WebCall – A Rich Context Mobile Research Framework

Zhigang Liu, Hawk Yin Pang, Jun Yang, Guang Yang, and Péter Boda

Nokia Research Center
955 Page Mill Road, Palo Alto, CA 94304, USA
{zhigang.c.liu,hawk-yin.pang,jun.8.yang,guang.g.yang,
peter.boda}@nokia.com

Abstract. The ever-increasing capability of mobile devices enables many mobile services far beyond a traditional voice call. In this paper, we present WebCall, a research framework on how to share and utilize the rich contextual information about users, such as phonebook, indoor and outdoor location, and calendar. WebCall also demonstrates a few services (e.g. human powered questions and answers) that can be built on top of user context. Third-party services can be integrated with WebCall through a simple API and potentially benefit from context filtering. An invitation mechanism is introduced to enable bootstrapping user base. Privacy concern is addressed by giving full control to users on how to share their information.

Keywords: Mobile social networking, user context, location based services, phonebook.

1 Introduction

Mobile applications are becoming more and more widely available to users, often aiming to provide similar functionalities as online applications. The ubiquitous features mobility brings to these applications can provide a new level of user experience. Location and proximity information, sensor inputs, activity-related information, etc. mashed up with on-device data and shared with other users in a privacy preserving manner opens up entirely new dimensions to connecting people, sharing and consuming, and social networking in general.

The research framework we present in this paper takes the most central element of all on-device applications, namely the contact list (or phonebook), as the starting point. It is our initial assumption that most of the important persons are already present in the contact list and it may provide an additional value to the user if contextual information appended to contact list entries are available. Originally, we planned to provide a visual channel additionally to the voice channel when two persons are having a phone call, showing relevant contextual information through the visual channel, such as current location and weather information, the last picture taken, calendar occupancy. Later, we extended this "my mobile page" to off-the-call cases too, and to other visual representations around location: showing a group of people, e.g. colleagues' indoor locations within the work place, occupancy of meeting rooms,

S. Ystad et al. (Eds.): MobiCASE 2009, LNICST 35, pp. 107–122, 2010.

time zone and city level information, etc. Furthermore, we extended the framework with a simple plug-in mechanism where opt-in services can be provided to the user. Instead of focusing on known and popular service, e.g. micro-blogging and status reporting, we created a novel small-scale service that enables users to send anonymous questions to their 1st, 2nd, and further level of social groups. Answers provided anonymously by others are presented in a summarized way.

The developed research framework is used for experimenting various use cases and novel ideas. It has not been, neither planned to be introduced for wider use as a product or service. This research tool enables us to quickly test technologies, prove concepts, and learn user acceptance and experience feedbacks in small-scale trials. It was partially derived from our previous work on social proximity networks [1] but with more focus on system implementation.

The paper is structured as follows. Section 2 explains the overall WebCall system architecture. Section 3 describes features and their underlying technical details. Section 4 deals with lessons we learned through our project. Finally, we conclude the paper with future outlook and directions of research.

2 System Architecture

We chose the traditional client-server architecture to prototype our WebCall framework. Below are a few rationales behind the approach:

- The client-server architecture meets the basic requirements of authentication and authorization. A centralized repository also simplifies contextual data mashup, new service deployment and universal user access anytime from any device.
- Persistent data storage. This is needed to store users' historical data (e.g. past locations and communications) that can be useful for creating more personalized services. Although some of the data may be stored in mobile devices and processed in a distributed manner, a persistent storage on the server simplifies the system. In addition, it is particularly valuable as a user may lose or damage his/her mobile device.
- Easy "plug-ins" of third party services, e.g. advertisement or feeds to/from other social websites. The WebCall server is the center of intelligence and handles data filtering and aggregation.
- Ease of development and deployment. There are many off-the-shelf open source modules for us to prototype WebCall. We can avoid developing everything from scratch and focus more on the feature design and experiments.
- Ubiquitous HTTP connectivity. This avoids all the technical issues caused by Firewall/NAT traversal. In addition, it solves a practical problem that wireless operators may block non-HTTP traffic in their networks.

Fig. 1 shows the overall software system architecture with the main logical components.

In our prototype, we developed a WebCall client on Nokia N95 devices. The UI is implemented as a Web Runtime (WRT) widget [2] which produces good UI on mobile devices with existing web technology (e.g. basic JavaScipt or more advanced AJAX). However, since a widget cannot access local resources on device such as contacts, calendar and Bluetooth due to security constraints, we also implemented a

local python module on top of Pys60 [3]. The python module acts as a local HTTP server. When the widget needs to access local resources, it sends an HTTP request to the python module which then processes the request and returns results in an HTTP response. This two-component client architecture is quite typical for developing web-based applications for mobile devices[1].

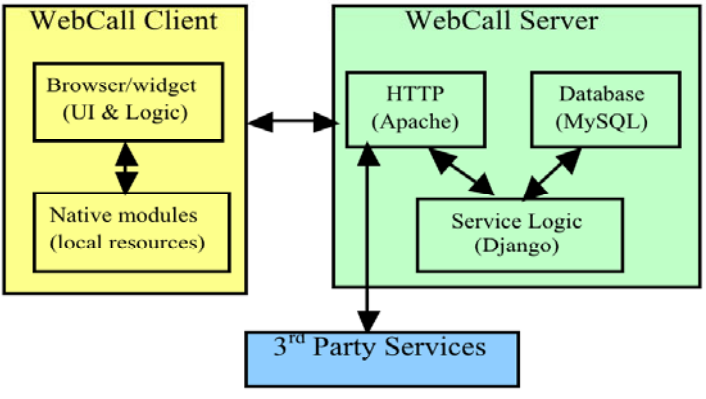

Fig. 1. WebCall system architecture

On the server side, we implemented the WebCall framework in a typical LAMP (Linux-Apache-MySql-Python) approach. Below is a quick overview on how the server processes each request from the client:

- The client sends an HTTP request to the server, triggered by either user interactions (e.g. menu selection) or background logic (e.g. location detection). Due to its simplicity and easy handling in Python, JSON [4] was chosen as the encoding format for the HTTP payload.
- The Apache HTTP server receives a request and dispatches it to the service logic unit if the URL prefix in the request matches that of WebCall.
- The service logic unit is the main part of the WebCall server. It is implemented in Python using the Django [5] framework. It parses each request received from the Apache HTTP server and dispatches it further to a particular Python function based on the request URL and request content (which is encoded in JSON). The python function processes the request and returns results in an HTTP response message.
- While processing a client request, the service logic unit will query and update MySql [6] databases as necessary. Fig. 2 shows a subset of tables in the database, in which virtual clipboard is a generic container for user related information (e.g. calendar, business card, messages, etc.)

[1] The latest version of WRT added a JavaScript device object through which a widget can access – and in some cases change – certain information on device. However, the capability is not without limit. A native proxy process is still needed for a widget if it needs to access certain resources such as writing to local file system.

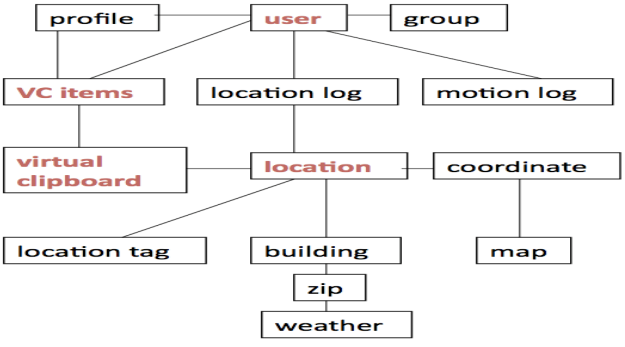

Fig. 2. Subset of database tables for WebCall

One important consideration in our system design is the capability to support third party plug-in services. As shown in Fig. 1, WebCall server can incorporate third party services through an HTTP interface. This is the way many features are implemented as described in the following section.

3 Features

The WebCall features are roughly categorized into channels: *People*, *Places*, and *Stories* (Fig. 3, left). After clicking on the people icon, a user will see a list of his/her contacts (Fig. 3, right), which the WebCall client reads from the local phonebook. A smiley face preceding a name indicates that the user is already registered with WebCall. This allows a user to know whom he/she can invite into WebCall system (see next section). Users are identified in the system by phone numbers in the international standard format. Usernames are allowed but optional.

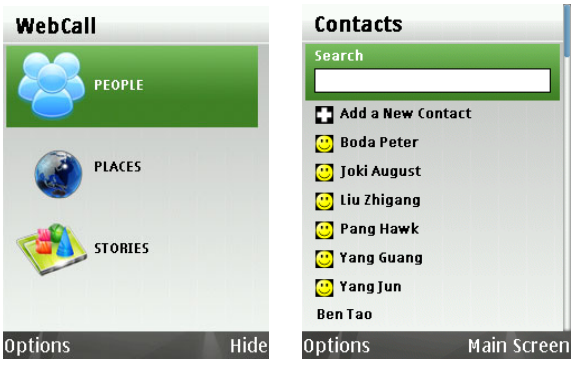

Fig. 3. WebCall top view and contact view

A user can initiate a phone call to a particular callee by clicking on his/her name. At the same time, information about the callee will be displayed on the screen as a dropdown list.

The "places" channel shows information about user locations, both indoor and outdoor. It can also show locations of a group of users.

The "stories" channel includes questions-and-answers and third party "plug-in" services.

The following sections will describe each of these services in detail.

3.1 Invitation and Registration

We provide an invitation mechanism in the WebCall framework to allow a registered user to invite his/her friends from the contact list who are not WebCall users yet. This is based on the concept of viral marketing where the number of users of a system could hit the tipping point and grow exponentially after initial success.

The invitation mechanism consists of two parts, one on the server side and the other on the client side. A WebCall user may select a contact from the phonebook and send an invitation to him/her if the intended recipient is not a WebCall user. The recipient will subsequently receive a Short Message Service (SMS) message with a short greeting text and an embedded link. Clicking on the link will lead the recipient to a registration/download page to be described shortly. Meanwhile the entire invitation process is recorded on the WebCall server such that various statistics can be retrieved later on.

3.1.1 Invitation

When the client sends out an invitation request, the server needs to handle this request accordingly. Under the Django framework, we first define a database schema for the invitation module that includes the InvitationID (a unique number to represent each new invitation), ApplicationID (optional), InviterName, InviterPhone, InviteeName, InviteePhone, Invitation_Status as well as different time stamps for status change. The invitation status can be "invitation started", "invitation cancelled", "invitation responded", "application installed" or "application removed", etc.

Then we divide invitation requests into five categories and handle them separately:

- Invitation_create: create a new invitation ID from inviter and the associated invitee information.
- Invitation_update: update an existing invitation ID with different status and corresponding time stamps.
- Invitation_retrieve: retrieve invitation status and corresponding time stamps information from an existing invitation ID.
- Invitation_track: for a given InviteePhone or InviterPhone, return a list of its inviters, invitees or a tree of invitees respectively.
- Invitation_stats: for a given InviterPhone, return some basic statistics like number of its direct invitees or indirect invitees.

On the client side, invitation functions are implemented as a library in JavaScript. The library exposes a single, uniform API that can be used not only in WebCall but also potentially in other third-party applications wishing for such functionality.

This single, uniform API is through an InvitationManager class, which internally consists of two building blocks: LocalComm and WebComm. LocalComm communicates via standard AJAX to the local HTTP server as described in Section 1. In the

current version of InvitationManager we use LocalComm to mainly send invitations through SMS to the receivers.

The other module, WebComm, is architecturally similar but used to communicate to the WebCall server. It is also based on AJAX and adopts a similar syntax. The five basic functions it provides are invite, cancel, update, retrieve and stats. With the exception that cancel and update are tied to the same server-side request category of Invitation_update, the rest is one-to-one mapped to the server-side categories. Note that we have not implemented the counterpart of Invitation_track in the client library intentionally, because it is very difficult to efficiently and effectively present such complex data on the small screen. We believe it is more useful on a personal computer (PC).

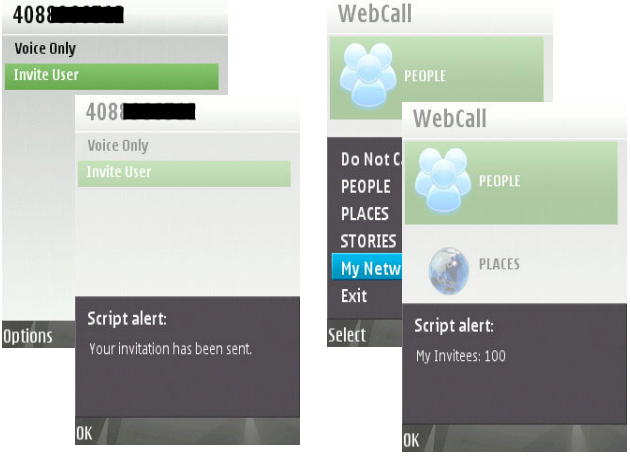

Fig. 4. Invitation module of WebCall

Fig. 4 shows several screenshots of the invitation module prototype in WebCall. On the left, a user is sending an invitation that involves notifying the WebCall server and sending an SMS message to the invitee. On the right, a user is checking statistics on his/her invitees, which involves checking the aggregated information with the WebCall server only.

3.1.2 Registration and Client Download/Installation

Before a new user can use WebCall, he/she needs to go through a simple three-step procedure: account creation, account activation, and client download/installation.

A new user can register with WebCall through a simple web interface from a browser either on a PC or on a mobile device. A registration can be triggered either by a user receiving an invitation in an SMS message, or by voluntarily visiting the registration web page. The only difference is that in the former case, the URL in the SMS message contains an InvitationID (as described in previous section) so that the system can track invitation statistics.

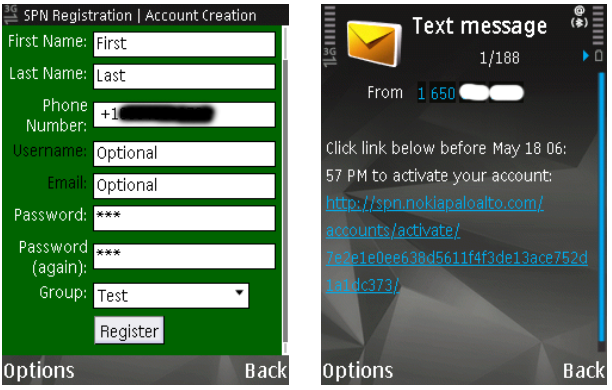

Fig. 5. User registration

The picture on the left-hand side of Fig. 5 shows a screenshot of the registration page as displayed in the Nokia N95 phone browser. To reduce the burden for a user to enter text from a mobile phone, all the fields can be dynamically pre-populated by the WebCall server in the case of an invitation-triggered registration. That is because the system can already retrieve information about the invitee from the inviter's contacts. Essentially the user only needs to fill in the password and group fields. Of course, the user may overwrite any field if he/she prefers to. A new user without invitation still needs to fill in the mandatory fields (in white). Note that the username field is optional. The system will generate a unique username from the phone number if the user does not provide one.

After the user fills in the registration form and clicks on the register button, the WebCall server will validate information received in the form and create a new user account. Then the server will send an account activation message (see Fig. 5, right) to the mobile device to verify both the phone number and the ownership. The final step of registration is for the user to click on the embedded URL in the activation SMS and launch the web browser. He/she can now proceed to download and install the WebCall client.

Note that although the procedure is optimized for phone browsers, the first step (i.e. account creation form) can be also done through any browser on a PC.

3.2 Location Based Services

3.2.1 Indoor Location Based Services

Location based services are rapidly growing due to available radio infrastructure as well as increased capabilities of mobile devices. These technologies allow various applications and services to be built from guide and tracking systems to location specific advertising. However, while outdoor location based services have been growing because of availability of GPS in portable devices, indoor location based services have not been keeping the same pace with its outdoor counterpart. The reason for this is that no single good solution exists for indoor positioning in which it can achieve high position accuracy and scalability in the sense of minimal cost, time and effort for deployment.

Other than the obvious radio signal triangulation to determine the indoor location of people, which suffers from signal fluctuation due to moving objects, academia are exploring the possibility of tracking the user with low-cost, inertial-sensor-based devices attached to shoes, such as accelerometers, gyroscopes and magnetometers. These devices are small and can monitor the distances of foot travels from real time acceleration data as well as the angle of rotation and/or the change of direction through the gyroscope and magnetometer information. There has been much progress in this field and results seem quite promising [7]. But this too is not a perfect system and requires frequent recalibration of the user's location due to limitation of the hardware performance.

The approach taken for the indoor location system was to find the quickest deployable technology to establish an experimental testbed [8] in our work environment that allows the demonstration of indoor location based services. Our indoor location system consisted of Bluetooth tags situated at all points of interest in the building. Each office, meeting room, laboratory and common area was tagged with a total of 64 Bluetooth tags. The Bluetooth tags acts as positioning beacons where the radiated powers from the tags are adjustable through software API control. Having the ability to control the radiated power allows for adjusting the range/granularity for Bluetooth discovery between the Bluetooth tags and mobile device. In this particular setup the mobile device does not use the Bluetooth received signal strength indicator information (RSSI), but rather senses by scanning to find whether it is within the range of a Bluetooth tag.

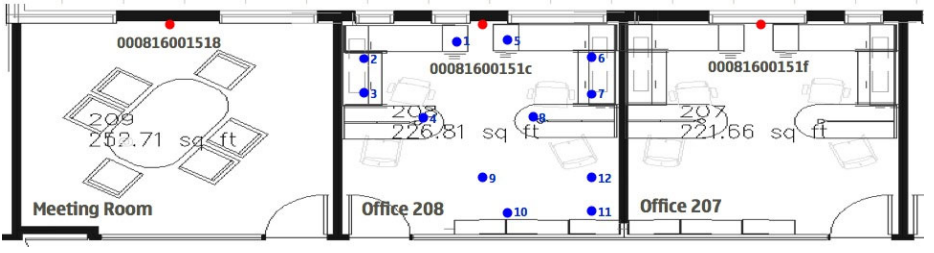

Fig. 6. Mobile device various measurement. Locations marked in blue for Bluetooth tag signal. Red dots represent locations of Bluetooth beacons in each office.

A number of measurements have been made to calibrate and verify the robustness of the indoor positioning system. Fig. 6 shows three Bluetooth tags [9] marked in red placed in each office and a mobile device positioned in 12 different locations within the center office marked in blue. Each of the Bluetooth tag's transmitted power adjustment was made in steps of 2dB to determine the level of Bluetooth signal penetration through the walls of the offices. In the case for Fig. 6, an optimal transmitted power from the Bluetooth tags was found to be -26dBm, equivalent to 2.5uW. Different room dimensions and wall material in buildings have a direct impact on the calibration of the transmitted power from the Bluetooth tags. Table 1 provides some approximations of the attenuation values [10] through common office construction at 2.4GHz (802.11b/g), the frequency range shared by Bluetooth and WiFi. A 3dB attenuation of radio signal is equivalent to a loss of half the power.

Table 1. Attenuation of radio signal at 2.4GHZ

Material	Attenuation
Plasterboard Wall	3 dB
Glass Wall with Metal Frame	6 dB
Cinder Block Wall	4 dB
Office Window	3 dB
Metal Door	6 dB

Shown in Fig. 7 is one set of measured results from the mobile device in location 5 of Office 208. Out of the 12 measured positions, the worst dataset has been selected as an illustration. The mobile device performed 5 Bluetooth scans searching for only the MAC addresses of the Bluetooth tags. These scans are then repeated 10 times, which are represented by legends as "try" in Fig. 7. For example for "try2", out of the 5 Bluetooth scans from the mobile device, there are 5 discovery of office 208, 2 discovery of the neighboring meeting room and no missed discovery. "No discovery" only occurs when the Bluetooth tag's radio signal are too weak for the mobile device to detect. As expected, variations are observed in each "try" due to movement of objects, including movement outside the room as well as possible interference from other radio signals in this frequency band. In addition, other scenarios have been taking into account such as the mobile device located in the user's pocket and when holding the mobile device with respect to the calibration of the Bluetooth tag's transmitted power.

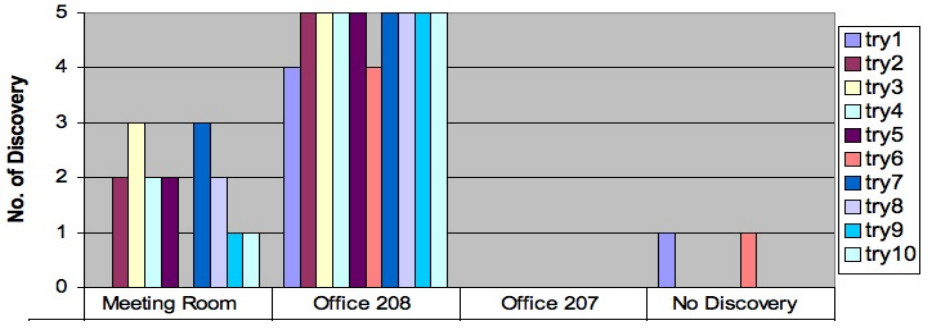

Fig. 7. One set of measurement results from mobile device at location 5 (Fig. 6, middle)

From these results, it does not seem reasonable to only make a single Bluetooth scan for the discovery of Bluetooth tags to determine the user's location, since there would be times when false or no location is detected. Instead, several Bluetooth scans are required to statistically compare the MAC addresses discovered and determine the correct location of the user.

To conserve the battery life on the mobile device, time interval for Bluetooth scanning and to report the user's location to the WebCall server can be adjusted by the user. In addition, the accelerometer sensor in the mobile device has been utilized to monitor whether the user is transitioning from one location to another. If this activity

is detected, Bluetooth scanning from the mobile device is automatically initiated, ensuring the user's location information is recent rather than waiting for the set interval time to elapse.

One main disadvantage of Bluetooth tagging is that the battery life on the tags only last from days to weeks. Another disadvantage is the Bluetooth discovery time, which is specified at minimum time of 10.24 seconds. To achieve low maintenance of the indoor location system, we need significantly lower power consumption as well as improvement in discovery time. It is anticipated that in the near future ultra low power (ULP) Bluetooth would be available. ULP Bluetooth in many cases makes it possible to operate low cost sensor-type devices, namely radio tagging for more than a year without recharging and also improves on discovery time. Hence, future standalone ULP Bluetooth tags will be able to operate without intervention for long periods of time.

Powered by the room-level accuracy in our indoor positioning mechanism, we developed three simple features in WebCall: map showing a user's or his/her team's location (Fig. 8), messages attached to locations, and meeting room occupancy.

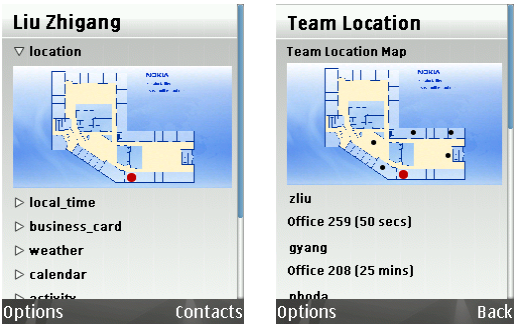

Fig. 8. Indoor location map. Red dot shows the target user's latest location while black dots (picture on the right) shows latest locations of his/her team.

3.2.2 Outdoor Location Based Services

We use a Cell-ID method as the basic technique to provide outdoor location services and applications. The method relies on the fact that mobile networks can identify the approximate position of a mobile device by knowing in which cell the device is operating at a given time. The main benefit of the technology is that it is already in use today and supported by most (if not all) mobile devices. The power consumption is much lower than GPS. However, the accuracy of the method is generally low (in the range of 200 meters), depending on the cell size. Generally speaking, the accuracy is higher in densely covered areas like urban places and lower in rural environments.

A Cell-ID usually includes four parameters: cell tower id (cellid), mobile country code (mcc), mobile network code (mnc), local area code (lac). When the WebCall server receives an outdoor location request from a WebCall client, it extracts these four parameters from the client's request. Then the WebCall server uses three APIs plus a local Cell-ID lookup table to obtain city-level location information and return it to the client. The ZoneTag API [11] allows registered developers to access the ZoneTag location services and get the best-known location directly (such as country,

state, city, zip code). The OpenCellID API [12] can return the geo-coordinates information (latitude and longitude) of a specific cell. Another reverse geo-coding API [13] can parse the latitue/longitude and return a list of postal codes and places.

In addition to querying above servers directly, the WebCall server also creates a local lookup table to cache the mappings between Cell-IDs and location information. A search in the local lookup table is always performed first by the WebCall server when processing a request. This can significantly improve the average response time. If a Cell ID is neither in the local lookup table nor in the three APIs' databases, a failure response is sent back to the WebCall client. The whole procedure is described as follows.

Procedure of query logic in outdoor location

```
Get a request of outdoor location with cellid parameters;
Outdoor_loc = cellid_local_lookup(cellid, mnc, mcc, lac);
If ourdoor_loc == None:
    OpenCellID_xml = OpenCellID_API(cellid, mnc, mcc, lac);
    Lat = OpenCellID_xml.lat;
    Long = OpenCellID_xml.long;
    If lat == '0.0' or long == '0.0':
        ZoneTag_xml = ZoneTag_API(cellid, mnc, mcc, lac);
        If ZoneTag_xml == []:
            Return('Fail');
        Else:
            Outdoor_loc =xml.dom.parse( ZoneTag_xml);
            Update_cellid_lookup_table(outdoor_loc);
            Return(outdoor_loc);
    Else:
        Geonames_json = Geonames_API(lat, long);
        If Geonames_json == {}:
            Return('Fail');
        Else:
            Outdoor_loc = json.read(Geonames_json);
            Update_cellid_lookup_table(outdoor_loc);
            Return(outdoor_loc);
Else:
    Return(outdoor_loc);
```

3.2.3 Local Weather and Time Services

Once the WebCall server obtains outdoor location information of a WebCall client, it can deliver a couple of additional services such as weather and local time. The outdoor location information includes city, state and country as well as zip code. An XML search API from weather.com [14] can be used to get a unique weather location ID. If the country is US, the WebCall server passes city and state as parameters to call the API; otherwise it passes city and county instead. Based on this weather location ID, the Yahoo Weather API [15] enables the WebCall server to retrieve up-to-date weather information for the corresponding location, as shown in Fig. 9.

There is also local time information contained in the Yahoo API XML response, which includes date, time and timezone as well. This allows a caller to know a callee's local time, a small but quite useful feature in practice (for example to avoid making a phone call during midnight at the callee's local time).

Fig. 9. Local weather information

3.3 Human Powered Questions and Answers

Another feature we added to WebCall is to enable users to ask questions from their mobile devices. Although people can get answers to many questions using services such as Google or Wikipedia, there still remains a large amount of information that is too transient or specific to be captured in the general web. The screenshot on the left-hand side of Fig. 10 shows a few examples of such questions. It is obvious that some of the questions are relevant only during certain time frames and to specific communities. The right-hand side screenshot in Fig. 10 shows a simple interface allowing a user to reply to a question and view the answers.

Besides the transience and specificity, a more important reason we offer the questions/answers (Q/A) feature in WebCall – in stead of using "traditional" web chat rooms or email – is to use contextual information collected by WebCall for optimal question routing. (Note: this is also how Q/A in WebCall is different than simple blogging applications such as Twitter). Such contextual information includes users' current and past locations, call logs (from which the system can infer social distances), calendar events, and even the Q/A exchange log itself. For example, a question about availability of a coffee machine in the kitchen can be routed only to people who are in the kitchen. Or, a simple question about whether you have seen John today can be routed only to people who is close to John, in terms of either social distance (from call logs) or physical distance (sitting in neighbor offices). Or, the same question can be routed to people who may have had meetings with John according to the calendar. In terms of the Q/A log, the WebCall server can analyze who replies to which questions most and use that information to rank users and construct target groups for question dispatching.

Another benefit of this Q/A mechanism is anonymity. Although the system tracks people who raise and answer questions, the identity is not shared among users unless a user opts to do so. This is to encourage answers from strangers.

We have implemented an experimental set of question routing rules including time, location, and group. Question routing is a critical factor for a successful Q/A service on mobile devices, given the constrained UI and limited user attention span on-the-go. It deserves more study in the future.

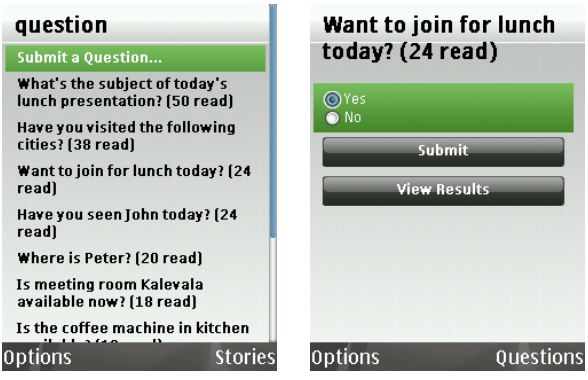

Fig. 10. Human powered question and answers

3.4 Other Features

WebCall also supports other features such as a user picture, business cards, and calendar event (Fig. 11, left). Another feature we want to emphasize is the capability to support third party "plug-in" services, such as news alert (Fig. 11, right) and advertisement. As many have discovered, content relevance is key to mobile services. One way to achieve that is to exploit user contextual information. This is the rationale why we combine third-party service with other WebCall features. We have implemented a simple framework in which a third party can submit content in the form of HTML and associated metadata (e.g. tags, target user criteria) to the WebCall server. The content will then be pushed to a user's mobile device after server validation and filtering. On the client side, hooks are already in place such that a new content item can be added to the menu and presented to users upon user selection.

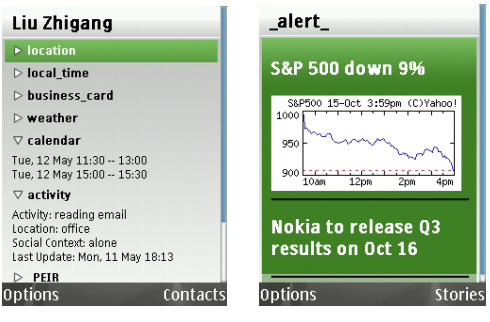

Fig. 11. Screenshots for calendar, activity, and news alert

WebCall can also act as the aggregator of user contextual information from sources other than mobile device. For example, Fig. 11 (left) shows a user's activities on a computer, as detected by context-sensing software named Pennyworth [16].

3.5 Privacy Control

Many of the contextual information described in previous sections are very sensitive in terms of privacy. The WebCall framework provides a few layers of privacy protection. First of all, only a user authenticated by the system can access other users' information. Second, we provide a simple privacy control interface (Fig. 12) that allows a user to specify what information can be shared with whom. User input will be translated into an entry to an access control table and enforced by the WebCall server when processing each request from a client. Lastly, every transaction is logged by the WebCall server so that any accident – if happened – can be analyzed afterwards.

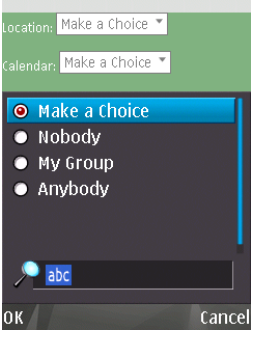

Fig. 12. A simple interface for privacy control

4 Lessons Learned

While trivial as it seems to use Bluetooth tags as location beacons, there are issues where Bluetooth signal is not completely attenuated by walls depending on the walls material. Hence, neighboring Bluetooth tag signal can easily penetrate the wall and give a false reading of a user's location. In addition, orientation and location of the mobile device can adversely affect the discovery of a Bluetooth tag that already operates at low output power.

As in any prototype of research concepts, we found that it is beneficial to have a clean and extensible system architecture. This is particularly crucial in our case as new ideas and features kept emerging even as we were prototyping the system. For example, one key component in WebCall is the database. The schema we chose at the beginning allows us to add new tables without loss of database integrity or having to modify existing tables. We found that the Django framework is quite capable and flexible, although documentation could be better and occasionally version upgrades break backward compatibility.

One thing we should have done but could not due to resource limitation is to involve real users for feature design. Although we selected WebCall features through brainstorming and common sense, it would be better to collect input from real users even before we implement a particular feature. Different perspectives, particularly from non-technical users, can provide balance to our probably biased view.

In terms of implementation, we found that the development environment on mobile devices still falls behind that on PC. While providing a good UI toolkit built on top of Javascript, WRT exhibits performance limitations as the WebCall client grows with more complexity. In addition, current version of WRT does not support dynamic loading of embedded Javascript. We had to work around the problem by creating a custom Javascript loader function that creates a DOM node on the fly. Hopefully the next release of WRT will address those issues and a few bugs we found during our implementation.

It requires extra effort to debug mobile applications. Even though an emulator on PC is very helpful, it does not cover all the test cases such as accessing local resources on device or setup network connections. The WebCall server logs every client request and server response in a table. It not only provides history of a user's transactions, but also serves as a good debugging utility for tracing down bugs. We also found that carrying query parameters in HTTP GET message header, rather than in the body of a POST message, is much more convenient for unit testing. One can simply tests the server with a web browser. Of course, it is a balanced judgment between API consistency and convenience.

5 Conclusions and Future Work

We developed WebCall as a generic research framework that extends traditional voice call with rich contextual information about users. The people component centers on the contact list on mobile devices, arguably the largest social networks in the world. The places channel provides location and location related information. Besides outdoor locations, we developed reliable indoor positioning system based on variable-powered Bluetooth beacons that can achieve accuracy to room level. The stories channel demonstrates a few promising features utilizing contextual information about people and places.

WebCall includes simple and intuitive invitation and registration mechanism to help grow the user base. We pay special attention to privacy and give users the full control on how to share their contextual information. Last but not the least, WebCall provides a simple API that allows third parties to integrate their services.

Still, there remain many interesting topics we did not have time to explore in depth and worth further study:

- First of all, as a research framework, we did not test the WebCall system with real users other than team members. It would be good to conduct user studies to validate the concepts and have a better understanding on whether and how user contextual information can benefit regular users without pushing them out of their comfort zone in terms of privacy.
- Similarly, we have not tried to build a real service out of this research framework. Although we believe the framework is quite robust and flexible, it remains to be tested on how it can be adopted by the industry and academic communities. In particular, we did not study in-depth the scalability of the system.
- Some components of the WebCall system are still "shallow". For instance, we only scratched the surface on the issue of Q/A routing based on users' historic

data. The topic is actually quite big and requires much more efforts. In addition, more study is needed on the third-party plug-in API.

- There are a few topics we left out from our system due to resource constraints and lack of real user data. One of them is user behavior predication based historical data. It would be a natural extension to the WebCall research framework by applying the latest machine learning techniques to the contextual information that could be collected by the system.

Acknowledgments

We would like to thank August Joki, who implemented most part of the widget UI on device and contributed to many of the ideas described in this paper. We also wish to thank Chris Karr for his assistance setting up Pennyworth for our system. The anonymous reviewers of this paper provided very good and constructive feedback, which we appreciate very much.

References

1. Yang, G., Liu, Z., Seada, K., Pang, H.-Y., Joki, A., Yang, J., Rosner, D., Anand, M., Boda, P.: Social Proximity Networks on Cruise Ships. In: MIRW 2008, Mobile HCI Workshop, Amsterdam, The Netherland (2008)
2. Web Runtime (WRT), http://www.forum.nokia.com/ResourcesInformation/Explore/WebTechnologies/WebRuntime
3. Python for S60 (PyS60), http://wiki.opensource.nokia.com/projects/PyS60
4. JavaScript Object Notation (JSON), http://www.json.org
5. Django, http://www.djangoproject.com
6. MySQL, http://www.mysql.com
7. Jeda, L., Borenstein, J.: Non-GPS Navigation with the Personal Dead-Reckoning System. In: SPIE Defense and Security Conference, Unmanned System Technology IX, Orlando, Florida (2007)
8. Cheung, K.C., Intille, S.S., Larson, K.: An Inexpensive Bluetooth-Based Indoor Positioning Hack. Massachusetts Institute of Technology (2006)
9. BodyTag BT-002. Bluelon, http://www.bluelon.com
10. Beating Signal Loss in WLANs, http://www.wi-fiplanet.com/tutorials/article.php/1431101
11. ZoneTag, http://developer.yahoo.com/yrb/zonetag/locatecell.html
12. OpenCellID, http://www.opencellid.org
13. GeoNames WebServices, http://www.geonames.org/export/ws-overview.html
14. weather.com API, http://xoap.weather.com/search/search?where
15. Yahoo weather API, http://developer.yahoo.com/weather
16. Pennyworth, http://pennyworth.aetherial.net