such an incentive mechanism makes the system “contribution-aware.” To simplify the analysis of the equilibrium of the system, we assume that if \( c_i \) is below a threshold value, \( i^{th} \) peer is a contributor and her request is always accepted, otherwise she is a free-rider and her request is accepted with a probability \( (1 - p) \). Now, the benefit a peer receives from the system is proportional to the contribution level in the system, thus the benefit is a function of the form \( \gamma c \), where \( \gamma \geq 1 \) denotes how useful our location-based service is to the users. For our simplified “contribution-aware” system, the equilibrium contribution level is the solution to the following fixpoint equation [7]:

\[
c = \text{Probability}(\pi_i \geq c + (1 - c)(1 - p) - p\gamma c)
\]

(4)

Assuming the decency level of the users to be uniformly distributed with \( \pi_i \sim U(0, \pi_m) \) we find the stable solution as, \( \max\left(\frac{(\pi_m - p) \pm \sqrt{(\pi_m - p)^2 - 4(1 - p)(\pi_m - \gamma p)}}{2(1 - p)}\right) \). If we set \( p = \frac{1}{\gamma} \), the contribution level \( c \) becomes independent of \( \pi_m \). We conclude that if our application is very useful (i.e., \( \gamma \) is big) to the users, rejecting a peer’s request (when her contribution is below the threshold value) with a small probability \( (\frac{1}{\gamma}) \) achieves a large amount of contribution regardless of the maximum decency level of the system. We evaluate the proposed incentive mechanism in section VII.

B. Privacy

In our system, as a service provider, the user can choose between “to be shared” and “not to be shared” options for a location. In a particular location, if there is only one neighboring peer, a malicious user can send multiple search keys to inquire about the places her neighbor visits, and then may use this information to exploit her peer. Our application could deny multiple frequent queries from a particular user, which requires each user to have a unique identifier (e.g., the phone’s IMEI). Also the application could deny multiple frequent queries from a particular user, which requires each user to have a unique identifier (e.g., the phone’s IMEI). Also the application could deny providing service when there are not “enough” users around, in keeping with the philosophy of \( k \)-anonymity [25]. Since our application uses Bluetooth or Wi-Fi based communication, a user may find that there are several users in her wireless range but the malicious peer may find only the user. In order to resolve this problem, a user may require the service-requesting phone to provide identities of its current neighbors.

C. Trust

How much trust to put on the information received from the nearby peers is another issue regarding our system. The contribution function we propose in the free-rider subsection
captures the quantity of service a peer provides, not the quality of her service. We may introduce one of the reputation-based schemes described in [6] to measure the quality of information provided by a peer. Using IMEI as the unique identifier of a user partially mitigates the problem of whitewashing attack where a user with low-quality information repeatedly joins the network under new identities to avoid the penalty, because acquiring new identities becomes costly.

D. Security

In our system, a set of malicious peers in collusion could influence a user to visit a place of their choosing. Though the randomized selection of peers by the Service Requester Component (SRC) makes such a collusion-based attack difficult, the possibility remains. In scenario 1 of “Motivation” section, even a single malicious peer could direct Bob to a particular location. To mitigate these problems, a user could inquire about a place she has been suggested by other peers of our application (before actually going there), in case the address seems dubious. Additionally a reputation-based scheme could curtail a malicious peers ability to misdirect the user. Colluding peers are likely to give high reputation-scores to their accomplices. As a result, the diversity of peers who provide reputation-scores to a set of peers could be used to determine the likelihood of a collusion-based attack to some extent.

E. Others

1) Battery-charge: The major source of battery-drainage is due to the periodic detection of location of the user. In our experiments, we set the desired accuracy for location to “kCLLocationAccuracyNearestTenMeters” using Core Location Framework of iOS 4.0, thus the battery-charge consumption due to location-detection was insignificant, as location-update is deferred until the user moves at least 10 meters [33]. In general, the battery-life of the smartphones running location-based services can be greatly improved by adopting the set of techniques proposed in [4]. They showed significant improvement of battery-life for navigation-based services, which require frequent location updates; for our case location-detection can be much less frequent as we are interested in finding the location of places only, and thus the battery-charge consumption can be reduced further.

2) HCI issue: When the user moves a lot and has multiple stops, the notifications issued by the Service Provider Component (SPC) may become bothersome for her. In order to remedy this problem, we implemented the notification as the bubble shown in Fig. 6(a). Thus the user can provide inputs to the application when she feels comfortable from a list of pending notifications.

3) Memory: The application saves the addresses along with the user-provided information in the on-phone database, which may fill-up the phone’s memory eventually. But a user does not always visit new places, rather she frequents a set of particular places and every once in a while visits new places. To save the information about each new place only 1.5 KB of memory is used, assuming keyword-length to be 100 characters and comment-length to be 200 characters. Thus if she visits even 150 new places in one month, the total memory usage is only 1.5 KB × 150 = 225 KB. Moreover, the places which the user has not visited for a while (e.g., six months) might be removed from the database with the user’s permission; again, the notifications for this process may appear as bubbles.

VII. IMPLEMENTATION AND EVALUATION

We have implemented the application on iPod Touch 4.0. The application connects with the nearby peers using Bluetooth or Wi-Fi (using Bonjour protocol) and determines location from Wi-Fi access points with an accuracy of roughly 10 meters [33]. We implemented the Rabin fingerprint-based Bloom filter library [31] in Objective C to use in our application.

In Fig. 6 a few screenshots show the operation of the application. In Fig. 6(a), the user provides information about the detected places upon receiving notification. The application uses reverse geo-coding to suggest addresses of the detected places, which the user can edit as needed. In Fig. 6(b), the user initiates a search by entering a search key, the application then connects to the nearby peers and shows the results returned from the peers in ascending order of distance. From the list of results, the user can choose one to see further details or see it on the map. The SQLite databases on the devices were populated by the application while we visited various places carrying the devices with us.

One of the authors carried one iPod Touch 4.0 while keeping the application running on the device for around 12 hours (8 AM to 8 PM) per day for one month. Fig. 7(a) shows the plot of the count of places he visited during this one month period along with the counts of new places (the places which he has not visited in any of the previous days).

From this one month’s data, the plot of his contribution (Eq.
2) and acceptance probability (Eq. 3) are shown in Fig. 7(b and c) for different values of $\alpha$ and $\lambda$; with $D = 7$ days and $t_j$ being the number of hours the application ran in the $j^{th}$ day. As the user kept the application running for 12 hours every day, the part in the brackets in Eq. 2 becomes constant.

If we assume that the author is a bit less generous letting the application run for a small random amount (with normal probability distribution having a mean of 3 hours and a standard deviation of 1 hour) of time each day, and starting the application each day at a random time, which is also normally distributed with a mean at 12 PM and a standard deviation of 4 hours; then using the one-month’s actual data, we have a different set of curves, which is shown in Fig. 8. From Fig. 7(b) and Fig. 8(b), we see that changing $\lambda$ affects the contribution more than changing $\alpha$ does; because for the generous user the time he lets the application run is constant and for the not so generous user, the time he lets the application run is small (Eq. 2). Thus visiting new places significantly increases their contribution-levels. From Fig. 7(c) and Fig. 8(c) we see that setting $\alpha = 0.8, b = 1$, and $\lambda = 0.7$ in Eq. 2 and setting $\mu = -3$ and $\psi = -0.5$ in Eq. 3, will mildly punish (probability of rejection is small) the (hypothetical) less generous user while the author’s requests will almost always be accepted, making the system “contribution-aware”.

We performed a survey among 100 students of Marquette University regarding the idea of our application. 53% of the students possess a smartphone and among them about 42% possess an iPhone. Among the students who own a smartphone about 85% have used location-based services to find a place of interest (e.g., nearby restaurant). While the students were new to some locality, they faced difficulty finding: a grocery store (33%), a food place (25%), a shopping mall (18%), a medicine store (4%), a barber shop (3%), and other places (17%). Among the 100 students, to find the location of a place of interest: 59% searched on the Internet while 41% asked people around them. This appears to weaken the validity of our application, but when we described the idea of our application to them, 75% of the students were enthusiastic about using our application. From this we may conclude that the students, who did not ask the people around them for the location and other information (e.g., quality) of a place of interest, felt uneasy asking strangers. Our application would make asking other people easier and hence their interest in using our application; as taking other’s opinions is inherent among humans (fourth observation in the “Introduction” section). The students who preferred asking surrounding people for information about a place gave many reasons for their preference, e.g., local people know the area well, they have real experience and provide reliable and personal comments, asking people at random results in less biased opinions, the people give ‘up to date’ information, etc. During the two weeks preceding the survey, the students visited 134 distinct places (excluding home and university), among these places were: 6 grocery shops, 32 food places, 5 shopping malls, 3 medicine stores, and 2 barber shops. This data shows the richness of location information that people gather during their day-to-day visits (social caching). Moreover, as the students are like-minded, the places some of them have explored and liked might be likable to other students as well with high probability.

VIII. Conclusion and Future Work

In this paper we have presented the idea of a pure P2P-based LBS application for smartphones, which provides pull-type location-based services, e.g., finding nearby Sushi restaurants or location of recent local events, to its users. Our application exploiting the facts that people sharing the same space-time context may have overlapping needs for location-based
information and that the places of interest of a particular locality might already have been explored by the local people during their day-to-day visits. As a consequence, a user may ask nearby people (including strangers) to get satisfactory location-based information. We have discussed issues like: free-riding, privacy, trust, security, battery-charge consumption, HCI, and memory, in the context of our application, along with their possible remedies. We have proposed an incentive mechanism to fight against the free-riders and evaluated the mechanism using real-life data and simulation. The remedies of the potential caveats in using our application presented in this paper (especially for security) are not flawless; we leave finding better remedies and their detailed discussion as a future work.

Presently we are in the process of deploying our LBS application on the smartphones of a large number of people. It is one of our future goals to evaluate various aspects of our application using the real-life data collected on a larger scale (than that of the presented evaluation) from this deployment, e.g., how the success-rate of the search-requests submitted by the users evolve over time, the validity of the proposed incentive mechanism against the free-riders, and whether the equilibrium contribution level is achieved. In the future, we also wish to analyze the social aspects related to our application; for example, the interplay between social relations and the consumption of available services in a certain area.

IX. ACKNOWLEDGEMENT

The work started as a project for a course taught by Dr. Praveen Madiraju. We would like to thank Dr. Madiraju for his continual enthusiasm for the idea.

REFERENCES


