IT 09 055
Examensarbete 30 hp
November 2009

A Mobile Distributed System for Personal Security

Yang Guo

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

A Mobile Distributed System for Personal Security

Yang Guo

The goal of this system is to provide a location based alarm system through which customers can broadcast their last known position in emergency situations. Imaging that you are hiking in the forest alone and don't know how to get back to your car, or you get lost driving to your friend's home. At times like these, how can you ask for help? With push of single button or pull off the hands free cord on this system, it will notify your friends with your latest locations immediately. The system is divided into two parts, the client side and the server side. The client side works on the iPhone and other mobile phone platforms and is implemented in Objective C and Java ME. The server side supports more than one server while running on the Linux platform and is implemented using Java EE.

There are two problems that need to be solved before we start. First, how can we get the user's location if their mobile phone does not have GPS device or they are in an area without GPS signals. As we all know, the mobile phone signal comes from the cell tower and every cell tower has a fixed cover area, The user's approximate location can be calculated by the cell tower's location if the tower's location which the mobile phone gets signal from is known. Second, since we have more than one server running, how do we synchronize the data on different servers? If we treat this problem as another kind of server/client communication, it can easily be solved.

In this thesis report, this system will be described based on the software development process, and how problems encountered during the system development are solved.

Handledare: Henric Ehrenblad
Ämnesgranskare: Arnold Pears
Examinator: Anders Jansson
IT 09 055
Tryckt av: Reprocentralen ITC

Table of Contents

1. Introduction

This thesis report describes the problems we have, how they are solved, how the system is designed and implemented. The second chapter is a discussion about what problems needed to be solved before start implementation. The third chapter explains the project requirements and analysis before the design phase. The fourth chapter is a description of the involved technologies. The fifth chapter describes the design of system architecture, database and communication protocol. The sixth chapter describes implementation detail of some important parts. The seventh chapter shows the test result of our solutions and the difference between our solution and existing solution. Finally, the last chapter discusses future improvements to system, location package and other concluding thoughts.

1.1 Background

Widespace AB hires two students to work on this project. The company intends to develop a new system for business which provides a safety method for their customers. The idea is, whenever a user feels that they are in trouble, they just need to pull out the hand free cord attached to the mobile or push a single button. By doing so, the system will immediately notify the predefined contacts of the user by sending their current location plotted on a map and some predefined messages.

The system is distributed[1] and based on client/server architecture. The client side works on the iPhone and other mobile phone platforms; it is implemented by Objective C and Java ME. The server side is a cluster running on Linux platform and implement by Java EE technology. The communication for client/server and server/server uses HTTP (Hypertext Transfer Protocol).

Before starting the system analysis, two problems which occurred in the system needed to be solved. The first problem is how will the system get the user's location if the mobile phone does not have GPS device or signal. The second problem is how to synchronize data among the servers that are working on a cluster.

The final goal of this thesis project is to develop a new distributed system to provide a safety method for the customer while the following requirements need to be achieved;

- The server side running on Linux platform provides required services to client side, using Java EE technology.
- The client side running on iPhone platform with the highest priority, using Objective C and iPhone SDK.
- The client side running on other mobile phones with lowest priority, use Java ME.

[1] Distributed System: A distributed system is one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.

1.2 Tools

- Glassfish V2.1 (container of Java EE)
- IDE
    - Netbeans 6.5
    - Xcode 3.1
- MySQL 5.1
- iPhone SDK 3.0
- Java Development Kit 6 (JDK6)
- Sony Ericsson SDK 2.5.0.3 for Java ME platform
- Matlab 2007a
- Hibernate 3.0

1.3 Planning

The Scrum[2] software development methodology is applied in our project, which divides the whole project into several sprints, each sprint is typically a 2–3 weeks period (depending on the task) and implements some use cases, Documentation is also included in the sprint.

- 2- 3 weeks of background study
    - familiarity with iPhone and MacOS
    - familiarity with Objective C
- 2-3 weeks of test programming
    - develop simple applications for iPhone & test functionality
    - draft database structure/schema
- 10-12 weeks of design and implementation
    - design protocol between client/server
    - design user interface
    - client and server implementation
    - redesign and implementation
- 3-4 weeks of testing, debugging and report writing

[2] Scrum: An agile software development framework

2. Problem Discussion

This chapter will describe and analyze the problems mentioned above, and present their solutions.

2.1 Location Problem

2.1.1 Mobile positioning

Mobile positioning determines the user's actual location information by mobile terminals and mobile networks, such as text messages or mobile signals to a complete a series of location information services. The mobile positioning method can be broadly classified into three types:

Positioning based on triangulation:

This approach is most widely used in mobile positioning technology. It uses the geometric calculation of the triangular relationship to measure the target object's location. This measurement of positioning technology needs to know three reference point locations and the distances from those reference point locations to the target object. Then using triangular knowledge, it can detect the target object's position.

Positioning based on environment analysis:

This approach uses the characteristics of environment surrounding the target object to measure its location. It detects the target object's location by comparing the characteristics of the surrounding environment to existing characteristics which have been stored in a database.

Positioning based on relationship:

This approach uses the relationship of the target object and one or more existing objects to measure the target object's location. It needs an identification system to identify each unique existing object such as the id of each cell tower.

2.1.2 Problem analysis

To solve the problem of locating the user without GPS signal, we considered using an existing cell tower or Wi-Fi[3] access point information. As we know, the mobile phone sends/receives signal from cell towers that are close to user, in other words, the user has a relationship with cell towers, and the mobile signal covers almost every corner in the city, if we knew the cell tower pinpoint location and service area, the approximate location of the user can be measured wherever they are. However most modern mobile phone has built in Wi-Fi technology, a wireless technology based on

[3] Wi-Fi: is a trademark of the Wi-Fi Alliance for certified products based on the IEEE 802.11 standards, it uses both single carrier direct-sequence spread spectrum radio technology (part of the larger family of spread spectrum systems) and multi-carrier OFDM (Orthogonal Frequency Division Multiplexing) radio technology. [4]

IEEE 802.11 standards. When the Wi-Fi device wants to access the wireless network, it will scan the Wi-Fi signal first. In our scenario, if the pinpoint location of each access point is known, your approximate location can be measured too, but the access point may be changed frequently. Therefore looking for a stable access point becomes the most important aspect.

After researching, we realized that there are existing solutions of pinpointing your locations without a GPS device by calculating Wi-Fi or cell tower location information, current services such as “MyLocation” from Google which works on the mobile phones platform, and “Loki” from Skyhook which is used on the iPhone platform. Our goal is not to create a new product to compete with these existing one, but to apply existing product into our project, and to overcome their limitations that we found from our study. For example, the solution from Skyhook is primarily used on the iPhone platform, but the service area of Nordic countries is not reliable enough. From the following map we can see the service coverage (blue area) of Nordic countries compared to other European countries. The Wi-Fi and cell tower data in the Nordic countries only include major cities, such as Stockholm, Uppsala, Oslo, and Copenhagen.

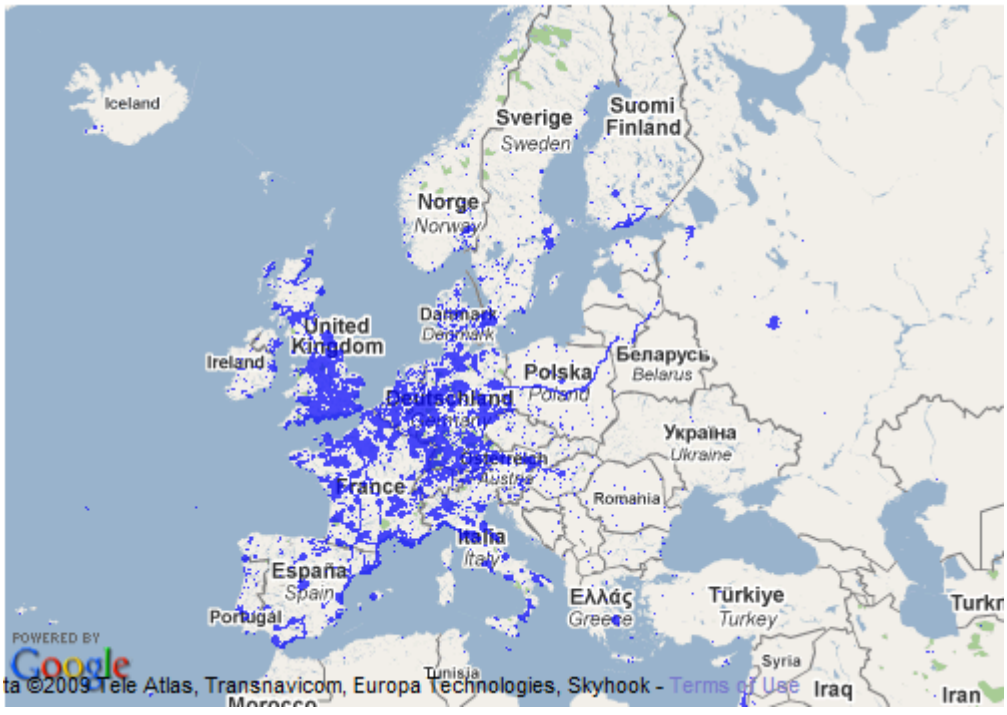


Figure 2.1

Solution from Google is also limited. The current version is the beta version. The coverage area is unavailable, but Google states that the cell tower technique will locate the user within about 1000 meters approximately; the positioning approach doesn't use triangulation, instead, it shows the range of the tower that the user's phone is connecting to [5].

2.1.3 Method for solving the problem

We considered creating a location package that combines the existing solution from Skyhook and our own positioning approach to compensate for the limitations of the existing solution. We focused on iPhone platform first, since it has the highest priority.

Now, the problem is how to get the cell tower and stable access point location information of Nordic countries. It would be a huge task to record all the location information into a database by ourselves for this project. Fortunately, our company offers an existing programming interface for getting cell tower and access point information of Nordic country; it can provide more cell towers and access points information than the Skyhook solution. The following figure shows our entire problem domain.

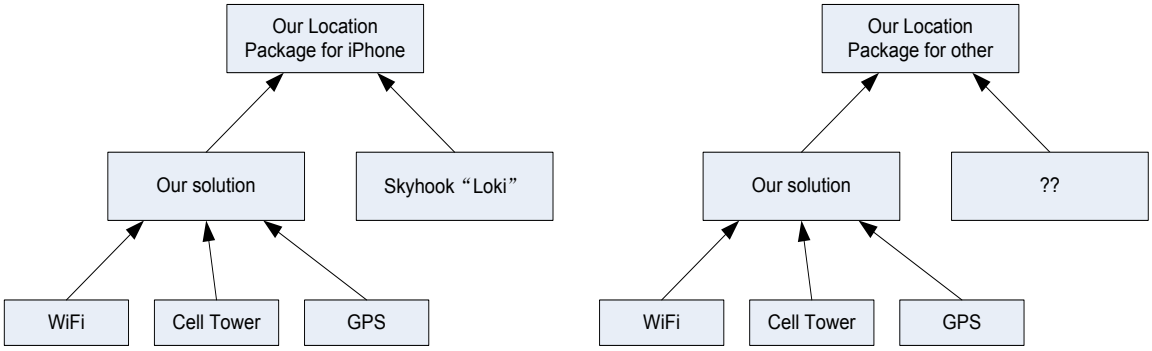


Figure 2.2

The mobile phone may receive cell signal or Wi-Fi signal from different cell towers or access points. Therefore we would apply the triangulation theory [6] in our solution to calculate approximate location. However, it cannot work so precisely as GPS. The figure2.3 shows triangulation theory in cell phone.

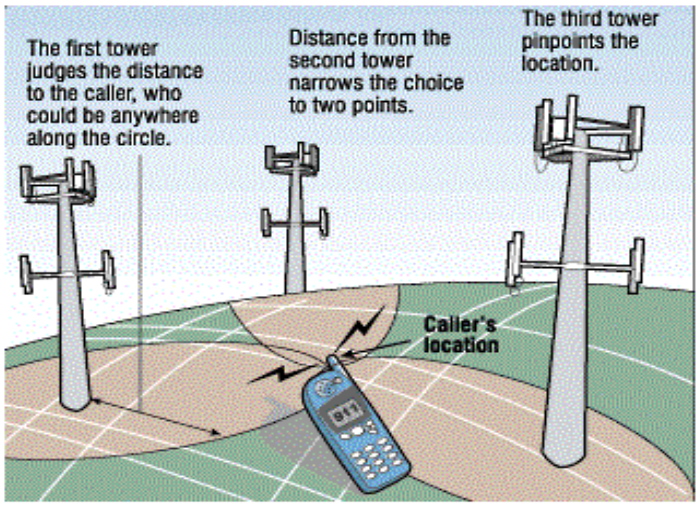


Figure 2.3 [7]

From a geometric or mathematical viewpoint, if the center location of three circle's and a point located at overlapping area of these three circles is known, then the approximate coordinates of that point can be calculated.

2.1.4 Solution analysis

We treated our location problem as a geometric and mathematical problem: the longitude and latitude of the cell tower can be considered as the coordinates of the center of the circle and the service area of cell tower can be considered as a circle with the service distance being the radius of circle. With these measurements, a geometric diagram can be drawn (figure 2.4). Here for instance, the cell phone could be picked up by three towers.

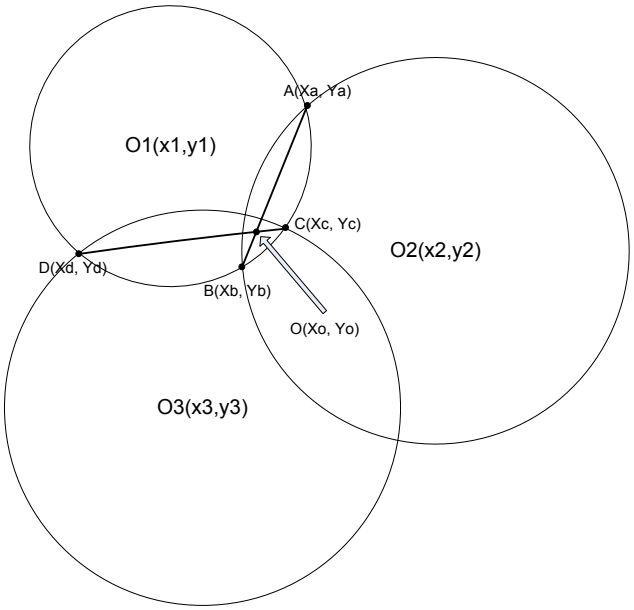


Figure 2.4

Mathematical calculation

Our goal is to calculate the coordinates of the approximate location O $(x_o y_o)$. Let $O_1(x_1, y_1)$, $O_2(x_2, y_2), O_3(x_3, y_3)$ be the coordinates of the three towers or access points, and set r_1, r_2, r_3 as a radius of each circle.
$A(x_a, y_a), B(x_a, y_b)$, $C(x_c, y_c), D(x_d, y_d)$ are points of intersection between $O_1 \wedge O_2, O_1 \wedge O_3$.

Using the second order equation group with two unknowns, we will get coordinate values of

$$A(x_a, y_a), B(x_a, y_b) \quad C(x_c, y_c), D(x_d, y_d)$$

Now we know coordinate of A, B, C, D, so there are two straight lines between A, B and C, D. The point of intersection between these two lines is what we need calculate. Therefore using the slope intercept equation[4], the slope K_1 K_2 and intercept B_1 B_2 can be calculated for lines between A, B and C, D by the equation group

$$\begin{cases} y_a = K_1 x_a + B_1 \\ y_b = K_1 x_b + B_1 \end{cases}$$

Since the cell phone must be located at the overlapping area of all three cell towers, two straight lines must intersect at a point in the overlapping area. The coordinates of that point is our goal. We apply the slope intercept equation group for both straight lines A, B and C, D.

$$\begin{cases} y_o = K_1 x_o + B_1 \\ y_o = K_1 x_o + B_2 \end{cases}$$

The O $(x_o y_o)$ are the approximate coordinates which we used to get the location without GPS.

For this solution, the system needs to communicate with the provider's service for getting the accurate tower or stable access point location. However, there will be issues with this process.

- Cost: in order for the user to access the Internet directly from their cell phone, they will need to pay for data communication. This cost depends on amount of kilobyte used.
- Speed: The speed is much slower if accessing the Internet from a cell phone compared to Wi-Fi.
- Signal cover area: currently, the Wi-Fi technology only works in the city areas.

Therefore, the best way to make up for each other's deficiencies is to combine them together. GPS has the best accuracy which the program will try to open first. If the cell phone does not have a GPS device or there is no GPS signal, the system will then scans and try to access wireless which is the next best because of the communication

[4] Slope Intercept equation: y = kx + b, where k is the slope and b is the y-intercept. Slope-intercept is the form used most often as the simplified equation of a line.

speed. If the system cannot find a wireless access point then will load the cell phone signal information. This order can be changed by the user's setting. In the iPhone platform, we will provide three options for the user. The first is "accuracy first mode" option which will access the GPS device first. The second is "save power mode" option which will access cell signal first, since it does not enable any extra device. The last option is "speed first mode" which will access Wi-Fi signal first, since Wi-Fi signal has the best network quality and network accessing speed. The basic pseudo code of solution is shown in the appendix.

2.1.5 Conclusion

Finding the user's location information without GPS is one of problems we needed to solve in this project. We solved this problem by combining existing technology and our own developments which makes up for the holes in the existing technology. Even though a solution has been found, it can still be improved upon. For example, we can improve the accuracy of the location. Currently, we only calculated the approximate coordinate. It would be better to calculate the accuracy information in meter, for instance telling the user "the precision is around 500 meters". The solution also has limitations, given that the location information of access points and cell towers are in limited cities only.

2.2 Synchronization Problem

Since we have more than one server deployed at different locations, the data needs to be synchronizing among the servers continuously. There are a number of reasons we implemented our server side in this way.

- Reliability: by placing data in more than one server, the client will not be dependent on the availability or accessibility of a specific server. It is better to keep at least two servers running to ensure guaranteed availability.
- Load balance: if the server is highly loaded, the client will have access to another lower loaded server.
- Localization: to ensure clients have accesses to a local or "reasonable local" server, the locality could provide high bandwidth and low latency access.

2.2.1 Problem analysis

Data synchronization is the process of establishing consistency among data on remote sources and the continuous harmonization of the data over time. It is fundamental to a wide variety of applications.

From research we found some existing solutions of data synchronization. For instance most database management systems provide a synchronization method to synchronize data, and most operation systems provide a file synchronization method and so on. However, we cannot apply these existing solutions into our application because of the following reasons:

- We cannot use the existing database system synchronization method because our servers are deployed in different places, and most of them are rented from different supporters. We cannot ensure that every host has enough authority level to modify database system configuration.
- The synchronization method from operation systems, such as the “rsync” service in Linux does suit our requirement either. The method is file based, meaning that the system will backup and synchronize updated file to the target location. Problem will arise if we apply this method. Due to the database file size of MySQL, this can be more than a hundred megabytes for a single file. Synchronization will take a long time for each change in the database. This issue makes this solution impossible.

With reasons illustrated above, we developed our own synchronization method based on existing algorithm in order to suit the unique requirements of the proposed system architecture. The cluster architecture has to be considered because it directly effects how we design our synchronization method. There are two candidate architectures from a distributed system point of view:

- Database centralization: this structure is similar to the client/server architecture. The front servers are independent from database, and the data does not need to be synchronized among the servers. This type of architecture provides a simple structure and high flexibility, but it can cause high load to the central database. The structure cannot ensure the speed of client accessing, because the database is not deployed locally. The structure diagram is shown as follows:

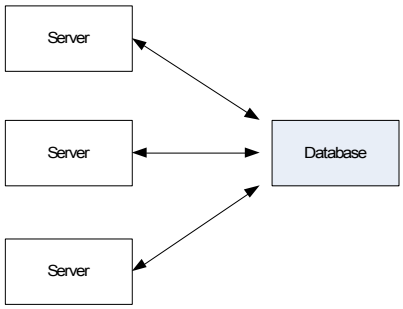
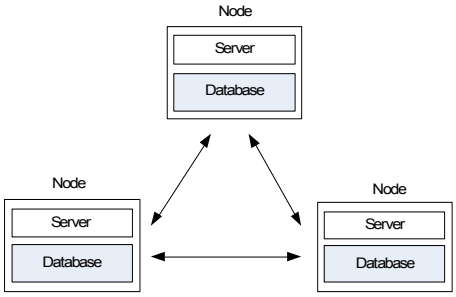


Figure 2.5

- Database decentralization: in this structure, each server connects to their database locally and data synchronization task is over network communication. This type of architecture may provide higher read/write data speed and higher network access speed than others. The client could access to a "local" server



which provides high bandwidth and low latency. Database decentralization structure is more difficult to implement than the database centralization structure. To implement this structure, we need to consider the crashing problem and other unpredictable problems while data is being transmitted over the network. The following figure shows the architecture diagram of this structure.

Figure 2.6

The distributed data synchronization is an atomic transaction among servers, a transaction that will either happen completely or not at all. This means our data has to be replicated to each server or rejected. We intend to apply database decentralization in our project because the advantages stated above. The two existing synchronization patterns [8] are:

- Broadcast (figure 2.7 a). The original node (sender) receives an update request and broadcasts it to the other node for voting. After voting the other nodes return their votes to the original node (sender) for request resolution.
- Daisy Chain (figure 2.7 b). The node receives the request votes and forwards the request along with its vote to another node. That node, in turn, votes and forwards the request and the votes along to another node that has not yet voted. This procedure continues until the request is resolved.

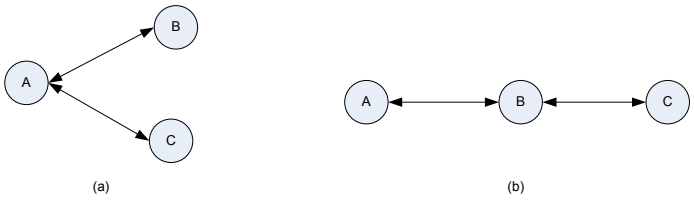


Figure 2.7

There are some general algorithms for atomic transactions in communicating systems. For example, two phases commit algorithm [9]:

The coordinator (original node, sender) side:

- Phase 1*:* When coordinator is ready to commit the transaction
  - Place *Prepare*(*T*) state in log on stable storage.
  - Send *Vote_request*(*T*) message to all other participants.
  - Wait for replies.
- Phase 2*:* Coordinator
  - If any participant replies *Abort*(*T*)
    - Place *Abort*(*T*) state in log on stable storage
    - Send *Global_Abort*(*T*) message to all participants

        - Locally abort transaction *T*
    - If all participants reply *Ready_to_commit*(*T*)
        - Place *Commit*(*T*) state in log on stable storage
        - Send *Global_Commit*(*T*) message to all participants
        - Proceed to commit transaction locally

The participant (target nodes, receivers) side:

- Phase 1: Participant gets *Vote_request*(*T*) from coordinator
    - Place *Abort*(*T*) or *Ready*(T) state in local log
    - Reply with *Abort*(*T*) or *Ready_to_commit*(*T*) message to coordinator
    - If *Abort*(*T*) state, locally abort transaction
- Phase 2: Participant
    - Wait for *Global_Abort*(*T*) or *Global_Commit*(*T*) message from coordinator
    - Place *Abort*(*T*) or *Commit*(T) state in local log
    - Abort or commit locally per message

Another Lamport's algorithm[5] works on distributed synchronization and time. The algorithm is given for synchronizing a system of logical clocks which can be used to order events. This algorithm is not suited for the requirements of our project, because the data items which it needs to consider for ordering includes its own timestamp. For example the alarm data item includes a time filed "20090915 17:26:17 +0200" and a unique email address for each user as the combined primary key in database. Therefore, the ordering task can be done within the database. We only need to focus on the transaction control.

2.2.2 Method of solving the problem

We applied database decentralization structure into our project, due to the performance issues and localization stated above. The solution is a modified version of two phase commit algorithms.

In coordinator side

- Copy "execute" command to cache.
- Broadcast a "execute" command to neighbors.
- Wait for feedback message from neighbors.

[5] "Time, Clocks, and the Ordering of Events in a Distributed System", Lesli Lamport. Communications of the ACM 1978, VOl. 21 No.7 558-565

- If all neighbors reply “success” message.
    - Execute the command locally.
    - If it is failure locally, send “undo” command to all neighbors.
        - Wait for “undo success” message from all neighbors.
- If any neighbor replies “Failure” message.
    - Send “undo” command to all neighbors.
        - Wait for “undo success” message from all neighbors.

In participant side

- Wait for command from coordinator
- Copy that commands to cache.
- If the command is “execute”
    - Execute the command locally, and then send “success” message to coordinator.
- If the command is “undo”
    - Execute “undo” command locally, and then send “undo success” message to coordinator.

The following figures have shown the sequence of our solution with three situations: figure (a) shows failure problem at coordinator, figure (b) shows the situation without any failure, and figure (c) shows the situation if failure message comes from any participants.

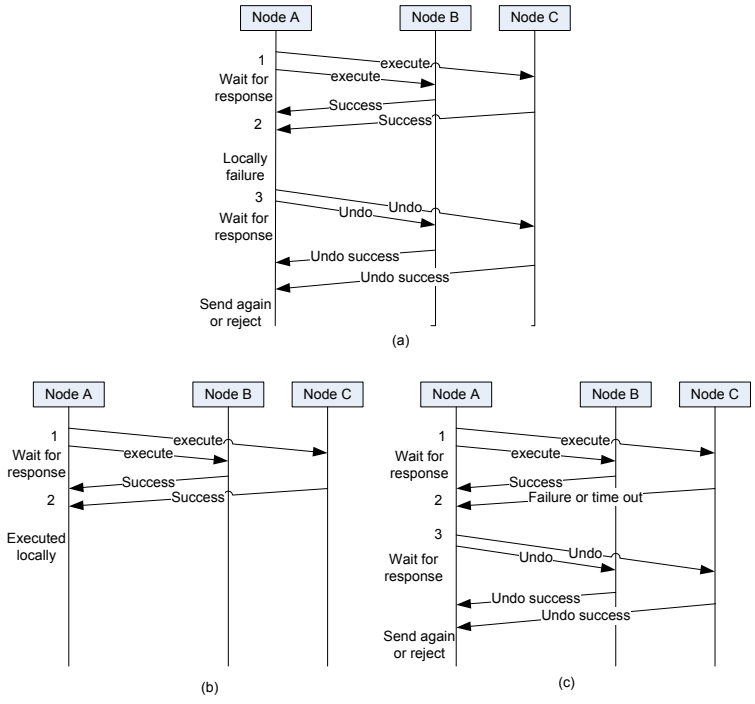


Figure 2.8

In our solution, the first handshake task of two phases commit algorithm is reduced. Instead of sending execute messages to other nodes in the beginning, we have a thread for updating all the servers' state lists periodically. In other words, the handshake task works in the background periodically. The system will not have concurrency problems because of our data definition and system design issues. The user can only login to a single server at the one time, in other words. Since a data item is only accessed by a single user, there is no critical data that needs to be locked. Synchronization task works on "user registration", "modify settings", "record location" and "notification" services. The data items of first two services are sequence independent which is updated depending on the unique id of each user's. The data item of last two services needs to be ordered, but each atom data item includes the timestamp from client as unique id; therefore the re-ordering task will be done in the database.

3. Analysis

3.1 System Requirement

The system has some sharp requirements from the company while the project is coming out. We divided them into functional and non-function requirements.

Functional requirements

- The system should be able to find the location
  - find location information at GPS mode
  - find location information without GPS
- The system should be able to send notification
  - send notification by SMS(Short Message Service)
  - the notification includes location information
  - the notification could show where you are on a map
  - the notification could show your track on a map
- The system should be able to record the location information
  - record location information with time stamp
- Statistic in server side
  - back office for statistic
  - log

Non-functional requirements

- The system is based on client server mode
- The client works on iPhone (primarily)
- The client works on other kind of cell phone (maybe in future)
- The system should be able to support multiple server
  - the server may be deployed on different place
  - the user may access any server
  - the data must be synchronized

3.2 Use Case

Use case is a description of the system behavior that originates from outside of that system. Based on the system requirements shown above, our system has two parts, the back office part and the client/server part.

The back office part is used to make statistics of the user's data, server management and server configuration that is mainly interactive with the administrators. The figure 3.1 shows the use case diagram of back office part.

- Login: it is the security portal for administrator access to back office system.
- Logout: it is the security way to exit office system.
- Statistic: the administrator could know all information about user's states, for example, how many users on line, number of users we have, number of issued alarm in last few minutes and so on.
- Manage user: administrator could change some parts data of users depending on the user's requirement.
- Settings: administrator could set properties of server, such as max notification per user and so on.

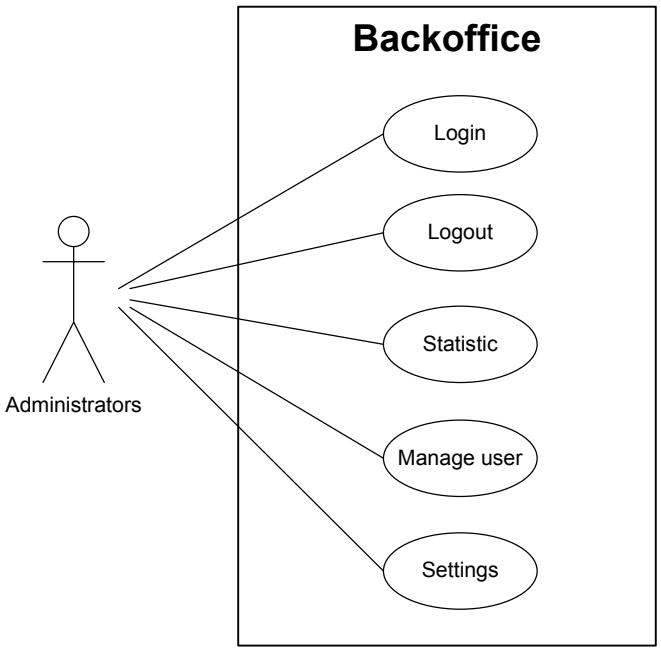


Figure 3.1

The other part contains the client side which is interactive with user and the server side which provides services to the user. This is the main part of this project and it includes all functionality for users. There are nine use cases in this part:

- Registration: register information when new user uses the system first time, such as email, pin code, name, phone number, their friend's information and so on.
- Login: secure portal for user to access to our system.
- Activate tracker: start tracking user's location, sending location information to server side and recording them into the database.
- Stop tracker: stop tracking user's location, need to enter pin code to stop tracker.
- Send notification: it means that, while the user is in need of help or is in danger, it sends an alarm to their friends through SMS to report where they are.

- View location on map: this function works for our users and their friends. Users could look for his location on map by clicking some buttons, and their friends could seek user's location by opening a link from the notification message.
- Settings: it is used to modify the user's information, such as changing the password and so on.
- Revoke: users can send message to tell their friends the last notification is invalid which means "I am fine".
- Logout: it is the safe way to exit from the system

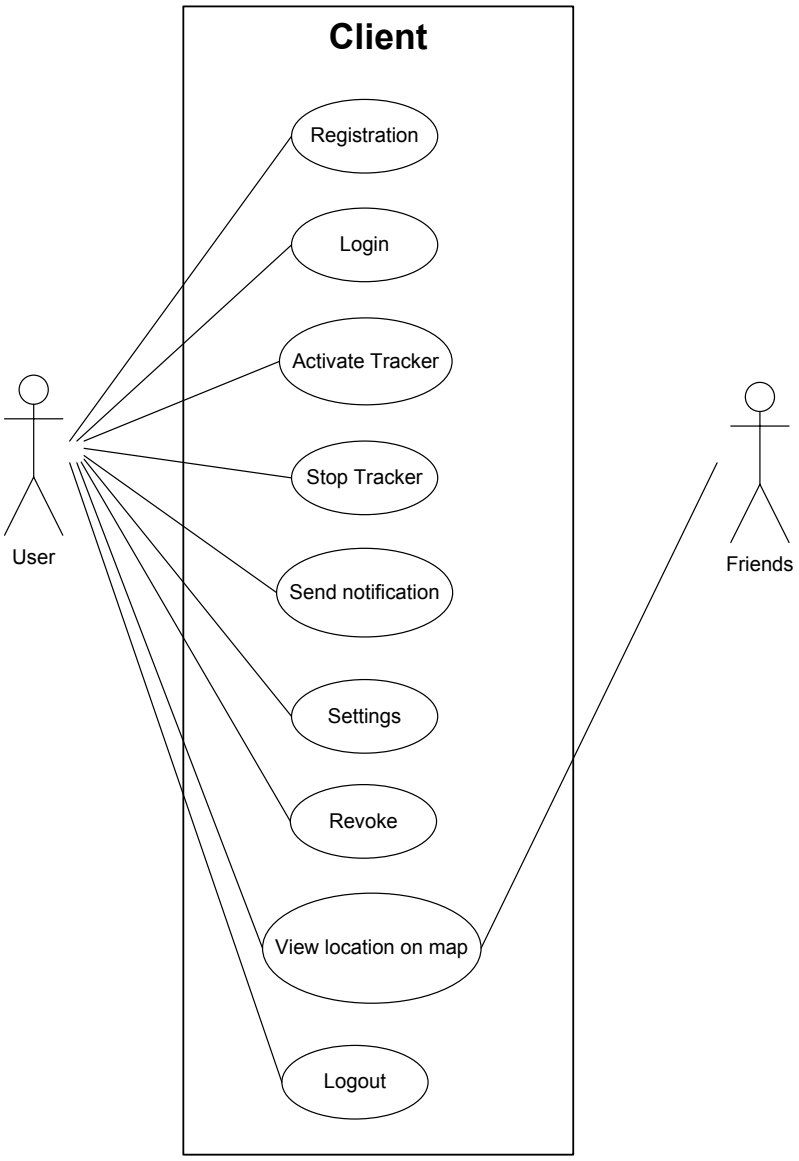


Figure 3.2

3.3 Services

Through the system requirements and study of use cases, the kinds of services we need to support at the server side become much clearer. Following services are core of entire system.

- Registration service: it supports registration of new user and record registration information into database
- Login service: user's information validation
- Notification service: it is used to send notification message with the link of map to user's friends by SMS.
- Revoking service: sending revoke message to user's friends by SMS.
- Signal information service: it depends on parameters from the client, location from cell tower or access point location information.
- Location calculation service: calculating user's location according to cell phone signal or access point information.
- Record location service: recording user's location information into database
- Maps generation service: generating a link of map that depends on the user’s coordinate
- Modify settings service: user could modify his settings of the system by invoking this service.
- Logout service: a safe way to exit system.
- Synchronization service: the service works for data synchronization among our servers.

4. Technology

From system overview and analysis, we had an outline of the whole system. In this chapter, we will introduce the technologies involved in our system.

4.1 Hibernate

Hibernate is a powerful, high performance object/relational persistence and query service. Hibernate allows you to develop persistent classes following object-oriented idiom - including association, inheritance, polymorphism, composition, and collections [11]. You can use object programming thinking to access and operate the database easier.

Hibernate applications define persistent classes that are "mapped" to database tables, and they adopt XML[6] file to specify the mapping between the object and database. The use of Hibernate requires creation of a Java class that represents the table in the database and then maps the instance variable into the class with correct columns in the database. Hibernate can be used to perform operations on the database like select, insert, update and delete the records in the table. It automatically creates the query to perform these operations.

4.2 Web service

Web service is defined by W3C[7] as "a software system designed to support interoperable machine to machine interaction over a network"[12]. It is a powerful programming language independent concept for data communication over a network. SOAP (Simple Object Access Protocol) is such a protocol used in web service. It is based on XML and HTTP, the kind of specification for exchanging structured information of web services in a network. It relies on XML for its message format and relies on HTTP for its message transmission. WSDL (Web Service Description Language) is a XML based language that provides a model for describe web service [13]. The basic structure of web service is shown in following figure.

[6] XML: Extensible Markup Language is a general-purpose specification for creating custom markup languages. XML's purpose is to aid information systems in sharing structured data, especially via the Internet, to encode documents, and to serialize data.

[7] W3C: World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential.

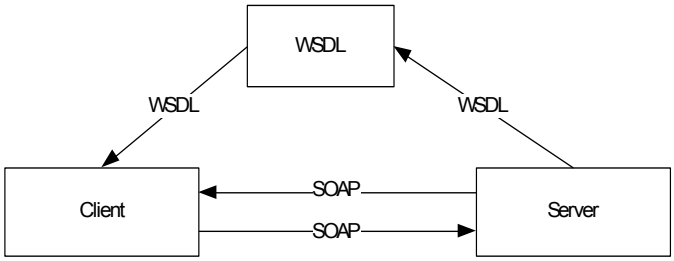


Figure 4.1

We choose web service as our data transmission, because it focuses on remote procedure calls. It presents a distributed method interface which is close to local method call. Java EE supports web service directly so there is no need to write our own codes for data communication. In other words, we would have needed to do a lot of work if we were building our own protocol upon transport layer, and this protocol may also be unsafe.

4.3 Enterprise Java Bean (EJB)

Enterprise Java Bean is defined in Java EE specification, and is a managed, server side component for local or remote application. It supports some services which are provided by the EJB container, such as automatic security, automatic transactions, life cycle management and so on. We do not need to code to perform such services. To implement those services, the code must conform to follow the rules that from the EJB container and implement an appropriate interface that allows the container to manage the component. There are three different types of EJB that are suited to different purposes:

- Session Bean - it is useful for mapping business process flow. It commonly represents pure functionality that is created as it is needed. There are two sub-types of session bean: stateless and stateful.
- Entity Bean - it maps a combination of data and associated functionality. The bean is usually based on an underlying data store and will be created based on that data within it.
- Message-driven Bean - it is very similar to the session bean, but is only activated when an asynchronous message arrives.

EJB is used as the server side since we cannot ensure the performance if the new server was written on our own, EJB can provide safe transaction and stable performance, as well as decrease our coding work significantly.

4.4 Security technology

As stated in the previous section, the communication part works with HTTP protocol. Since HTTP protocol does not contain security measures, the data can be attainted very easily by an unauthorized user. To protect the data, it has to be encoded into another format that only the client and server can understand. A basic security solution is shown in the following figure.

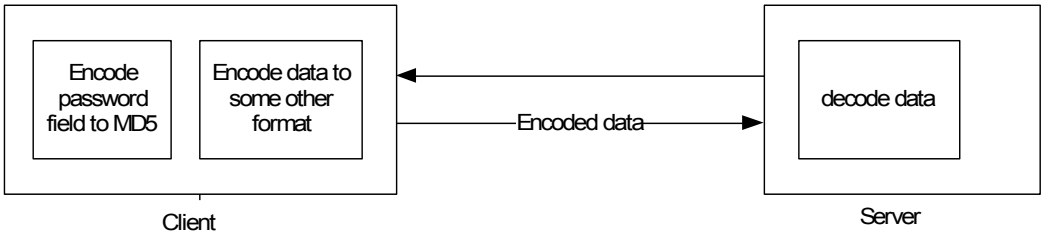


Figure 4.2

In our solution, the critical field, like a user's password, will be applied with MD5 (Message-Digest algorithm 5) cryptography algorithm. MD5 algorithm is one way pass over the data [14], this means that only the user knows the password and server just compiles two md5 codes to validate user's information. For the data of system transmitted, the client and server maintain a list that contains a set of numbers. Our program takes one of the numbers from the list and pluses that number to each char of the string. Then it passes the encoded string and the index of that number to server. The server side will de-code that data depending on the index of list. This process is illustrated below for more clarification.

a string "Hello world" in client side

a list A in both server and client side: A {1, -2, -1, 3}

take a random index : 0

pick up the number from that list; A[0] = 1

after plus 1 to each char of "Hello world"

we get "Ifmmp!xpsme", this is the encoded data

As for encryption, security will be ensured even if the data is compromised, because the list is hidden in the client and server side. The hacker can only know the index of lists which alone cannot crack the data.

5. Design

In this chapter, the details of system architecture will be presented. This includes the design of the client and server side, the database design, as well as the protocol between client/server, server/server.

5.1 System architecture

The entire system is based on client/server model. This includes the server/server communication for data synchronization which can also be regarded as another kind of client/server model. From the user's perspective, the client is the cell phone which is installed our system, and the server is one of the nodes. Using data synchronization, nodes connect to each other. One of the nodes can be viewed as a “client” and the other nodes can be considered as “servers” which are the destination of the "client". The following figure shows the overview of the system structure.

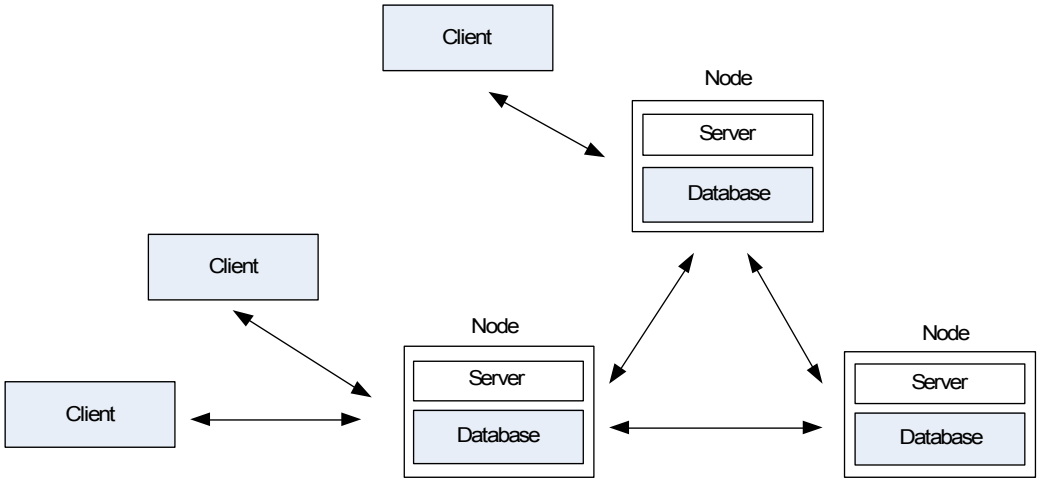


Figure 5.1

Web service is used to communicate between client/node and node/node. Our service is deployed on the nodes. The client can access any one of the nodes because these nodes are exactly the same.

Client Architecture

This section explains how the client architecture was designed and why it is manipulated in that way. Several design patterns were applied in the client part because patterns could make the codes more understandable and may be more efficient. The model-view-controller (MVC) design pattern is used in the client side especially in user interface, because it brings more extensible and readable coding manner which is better than programming without MVC pattern. There are three layers in this design pattern:

- Model layer: consists of objects that represent the data in the application. For example, representing the user's name, phone, and email to the model.

- View layer: presents layout and appearance of the system user interface, such as application's windows, views, and buttons and so on. Additionally, the view objects need to present notifications coming from the events.
- Controller layer: is a bridge between the model and view layer. It receives the notification from view layer and processes the action according to the data from model layer.

The delegation pattern is another important pattern used on client side. It is a technique that an object expresses certain behavior to the outside, but in reality, it is the delegate's responsibility for implementing that behavior to an associated object. [15]

The figure5.2 shows the client side architecture. The "user interface" in the diagram is the view layer of MVC pattern and the "function controller" in the diagram is the controller layer of MVC pattern. The controller layer receives UI event from the view layer such as button clicking event, and it is also responsible for sending notification to the view layer. The web service layer is mainly responsible for processing raw data and data communication. It consists of four parts: SOAP parser, protocol, security and network controller.

- Security: encrypt or decrypt data.
- Protocol: it's our own protocol used between communications, this defines the format of our operation or service calls.
- SOAP parser: responses for generating message of web service call following the soap protocol, or getting information from web service response.
- Network controller: it's the bottom layer of client side, which responses for sending and receiving HTTP request and response.

The "location timer" is a timer used for recording locations periodically, and it works on a thread. The "exception" defines our exception in the client side. If program gets any exception, the exception messages will be logged by "Logger".

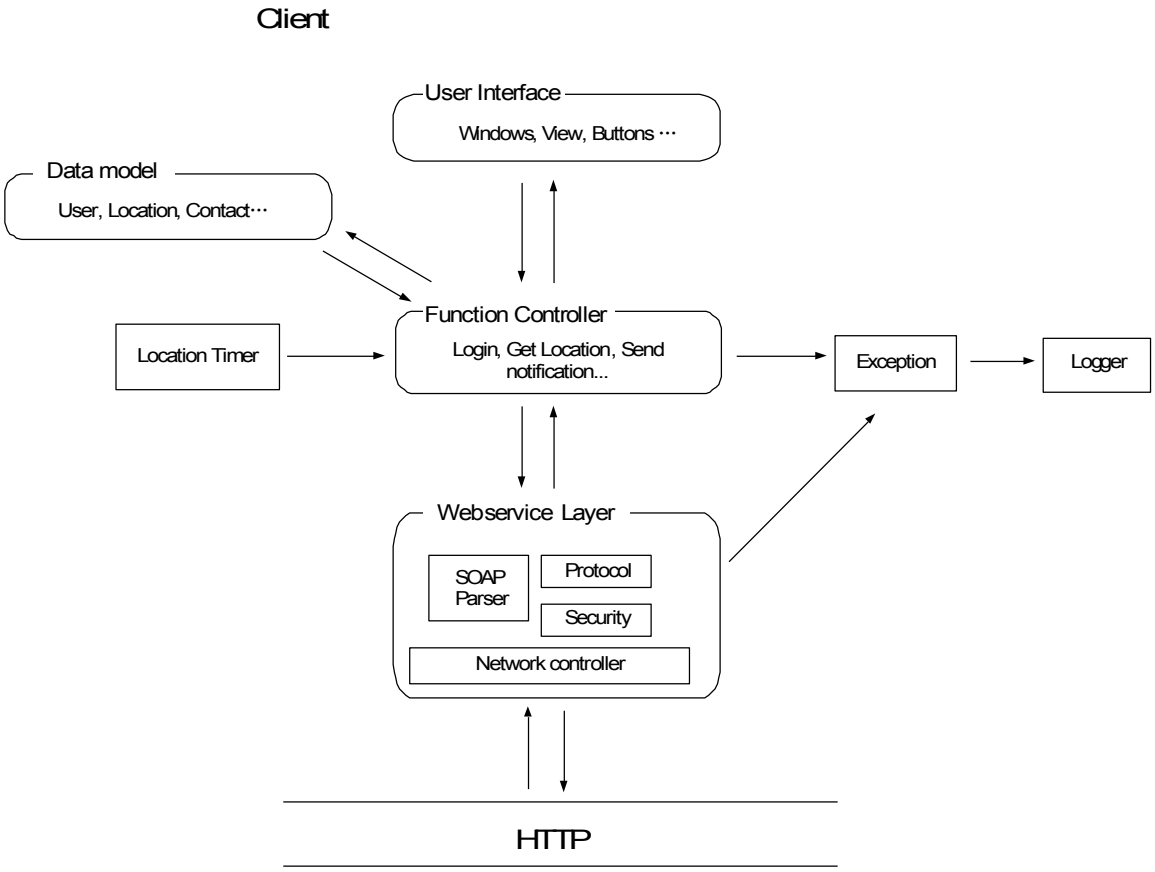


Figure 5.2

Server Architecture

The server side architecture is a bit more complex than the client side. We need to consider the server/server communication. Figure1.1 below shows the architecture of server side.

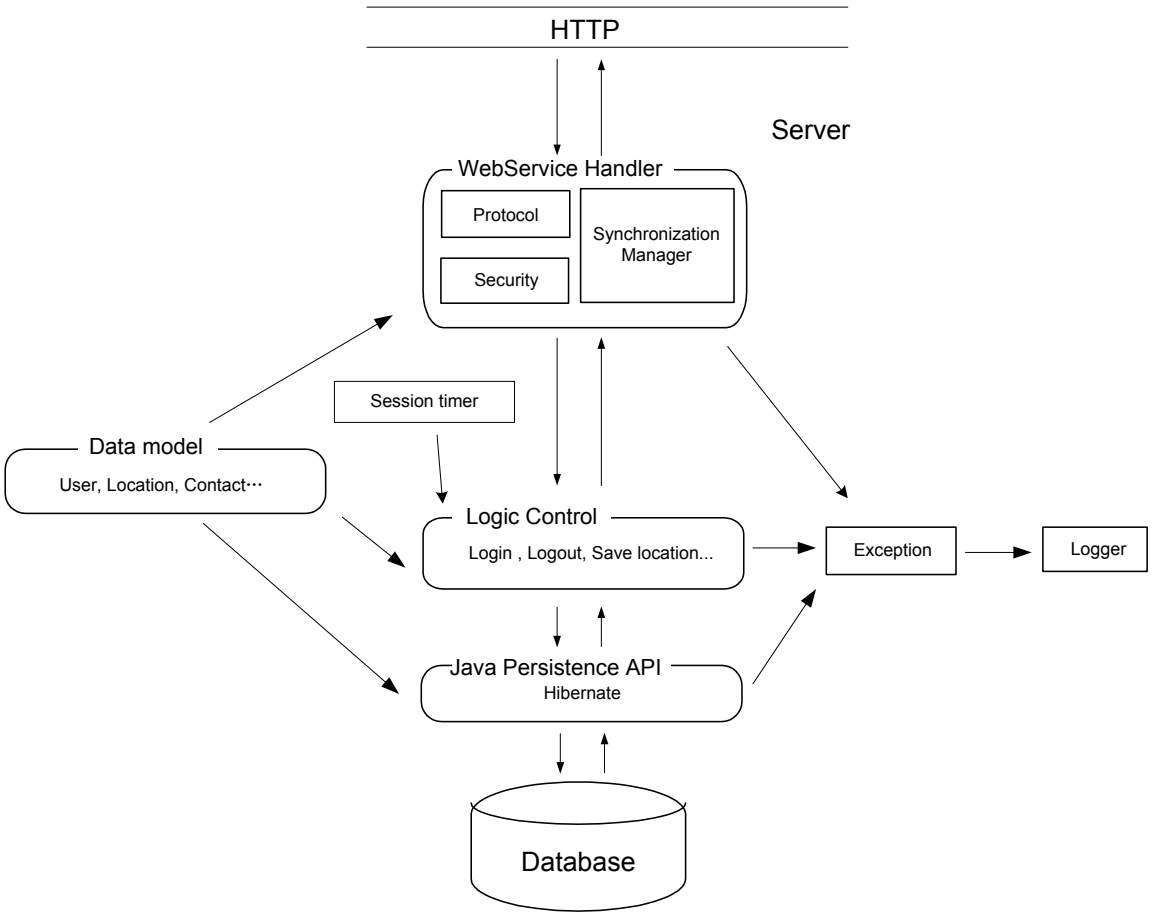


Figure 5.3

The "Web service handler" is responsible for supporting web method. Data format is received by "Protocol" the same as client side, and then decrypted securely. The "Synchronization Manager" plays the most important role in achieving data synchronization among servers. If an action comes from the client, the data is updated on local database first, and then invoke web method of other servers to synchronize data at the same time. If that action is not executed successfully in other servers or a time out error is raised, it will be stored in a "failed cache" in order to do it again later. The "failed cache" follows first-in-first-out principle to keep data in sequence. There is a thread which checks the cache and executes the action in the cache periodically. This way, the data is not the same in short periods, Furthermore, since the program decides which server the user will be connected to, it is impossible to change the connected server immediately.

The "Logic Control" defines what an action does. The actions such as login, logout, send alarm actions and so on. This part also maintains a session for each user. The session is created when the user logs in and a timer keeps track how long the user is not interactive with server. If this time passes a set limit and then server will force user to logout automatically.

The Java Persistence API manages the data and object/relational between the database and the Java program. It comes from Java EE specification, and is used for mapping data from object to database; for example, mapping "User" object to "User" table in database. It contains a full object/relational mapping specification supporting the use of Java language Meta data annotations or XML descriptors to define the mapping between Java objects and a relational database. It supports a rich, SQL (Structured Query Language) query language and EJB QL[8] query language for query data from the database. It also supports the use of pluggable persistence providers. There is many technologies, such as Hibernate, Toplink, and JDO to support the API, and we choose Hibernate in our project.

The "Data model", which is different from client side, defines all objects that we need to map to database. It will be used everywhere on the server side. The "Exception" defines our own type of exception for the server side. The "Logger" is responsible for recording exception messages for debugging.

5.2 Database Design

This step in the design phase is to determine what kind of data needs to be stored in the database and find out the relation between them. This data also needs to be present as objects in the "Data model". In other words, the user's information must be stored in the database. Information that should be stored includes the user's email as primary key, name, phone number, and the user's friends' information which includes their name and phone number. The location information must also be stored into database to keep track of the user's location. This information should consist of the user's longitude and latitude, as well as the time when this information was received. The alarm also needs to be stored into the database. The following figure shows the entire database diagram.

[8] EJB QL: Enterprise JavaBeans Query Language defines the queries for the finder and selects methods of an entity bean with container-managed persistence.

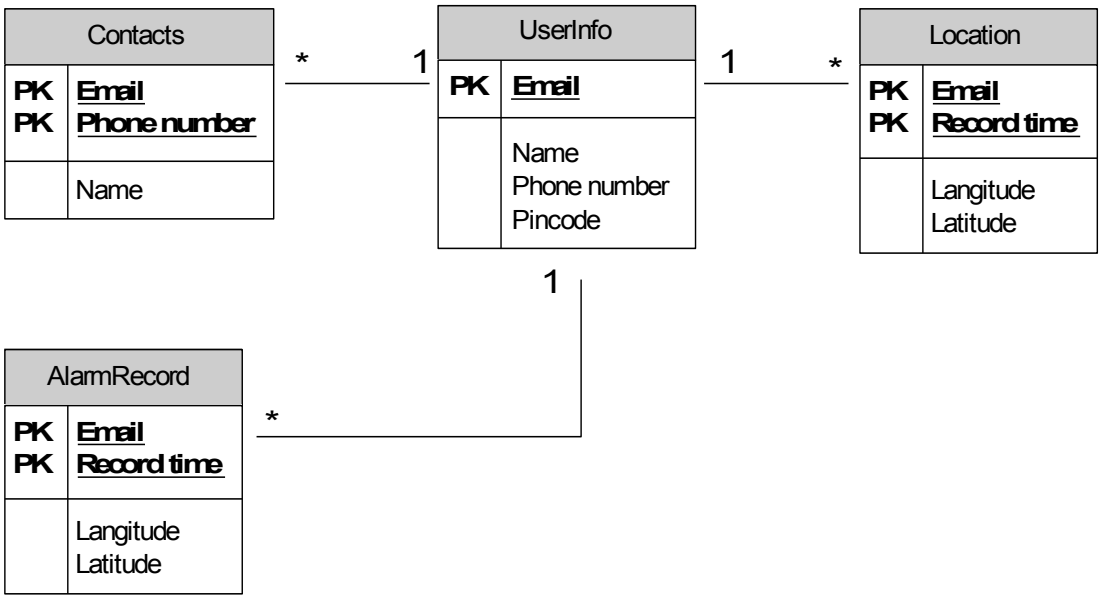


Figure 5.4

5.3 Protocol

The protocol defines a set of rules between the server/server and server/client communication. We used XML as our protocol format because following reasons:

- It is an extendable language, meaning that we can create our own tags, or use the tags which have already been created.
- It is fully compatible with applications like JAVA, Objective C, and it can be combined with any application which is capable of processing XML irrespective of the platform it is being used on.

The root tag of protocol specifies the kind of services from a request, and the child tag specifies the provided data set to service. For example:

```
<login>
        <email>.. </email>
        <pincode>..<pincode>
</login>
```

The root tag “login”, means user require login to system, the tag email and pin code are information provided to login service. The feedback message from server, we attached "return" at each root tag, for example <login-return>...</login-return>.

Protocol description

User registration service:

```
<reg>
 <email></email>
 <pin></pin>
 <users_name></users_name>
 <users_phone></users_phone>
 <contacts> (optional)
  <friend>
        <name> </name>
        <phone></phone>
  </friend>
        ….
 </contacts>
</reg>
```

```
<reg-return>
        <state>true|false</state>
        <info ></info>
</reg-retrun>
```

This service is used for registering new customer by his/her email, pin code, name and phone. The tag “contacts” defines the contact list of the user which the system will send notification to. The user may also update their contact list after registration as well. The feedback message includes tag “state”, which may include an extra message depending on whether the service is executed or not. For example, the extra message could be an error information in the tag “info”.

Login service:

```
<login>
        <email>.. </email>
        <pincode>..<pincode>
</login>
```

```
<login-return>
 <state>true|false</state>
 <info></info>
 <success> (optional)
  <session></session>
  <credit></credit>
  <contacts>
   <firend>
    <name></name>
    <phone></phone>
   </firend>
    ....
  </contacts>
 </success>
</login-return>
```

This service is used for user to login to server, which requires user’s email and pin code information. The feedback message provides session id, credit information and contact list from server if login service is executed; otherwise it provides error information inside “info” tag.

Alarm service:

```
<alarm>
  <session></session>
  <test>true|false</test>
  <localtime></localtime>
</alarm>
```

```
<alarm-return>
  <state>true|false</state>
  <credit>20</credit>
  <info></info>
</alarm-return>
```

This service is used for user to send alarm to his friends. The tag “session” specifies the session id from the login service. The “test” tag tells server it is the regular alarm or just user’s test. The tag “localtime” specifies user’s local time from client mobile phone. Since credit is spent when the user sends the alarm, the updated credit information should also be attached in the feedback message.

Find and save current location service:

```
<findlocation>
  <session></session>
  <locationset>
    <location>
      </lot>
      </lat>
      </dis>
    </location>
    ......
  </locationset>
</findlocation>
```

```
<findlocation-return>
  <state>true|false</state>
  <info></info>
</findlocation-return>
```

This service is used for recording the user’s current location, as well as calculating and saving their current location if the mobile phone client can only get cell tower or access point information. The tag “locationset” includes the location information list (longitude, latitude, cover distance) of the cell towers or access points. These locations will be used for calculation.

Save current location service:

```
<location>
  <session></session>
  <lot></lot>
  <lat></lat>
</location>
```

```
<location-return>
  <state>true|false</state>
  <info></info>
</location-return>
```

This service is used for saving the user’s current location if the location can be found by mobile phone built in location service. The session id, longitude and latitude information should be provided to the server.

Synchronization service:

```
<syn>
 <from></from>
 <command>
  <![CDATA[<location>...</location>]]>
 </command>
  …..
</syn>
```

```
<syn-return>
 <state></state>
 <info></info>
</syn-retrun>
```

This service is used for server/server data synchronization. The “from” tag specifies the name of the original server. The tag “command” specifies the service protocol such as “Save current location service”.

6. Implementation

This chapter will present implementation details that are extended from design phase, including design pattern, web service, session control and data synchronization parts of server side, and the implementation detail of the location part is also presented in this chapter.

6.1 MVC design pattern

MVC pattern has to define three parts for its implementation; user interface definition, data model definition and controller definition which is used for handling and processing action from user interface.

```
@interface NewUserView: UIView{
        IBOutlet UITextField *name;
        IBOutlet UITextField *phoneno;
        IBOutlet UITextField *email;
        IBOutlet UITextField *pin;
        IBOutlet UITextField *retypepin;
        IBOutlet UIButton *acceptence;
        UIButton *submit;
}
@end
```

The UIView object is a window definition in Objective C, our interface "NewUserView" is extended from UIView object, which means a window shows on iPhone for new user registration and consists of several text fields for input.

```
@interface User: NSObject {
        NSString *email;
        NSString *name;
        NSString *phoneNo;
        NSString *pincode;
}
@property (nonatomic, retain) NSString *email;
@property (nonatomic, retain) NSString *name;
@property (nonatomic, retain) NSString *phoneNo;
@property (nonatomic, retain) NSString *pincode;
@end
```

For data model definition, we create an object called "User" used for modeling user's information, email, name, phone number and pin code, which are user’s properties.

```
@interface NewUserViewController {
        UIView *newuserview;
```

```
        User *userinfo;
}
-(void) submitAction:(id)target
@end
```

Here we define controller for new registration, and the property “newuserview” is the reference of view layer. The property “userinfo” is a data model to model user’s information from view layer, and the method "submitAction" is used for catching submit button action from "NewUserView". Inside the method, we firstly collect user's input from view layer, and then compose that information to "User" model.

6.2 Delegation design pattern

Given that delegation pattern is something about method execution outside object, the implementation idea is to use program interface. Inside object, it only assigns the interface that needs to invoke, and the real code of implementation is defined in the other objects which extends from that interface. For example, in our network communication part, the interface "NetworkDelegate" defines a method "response" with a string parameter, which is used for processing income message from network, and the "NetworkController" responds for HTTP communicates operation.

```
@protocol NetworkDelegate <NSObject>
-(void) response:(NSString *)incoming;
@end


@interface NetworkController : NSObject {
        id<NetworkDeleaget> delegate;
                .....
}
        ......
@end
```

When the data comes from network, the program only invokes interface's method “response” from "NetworkController".

```
[delegate response: incomedata];
```

6.3 Web service

The program definition of server side web service is very simple, for the new version Java EE[9] supports annotation definition for web service. The basic structure is shown below.

[9] Java EE 5: new version of Java EE specification, it add some major features of the new specification, such as EJB 3.0, Java Persistence Architecture (JPA), Web services, JAX-WS.

```
@WebService()
@Stateless()
public class RegistrationService implements RegistrationServiceIF{
        .......
   @WebMethod(operationName = "registration")
   public String registration(@WebParam(name = "protocolXML")
                                          String protocolXML) {
        .....
   }
```

The parameter of web method stands for the incoming data from client, the returned value of each web method means the data needs to be sent to client. There is only one parameter for each web method, because transmitted data has already been defined in our protocol. The parameter "protocolXML" is a string with our protocol format. The type of returned value is also a string with our protocol format. Each web method presents what service we have in the server side. The definition of all web service is shown on appendix.

In iPhone client side, we use TCP/IP connection API to realize web service request. The following HTTP protocol code shows details of web service called from client. The first line defines request method, path of web service and protocol type. The header HOST define address of server, the header Content-Type that describe format of body, we use XML format based on SOAP protocol, while the header Content-Length tells length of body message to server. The format of body completely matches SOAP protocol.

```
POST /WebserviceUserRegistration HTTP1.1\r\n
HOST: 192.168.0.125\r\n
Content-Type:text/xml\r\n
Content-Length:[bodylength]\r\b\n
\r\n
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header/>
   <S:Body>
      <ns2:login xmlns:ns2="http://ws.isafety.com/">

<protocolXML><![CDATA[<login><email>benqyang_2006@hotmail.com</email><pin>12345
</pin></login>]]></protocolXML>
      </ns2:login>
   </S:Body>
</S:Envelope>
```

6.4 Session control

In the server side, the session control part is written by stateless session bean, it sounds bad because Java EE contains stateful session bean for user's session control. But the problem is that the stateful session bean does not support session control

which invoked from local. For instance, our session bean is invoked from the web service handler, which is a local method invocation. That is why exiting solution from Java EE cannot fit our requirements.

```
Final Map<String,UserTmp> onlineList = Collections.synchronizedMap(
                                        new HashMap<String,UserTmp>());
```

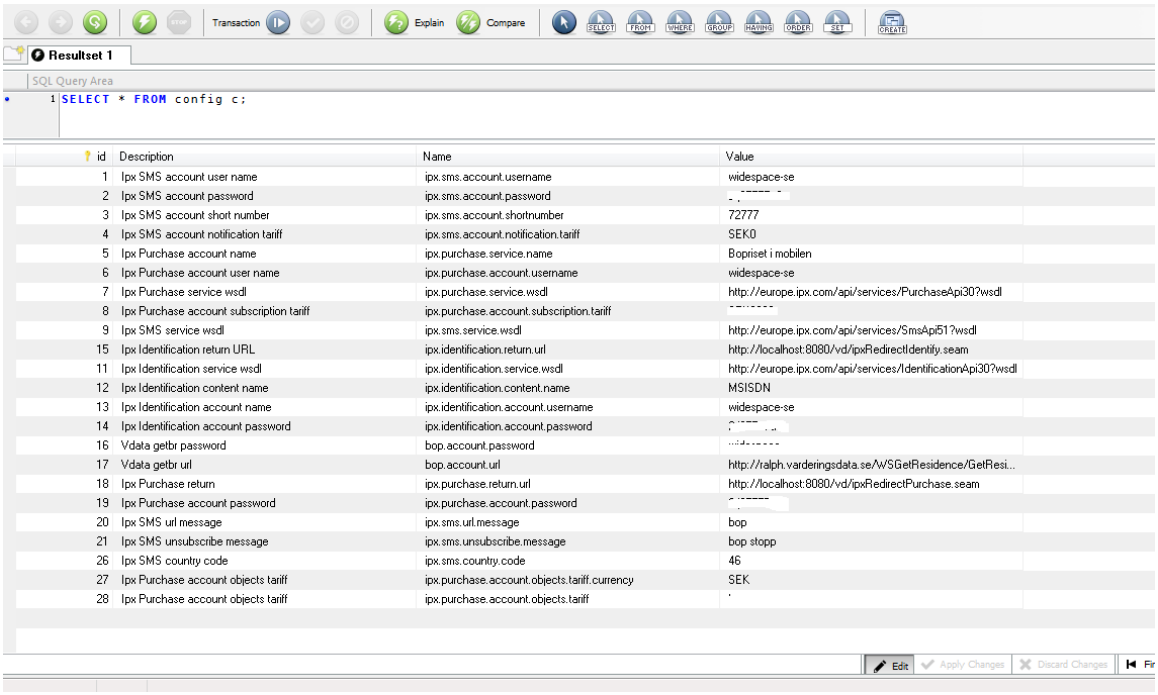
First of all, we create a synchronized map (the constant onlineList) to store on-line user's information, the key of the synchronized map is the session-id we generated, and the value is an object we created to store the user's data and timer. The session id format is like this: bde29cfe-a5bd-3cb5-a074-d88a8921edc6.

When user logging in, the system creates an unique identifier as user's session id, which depends on local server time and unique email address of user, and sends it back to client. The server also creates a timer to keep tracking off line duration. The session id is used for client/server interaction as the user identifier in the future and will be invalidated when user logging out. The timer is updated when user invokes a web method after login, if user does not invoke web method during a period, 20 minutes take an example, the timeout action will be activated to force user log-out automatically.

```
@Timeout
public void userTimeout(Timer timer){
   log.info("User time out : "+timer.getInfo().toString());
   // logout user
   this.logout(timer.getInfo().toString());
}
```

6.5 Sending SMS notification

The SMS message includes information of the user who sends this help message, a time of issued alarm and a web link showing user's latest three locations on the map. The service "sendAlarm(String protocolXML)" defines the code of sending SMS to user, we used Ericsson IPX-Messaging service[16] to send SMS to user, which is a web service provided by Ericsson enables to send and receive SMS and MMS messages connecting to more than one billion mobile subscribers in 25 countries. This API is not free, user needs to create an account and pay the service. Then user will get a list of properties that used for invoking Ericsson API. The following figure shows the list of properties which we maintain in our database.



id	Description	Name	Value
1	Ipx SMS account user name	ipx.sms.account.username	widespace-se
2	Ipx SMS account password	ipx.sms.account.password	
3	Ipx SMS account short number	ipx.sms.account.shortnumber	72777
4	Ipx SMS account notification tariff	ipx.sms.account.notification.tariff	SEK0
5	Ipx Purchase account name	ipx.purchase.service.name	Bopriset i mobilen
6	Ipx Purchase account user name	ipx.purchase.account.username	widespace-se
7	Ipx Purchase service wsdl	ipx.purchase.service.wsdl	http://europe.ipx.com/api/services/PurchaseApi30?wsdl
8	Ipx Purchase account subscription tariff	ipx.purchase.account.subscription.tariff	
9	Ipx SMS service wsdl	ipx.sms.service.wsdl	http://europe.ipx.com/api/services/SmsApi51?wsdl
15	Ipx Identification return URL	ipx.identification.return.url	http://localhost:8080/vd/ipxRedirectIdentify.seam
11	Ipx Identification service wsdl	ipx.identification.service.wsdl	http://europe.ipx.com/api/services/IdentificationApi30?wsdl
12	Ipx Identification content name	ipx.identification.content.name	MSISDN
13	Ipx Identification account name	ipx.identification.account.username	widespace-se
14	Ipx Identification account password	ipx.identification.account.password	
16	Vdata getbr password	bop.account.password	
17	Vdata getbr url	bop.account.url	http://ralph.varderingsdata.se/WSGetResidence/GetResi...
18	Ipx Purchase return	ipx.purchase.return.url	http://localhost:8080/vd/ipxRedirectPurchase.seam
19	Ipx Purchase account password	ipx.purchase.account.password	
20	Ipx SMS url message	ipx.sms.url.message	bop
21	Ipx SMS unsubscribe message	ipx.sms.unsubscribe.message	bop stopp
26	Ipx SMS country code	ipx.sms.country.code	46
27	Ipx Purchase account objects tariff	ipx.purchase.account.objects.tariff.currency	SEK
28	Ipx Purchase account objects tariff	ipx.purchase.account.objects.tariff	

Figure 6.1

6.6 Synchronization

Data synchronization is the most important part in the servers. The “queue” data structure for "failed cache" is used due to first-in-first-out principle. The elements with xml format to store the protocol defined before.

```
Queue<String> failedCache = new SynchronousQueue<String>();
```

Data synchronization just works for some methods which are used to modify database such as user registration method, user information setting method, record location method, and record alarm method. A new thread is created in order to handle the operation on “failedCache”, it is activated while a command putted into that cache. If there is no more command in the queue, which means the cache are empty, and then the thread become to waiting state. The following codes show part of synchronization method.

```
public void dataSyn(String action){
     /* call syn web method */
    try{
        ......
        if( responseCode == ERROR){
                failedCache.put(action);
                cacheThread.notify();
        }
        ........
    }catch(TimeoutException e1){
```

```
            failedCache.put(action);
            cacheThread.notify();
        }

Thread cacheThread = new Thread(new Runnable(){
      public void run(){
            for(;true;){
            String action = failedCache.poll();
            if(action == null){
            try{wait();}catch(InterruptedException e1){}
            }
            if(action != null){
                        dataSyn(action);
                        .....
});
```

6.7 Implementation of Location Info

The mobile phone keeps track four values that loaded from cell tower signal. They are MCC (Mobile Country Code), MNC (Mobile Network Code), LAC (Location Area Code) and CellID. The values are parameters of location API, for example, wireless access point information, the IP address is parameter of location API.

An interface is defined for getting location in iPhone and Java ME client. The codes are shown below. The first method has no parameter, which indicates getting location information from GPS if the device supports that, and the other two overridden method is invoked from first method. The second method with one IP parameter indicates getting location information of access point. The last method with four parameters indicates getting location information of cell tower.

```
@protocol LocationIF <NSObject>
-(NSString *) findLocationInfo;
-(NSString *) findLocationInfo:(NSString *) ip;
-(NSString *) findLocationInfo:(NSString *)mmc
                        andMNC:(NSString *)mnc
                        andLAC:(NSString *)lac
                        andID:(NSString *)cellid;
@end

public interface LocationIF{
public String findLocationInfo ();
public String findLocationInfo (String ip);
public String findLocationInfo (String mmc, String mnc,
                                String lac, String cellid);
}
```

These methods return a string with XML format that consist of location request type, longitude, latitude, and cover distance. The XML code is shown on below.

```
<location>
      </type>
      </longitude>
      </latitude>
      </coverdistance>
</location>
```

For this interface design purpose, it creates a standard for different providers in order to adapt providers towards, the same input and the returned value. The following code is used in iPhone client:

```
LocationIF *skyhook = [[Skyhook alloc] init];
LocationIF *mylocation = [[MyLocation alloc] init];
….
….
@try{
   [skyhook findLocationInfo];
   ……
}
@catch(){
   [mylocation findLocationInfo];
   ……
}
@finally{
   ……..
}
```

The “Skyhook” class and “MyLocation” class are both extended from “LocationIF” interface; these classes implemented for different providers. The program will ask “Skyhook” provider first, if there is no result returned, it will ask our own solution.

Location calculation

The calculation part works on server side, the client requests web service with parameters, and the service returns calculation results. The main task of location calculation is to solve the equation group, but it is hard to write our own Java code to do that, the solving performance cannot be ensured if this part is implemented by our self.

Matlab is a powerful tool for solving mathematic problem. Therefore the solution for the equation group is written in Java and executed in Matlab. In other words, the program executes the Matlab code outside Java program. It is much simpler than writing our own code for solving equation group with Java, of course the Matlab must be installed in the server side.

The function "solve" in Matlab is used for solving the equation group, the Matlab code is shown on following:

```
[xa,ya] = solve ('(xa-(-122.03))^2+(ya-37.33)^2=10','(xa-(-120.03))^2+(ya-36.33)^2=10')
```

```
xa =
 -122.35287565553229529525080787682
 -119.70712434446770470474919212318
ya =
 34.184248688935409409498384246361
 39.475751311064590590501615753639
```

xa and ya are the result of equation group with two different values.

7. Testing

In this chapter, the testing result of the solution and the comparison with the existing solution will be described, and some pictures about what the real system looks like will be shown.

7.1 Speed testing

After phase implementation, a speed tests for how long the system can get the location in both the solution and iPhone existing solution. In order to compare the difference between solutions is made, and the test method must be the same for different locations. The method is that in different places where both the cell phone signal and Wi-Fi wireless signal exist, and then the application program is run to load the location with each solution. The texts ware repeated for five times, and the average result of the tests is shown below. The result shows that the speed of getting the location from cell phone signal is unstable and slower than Wi-Fi, therefore the program is created to access the Wi-Fi network first.

	Cell phone 3G	Wi-Fi wireless
Time of first fit	25.87s	15.34s
	30.45s	10.12s
	-	20.23s
	50.85s	17.80s
	45.91s	9.67s

The same testing for the built in iPhone solution at the same place is used; the result is better than the solution and shown below. The loading speed increases immediately after the second testing for both the cell phone signal and Wi-Fi signal. It is assumed that iPhone solution has a timer to save customer request time, if the period between two requests is too short, it will use the customer previous location.

	Cell phone 3G	Wi-Fi wireless
Time of first fit	30.10s	9.31s
	18.67s	4.86s
	1.62s	1.23s
	1.39s	1.17s
	1.27s	1.29s

7.2 Precision testing

The precision of the testing is very import for this project, because the precision determines whether user's friends know range of the area where the accident place is, therefore a lot of testing at different areas have to be done. The precision test method is almost the same as the speed testing in different places where the cell phone signal

or Wi-Fi signal can be received, and then the application program is run to load the location with each solution, and finally the solution close to customer's current positioning is found.

In the solution, the precision result depends on how many cell tower signals or Wi-Fi signals can be received. The approximate location just equals to a cell tower location if there is only one signal from the cell tower, but in this situation, the Wi-Fi result is much better than the cell tower's result, because the Wi-Fi signal coverage is much smaller than the cell tower signal.

Because of the time and environment limitations, several tests of precision in Stockholm city area were only done, and the average result shows that the solution is almost similar as the built in iPhone solution. The maximum error in accuracy of the solution is around one kilometer with two 3G signals, but the average accuracy is around 500 meters. Here, one of testing results in Stockholm city area is shown. The following data show that the mobile phone gets signal information from two cell towers in the solution.

Network : 24005 Sweden - 3G/Tele2/Telia
Area : 002A STOCKHOLM
Cell : 002A0003 Karlavägen Engelbrektsgatan
Info :
Latitude : 59.340431
Longitude : 18.076065
Distance: 2.6

Network : 24005 Sweden - 3G/Tele2/Telia
Area : 002A STOCKHOLM
Cell : 2DAC Centralbron Tegelbacken
Info :
Latitude : 59.328724
Longitude : 18.059656
Distance: 2.6

The following figure shows where I am, the calculation results from cell phone signal and the Wi-Fi location. The coordinate (latitude, longitude) is translated into the Google map.

- A: shows where I am (59.340331, 18.082065)
- B: shows calculated location from two cell tower signals (59.335928, 18.072078)
- C: shows the location from Wi-Fi signal (59.339087, 18.0835639)

Figure 7.2

The method of iPhone existing solution testing is the same. The following figure shows the result of precision testing for iPhone built-in solution.

- A: shows where I am (59.340331, 18.082065)
- B: shows the location from cell phone signal (59.336509, 18.0813527, 85.129587, 483.598471)
- C: shows the location from Wi-Fi signal (59.3393193, 18.082432, 34.987167, 92.867415)

Figure 7.3

From the testing result, the precision of iPhone solution is similar to our solution. However, they are not good enough. The iPhone solution provides the value of horizontal accuracy and vertical accuracy which we do not have. Therefore our program tries to ask iPhone solution first, if it cannot find where you are, it will invoke our solution.

7.3 System functionality testing

The main function of the system is for tracking user's location and sending a notification to their friends while they get trouble or they need help, and other functionalities will be shown on appendix. The following screenshot shows what is inside the SMS message and how the map looks like.

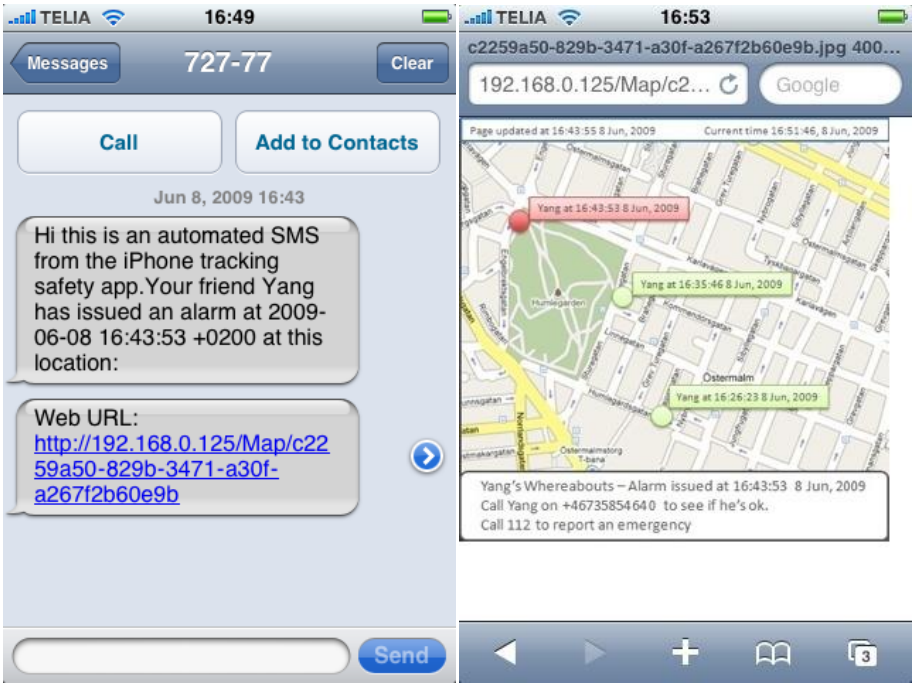


Figure 7.2

7.4 Testing Conclusion

According to the testing result, the exiting solution is a little bit better than ours due to it supports accuracy value. The testing task plays a vital role in this project. Unfortunately, the testing cannot be executed at many different areas and environments due to limited time. However, the company will publish a beta version first for testing the application. In this way, the system can be tested in different area by customers, and can also get some feedbacks from customers in order to improve the system according to customer suggestions.

8. Conclusion

Through this project we have gained a lot of project experience and understood related theory from the books more fundamentally.

The project belongs to rich client distributed system, it has solved the problem of getting location without using GPS, data synchronization and how we realize a software product from the beginning. The system has been done incrementally, from requirements study, problems analysis, design the method to solve problem, to final implementation and testing. Even though the development process is not quite in order, the solution rises up from a lot of re-analysis, re-design and re-implementation.

The system is not perfect, it would keep working on many improvements, the following issues could be considered as future works:

- Location database: our service provider which provides cell tower and access point location information still works on some part of Nordic countries. It is a real problem for this system, and we will try to add more providers to our solution.
- Java ME client: the location package only works on some SonyEricsson mobile phones, and we still need to take much effort on other kinds of mobile phones.
- Back office: a way of watching server state and statistic, we still work on that.

Reference

[1] Java EE, http://java.sun.com/javaee/

[2] GPS, http://en.wikipedia.org/wiki/Global_Positioning_System

[3] Cell phone signal, http://en.wikipedia.org/wiki/Radio_waves

[4] Wi-Fi, http://en.wikipedia.org/wiki/WiFi

[5] Google solution, http://www.cio.com/article/159852/Google_Maps_With_My_Location_Service_Uses_Cell_Towers_to_Find_Users

[6] Triangulation theory, http://scholar.ilib.cn/A-xfxyxb200203019.html

[7] Triangulation method, http://www.al911.org/wireless/triangulation_location.htm

[8] A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases, ROBERT H. THOMAS, Bolt Beranek and Newman, Inc.

[9] Two phases commit algorithms, Distributed Systems: Principles & Paradigms, 2nd ed. By Tanenbaum and Van Steen

[10] Hibernate documentation, https://www.hibernate.org/5.html

[11] W3C Web service, http://www.w3.org/TR

[12] W3C web service description language, http://www.w3.org/TR/wsdl

[13] Berson, Thomas A. (1992), "Differential Cryptanalysis Mod 2^{32} with Applications to MD5"

[14] Craig Larman, Applying UML and Patterns, Prentice Hall

[15] Ericsson IPX-Messaging service, http://www.ericsson.com/solutions/ipx/ipx-messaging.shtml

Appendix A: Web service definition

```
public interface RegistrationServiceIF {
public String registration(String protocolXML);
public String forgetPassword(String protocolXML);
}

public interface UserControlServiceIF {
public void login(String protocolXML);
public void logout(String protocolXML);
public void findLocation (String protocolXML);
public void saveLocation(String protocolXML);
public void sendAlarm(String protocolXML);
public void sendRevoke(String protocolXML);
public void update(String protocolXML);
}

public interface SynServiceIF {
public void synchronizeData(String protocolXML);
}
```

Appendix B: Protocol description

--- registration ---
<reg>
<email></email>
<pin></pin>
<users_name></users_name>
<users_phone></users_phone>
<contact>
<friend>
<name></name>
<phone></phone>
</friend>
</contact>
</reg>

------ login ------
<login>
<email></email>
<pin></pin>
</login>

------ logout ------
<logout>
<session></session>
</logout>

---- find location ----
<findocation>
<session></session>
<locationset>
<location>
</lot>
</lat>
</dis>
</location>
.....
</locationset>
</findlocation>

---- save location ----
<savelocation>
<session></session>
<lot></lot>
<lat></lat>
</savelocation>

---- send alarm ----

<alarm>
<session></session>
<localtime></localtime>
</alarm>

--- send revoke ---
<revoke>
<session></session>
</revoke>

--- synchronization ---
<syn>
</from>
<command><![CDATA[….]]></command>
…..
</syn>

--- update ----
<update>
<email></email>
<pin></pin>
<item></item>

</phone>
or
</pin>
or
<contact>
<friend>
<name></name>
<phone></phone>
</friend>
</contact>
</update>

Server ----------> Client
------ Regsitartion return -------
<reg-return>
<state></state>
<info></info>
</reg-retrun>

----- Login return ----

<login-return>
<session></session>
<contacts>
<firend>
<name></name>
<phone></phone>
</firend>

....
</contacts>
</login-return>

---- logout return ----
<logout-return>
<state>true|false</state>
<error></error>
</logout-return>

---- find location return ----
<findlocation-return>
<type></type>
<longitude></longitude>
<latitude></latitude>
<coverdistance></coverdistance>
</findlocation-return>

---- save location return ----
<savelocation-return>
<state></state>
<error></error>
</savelocation-return>

---- send alarm return ----
<alarm-return>
<state></state>
<error></error>
</alarm-return>
---- send revoke return ----
<revoke-return>
<state></state>
<error></error>
</revoke-return>
---- update return -----
<update-return>
<state></state>
<error></error>
</update-return>

Appendix C: Pseudo code of location solution

The pseudo code :

try GPS
if existed
    return location information from GPS
until time out or cannot find GPS device
try wireless
if existed
    save all wireless access point information
    get location information of these access point
    if number of access point = 1
        return coordinate of cell tower
    if number of access point = 2
        calculate intersection point
        calculate slope and intercept
        return center point coordinate of straight line
    if number of access point > 2
        take three of single information
        calculate intersection point $A(x_a, y_a), B(x_a, y_b)$
        calculate intersection point $C(x_c, y_c), D(x_d, y_d)$
        calculate slope and intercept K_1B_1
        calculate slope and intercept K_2B_2
        return coordinate of intersection point of both straight line
until time out or cannot find wireless device
try to get list of single information from cell phone
if success
    if number of single = 1
        return coordinate of cell tower
    if number of single = 2
        calculate intersection point
        calculate slope and intercept
        return center point coordinate of straight line
    if number of single > 2
        take three of single information
        calculate intersection point $A(x_a, y_a), B(x_a, y_b)$
        calculate intersection point $C(x_c, y_c), D(x_d, y_d)$
        calculate slope and intercept K_1B_1
        calculate slope and intercept K_2B_2
        return coordinate of intersection point of both straight line
until time out or some error occurred
fail - cannot find location