Journal of Mobile Multimedia, Vol. 6, No.1 (2010) 073-094

G2G: LOCATION-AWARE MOBILE SOCIAL NETWORKING WITH APPLICATIONS IN PERSONALIZED RECOMMENDER SYSTEMS

IOANNIS T. CHRISTOU SOTIRIS MICHALAKOS

Athens Information Technology, Athens, Greece

ichr@ait.edu.gr s.michalakos@gmail.com

Received March 1, 2009
Revised September 3, 2009

We present G2G, a proximity-aware social networking platform for Mobile Information Device (MID) users and its applications in personalized travel assistance via location-aware recommender systems. User proximity to a friend or a place or event is determined via GSM cell information and enhanced via GPS information if the mobile device has such capability. GSM cell information alone is greatly enhanced through the concept of user-defined "hotspots" which is the means that allows the system to expand the notion of proximity to include not only users covered by the same GSM cell but also users and/or hotspots covered by geographically neighboring cells. The system can be thought of as a Selective Dissemination of Information platform where users subscribe to it and are willing to receive selectively information coming from their own personal network of friends and acquaintances, and if they opt for, from other system users as well, in the form of location-aware notes about nearby places. G2G continuously and actively disseminates to users information relevant to their current location within a user-defined radius. This information is published into the system by the users themselves who can recommend a place or event. To bootstrap the system with sufficient content however that makes the use of the system appealing to users, we have developed Web Information Extraction methods that extract relevant data from web pages that contain recommendations about public places and/or events that can be geo-located using publicly available GIS services such as Google Earth APIs. Regarding its Social Networking aspects, the system allows users to see nearby friends and chat with them, as well as block selected friends from seeing them. The system also acts as a personalized tourist guide providing visitors of outdoors cultural heritage sites multimedia presentations about the site on the spot.

Key words: Wireless computing, Mobile Social Networking, Location-based Services
Communicated by: D. Taniar

1 Introduction

Social Networking and its supporting tools on the web, Social Networking Sites (SNS), are among the greatest successes of the Web 2.0 paradigm. The roots of SNS can be traced back to 1997 when SixDegrees.com opened for the public. Today, Facebook (www.facebook.com), MySpace (www.myspace.com), LinkedIn (www.linkedin.com) and Twitter (http://twitter.com) are only a few of the most successful examples of this social phenomenon, with each of the first three sites having well over 10 million subscribers (Facebook has more than 150 million subscribers world-wide!). Taking into account the very high penetration rates of mobile telephony in the global populace as well, it is almost obvious that social networking capabilities ported into mobile phone handsets would be of great value to users, and indeed a good number of social networking applications are available on mobile

phones today. Some manufacturers (e.g. Nokia) even ship their phones with pre-installed such applications.

In this paper, we present G2G, a client/server platform for mobile social networking that offers a number of features to the users that are only possible in the mobile world, and with the added benefit that these features are either absent from other existing systems or are of much improved quality as we shall see in the next sections. In particular, we describe a system that is capable of detecting proximity of a user to another friend or to one of his/her previously declared hotspots and notify the mobile user accordingly. It also allows for location-related notes to be published by any user to his/her friends giving rise to a number of application scenarios, from grocery-shopping lists, to location-aware recommendations, to treasure-hunt team-games, to multimedia presentations to visitors of cultural heritage sites.

1.1 Related Work

DodgeBall [4] is one of the first systems which used mobile phones as client platform that can be deemed as a location-based social networking tool which allows individuals to stay in touch and meet while moving in towns, but location-awareness must be manually provided by the user. The system works by sending and receiving a number of SMS messages, which are then processed by DodgeBall's back-end mechanism, so that user' s friends or friends of friends can be notified, given that they are at most 10 blocks away from the point where the user initially checked in. DodgeBall is available only in 22 U.S cities and lacks universal application. In the context of DodgeBall, popular places including bars, restaurants, malls, coffee shops etc. are geo-coded, in the sense that their location as described by the country, state, city, address, zip code etc has been mapped to GPS coordinates, and so the system can eventually notify all user's friends/acquaintances. However, the notification mechanism and the way the position of a user is determined suffers from some weaknesses. First, due to the SMS-based notification mechanism, the subscribers of the system and especially those who are well-connected can see their mobile phone become “bombarded” with SMS messages notifying them of friends’ whereabouts and activities. Furthermore, since the places that can be used by the users as check-in places are manually geo-coded in the DB of DodgeBall, obviously users cannot notify their friends about their arrival to any place that is not in the system database yet.

ZYB is a new project whose goal is to create a smartly connected phonebook for users' phones and web browsers. As the project's goal denotes, even though it cannot be considered and categorized as a social networking application, still it provides relevant social networking functionality. After a user sings up to ZYB, the user is called to back up all of his/her contact information, as well as his/her calendar, photos and text messages. In this way this sensitive for the user information is always secure, even if the user's cell phone is lost. Moreover, ZYB can automatically update the contact information stored in the user's handset, in case any of user's contact details happen to change. Lots of people would likely object to this idea, especially those that are more sensitive concerning privacy issues. A newly created ZYB user, by backing all of his contact details, calendar and messages to a ZYB server, essentially “reveals” sensitive information not only about himself/herself, but also about his/her friends/relatives/acquaintances; even if the user is not concerned about it, does the same hold for all of his/her contacts?

ZYB notifies a user about his/her friend social updates by reviewing his/her friends material (posts, photos etc) in social networking sites including Facebook, Flickr, and Twitter. Even more, ZYB offers some kind of location detection functionality by pinpointing the "exact" location of user on a map assuming the phone has built-in GPS receiver. However, it's worth mentioning that early tests reveal that the location detection mechanism of ZYB phonebook is not accurate at all when tested indoors. In particular, in one case, it gave to a user an estimate about a location that was 190 km away from his/ her actual location. So, it seems that the supported location awareness mechanism is mostly accurate when in manual mode, in which case the user specifies manually his/her exact location on a loaded map. The ZYB Phonebook application is currently only available on a number of select Sony-Ericsson devices.

Place Lab ([2], [7]) is a platform for location detection working on a range of mobile computing devices (from laptops to PDAs to smart phones running Symbian OS Series 60 2nd Edition only), which focused on combining a number of wireless radiation sources such as Wi-Fi beacons or GSM Access Points to show that accurate location detection is possible in nearly all of metropolitan Seattle WA, USA area, but to provide this accuracy, Place Lab requires a database of the radiation sources' geographic coordinates. Our experimental results confirm that high location detection accuracy is possible for the metropolitan Athens, Greece area as well utilizing only a single provider's GSM cell beacon information without maintaining anywhere the geographic coordinates of the GSM cell locations. Now, in rural areas where cells cover large distances, for any given definition of proximity, precision will necessarily be much lower than in an urban setting, as we prove in section 4. However, one can argue that proximity in a rural setting in the minds of most people would most likely cover a much larger distance than in an urban setting; indeed, in an isolated mountain range, one would consider a friend to be nearby, even if that friend is on the next peak, a few kilometers away, and not only if the friend is located within half a kilometer away. In fact, we compute lower bounds on average precision of proximity-detection which is inline with all experimental results. Further research work on location-aware and location-based research is described in [5], [8], [11], and [9]. A broader view in a framework for location-based services provisioning is presented in [12], while energy efficiency issues are dealt with in [13].

Live Contacts! (www.livecontacts.com) is another system that allows mobile phone users to find nearby friends, but again, it only works with mobile handsets that have a built-in GPS receiver. The system plots friends' positions on a map, but it has no concept of registering location-aware notes.

Similar systems have been recently developed at MIT as show-case demonstration projects (loco project, loco@mit.edu, Geo-Life project) for the recently released Android Mobile Phone Operating System (http://code.google.com/android). Loco in particular, is a mobile social networking application built on top of the Android phone's contact manager, meaning that anyone in the user's contacts is already his/her friend. Loco users can search within the application for an event like a party and also see other events tagged by their friends as a "party". Upon selecting a specific event a user can also browse among photos associated with this specific event, assuming that any of the participants have taken and uploaded photos related to this event. As it will further described in a latter section the event searching functionality proposed by loco was already supported and it has been further expanded in the context of G2G, supporting public and private messages which can take the form of a simple note or an alert.

Google Latitude is a recent addition to Google Maps that allows users to let friends see their location on the Google map. However, in many countries, data traffic costs can incur significant charges to users that make even modest use of the application as just one positioning request in Google Maps causes the transfer of more than 44KB downstream; in comparison, continuous use of G2G for one hour using the default settings results in a combined upstream and downstream data traffic that is less than 10KB.

In contrast to Loco and most other social networking applications, G2G does not require GPS information, but utilizes GSM cell information to its full extent to deliver proximity-detection of users to friends and/or places with enhanced reliability. It also takes the concept of location-aware notes/posts to its full potential allowing user communities to build personalized recommendations for places or events.

In the last few years, GPS-based location detection mechanisms are gradually becoming a trend, especially after the release of Google Maps and the introduction of the Google Earth API based on web services, which caters for the smooth integration of the maps functionality to almost any application. Obviously, such functionality is an "eye-candy" in the context of mobile social networking applications, since it turns such an application into a fancy "friend tracking" tool. True as that may be, we argue that systems utilizing GPS-based location detection only, despite the higher accuracy that GPS can provide, are in fact suffering from a number of serious limitations:

- ✓ They are forced to work only on handsets connected to GPS devices, or even worst require that the mobile phone handset has a built-in GPS antenna.
- ✓ They suffer from the known "open-sky" problem, i.e. they cannot pick GPS signals when the user is in an indoors environment or even when the user is under thick trees covering the sky and so on; in all such situations the system cannot work at all.

GSM cell information is nearly omni-present in the world, and as we shall see in the next sections, it more than suffices to provide proximity-awareness. In fact, it is possible to let the user specify, though in an approximate in geographic terms manner, how close he/she must be to a hotspot or friend in order to receive appropriate notification from the system.

As a final notice we mention that there exist commercial Symbian applications (Mini-GPS, GSM-navigator) that allow users to name and even group GSM cells their mobile picks up but none of them has social networking features of any kind, which is the main focus of this work.

2 Basic Concepts and Application Scenarios

Early on in the specification of the G2G system, it was decided that GPS or WLAN Access Point information should not be a pre-requisite for the mobile client platforms. This in effect poses a hard requirement that location awareness (a better term would be proximity awareness) must be provided either through GSM cell information alone or combination of GSM cell information and GPS coordinates when the latter is available. A major obstacle to detecting users' proximity to each other however when no GPS information is available, is that, a system without access to the telephony provider's internal databases has no information at all about the geographical coordinates of the Access

Point identified by a particular cell-id and area-code. Proximity is then defined only as a Boolean condition of two users or places covered by the exact same GSM cell.

On the other hand, given that very often the same place is covered by more than one cell and that hand-offs are frequent and unpredictable in such situations, it is entirely possible for two users to be in very close proximity to each other and still the system to be unable to detect their proximity because they are covered by different cells. The same holds for users who are being serviced by different mobile telephony companies, since the latter maintain their own cell network to service their clients. In the case of two such users, denoted as u1 and u2, who have a different network operator and thus are being serviced in a specific location by two different cells, denoted as c1 and c2, the system cannot detect their proximity even if the location of user u1 is within the range of cell c2 and vice versa.

2.1 Hotspot Definition and Cell Correlation

To overcome the obstacle of the unpredictable hand-offs, and let users add by themselves greater proximity-detection accuracy to the system, we introduce the notion of hotspots. The system lets users "bookmark" at any point in time or space a favorite place, to become a hotspot.

Whenever the user bookmarks a place, by providing a "hotspot-name" and "hotspot-area", the system records the hotspot information entered by the user together with the current cell-id of the Access Point covering the user's handset (and GPS coordinates if the user's phone provides such capability and the user has chosen to turn on the GPS location detection functionality). Whenever the user is in the vicinity of a hotspot (more accurately defined in a later section) he or she may denote his/her arrival to that hotspot so as to let his/her friends know where he/she is so that they join him/her. He/she does so by selecting that hotspot of choice from a Form on the mobile client's screen. If the user does not declare his/her arrival to a hotspot, he or she still remains visible to nearby friends but the friends cannot tell exactly where she/he is. In order to join the user, his/her friends would have to send an instant query message through the embedded G2G chat application, or simply call her/him and ask.

Upon announcing arrival to a hotspot, the mobile client starts to continuously collect information about the cell-id of the Access Point to which it is currently assigned. This information essentially represents overlapping Access Points at the particular location of the user. As soon as the user declares their departure from a hotspot, the system –after asking the user to verify their stay in the hotspot's area for the whole interval of time between the announced arrival and announced departure- submits to the G2G server the collected cell-id information of the Access Points that covered the mobile handset. This information allows the system to determine overlapping and thus geographically neighboring Access Points, and to determine that two users are "nearby" when they are covered by such neighboring Access Points.

Consider the system's recall R, defined as

$$R = n_{ndf} / n_{nf}$$

where n_{ndf} denotes the number of nearby friends *detected* as being near-by and n_{nf} denotes the number of *true* nearby (on-line) friends. Given the cell correlations for a particular hotspot, the system recall at the hotspot is then greatly increased since the system no longer misses friends because their covering Access Points do not match; this statement is made more precise in section 4.

Notice also that the concept of hotspots also serves indirectly as a means of recommending public places, in the sense that the hotspots that a user defines are, by definition, places that (s)he hangs out, so they carry an implicit recommendation by the user; therefore, they appear as such on all the user's friends when they are nearby.

To overcome the obstacle of proximity detection of users who are being serviced by difference mobile telephony operators we have deployed a mechanism which allows two users to manually correlate cells of different network providers when they notice that the system cannot identify them as being "near by". In order to clearly highlight this mechanism we describe a use case scenario. Two friends, Bob and Alice are hanging out in a coffee shop and they notice in their main screen of their G2G application that the system cannot detect their proximity to each other –at least one of the two has a phone with no GPS receiver. They decide to manually correlate the cells and in order to do so they both select the "Correlate Cells" option of the application's main menu. Figure 1 shows the "Cell Correlation" screen of the application.

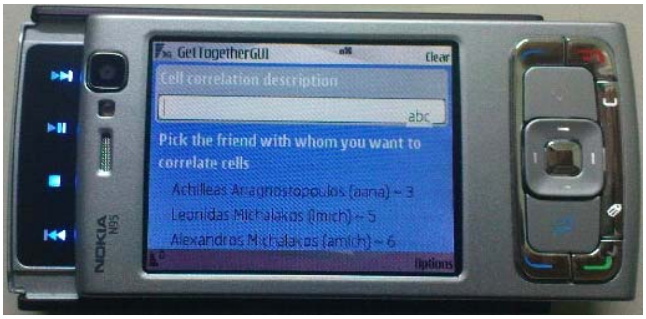

Figure 1 Cell Correlation Screen on the Nokia N95

The Cell correlation screen allows Bob to provide a description of the upcoming cell correlation and also displays a list of all of his friends. Bob selects Alice with whom he wants to do the cell correlation and then chooses the "Correlate Cells" option of the menu. After both friends follow the same steps, their current cell-id is saved in the server and a Java Servlet performs the cell correlation by storing the received information in a DB table. Each cell correlation entry consists of the following fields:

u1, Bob's id, who initiated the cell correlation handshake

u2, Alice's id , who completed the handshake by sending her own GSM information to the server

cid1, the cell id of Bob

cid2, the cell id of Alice

Figure 2 highlights the cell correlation hand-shake that takes place in the server side.

After a system-defined threshold number of pairs of users of different mobile telephony providers correlate their cells at one location, this correlation is available for all G2G users (the threshold is greater than one to avoid false correlation attacks on the system.)

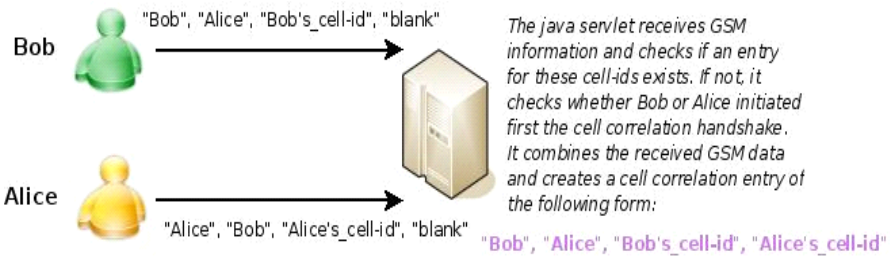

Figure 2: Cell Correlation Mechanism

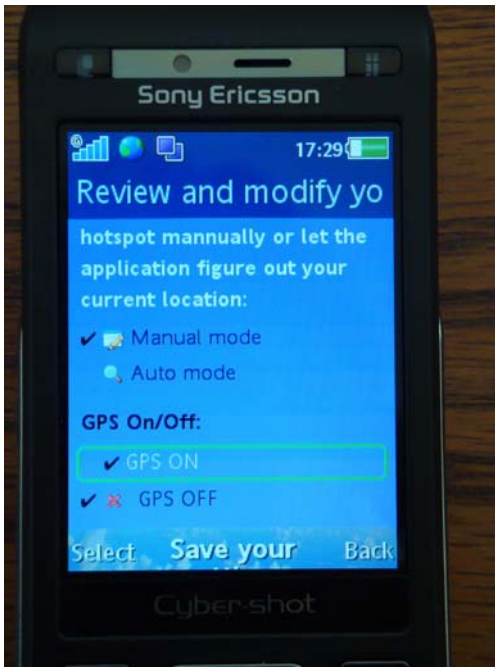

Figure 3: System Settings Enabling GPS Support on a Sony-Ericsson C905

Apart from the mechanism we presented to solve the proximity detection issue of users who are being serviced by different mobile telephony providers, another mechanism allows users having a handset with an integrated GPS antenna or even an externally attached Bluetooth GPS device, to customize their settings in the G2G client to enable GPS-based location detection (see fig. 3); in such a case, the application sends GPS coordinates along with the GSM data each time it communicates to the G2G servers location information.

This means that every time:

- ✓ a “heart-beat” of a user’s phone is sent to the server, or

- ✓ a new "hotspot" is defined, or
- ✓ an arrival to an already declared "hotspot" is declared, or
- ✓ a cell correlation of cells belonging to different providers is attempted

a GSM info (cell-id) – GPS data (longitude, latitude) correlation is also performed. In this case, not only are the search capabilities of the system enhanced and the detection proximity becomes more accurate, but also we can have an actual distance calculation between two friends. Also, after a sufficiently large number of GPS coordinates and GSM cell id correlations have been submitted to the system for a particular GSM cell, we can have an accurate estimate of the location of that cell (by taking an appropriate bary-center), even though the cell location is not provided by the mobile telephony provider.

2.2 Neighbor-of-a-Neighbor Cell-Proximity Enhanced Search

The concept of proximity search based on GSM cell information can be easily extended to a "Neighbor-of-a-neighbor" proximity search once the system knows about overlaps between the geographic coverage of GSM cells. Consider the graph in fig. 4

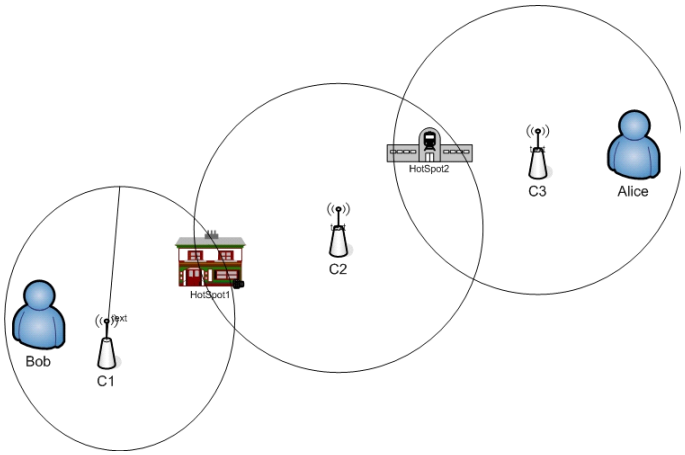

Figure 4: Neighbor-of-a-Neighbor Search: Bob is notified of Alice's Presence and Vice-Versa

In the figure, there are two locations ("HotSpot1" and "HotSpot2") which some user bookmarked as being his/her hotspots, and has let the system record the GSM cell info of the Access Points covering each hotspot. As a result, the system knows that HotSpot1 is covered by cells C1 and C2, and that HotSpot2 is covered by cells C2 and C3. The neighbor-of-a-neighbor search allows user Bob covered by cell c1 to be notified about the presence of his friend Alice who is covered by cell C3 as being nearby to him. Notice that *it is not necessary for either Bob or Alice to have been the ones that bookmarked any of the two hot-spots*.

This extended search coverage greatly enhances recall capabilities, and also precision capabilities in cases where the mobile service provider uses micro-cells or Access-Points with very small range (in the order of 100 – 200 meters). In the latter cases, most users would like to see friends covered by cells "close-by" as well.

Consider a graph whose nodes are the GSM cells that have been recorded by the system; in this graph an arc exists between every pair of neighboring cells as shown in fig. 5. The "neighbor-of-a-neighbor" proximity search can now easily be extended into a more general "neighborhood" search whereby the user specifies the maximum distance of the location of friends in terms of the number of "hops" that exist in the shortest path in the GSM cell graph connecting her/his GSM cell to their friends' cell, as shown in fig. 5.

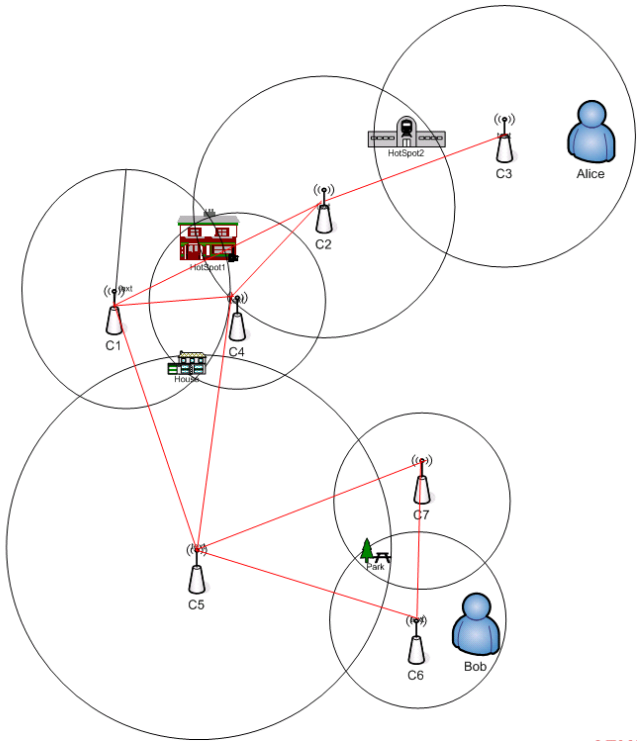

Figure 5: Alice can see Bob in her screen if she sets the Max. Number of Cell Hops for search to four or more

At this point we notice that in [10] the authors present a conceptual framework for automatic learning of hotspots (named bases and areas) based on the amount of time a user spends within a cluster of cells presumably covering the hotspot. The problem with their "on-device machine learning" approach is that even if a cluster of cells can be identified as a hotspot, for most applications this information is useless unless the hotspot has an attached semantic label, such as a hotspot name, something that only the user himself/herself can provide. If the user is to provide such information, then our approach is applicable and also more advantageous as it avoids consuming battery power on the mobile handset in order to run complicated algorithms, and it allows all users to take advantage of the cells' geographic proximities discovered by their friends.

2.3 Application Functionalities

A number of application functionalities have been built-in the G2G system.

- ✓ Spontaneous Meetings between Friends.

The basic functionality of the system is to let mobile phone users (and in particular, smart-phone users) discover seamlessly nearby friends, and vice-versa, make their own presence known to near-by friends as shown in fig. 6 –but also maintaining the option to prevent at some point in time friends from detecting that they are near by. This functionality is perfectly appropriate for the emergence of "spontaneous meetings" between friends. Nevertheless, once this proximity-detection mechanism is available, many other applications are possible, and we describe the ones we have already built-into the system.

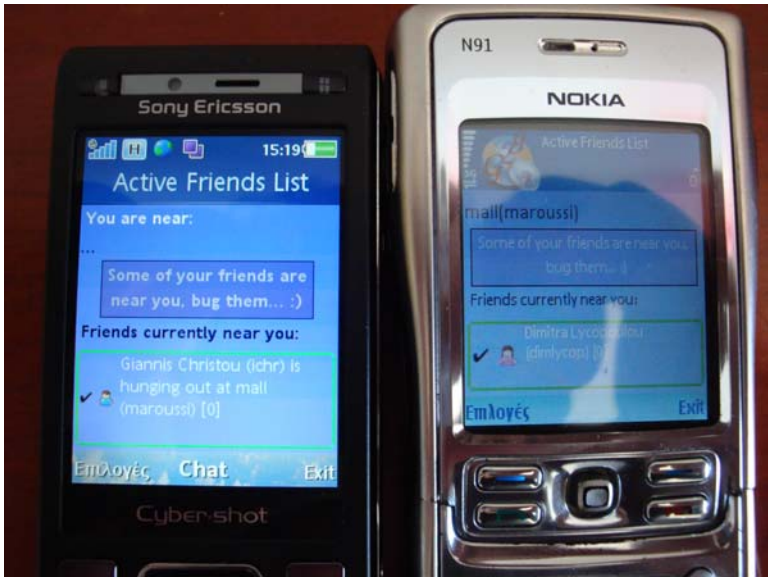

Figure 6: Detecting the Presence of Near-by Friends on a Sony-Ericsson C905 and a Nokia N91

- ✓ Chat and Instant Messaging

In order to allow for efficient verbal communication between nearby friends, we have embedded a chatting mechanism which enables users to chat in a one-to-one mode as shown in fig. 7. This mechanism allows nearby friends to quickly determine what each party is doing and to decide whether or not to meet.

- ✓ Location-Aware Multimedia Notes

Location-aware notes management is a mechanism that allows a user of the system to post a multimedia or textual note (of up to 512 characters) and determine its publication and expiration time and date intervals (figs. 8 and 9). The note becomes visible to the user and his/her friends only when they are located in the area where the note was posted and at the specified times and dates. Notes are categorized along two categorical dimensions, (I) "private" or "public" and (II) "normal" or "alerts". Private notes serve only as reminders to the user, and essentially allow for "location-based TO-DO lists", whereas public notes can serve as a recommendation system: users can provide recommendations regarding restaurants, bars, public places etc. that will be visible by a friend when

they are in the vicinity of the place where the note was originally posted. This location-aware recommender system that only shows recommendations of friends can add significant value to the user experience when the user visits a place for the first time. As already mentioned before, any hotspot that a friend of the user has previously defined will also appear as a public note in the user's notes screen albeit it will appear after any other notes that friends have explicitly defined for the area. Finally, alerts appear on the users' mobile handsets as "tickers" in the J2ME client application, and their purpose is to raise the user's attention to a particular issue.

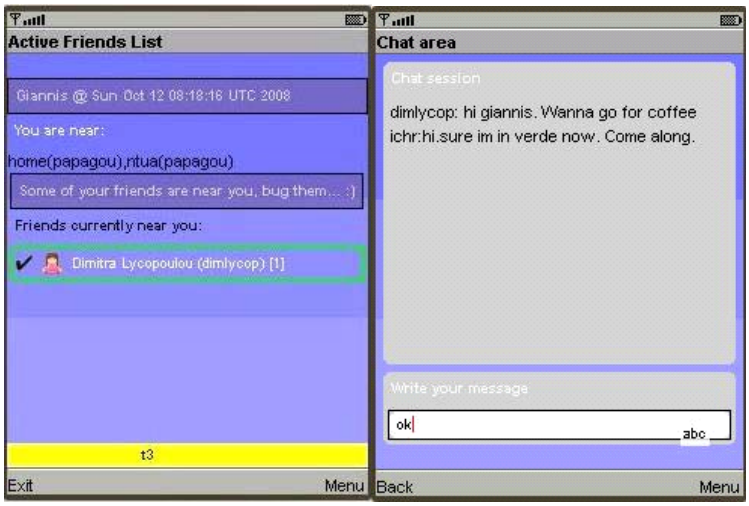

Figure 7: After Dimitra "sees" Giannis being nearby, she sends a message which is received by Giannis (left picture) and indicated in his screen by a non-zero number of unread messages from Dimitra. He then replies with a response chat message that is shortly afterwards shown in the chat area of Dimitra's mobile (right picture).

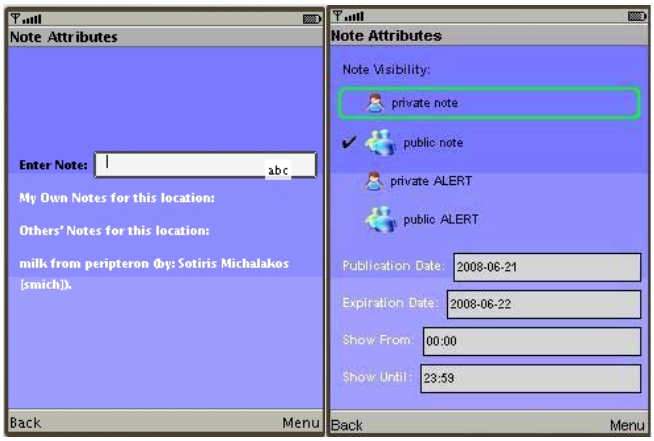

Figure 8: Users can view/post location-aware textual or multimedia notes of/for their friends; on the right is the data entry form for a note.

Overall, the notes that a user receives for any place in particular are automatically filtered by the facts that on average a user would like to see what their friends recommend and by the order in which these recommendations appear on the user's handset. This is the essence of the personalization the system provides through the Social Networks of its users.

Multimedia notes posting (see fig. 9) is of course possible through a multimedia-capture mechanism that we have implemented on top of the Java Multimedia API (JSR-135), which is at its core an event-driven API comprising a set of call-back methods. When the user chooses to record a multimedia object, a new form appears with the live camera feed shown on a canvas in their screen. At this point, the user has the option to either record video, take a snapshot picture or go back.

Figure 9: Capturing Multimedia Location-Aware Notes

The state-chart diagram in fig. 10 shows the options the user has at any moment during a multimedia capture session. Each transition from a state shown in the figure is triggered by one of the commands available to the user when they are in that state. As soon as the system reaches one state, commands are registered and/or unregistered in the appropriate form associated with the corresponding state.

Finally, a number of innovative application scenarios can be realized on top of the embedded application functionalities described so far:

- ✓ Location-Aware Multimedia Tours of Outdoors Cultural Heritage Sites

Multimedia public notes can also serve as tourist guides in outdoors cultural heritage sites for users of phones equipped with GPS receivers. Indeed, appropriate multimedia content about a cultural heritage site can be geo-located and then automatically downloaded to interested users that wish to receive such content while they are visiting a site according to the principles set forth in [15]. Also, a more recent tour guide allowing personalized recommendations is presented in [6].

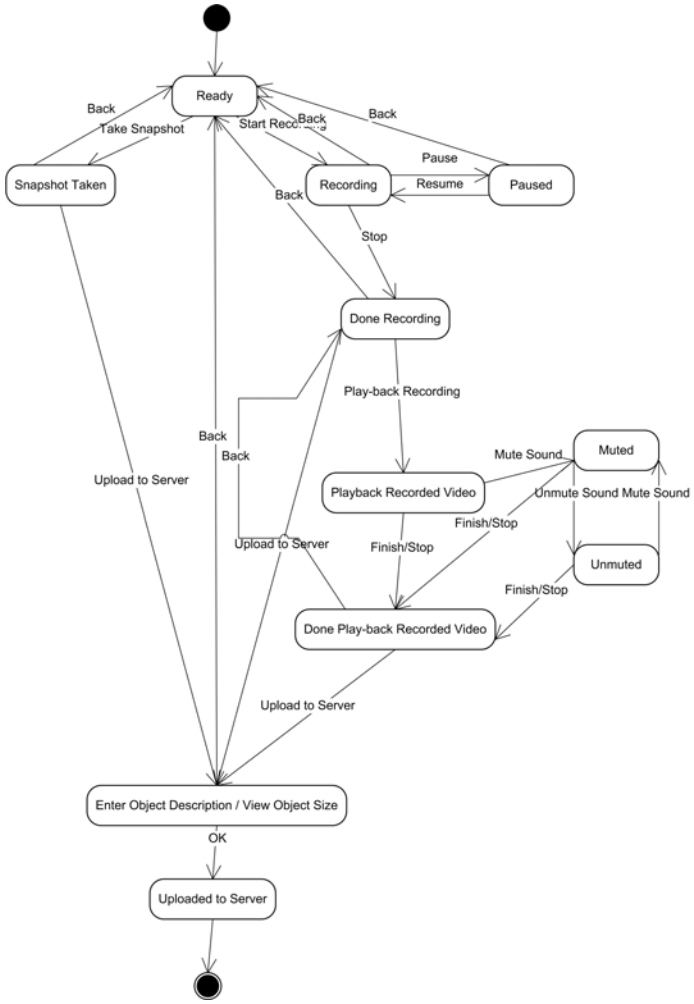

Figure 10: UML State-Chart Diagram showing the interactions of the user with the system during multimedia capture. Each transition from each state represents a command available when the user is in that state.

- ✓ Treasure-Hunt Games

The combination of these two proximity-aware mechanisms (detection of nearby friends and posting of notes) allows for setting-up treasure-hunt or other adventure games! A game designer can visit a number of places where clues are to be found for the game progress, and post various notes for teams of his friends to pick up in their mobile phones at later specified times. Each note would be a clue as to the time and location of a place that the gamer should be at in order to receive the next clue. The goal of the game could be to detect in the mobile phone the presence nearby of the "missing person" or some other target in a particular location that will be known to the game participants only by following the clues.

3 System Architecture

The system architecture, shown in fig. 12, follows the client/server model with HTTP serving as the message transport protocol for communication between clients running on smart phones and the G2G services which are hosted on a Tomcat container engine (http://homer.ait.gr:8080/GetTogether). The G2G central servers store all information (such as hotspots definitions, cell correlations, user relationships and so on) on their databases (MySQL), and are responsible for all processing needed in order to disseminate information to the users' handsets. The central G2G servers also provide a web user-interface, through which users subscribe to the G2G platform and invite friends to join them, as well as download the client applications. Finally, a dedicated server component ("Focused Crawling") crawls targeted sites using focused crawling techniques and extracts information about public places/events that are subsequently geo-coded and stored in the G2G databases in order to serve as additional recommendations for places or events to visit for all G2G users [1]. The system crawls a number of sites initially provided manually by the system administrators, and then gradually learns whether links where the base URLS point to contain relevant information or not, using semi-supervised continuous learning techniques. The focused crawling module has four distinct states that it can be in: (a) preparatory, (b) training, (c) crawling and (d) reporting state, and it directly integrates a number of machine learning algorithms implemented in the WEKA suite of machine learning tools and it of course can directly use any algorithm implementing the WEKA interfaces; in our setting, using standard TF-IDF based representation of the web pages crawled and decision tree and SVM ensembles of classifiers results in a system precision of more than 80%; standard Information Extraction methods then allow the system to automatically retrieve the actual address (and time-stamp) of the place (or event).

The System Configuration Component on the server side also allows configuring the system so that it connects with external social networking sites in order to get the circles of friends of users already established in other sites. Users have the option to declare their membership in external Social Networking Sites (SNS). In such a case, a user is notified of another user's proximity even when the two users have not explicitly declared they are friends in the G2G system, assuming of course they are friends on the external SNS. This is also very important because existing social relationships between people established in an SNS do not have to be explicitly re-established in our servers by the G2G users.

The client modules are a combination of servers written in C++ for the Symbian OS Series 60 [3] and J2ME applications [14]. A special server for getting GSM cell-id information was written in Symbian C++; this server can be queried via sockets from any J2ME application in order to provide GSM cell-id, and is only necessary for certain Symbian Series 60 devices that do not offer a native J2ME mechanism for retrieving this crucial information. In the newer releases of Java Platforms for most smart-phone devices from most manufacturers a Java System property allows retrieving cell-id. Consequently, the G2G platform can run in almost all new generation cell phones, since the client midlet can run as a stand-alone application without the support of the previously described Symbian daemon.

Having the needed location information (GSM, GPS, or both), the client application notifies the central G2G servers of the position of each G2G user, since a J2ME heart-beat thread periodically initiates an HTTP request to the server sending it the location information of the user's handset. In

return, the server replies with a response including all nearby friends of the user that have authorized the user to "see" them. The retrieved nearby friends must have sent their last heart-beat within a specified time period in order for the system to consider them as "online" and let the user detect their presence.

In total, nine or ten different "worker" threads are spawned after the main J2ME MIDlet starts running. Each of these threads is responsible for communicating with the G2G servers for different tasks. Five of the threads upon start-up, enter the wait state, waiting to be notified by the main UI thread in response to a user action so as to perform an HTTP-based communication according to the specific user action. The remaining five threads, namely the FetchCellInfoThread, the FetchGPSInfoThread (optional), the SendHeartBeatThread, the FindCurLocationThread, and the NotesManagementThread sleep for a user-defined period of time and then initiate a communication with a server (the first thread possibly initiates a socket-communication with the Symbian C++ server running on the phone if this is necessary) in order to do accordingly: (1) find the current GSM cell information, (2) (optional) retrieve the latest valid GPS coordinates (3) send the current GSM cell information to the server and get nearby friends, (4) automatically detect any nearby hotspots, and finally (5) fetch any related notes and/or multimedia presentations for the area.

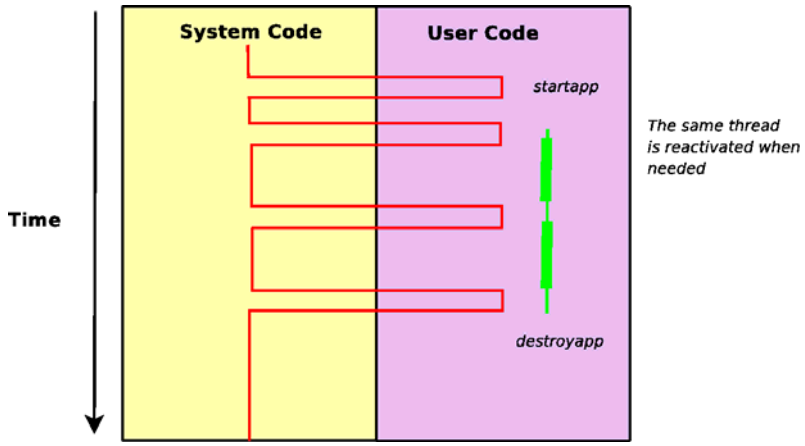

Figure 11: G2G Threading Model Mechanism

The thread mechanism implemented is specifically designed to serve the needs of such a resource intensive application. The UI thread handles events that can be Operating System events or user actions by responding to user commands by determining which of the running threads is responsible for handling the event and acquiring a lock on the sleeping thread; it then sets the appropriate data for the thread to use and notifies (wakes) the thread which in response performs the action dictated by the data set by the UI thread. Figure 11 explains this behavior. The important benefit of this mechanism is that there is no need for spawning and killing threads during the application's run-time. Each thread is only spawned once, and killed during shutdown.

The client MIDlet is approximately 10,000 lines of Java code –a relatively small code base-, and was written using J2ME-Polish [16], an Open-Source suite of tools for developing professional-looking Graphical User Interfaces in J2ME.

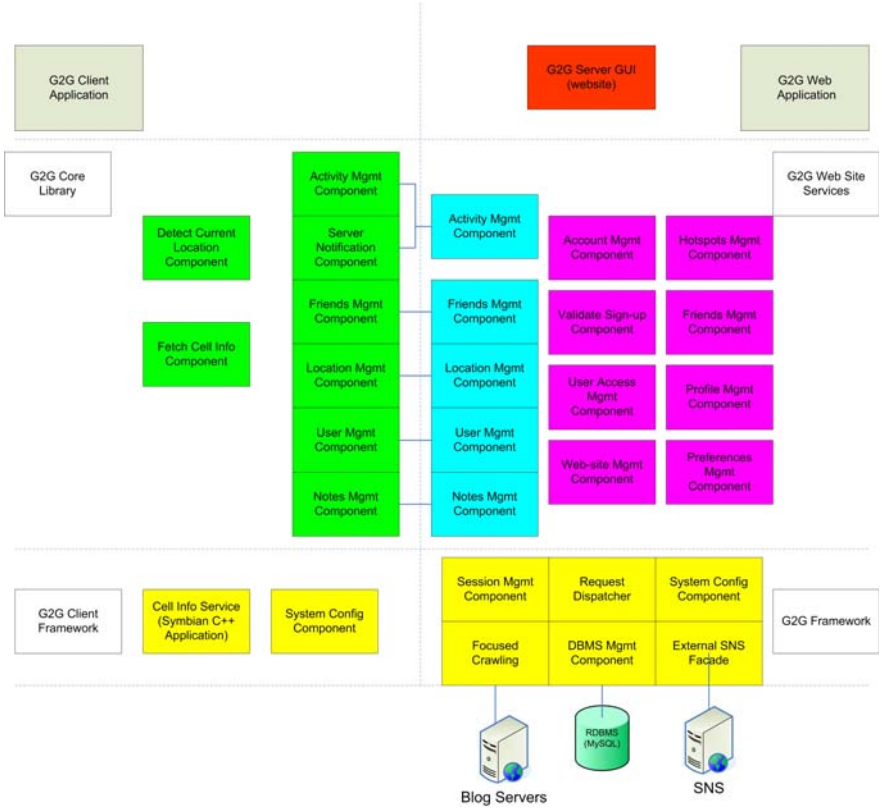

Figure 12: Overall System Architecture

3.1 Security Considerations

As a counter measure against DB flooding by malicious users who may try to always correlate the same pair of cells or even by a malicious user who goes from location to location and performs only the one part of the cell correlation hand-shake, resulting in incomplete cell correlation entries, we have developed an appropriate back-end mechanism. More specifically a server-side daemon is running periodically to remove incomplete entries stored in the DB at least half an hour before the actual running time of the daemon.

The mechanism for detecting cell correlations could be potentially misused by malicious users by submitting to the system correlations between cells that are geographically far apart –by simply traveling a large distance so they capture distant cells, and then registering them as if they never left the hotspot. The simple counter-measure of never allowing any user to correlate more than a small fixed number (set to 10 after experimentation with the characteristics of the topology of the mobile

networks in Athens, Greece) of cells from any hotspot at any time suffices as in the scenario the user would collect much more than 10 cells and the system then blocks the proposed correlations.

Also, a number of security issues spring from the fact that the system uses a client/server architecture for information dissemination to the mobile clients. For this reason, communications between the mobile handsets and the user make use of the secure http protocol. Location information is kept in one database encrypted, whereas user-related information resides on a different database on a different server as shown in fig. 13.

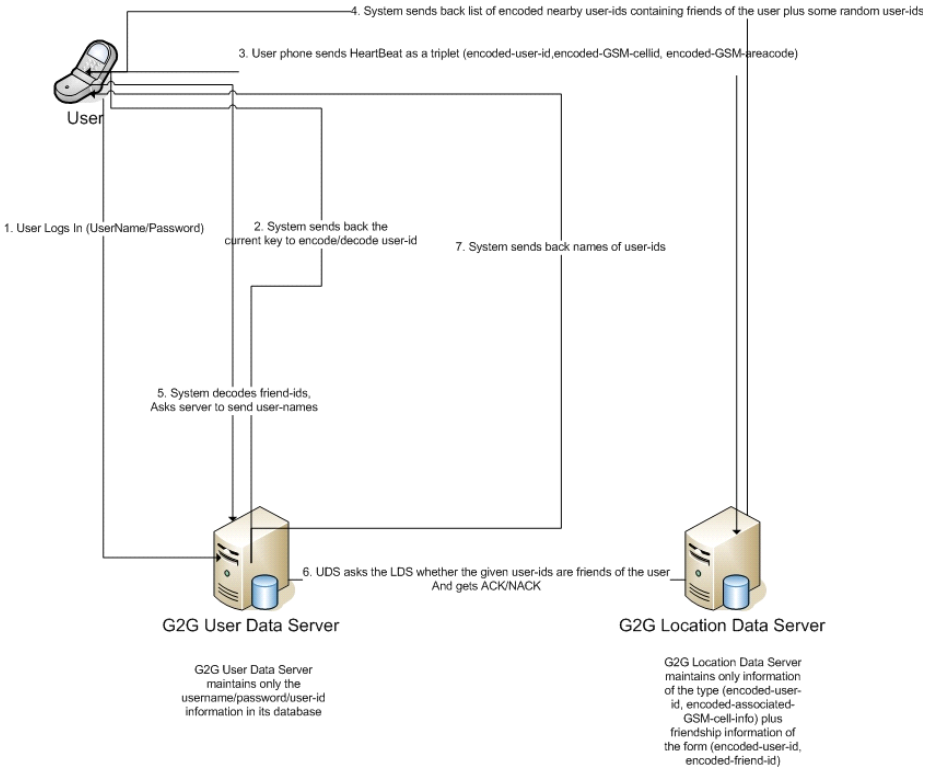

Figure 13: Securing Location Information on the Back-End

An attacker would have to compromise two different servers and databases to be able to find out a specific user's location information. We are in the process of investigating the use of multiple distributed servers in a Super-Peer network fashion for enhancing system security.

Finally, privacy concerns are dealt with by the facts that

1. the system never discloses user location-related information unless the user has his/her mobile handset turned on, has installed the application and runs the application, and has indicated that (s)he wants her/his friends to be notified when they are near; the system only notifies the selected friends of the user in the user's vicinity.

2. the system provides the option to become "invisible" to certain friends whenever the user wishes to do so, and for as long as the user wishes.

4 Accuracy of Cell-Id-based Proximity Detection

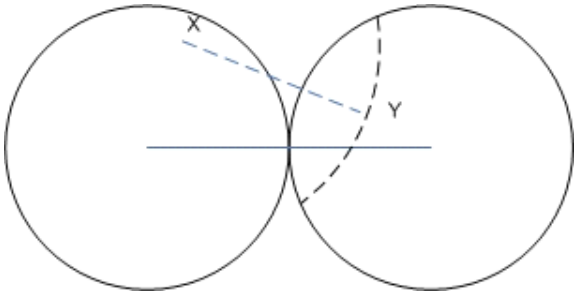

Figure 14: Worse-case scenario for Precision Accuracy: Hotspot at the intersection of the range of two Accesss Points. For a user located at point X, only users located to the left of the dotted curve in the picture are nearby, but the system will fetch all friends located in any of the two plotted circles

Since accuracy of proximity when the user device sends GPS coordinate information is very high – in the order of a few meters-, we are only concerned here with the case where GPS coordinate information is not available. Consider the system's precision P for normal queries (as opposed to neighbor-of-a-neighbor extended search queries), defined as

$$P = n_{ndf} / n_{df}$$

where n_{df} denotes the number of friends detected as being near-by (but not necessarily near-by). Clearly, precision depends on the definition of nearness. If we define two users as being nearby when their absolute distance R_M from each other is no longer than 2 km. (essentially a "walking distance" definition) and assuming that mobile telephony Access Points in urban areas have a radius of coverage never exceeding r=0.5 km, we see that the system precision will always be 100% in these areas, assuming of course that the user has set to 1 the "Max. Number of Cell Hops Away" that the system is allowed to search for friends! Any relaxation of the maximum distance allowed in the definition of users nearness will of course maintain the 100% accuracy of precision in such areas. On the other hand, if we restrict the maximum allowed distance of two nearby users to be R_M=1 km. then the worst case for system precision will occur when the hotspot is covered by two Access Points whose circles of coverage just touch at the hotspot. In this worse-case scenario (see fig. 14), average precision would be

$$P = (Pr[\,||X - (Y+S)|| < R_M \mid X,\ Y \in R(0,r)\,] + Pr[\,||X - Y|| < R_M \mid X,\ Y \in R(0,r)\,]) / 2$$

where X and Y are random vectors in 2D with each component following the uniform distribution in $[-r, r]$, $S=(2r,0)^T$ is a vector in two dimensions, and $R(c,d)$ denotes the circle centered around c with radius d. The two terms of the sum correspond to the cases where two users are in different circles or in the same one. Carrying out this computation, the formula for average precision in the worse case scenario becomes

$$P(r,R_M) = \frac{1}{2\pi^2 r^4}\int_0^{R_M}\int_0^{r}\int_0^{2\pi}\int_0^{2\pi} I_1(\rho,\varphi,\sigma,\vartheta)\rho d\varphi\sigma d\vartheta d\rho d\sigma +$$

$$\frac{1}{2\pi^2 r^4}\int_0^{R_M}\int_0^{r}\int_0^{2\pi}\int_0^{2\pi} I_2(\rho,\varphi,\sigma,\vartheta)\rho d\varphi\sigma d\vartheta d\rho d\sigma$$

where

$$I_1(\rho,\varphi,\sigma,\vartheta)=\frac{\left[r^2-(\rho\cos\varphi-2r+\sigma\cos\vartheta)^2-(\rho\sin\varphi+\sigma\sin\vartheta)^2\right]_+}{\left[r^2-(\rho\cos\varphi-2r+\sigma\cos\vartheta)^2-(\rho\sin\varphi+\sigma\sin\vartheta)^2\right]}$$

$$I_2(\rho,\varphi,\sigma,\vartheta)=\frac{\left[r^2-(\rho\cos\varphi+\sigma\cos\vartheta)^2-(\rho\sin\varphi+\sigma\sin\vartheta)^2\right]_+}{\left[r^2-(\rho\cos\varphi+\sigma\cos\vartheta)^2-(\rho\sin\varphi+\sigma\sin\vartheta)^2\right]}$$

and $x_+ \equiv \max(x,0)$.

The center of the left circle in fig. 14 is at the coordinate system origin, point X is represented in polar coordinates $(\rho\cos\varphi,\rho\sin\varphi)$, point Y is located at $(\rho\cos\varphi+\sigma\cos\vartheta,\rho\sin\varphi+\sigma\sin\vartheta)$, and the center of the right circle is located at (*2r*,*0*). The functions $I_i(\rho,\varphi,\sigma,\theta)$, $i=1,2$ are index functions that take the value 1 when the points X and Y are "nearby" according to the definition of "nearness", and are zero otherwise. The $I_1(\rho,\varphi,\sigma,\theta)$ function in particular becomes 1 when the two points X and Y are in different circles and are considered "nearby" and is zero in the rest of the integration domain, whereas the $I_2(\rho,\varphi,\sigma,\theta)$ function is zero everywhere in the domain of integration except when the two points X and Y are in the same circle and at a distance less than R_M where it assumes the value 1. Numerical integration using MATLAB then gives a worse case average precision for $r=0.5$ *km* and $R_M=1$ *km*. of 0.71, and when we increase the maximum allowed distance to 1.5 km. the worse-case average precision becomes 0.95, as can be seen in fig. 15.

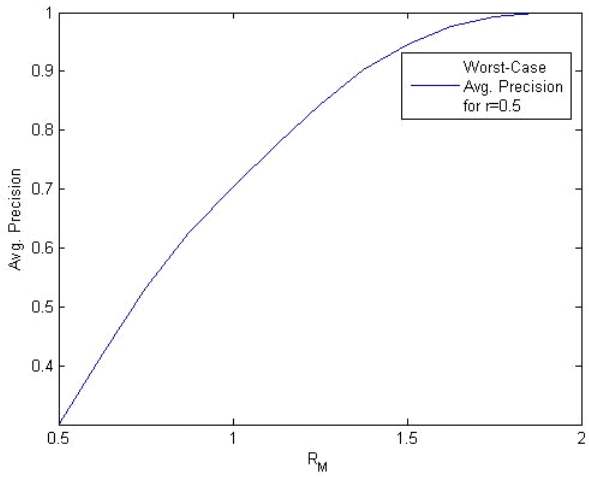

Figure 15: Plot of the worst-case average precision $P(r,R_M)$ for r=0.5 km.

Notice that *in the real-world the average precision will be much better* than this because cells do not cover a spherical area but instead form cones of coverage. Similarly, the average Recall $R(r, R_M)$ of the system as defined in section 2.1, in this case becomes

$$R(r,R_M)=\frac{\pi r^2+\frac{1}{2\pi r^2}\int_0^{R_M}\int_0^{r}\int_0^{2\pi}\int_0^{2\pi}F(\rho,\varphi,\sigma,\vartheta,r)\rho d\varphi\sigma d\vartheta d\rho d\sigma}{\pi R_M^2}$$

where

$$F(\rho,\varphi,\sigma,\vartheta,r)=\frac{\left[r^2-(\rho\cos\varphi-2r+\sigma\cos\vartheta)^2-(\rho\sin\varphi+\sigma\sin\vartheta)^2\right]_+}{\left[r^2-(\rho\cos\varphi-2r+\sigma\cos\vartheta)^2-(\rho\sin\varphi+\sigma\sin\vartheta)^2\right]}+\frac{\left[r^2-(\rho\cos\varphi+\sigma\cos\vartheta)^2-(\rho\sin\varphi+\sigma\sin\vartheta)^2\right]_+}{\left[r^2-(\rho\cos\varphi+\sigma\cos\vartheta)^2-(\rho\sin\varphi+\sigma\sin\vartheta)^2\right]}$$

which after numerical integration gives values $R(0.5, 1)$=0.427. (The function $F(\rho,\varphi,\sigma,\theta,r)$ semantics are as in those of the above defined index functions $I_i(\rho,\varphi,\sigma,\theta)$.) This value is significantly higher than what it would be in the case where only a single cell of radius r covers the area (0.25). Of course, in rural areas, where cells have larger range *r*, recall improves a lot, e.g. $R(1,1.5)$=0.69. Recall also improves drastically with the number of correlated cells in any area.

In practice, our experimental results indicate that for most convenient definitions of maximum allowed distance for nearness, the average precision in the various suburbs of Athens Greece is always very good. Table 1 shows the results in detail.

Suburb	**Avg. Prec. for 1km max. distance**	**Avg. Prec. for 2km max. distance**
Agia Paraskevi	90%	100%
Peania	70.5%	85%
Cholargos	97.7%	100%
Papagou	95%	100%
Kalyvia	83%	91.3%

Table 1: Average Precision of Nearby Friends Detection

The less-than-one-hundred percent precision accuracy observed in the Peania and Kalyvia areas is due to the fact that they are remote, almost rural areas, far from Athens and they are served by long-range Access Points, effectively increasing the distance at which two users may be located and still be considered by the system as being „nearby“. For one mobile provider, 21 different cells were detected in the Papagou municipality (an Γ-shaped area covered by a diameter no longer than 3km) whereas in Peania, a rural area in the outskirts of Athens that used to be considered a village, only 4 different cells were covering an area of more than 4 km diameter.

5 Conclusions and Future Work

We presented G2G, a proximity-awareness platform for Mobile Information Device users that allows them to detect the presence of their friends when they are nearby, to post notes that are localized in

time and space, and using these basic mechanisms, to essentially create recommender systems for places and events and even set-up team-based adventure games, or personalized multimedia tours of outdoors cultural heritage sites. We have described the details of a "Neighbor-of-a-neighbor" mechanism for extending the range of search of nearby user-friends and hotspots in the system in the absence of GPS coordinate information, and we have seen how the introduction of user-defined hotspots allows for the system to collect geographic-proximity information about GSM cells without the need for detailed charting of those cells during initial set-up. Nevertheless, more accurate geographic proximity of hotspots and cells will eventually be provided by the G2G user community itself, because the G2G platform is able to retrieve and submit GPS information from clients that are equipped with GPS antennas. Eventually, as our user community keeps using the system and going places with GPS-enabled phones, the cells covering those places will be tagged with geo-location information, thus allowing the system to acquire a geographic notion of proximity of the various cells the user community visits.

References

[1] Ammolochitis, E.: "Design & Implementation of a Focused Crawler", Master's Thesis in Master of Science in Information Technology and Telecommunications, Athens Information Technology, 2009.

[2] Chen, M., Sohn, T., Chveler, D., Haehnel, D., Hightower, J., Hughes, J., LaMarca, A., Potter, F., Smith, I., & Varshavski, A., "Practical Metropolitan-scale Positioning for GSM Phones", In Proc. of the 8th Conf. on Ubiquitous Computing, Orange County, CA, USA, Sep. 17-21, 2006.

[3] Coulton, P., Edwards, R., and Clemson H., "S60 Programming: A Tutorial Guide", John Wiley & Sons, Hoboken, NJ, 2007.

[4] Dixit, J., "The Artful & Mobile Dodger", IEEE Spectrum, March 2005.

[5] Fongen, A., Larsen, C., Ghinea, G., Taylor, S.J.E., Serif, T.: "Location based mobile computing - A tuplespace perspective". Mobile Information Systems 2(2-3): 135-149, 2006.

[6] Gulliver, S.R., Ghinea, G., M. Patel, Serif, T.: A context-aware Tour Guide: User implications. Mobile Information Systems 3(2): 71-88, 2007.

[7] Hightower, J., LaMarca, A., and Smith, I. "Practical Lessons from Place Lab", IEEE Pervasive Computing, vol. 5 (3), 2006.

[8] Jayaputera, J., Taniar, D.: "Data retrieval for location-dependent queries in a multi-cell wireless environment". Mobile Information Systems 1(2): 91-108, 2005.

[9] Kang, S.-W., Song, M., Park, K., Hwang, C.-S.: "Semantic prefetching strategy to manage location dependent data in mobile information systems". Mobile Information Systems 1(3): 149-166, 2005.

[10] Laasonen, K., Raento, M., Toivonen, H. "Adaptive On-device Location Recognition", Lecture Notes in Computer Science, vol. 3001, pp. 287-304, Springer-Verlag, Berlin – Heidelberg, 2004.

[11] Lee, D. L., Zhu, M., Hu, H.: "When location-based services meet databases". Mobile Information Systems 1(2): 81-90, 2005.

[12] Priggouris, I., Spanoudakis, D., Spanoudakis, M., Hadjiefthymiades, S.: "A generic framework for Location-Based Services (LBS) provisioning". Mobile Information Systems 2(2-3): 111-133, 2006.

[13] Song, M., Kang, S.-W., Park, K.: "On the design of energy-efficient location tracking mechanism in location-aware computing". Mobile Information Systems 1(2): 109-127 2005.

[14] Riggs, R., Taivalsaari, A., and VandenBrink, M. Programming Wireless Devices with the Java 2 Platform, Micro Edition. Addison Wesley, Upper Saddle River, NJ, 2001.

[15] Stricker, D., Daehne P., Seibert, F., Christou, I.T., Almeida, L., Carlucci, R., and Ioannidis, N.: "Design & Development Issues in ARCHEOGUIDE: An Augmented Reality based Cultural Heritage On-site Guide", Proc. of the Intl. Conf. on Augmented Virtual Environments and 3D Imaging (EuroImage 2001), pp. 1-7,IEEE Press, Myconos, Greece, May 2001.

[16] Virkus, R. Pro J2ME Polish: Open Source wireless Java tools suite. Apress, Berkeley, CA, 2005.