Sbuddify – A real time location tracking application

Tianlong Li, Songyan Hou, Enrui Liao, {tl2493, sh3348, el2756}@columbia.edu

***Abstract*—Location-based service is one of the most popular topics of today's software business, which is being utilized in almost every possible way it could be exploited. Meanwhile, media and method of communication, urged by growing reachability of telecommunication, are also evolving frequently. We decided to focus on the very simple form of these two kinds of need with one application that mixes LBS and communication together..**

This paper presents the design and implementation of Sbuddify, an iOS application to make use of LBS and implement a responsive and fully expandable feature of sharing real-time location with other users in a secured way. Sbuddify is designed to serve the very basic need yet in a very effective and robust way. Related tests have shown the responsiveness and expandability of this application are discussed as well.

***Index Terms*—Location Based Service, LBS, iOS, HTTP, Flask**

I. General Information

A. Location Based Service

Location based service application has gained tremendous momentum in recent years. A location-based service (LBS) is usually considered to be an application for mobile devices that requires knowledge about geographical information of the device. At one hand, LBS is used in the context of streaming location based information to the user, such as recommending nearby restaurants or nearby users who may share similar interest towards some activity. LBS is effectively merging ever. On the other hand, LBS is very useful in the context of dialing emergency service such that the FCC requires that all carriers meet certain criteria for supporting location-based services for emergency[1].

B. Location Based App

LBS customer received customized service based on their current location. Compared to the traditional networking service model that notifies indiscriminate service to all the clients, a LBS customer does not need to waste a lot of time to search relevant service around manually. Instead, the LBS serve-as-you-move model enables customers to focus what he or she actually cares.

A growing number of companies are riding this wave to provide public location based services, such as Tinder, Yelp, Foursquare, Scvngr, and Uber. These LBS providers offer a variety of options in transportation, food, and social networking. For instance, some of the applications will learn what you like and lead you to similar place in the neighborhood. Another example is Starbuck's app, simply showing customers the nearest Starbuck to enjoy some afternoon coffee and leisure.

C. Sbuddify

The convenience of LBS service leads us to an interesting idea: How well can we have fun with a LBS app when most of the apps are offering recommendation?

Our team was trying to answer this question with our iOS app, Sbuddify. Sbuddify allows you to know the location of your "buddy" at any time. Now you can use the GPS technology on your iphone to give you the peace-of-mind that you know where everybody is located. Wouldn't it be nice to see where your kids have been throughout the day and know that your spouse has safely arrived during his or her travel? Sbuddify will take care.

Sbuddify provides users with the following features:

1) Self tracking of newest location of the user.
2) Insert photo and descriptions along the path (long press the map to add check point).
3) Locate another iPhone and share path information based on secure ID.

The remaining part is organized as follows. Section 2 introduces some related works. Section 3 describes the detailed architecture of our application and technical details. Section 4 shows the test results of Sbuddify and Section 5 describes the desired future work of this application. Section 6 presents our conclusion.

II. Related Work

Currently, there are many tracking applications built on a mobile platform such as "Family Tracker", "Find my friends", "Trick or Tracker" etc [2]. Some of these apps, like "Find My Kids", are aimed for security where the real time locations of vulnerable people are of vital importance. And most of tracking apps focus on offering a brand new way for communication between users with intimate relationships.

Technically, the ideas under the hood are quite similar for these apps: utilizing the GPS embedded in mobile devices to gather the real time to get location information of the tracked user and providing the processed data to others. Both mainstream mobile platforms - iOS and Android, have fully supported location-based services, where the location data can be readily accessed via built-in frameworks or APIs. Native map components are also well established so that the location info can be displayed intuitively.

III. Implementation

Our system is mainly composed of two parts: the iOS client and the backend server deployed on cloud (client-server computing model). We use MongoDB, which has better performance on storing geo-location data than traditional relational database to store the location data. We apply HTTP protocol

to build the connection between both sides. Figure 1 shows the skeleton of our architecture, which will be explained in detail later.

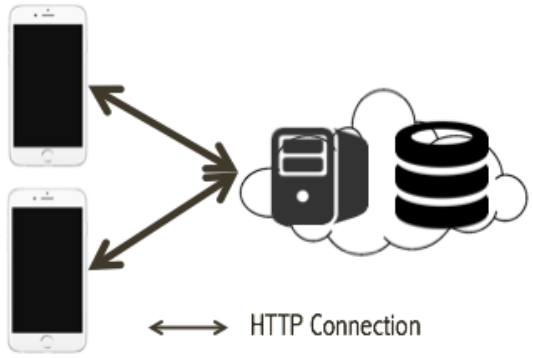

Fig. 1: Skeleton of system architecture

A. iOS Client

iOS client side is based on UITabBar model, where two tabs of tracking self and tracking others are presented. Each tab bar serves as containers for several other viewcontrollers so that we are allowed to add extended features which require further view transitions and controller logics. Both tab bars are containing a UINavigationController sub-class that controls transition between DisplayMapViewController and tab-specific sub-viewcontrollers, with added functions to deal with respective logics in each tab. The top menu where user's unique ID is shown is updated upon synchronous http response with server-generated ID, and shared between tabs.

DisplayMapViewController draws a path based on the newest location information received from delegate of built-in location manager in self-tracking mode, or requesting server in other-tracking mode, utilizing the Breadcrumb library provided by Apple and following the corresponding developer guide [3]. The background worker thread will constantly calculate a path region that covers all location points captured so far, and expand current rendered area to indicate new region just passed. The threshold for expanding rendered area can be modified for more responsive and faster rendering, at the cost of performance.

In the "self" tab, RouteViewController controls the transition between DisplayMapViewController and AnnotationSettingViewController which handles the setting of new annotation such as title, comment and photo to be added inside the annotation on the map. RouteViewController is also relating the tap status of track button to starting or ending tracking and uploading location data to server.

In the "other" tab, OtherViewController communicates with DisplayMapViewController in terms of the user actions such as entering and verifying target IDs or refreshing to enter new target ID to track. Before entering DisplayMapViewController, OtherViewController will ensure that target ID is valid and sending location to server, so that DisplayMapViewController can start drawing something immediately. Refreshing button will cause DisplayMapViewController to yield its control of the screen back to OtherVIewController so that the new verification process can initiate for new target ID.

B. Data Interface

In order to connect the frontend client and backend server, we create a data access object (DAO) for data sending and retrieval, acting as the data manager for the whole application. To make our implementation simple and clear, we apply singleton design pattern that is mainly used for centralized management of internal or external resources [4]. Singleton involves only one class, which is responsible to instantiate itself to make sure it creates no more than one instance; the same time it provides a global point of access to that instance. In this case, the DAO instance can be used from everywhere inside the client logic.

As for network connection with server, we use basic HTTP protocol, supported by Foundation framework from iOS *(NSURLConnection, NSMutableURLRequest, NSURLResponse, etc.)*[5]. There are three main interface methods used in our application:

1) *(NSString *) getID*, which will send an HTTP GET request, asking for the user id generated by server. This id will act as a passcode for future authentication.
2) *(NSArray *) sendKeywords: (NSString *) keywords*, which will send the passcode user typed in to server for authentication. Only matched id can be recognized as successful. This method will return the real time location information (in form of [latitude, longitude]) to caller only if the passcode is correct. Otherwise, server will send invalid geo-information back, discarded by client side. To distinguish between different kinds of GET request, we add a new field in message header named "Message-Type".
3) *(void) sendLocationwithLat: (float) lat andLon: (float) lon*, which will send a HTTP POST request, sending the geo-location information in form of [latitude, longitude] to server. In order to provide better user experience, we apply asynchronous version of POST method with a delegate listening to the response from the server, which will not block the caller's thread before receiving the response.

We use JSON as our uniformed data transmission format, which is simple and lightweight, fitting better for data exchange format than XML[6]. iOS supports JSON serialization method well with *NSJSONSerialization* class converting between native Cocoa class and JSON objects.

C. Server Framework

The two requirements of the frontend are low latency and high security. Here we chose python flask as the server framework, because it is a "light"framework compared to Django or more full-fledged web frameworks. It offers a good interface to database and requests.

As showed in figure 2, server consists of a Python program deployed on EC2 instance to be available to public. For the database we chose MongoDB for its fast speed with flask.

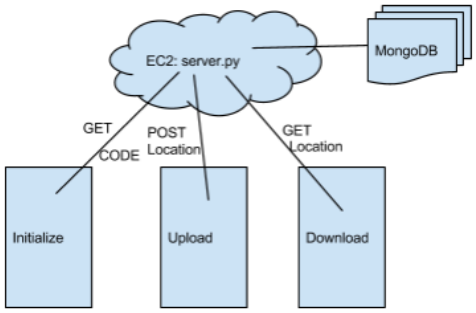

Fig. 2: Backend Logic

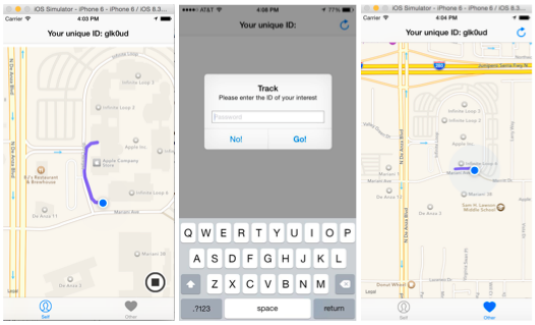

Fig. 3: Core functions of Sbuddify

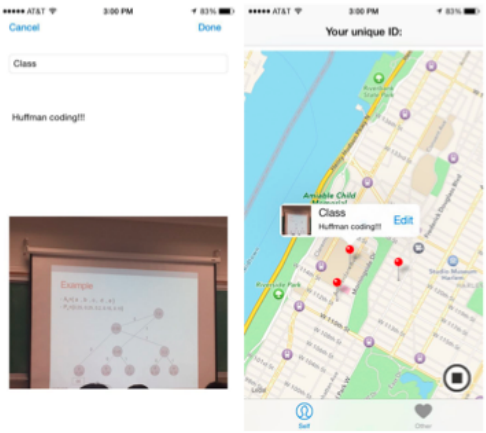

Fig. 4: Add key points in the map

AWS offers a fast deployment tool Elastic Beanstalk to install all the dependency for server.

The first problem is that AWS Elastic Beanstalk is not compatible with MongoDB. One needs to remote log into the EC2 instance and configure the VM manually. After setup MongoDB on the instance, install gcc and upgrade pip version to the latest. Then install flask, Pymongo and Cryptography libraries. At last runs the program with root permission.

Secondly, server cannot reach out to the client actively, though http request is safe and easy to use. Without compromising the performance, we make the client request for up-to-date location information every second. Server is passively listening on port 5000 to generate code, maintain the relationship between users identified by code and switch location data between clients.

When the app initializes, it sends server a request of initialization. Server read Http request with flask request handler and provide an identity code generated by Fernet encryption. To facilitate human memory, the code has eliminated punctuations. Server keeps a relation dictionary for fast access. Approximately, when the concurrent user is less than 10000, this works much better than putting it into database. Data from client are all location tuples in http request. Server backup location data into database for track recovery. At the same time, the latest location data is cached into relation dictionary. When a client pass the authentication code to server, server search the entry in relation dictionary in O(1) time and return the location data.

IV. Test Result

We have tested our application both on Xcode simulator and real iOS device (iPhone 6). The screenshots of the app are as follows:

Figure 3 displays the core functions of Sbuddify. On ”self tab”, user’s own route will be drawn on the map and on ”Other” tab, you can check people’s real time location after typing right user id in the prompt field.

Figure 4 describes the process of adding a new keypoint in the user’s own path. After long pressing some specific point on the map, a new interface will come out and ask user to edit the content at this point. User can type in title, content, and add photos from the gallery. A new pin point will appear on the map after editing.

To test the responsiveness of our application, we recorded the average time differences between showing same location slot on user’s ”self” tab and tracker’s ”Other” tab. The network connection environment has a significant influence on the performance of our app. Therefore for each test, we also examined the network delay using tools like ping or tracert at that specific time and location. The results are displayed in table. 1

Test Number	Average Time differences (ms)	Average Network Delay (ms)	Response time in Sbuddify (ms)
1	36.9	34.8	2.1
2	45.9	40.5	5.4
3	31.4	30.4	1.0
4	36.8	33.7	3.1
5	34.6	29.3	5.3
6	38.2	32.3	5.9
7	43.8	41.2	2.6
Avg	38.2	35.2	3.0

Table. 1: Test result of application delays

According to the result shown above, the in-application delay is always less than 10 ms, which is negligible compared to the practical using scenario, proving that the responsiveness of our application is satisfying.

V. Future work

Due to the limit of time, our current prototype is still very simple. However based on that, we are considering building more features as follows:

1) Currently the user can create check points along the way, decorating their routes with some keywords and photos. It would be more interesting if the users being tracked can share the routes as well as their checkpoints. In that case, the interaction between users will become more intimate.
2) Right now our app allows the user to track only one user at a time. To make it more popular, we will try to let user track multiple people at the same time.
3) Since the main focus of our application is to build a new way for users with intimate relationship to see each other. It would be better if the matched users can have more channels to communicate in our app. Besides sharing the key points along the path, we will also try to import commenting and chatting in the future.

VI. Conclusion

We have designed, implemented and operated "Sbuddify", an iOS application integrating location based service and providing users with a brand new way to share the real time location in a simple, secure and elegant method. Every user is identified by an encrypted passcode. Once two users are matched, their real time location info will be available to each other and drawn on the map. In addition, the users can also mark some interesting points along the way with text and photos decorated.

In our implementation, we try to gain a balance among the security, user experiences and performance. The first concern we took was the security, to make sure that one cannot randomly guess other's ID to access his location. This was achieved by Fernet encryption. There are already a lot of encryption algorithm such as AES and RSA. But a code of thousands character is infeasible to visualize and memorize. We implement Fernet encryption because of trade-off between user experience and security.

The second thing we noticed during development was performance issue when we tried to render every newly updated location in a very frequent manner. We resolved this by adding a threshold for rendering so that location changes need to accumulate to some considerable level to trigger the redrawing. We took a similar approach to reduce the server load of exchanging location updates between many clients. Only location change that is far away enough from the last location will be transmitted to server and published to subscribed user, instead of every location.

Appendix A
Source Code

https://github.com/Alomin/Tracker

Appendix B
Contribution

Songyan Hou: Project Manager. Write data interface between iOS client and backend server.

Tianlong Li: Implemented iOS client running on iPhone6

Enrui Liao: Programmed Server and database on EC2

The other works are shared on average.

References

[1] FCC ADOPTS RULES TO IMPLEMENT ENHANCED 911 FOR WIRELESS SERVICES
https://transition.fcc.gov/Bureaus/Wireless/News_Releases/1996/nrwl6026.txt
[2] *5 apps for spying on your spouse*, New York, USA: MarketWatch, 2014.
http://www.marketwatch.com/story/5-apps-for-spying-on-your-spouse-2014-03-10
[3] *Breadcrumb*, Florida, USA: Apple Developer, 2013.
https://developer.apple.com/library/ios/samplecode/Breadcrumb/Introduction/Intro.html
[4] *Singleton Pattern*, Cambridge, England: OODesign.com, 1999
http://www.oodesign.com/singleton-pattern.html
[5] *Making HTTP and HTTPs requests*, Florida, USA. Apple Developer, 2014.
https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/WorkingWithHTTPAndHTTPSRequests/WorkingWithHTTPAndHTTPSRequests.html
[6] *JSON: The Fat-Free Alternative to XML*, USA: ECMA International, 2013.
http://www.json.org/xml.html