

**MOBILE LOCATION BASED INDEXING OF  
DISASTER NOTIFICATION SYSTEM**

**THU THU ZAN**

**UNIVERSITY OF COMPUTER STUDIES, YANGON**

**JANUARY, 2019**

# **Mobile Location Based Indexing of Disaster Notification System**

**Thu Thu Zan**

**University of Computer Studies, Yangon**

A thesis submitted to the University of Computer Studies, Yangon in partial  
fulfillment of the requirements for the degree of

**Doctor of Philosophy**

January, 2019

## **Statement of Originality**

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

.....  
Date

.....  
Thu Thu Zan

## ACKNOWLEDGEMENTS

First of all, I would like to thank the Minister, the Ministry of Science and Technology for full facilities support during the Ph.D. course at the University of Computer Studies, Yangon.

Secondly, I would like to express very special thanks to Dr. Mie Mie Thet Thwin, Rector, the University of Computer Studies, Yangon, for allowing me to develop this thesis and giving me general guidance during the period of my study.

I would like to extend my special appreciation to Prof. Dr. Nwe Nwe Win, Vice-President, Myanmar Computer Federation (MCF) for the useful comments, sharing knowledge, giving advice, and insight which are invaluable to me.

I would also like to express my respectful gratitude to Dr. Khine Moe Nwe, Professor, Dean of the Ph.D. 9<sup>th</sup> batch, the University of Computer Studies, Yangon, for her excellent guidance, caring, and providing me to join EBA internship fieldwork with foreign exposures during the Ph.D. study.

I would like to express my deepest gratitude to my supervisor, Dr. Sabai Phyu, Professor, the University of Computer Studies, Yangon, for her patient supervision, tenderness, encouragement and providing me with excellent ideas throughout the study of this thesis. I will always remember her for being a mentor to me.

I would like to express my respectful gratitude to all my teachers for their encouragement and recommending the thesis. To the reading committee teachers, especially Daw Ni Ni San, Lecturer, English Department, I would like to thank her for valuable supports and editing my thesis from the language point of view.

I also thank my friends from the Ph.D.9<sup>th</sup> batch for their co-operation and encouragement.

Last but not least, I am very much indebted to my family for always believing in me, for their endless love and support. They are always supportive of me during my period of studies, especially for this Doctorate Course.

## ABSTRACT

Myanmar is one of the countries that geographically located in the disaster-affected area because of the climate changes and environmental conditions. There is no way to prevent the natural disaster, but its impacts can be reduced or rescued. Adequate prior disaster information can save a significant number of lives and properties. Therefore, the accurate alerts or notifications about disasters are needed for Myanmar people. Now, there are many mobile phone users along with the technological progress around the world including in Myanmar. Thus, mobile devices become the most convenient communication tools which have not time and place limitations. A suitable disaster notification system based on mobile phones is one of the useful things as it is also a requirement of Myanmar people. Because of the fast and easy way, notification messages via mobile phones take benefits in the communication activities.

The main service task of this system is the delivery of possibly disaster information to mobile devices which are in the imminent disaster region. The system finds whether mobile is within a predefined area using its current location. The users who are in the imminent disaster area will receive the required notification messages. In this system, the server gets the current mobile position and keeps their location update structurally are a great challenge along with the continuously changing in the position of mobile devices. Therefore, a suitable technique is needed to store and update mobile positions. Moreover, when a message delivery to mobile devices within the specified range, the factor to be considered which mobile should be sent the message first. One of the most suitable solutions is sending the nearest mobile locations as fast as possible.

In this system, a two-dimensional balanced structure, a presort-nearest location index tree is proposed that allows maintaining, updating, and circular range querying mobile objects within the required time. It also supports generating nearest locations by index structure from the desired query point. In this structure, all of the location nodes are placed by level order thus nodes at any distance can easily find without traversing the whole tree and the searching time may reduce greatly. It is harmonized to solve nearest neighbor location queries since the locations of the data

points are based only on their relative distances from each other. In addition, all mobile locations in the range will be available by the distance at one time.

Then, the system architecture is built for sending notifications by connecting with firebase cloud messaging (FCM), application server and mobile devices in the imminent disaster area. To overcome being unnecessary updates at the server, Hybrid Update Algorithm is proposed in this system. In this case, a virtual mobile dataset is needed to access several of moving mobile locations for performance evaluation. Thus, a synthetic mobile location generator is proposed that is based on the creation of two-dimensional mobile locations. As a result, this generator is free from location privacy and confidentiality.

The necessary performances are tested by using a JUnit testing schema, which can automatically apply to run in testing functions. For performance evaluations, the execution time, updating time and CPU usage are measured by comparing between proposed presort-nearest location index tree, presort range tree and KD tree according to the evaluation of tree construction, range searching and neighbor searching over moving objects. Besides, the distance-based method is applied for comparison of two of index structures.

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>i</b>
<b>ABSTRACT</b>	<b>ii</b>
<b>LIST OF FIGURES</b>	<b>ix</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF EQUATIONS</b>	<b>xii</b>
<b>1. INTRODUCTION</b>	
1.1 Problem Definition .....	3
1.2 Terminology .....	5
1.3 Requirements.....	7
1.4 Motivation of the Research .....	7
1.5 Objectives of the Research.....	8
1.6 Contributions of the Research .....	9
1.7 Organization of the Research .....	10
<b>2. LITERATURE REVIEW</b>	
2.1 Android Location-Based Services.....	11
2.1.1 Challenges in Location-Based Services.....	11
2.1.1.1 Components of Location-Based Services.....	12
2.1.1.2 Location-Based Applications.....	13
2.1.2 Location Providers.....	14
2.1.3 Issues of Location Update Policies.....	15
2.2 Structure of Moving Objects.....	16
2.2.1 Moving Object Database.....	16
2.2.1.1 Location Management Perspective.....	17
2.2.1.2 Spatio-Temporal Data Perspective.....	18
2.2.2 Indexing Based Moving Object Structure.....	18
2.2.2.1 Indexing Trajectories of Moving Objects.....	19
2.2.2.2 Nature of Index Structures and Their Issues.....	20
2.2.2.3 Indexing Current and Future Movement.....	22
2.2.3 Querying Moving Objects.....	24

2.2.3.1	Query Types.....	24
2.2.3.2	Range Monitoring Queries.....	25
2.3	Synthetic Dataset Generation.....	26
2.3.1	Synthetic Dataset for Moving Objects.....	26
2.3.2	Advantages and Issues of Using Synthetic Data.....	27
2.4	Notification Process.....	28
2.4.1	Disaster Notification Techniques.....	28
2.4.2	Types of Notification.....	29
2.4.3	Categories of Location Based Alerting System.....	30
2.4.4	Importance of Notification.....	30
2.4.5	General Components and Capabilities of Notification.....	31
2.5	Summary.....	32

### **3. THEORETICAL BACKGROUND**

3.1	Trends in Mobile Technologies.....	33
3.2	Cloud for Mobile.....	34
3.2.1	Context-aware Services.....	35
3.2.2	Location Based Services (LBSs).....	37
3.2.2.1	Reactive LBSs.....	38
3.2.2.2	Proactive LBSs.....	38
3.2.3	Categories of Location Update Policies.....	39
3.2.3.1	Always Update Strategy.....	39
3.2.3.2	Never Update Strategy.....	39
3.2.3.3	Distance-based Location Update.....	40
3.2.3.4	Time-based Location Update.....	41
3.2.3.5	Movement-based Location Update.....	41
3.2.3.6	Profile-based Location Update.....	42
3.2.3.7	Deviation-based Location Update.....	42
3.2.3.8	Hybrid Location Update.....	43
3.2.4	Android Operating System in Mobile Technology.....	43
3.2.4.1	Advantages and Limitations of an Android.....	43
3.2.5	Cloud to Device Messaging.....	44
3.2.5.1	Firestore Cloud Messaging.....	46

3.3	Scope of Moving Object Database.....	48
3.4	Indexing and Its Factors.....	49
3.4.1	Range Searching Process and Types.....	50
3.4.1.1	One Dimensional Range Query.....	50
3.4.1.2	Two Dimensional Range Query.....	52
3.5	Presort Range Tree.....	53
3.5.1	Procedure of Presort Range Tree (PRTree).....	54
3.5.2	PRTree with Circular Range Searching.....	55
3.6	Example: Proposed Tree with center and service distance.....	55
3.7	Nearest Neighbor Techniques.....	56
3.7.1	Structure Based Techniques.....	56
3.7.1.1	Ball Tree k Nearest Neighbor (KNS1).....	57
3.7.1.2	KD Tree Nearest Neighbor (kdNN).....	57
3.7.1.3	Cover Tree.....	58
3.7.2	Structure Less Techniques.....	59
3.7.2.1	Brute Force Method.....	59
3.7.2.2	k Nearest Neighbor (kNN).....	59
3.7.2.3	Weighted k Nearest Neighbor (WkNN).....	60
3.7.2.4	Condensed Nearest Neighbor (CNN).....	60
3.7.2.5	Reduced Nearest Neighbor.....	60
3.7.2.6	Clustered k Nearest Neighbor.....	61
3.7.2.7	Rank Nearest Neighbor .....	61
3.6	Summary.....	61

#### **4. THE PROPOSED SYSTEM ARCHITECTURE**

4.1	Getting Mobile Locations.....	64
4.2	Updating Client Application's Location.....	65
4.3	Index Based Structure.....	66
4.3.1	Presort-Nearest Location Based Application Server.....	67
4.3.1.1	Proposed Presort-Nearest Location Tree.....	68
4.3.2	Presort-Nearest Location with Circular Range Search.....	69
4.3.3	Example: Calculating Proposed Index Tree.....	70
4.4	System Model.....	72

4.5	Multicast Messaging System.....	72
4.6	Notification System Steps.....	73
4.7	Scheduler Process Module.....	74
4.8	Process Flow Using Scheduler.....	75
4.9	Virtual Mobile Dataset Generation.....	76
4.10	Moving Object Generator.....	77
	4.10.1 Mobile Location Generator .....	78
	4.10.2 Process Flow of Generating Mobile Dataset.....	79
4.11	Distance-based Range Searching.....	80
4.12	Summary .....	80

## **5. IMPLEMENTATION OF THE PROPOSED SYSTEM**

5.1	Program Demonstration.....	83
	5.1.1 Main View of Program Demonstration.....	83
	5.1.2 Building Range Users Index Tree.....	84
	5.1.3 Sending Level Order Messages.....	84
5.2	System Implementation.....	84
	5.2.1 Prerequisites for Using Firebase Cloud Messaging.....	86
	5.2.2 Client Side Application.....	86
	5.2.2.1 Prerequisites for Client Application.....	86
	5.2.3 Server Side Implementation.....	87
5.3	Summary.....	88

## **6. EXPERIMENTAL RESULTS**

6.1	Means for Evaluation Environment.....	91
	6.1.1 Synthetic Dataset.....	92
	6.1.2 Experiments and Test Cases.....	93
6.2	Evaluation of Response Time by Location Providers.....	95
	6.2.1 Evaluation of Response Time by GPS.....	95
	6.2.2 Evaluation of Response Time by Network Provider.....	96
6.3	Evaluation of Message Arrival Time for Users .....	97
	6.3.1 Evaluation of Message Arrival Time for Online Users.....	97
	6.3.2 Evaluation of Message Arrival Time for Offline Users .....	98

6.4	Performance Evaluation Metrics .....	98
6.5	Evaluation of Processing Time .....	99
6.6	Continuous Range Search .....	100
6.7	Evaluation of Computational Responsiveness Time.....	101
6.8	Discussion.....	102
6.9	Comparison of Tree Construction Time.....	102
6.10	Comparison of Range Searching Time.....	103
6.11	Comparison of Nearest Neighbor Search within a Range.....	104
6.12	CPU Time of Continuous Range Query.....	105
6.13	Discussion.....	106
6.14	Summary .....	106
<b>7.</b>	<b>CONCLUSION AND FUTURE WORKS</b>	
7.1	Advantages and Limitation of the Proposed System.....	109
7.2	Conclusion.....	110
7.3	Future Works.....	110
	<b>AUTHOR'S PUBLICATIONS.....</b>	<b>112</b>
	<b>BIBLIOGRAPHY.....</b>	<b>114</b>
	<b>LIST OF ACRONYMS.....</b>	<b>124</b>

## LIST OF FIGURES

2.1	General Architecture of Location-based Services.....	12
2.2	Architecture of Mobile Object Database System.....	17
2.3	Objects and Index Structure.....	19
3.1	Mobile Cloud Computing Model.....	35
3.2	Dynamic Location Update Policies in Various Locations.....	40
3.3	Activities of FCM between App Server and Client App.....	45
3.4	App Server, Android Mobile and FCM relations.....	47
3.5	Sample one dimensional Range Search between 20 and 60.....	52
3.6	Sample Structure for Two-dimensional Range Tree.....	53
4.1	Architecture of the Proposed System .....	63
4.2	Getting Location of Client Application.....	65
4.3	Hybrid Location Update for Mobile Application.....	65
4.4	Sample Index Tree with Circular Range.....	67
4.5	Presort-Nearest Location Index Structure by Level Order.....	70
4.6	Sending Message by Level Order.....	71
4.7	Client-Server System Model.....	72
4.8	Communications between Mobile Phone, FCM and Server.....	73
4.9	Process flowchart of System with Scheduler Process.....	75
4.10	Architecture of Moving Object Generator.....	77
4.11	Process Flow of Moving Object Generator.....	79
6.1	Client Application's Location Update by GPS.....	95
6.2	Client Application's Location Update by Network Provider.....	96
6.3	Sending Continuous Messages to Mobile Users in the Range.....	97
6.4	Sending Messages to Mobile Users Who are Offline.....	98

6.5	Comparison of Processing Time.....	99
6.6	Comparison of Continuous Range Search Time.....	100
6.7	Comparison of response time.....	101
6.8	Comparison of Tree Constructing Time.....	102
6.9	Evaluation of Range Searching Time.....	103
6.10	Nearest Searching Time between proposed tree and KD tree.....	104
6.11	CPU Time for Continuous Range Query Over Moving Objects.....	105

## LIST OF TABLES

Table 2.1	Key Capabilities of Notifications .....	31
Table 3.1	Credential and Parameter Explanation.....	47
Table 5.1	Features and Their Explanations .....	87
Table 6.1	Parameters and Values .....	91

## LIST OF EQUATIONS

Equation 4.1.....	66
Equation 4.2.....	80
Equation 6.1 .....	99
Equation 6.2 .....	101

# CHAPTER 1

## INTRODUCTION

Nowadays, mobile phones are used as a daily need with desired information and search queries. Most of the users required queries are usually based on current or anticipated locations. For example, searching nearest restaurants, viewing the route and inquiry information are the user required queries which are supported by pop technology [17]. This technology normally based on static or quasi-static moving data. In addition, the services such as receiving weather information, emergency alerts, and advertisement getting access by automatic notifications are push technology [84]. This service is usually based on moving positions or mobile locations. Both of the two technologies come from location-based services (LBSs) which are one of the active research areas till now. Moreover, the two variants of the push and pop are two supportive sub-technologies along with the user requirements. They can provide vital support for disaster notifications. With advances in location-based services (LBSs), mobile and wireless devices have facilitated the tracking and monitoring of mobile objects. In such environments, mobile devices regularly receive their current positions from the location providers and send them to a server. The server maintains the location information and store in the database. The required queries such as "which mobiles are currently located within the imminent disaster area" are processed and results are sent to mobile devices. For these queries, the server has to search all of the current mobile locations that are in the disaster area. Therefore, an appropriate structure and nearest searching method are required.

Generally, most of the indexing structures with nearest neighbor search are separately taken by two parts: building index structure and searching nearest locations of the desired query point  $p$  [23]. The first part of building a tree also includes appropriate structure for index tree along with receiving and collecting data points. Then, an appropriate algorithm is used to find the nearest points from a given query point. It is totally good for static or non-moving objects structure but it may issue in moving locations or objects indexing. Moving object index structure with the regular update will also need to cooperate updating of the nearest points searching algorithm. It may think consistency between index structure and searching algorithm for discretionary data points query and search will be required concurrency control.

Normally, the nearest searching technique can be divided into two parts: structure-less technique and structure-based technique. For example, one of the most well-known techniques such as k nearest neighbor is a structure-less technique and it is very easy to implement. The general work of structure-less technique is that distance is calculated from all nodes to the sample nodes of a query and the node with the closest distance is regarded as the nearest position of nodes [27][34]. These techniques are very simple but the value of k affects the result. To improve the speed of query and memory requirement, a variety of tree-based index structures called structure-based techniques are applied in many areas. Recent advances in moving objects based research and applications, the positioning technologies and indexing based structures have collaborated together thus it leads to a perfect technique for maintaining and updating mobile position in the dynamic environment. Normally, the purpose of the traditional index structure is generating queries faster and easier with the required information. However, a good mobile objects' indexing needs not only faster query but also ability to update regularly. It is not easy to keep and apply for dynamic data as it needs to be reasonable for updating of an unusual and unpredictable number of moving positions.

In this system, the presort-nearest location index tree structure is proposed combining the concept of the nearest neighbor algorithm and tree-based index structure. Thus it gives the advantage of an index structure to easy data access and fast query along with the retrieving nearest locations from a location point that can be learned by the second closest to the nearest third and fourth by each location in the index structure.

This system explains how to combine mobile location attributes in the proposed index tree and system architecture is built to deal with the overall system. As mobile positions are basically used in this system so that it requires continuous checking to the update positions and the result is tested to valid within a suitable period of time. When the use of indexing based on location or position, the three basic positions are normally analyzed such as historical, current and anticipated future position. This system takes the second one which is based on the current position thus it has to get its location with the appropriate update will be there. To reduce the unnecessary and frequent update position in mobile, the location update policy is proposed by Hybrid Update Algorithm. It combines time and distance-based threshold

to reduce traffic cost. A synthetic mobile location generator is proposed to produce appropriate synthetic dataset.

In summary, the system is intended to send the notification to mobile devices which are in the imminent disaster area by keeping current mobile locations and maintaining them consistently. For example, when a disaster such as an earthquake occurs, the closest region to the epicenter location is regarded as the dangerous region. The notification message can be sent fast and easy if the closest locations to the epicenter are known. Messages can send successively through areas that really need notification based on the proposed index structure. In this system, circular range query and presort-nearest location index structure are proposed to send multicast notification starting from the nearest locations. All of the messages are sent by push technology which is supported by firebase cloud messaging (FCM). This system focuses on Android because it is used by the majority of smartphone devices in the world.

### **1.1 Problem Definition**

With the extending of mobile technology, the new areas are emerged by integration of getting mobile locations and handling their moving positions. In addition, the acquiring and maintaining of moving mobile positions widely spread in a wide variety of location-based applications. Therefore, the storage structures are required to allow continuously access and fast retrieval of information on these mobile objects. Some of the index based methods can be used for these objects. Normally, indexing has its own relevant structure that needs time to build it. It is saved by building the structure only for points that need to receive information and it can be used in the multicast notification.

Besides, there is an increasing interest in storage and indexing based on tree structures. Some previous spatial tree structures are not appropriate for keeping moving mobile locations because of their unbalanced structure. Searching for an unbalanced tree may require traversing an unusual and uncountable number of pointers and positions. Therefore, a suitable balanced two-dimensional tree structure for moving objects (mobile locations) is required. When the message is sent, it requires sending the message first to the mobile devices where they are nearest than the other devices in the range.

Since the variation of mobile locations continuously, tracking the changing position of mobile devices becomes a new challenge. All of these objects may not be the same in movement behaviors. Thus, the continuous location queries are the great challenge and common service demanded by mobile devices. Generally, a repeated query is a query which is worked in the database only once and remains active over a predefined query lifetime. It has to continuously update the query to the database with the predetermined time period. Besides, in moving location queries, not only a query point itself is moving, but also spatial points such as latitude and longitude stored in a spatial database can move too. It requires updating both the content and the organizational structure of the spatial database constantly.

In a client-server system of mobile location, a server usually keeps track of the location and its update whenever the mobile gets its location. As a result, the current and update location of each user would always be received at the server side whereas it would create a problem. There is nothing special for storing static objects but moving objects will need to be adaptive and dynamic. Moreover, it needs to maintain the update position consistently and faster query dynamically. If the mobile movements are small and frequent, needless updates would be received at the server. Moreover, it is not easy to update the database continuously whenever the locations change every time. In order to keep moving objects (e.g. mobile locations) in a database with the updated locations thus, an updating policy that supports a reasonable update for mobile objects is required.

According to environmental needs, many datasets are generated synthetically such as animal, mobile user, bus, Hurricane dataset, and so on. A synthetic dataset usually comes from application-dependent generation. Therefore, a dataset generator is required about their domain of interest before making any dataset synthetically. In reality, the millions of mobile positioning data are unavailable for performance evaluation of index structure. If the mobile location dataset can create synthetically, it would be free from location privacy and confidentiality. It can be used to get location data for performance evaluation of proposed index structure. Therefore, the requirement of generating a synthetic dataset for dynamic mobile locations that seem realistic is a challenge.

To define notable problem definitions related to this work:

Problem1. Getting current position of mobile devices with necessary updates as well as reducing the server bottleneck.

Problem2. Sending notification message continuously that begins from the nearest mobile devices in the range.

Problem3. Generating a large number of mobile locations for performance evaluations synthetically that seems realistic.

It should be noted that these problem definitions are closely related to each other and all of them have the same destination. The main idea of this system is sending notifications to mobile devices which are in the imminent disaster area.

## 1.2 Terminology

In order to get mobile locations and keep them into server consistently, the following general terminologies are needed.

**Location Based Services (LBSs):** Location-based services (LBSs) are used for the services based on the geographical location of mobile devices. It is also a vital role in finding the necessary information and querying the traffic areas. There are two types for these functions that automatically receive information from the service providers and inquiry or request from the user. These types of services are called push type of service and pull type of service. Both of them have basically come from location-based services which have their own advantages and applied in many areas [41].

**Global Positioning System (GPS):** It determines location using satellites. It does not need any kind of internet or wireless connection. One of the main features of the GPS is obtaining a current location for GPS-equipped devices or materials. GPS system consists of obtaining location from satellites, sending it to receivers, monitoring the accuracy and continuous changes [74].

**Cellular Network:** It supports receiving locations without fixing the initial location. It returns coarse fixes when the GPS is not enabled. These systems provide to support unlimited numbers of users based on the radio frequency (RF) channels that have the same limit in a group of cells [100].

**Network Provider:** It finds the user location with the help of cell tower and Wi-Fi access points. An accurate location is released after checking the status of the results obtained by network lookup. It can work both indoors and outdoors even it is less accurate compared to GPS. The results are fast and consume less battery [45].

**Google API:** Generally, most of the mobile phones receive their locations with the help of GPS. The activity of their accessing locations is different depending on the type and the features or functions of mobile devices. Some devices have complete

features of getting the location from a cellular network. Likewise, some devices receive their accurate and fast locations from the network provider. There is an optimal solution that can switch location providers such as GPS, network provider, and cellular network. This is called Google API which exists as a tradeoff provider between devices' features, type and functions [81].

**Google Cloud Messaging (GCM):** Google cloud messaging (GCM) provides for sending the message from cloud to Android device. It has no cost that exists as a persistent connection between a third-party server and Android devices to send data. It maintains queuing of messages when the target application of the mobile phones is offline. It has the ability to send push notifications and supports multicast message system [77].

**Firestore Cloud Messaging (FCM):** Firestore cloud messaging (FCM) is a free service that provides delivery of the message with reliability and security. It is the upgrade version of Google cloud messaging (GCM). The main objective of this version is to increase reliability and update the infrastructure of current GCM. It is well integrated with other Google services so that it occupies fast input/output and less access time [82].

**Android:** Android is an operating system for Linux-based mobile devices that have an operating system, middleware, and applications [75]. It provides for developers to create open platform applications. It supports a set of location-based applications and services which helps the users to access the multiple services based on the user location.

**Push Technology:** It is an Internet-based technology that the initial transaction or the request state is started from the server or the publisher. Push is also known as "Webcasting", "Net casting" or "Point casting" [18]. Push notification is a kind of notification that has been sent out to a device by a central server. Push has more capacity than pull technology such as immediacy, efficiency, reduced latency, longer battery life, shorter learning curve.

**Indexing:** It is a special data structure that allows having quick access to data or objects systematically. Indexing is usually used for users to display results that match the desired user criteria. In recent years, there has been a survey on indexing algorithms together with the activities of moving object databases to fast and efficient processes [51]. An indexing based structure is very suitable for continuous queries within a specified time by sending multicast messages, reports or notifications. It is

more suitable as location points are taken repeatedly within a range by only once the query.

**Location Update Policies:** Advancement in mobile computing technologies, location update policies or strategies has emerged in various location-based areas. Each of them has suitable applied areas with specific calculation. When the location update is done, the main concept is how many times of update process are required in each device. If the process is slow, it is difficult to get the latest position of this device. On the other hand, when the process is fast, an extra update occurs repeatedly and it consumes not only update cost and battery power [69].

### **1.3 Requirements**

Throughout this work, the following objectives are pursued:

- To receive the user's mobile phone from a special location provider called Google API
- To register to FCM that generates unique token ID to mobile phone
- To communicate to a server not only sending token ID but also the latitude and longitude of the current location
- To fetch notification from the server and sends to the mobile application with FCM
- To solve frequent or unnecessary update in both client and server side
- To support fast range query with a relevant index tree structure
- To send message starting from the nearest to the remote locations of mobile devices in the range
- To solve the performance of proposed approaches by comparing other existing index structure and without using indexing.
- To conclude the work an interpretation of the evaluation's results

### **1.4 Motivation of the Research**

People may be unobserved about the imminent natural disaster so that the lack of adequate news or notifications causes the major damage during the disaster. Especially, people in the disaster zone will be more difficult for those who do not know imminent disaster notifications. The concise disaster information can save a significant number of lives in some affected situations such as train derailments, head

injuries and gas-related fire damage. In this system, multicast notification service is utilized to send notifications only to those who need to receive them.

### **1.5 Objectives of the Research**

The main reason for this research is building presort-nearest location index tree structure for range and nearest queries by incorporating dynamic attributes (mobile locations). The next aim of this research is to reduce unnecessary update, communication cost, and server bottleneck problem by using Hybrid Update Algorithm. The last one is proposing a synthetic mobile objects generator to produce many mobile locations that can be used in performance evaluation of moving objects indexing.

The objectives of this research area are as follows:

- To be a balanced frequency of update location to the server and reduce traffic cost
- To support continuous range query for registered mobile locations
- To send message continuously for all mobile devices starting from the nearest mobile location in the range
- To generate realistic synthetic mobile object dataset for performance evaluation
- To be aware of the disaster-prone area and give notification message to mobile phones around the disaster area
- To conduct disaster preparedness and management plan
- To save life and property affected by the disaster and reduce disaster impacts
- To send notification messages in desire time using Android Push Notification approach
- To reduce thousands of network connection by FCM that supports persistent connection between mobile applications and application server.
- To take disaster recovery and rescue actions
- To provide empirical studies of using index tree and without index tree for mobile objects
- To send a message starting from the nearest mobile locations as fast as possible

## 1.6 Contributions of the Research

This research is concerned with multicast disaster notification system which is sent by using Google API, firebase cloud server that has firebase cloud messaging service and location-based services (LBSs). The results of this research are the web-based notification system and the implemented Android based application. The proposed index structure in this research is a mobile object based index tree that supports continuous range queries and nearest locations in the range. In fact, the range query is available without using index tree based on either a distance-based method or other measurements. But, it has to search along with a sequential or order match pattern by each object in the dataset. Therefore, two issues are found in this way.

- i. Such scanning of the large dataset can be very high cost.
- ii. The larger dataset, the more execution time is required by sequential searching.

Especially for repeating a single range, indexing based range searching is faster than each of the single match of range searching. The proposed index tree provides to search range queries in conformity with mobile objects. Then, the relevant update policy based on distance and time threshold values is proposed and it is called Hybrid Update Algorithm. It provides to be balanced update frequency for locations of mobile objects between mobile applications and service provider. Finally, the required synthetic mobile location generator is proposed by the process flowchart, the architecture, program demonstration and application test with performance evaluation.

The main contributions of the proposed system are as follows:

- i Proposing presort-nearest location index tree that is suitable for continuous range query and nearest query of moving mobile objects.
- ii Producing nearest nodes by level that supports to send message starting from the nearest locations of epicenter.
- iii The dynamic index structure that can handle both update and retrieve location information within the desired response time.
- iv Hybrid Update Algorithm is proposed to solve unnecessary frequent update position of moving mobiles at the client side.
- v A procedure to generate the virtual mobile dataset is proposed that is the creation of two-dimensional mobile locations.
- vi Generating location objects using random functions, categorizing three behaviors that they seem realistic by synthetic mobile location generator.

## **1.7 Organization of the Research**

This dissertation is organized with six chapters. This chapter includes an introduction, the motivation of the thesis, the problem statements, objectives, focuses and contribution of the research work. Chapter 2 surveys the challenges and components of location-based services, different sources to handle moving mobile objects and index tree structures on literature that deals with the dissertation. The theoretical background of moving objects database structure, types of range search and nearest neighbor search, location update strategies, and notification methods are described in Chapter 3. The architecture of the proposed system and the proposed algorithms for moving mobile objects is discussed in Chapter 4. The design and implementation of the proposed system are represented in Chapter 5. Chapter 6 describes the evaluation of the experimental results by measuring with the usage of the proposed presort-nearest location index tree, presort range tree and KD tree along with the comparison of distance-based range searching. Finally, Chapter 7 presents the conclusion extracted from this research works and depicts the future research lines.

## **CHAPTER 2**

### **LITERTATURE REVIEW AND RELATED WORK**

This chapter categorizes into four parts: location-based services (LBS), structure of moving objects and their databases, indexing based moving objects structure, the importance of notification, its related researches, and description of disaster notifications.

#### **2.1 Android Location-Based Services**

This section aims to categorize challenges in Location-based services (LBSs) over Android mobile phones, Android location providers and issues in location update policies. The challenges, types, components and applications of location-based services will be discussed and studied with respect to its related works. Then, various location providers and their features are studied together with the issues of location update policies.

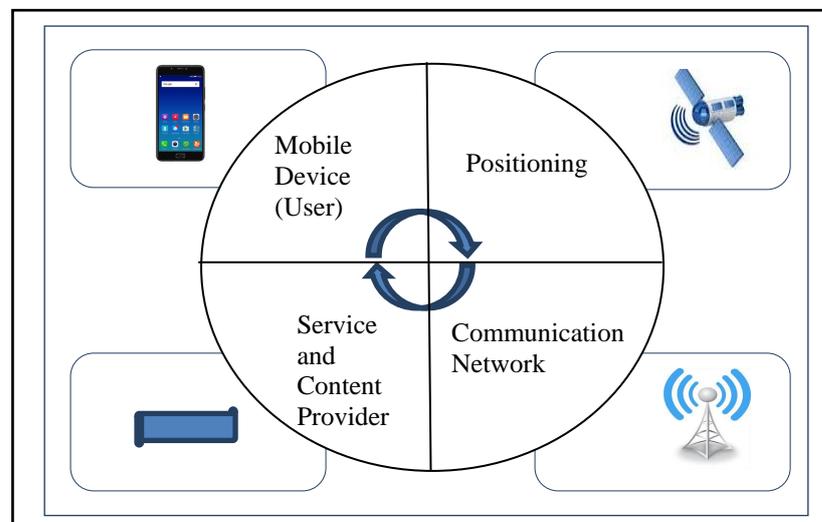
##### **2.1.1 Challenges in Location-Based Services**

Location-based services are services based on the device's geographical location given to the mobile phones. It typically provides information or entertainment. It largely relies on the mobile devices' location. Normally, the special issues and requirements related to LBS have grown since LBS together with the variety of research areas of positioning, modeling, and analysis of location-based data are rapidly increase [31]. In order to make the LBS services possible, some infrastructure elements are necessary, including:

- mobile devices,
- applications,
- communication network,
- positioning of components,
- servers,
- services.

Push notification, one of the location-based technologies is a clickable real-time message system which appears on the screen even mobile is in an idle state. For this technology, there is an inexpensive way of communication that interacts between

a third-party server and application. Moreover, it can create rich push messages with valuable notification and information. This is called firebase cloud messaging that allows queuing and delivering the message to mobile devices to the server [82]. Besides, as push comes from location-based services, it also has a similar service called pull [19]. However, apart from each of functional differences, the results of two types are quite different in energy consumption [18]. In pull type, the mobile application pulls the server whenever it requires new messages. In this way, pull may suffer battery consumption and message delays when the pulling frequency is too high or too low [42]. Nevertheless, both approaches are used in risk reduction management, such as natural disaster alert system and other information accessing systems such as advertising, product promotions and so on. A general architecture of the location-based services system is shown in figure 2.1.



**Figure 2.1 General Architecture of Location-based Services**

### 2.1.1.1 Components of Location-Based Services

Each of the location-based services is composed of several components and their connections. The required components for building location-based infrastructure are the following [93].

- (1) Communication Network: It provides a connection between the mobile device and service provider. All of the requests from mobile devices are transferred to the provider and the result information is delivered back to the mobile devices.

- (2) Mobile Devices: It serves as an interface tool that includes positioning technique. It is mainly used for entertainment, sharing news and location-based services. Today, the various mobile devices are emerged along with the improvement of technologies.
- (3) Positioning Component: It processes acquiring position, providing location and determining the current position. The positioning can be classified into two types: indoor and outdoor. Indoor positions are obtained by using radio beacons or active badges. On the other hand, outdoor positions are easily received with the help of location providers such as GPS, cellular network and so on.
- (4) Service and Location Provider: It mainly supports the accurate result requesting for desired location or position of mobile users. Each of the location providers has its own benefit according to the features and functions of mobile devices. It provides a number of different services especially for requesting and offering a location for mobile devices.
- (5) Data and Content Providers: The responsibilities of these providers are maintaining the location data and information in which the results are satisfied and applied in many areas.

### **2.1.1.2 Location-Based Applications**

The location-based services are greatly used in several different applications. It enables user desired applications like local advertisements, emergency notifications, tracking devices or moving objects, and finding nearby friends etc. There are also military and even terrorist location-based applications. The LBSs applications can be categorized into five groups.

- (1) Finding positioning and location services
  - a. application services (shops, hotels, ATMs, ...)
  - b. connector services (hot spots, adapters, ...)
- (2) Searching location to increase (network) services
  - a. incoming or outgoing connection improve by location
- (3) Providing information based on location
  - a. traffic guides, product promotions
- (4) Making others check of user current location
  - a. user (individual), range search, activities (group)

## (5) Security

- a. access users' location and care of their security

### **2.1.2 Location Providers**

Today mobile techniques or services are mostly based on the current location of the user that receives from the location providers. There are three main types of location providers that support for getting locations. They are GPS, Network Provider, and Cellular Network. They have the following features.

- (a) GPS features (GPS, AGPS)
  - i. It determines location using satellites.
  - ii. It does not need any kind of internet or wireless connection.
  - iii. This provider may take a while to return a location fix depending on the condition.
- (b) Network provider (AGPS, CellID, and Wi-Fi MACID)
  - i. It determines location based on the availability of cell tower and WiFi access points.
  - ii. It results are retrieved by means of a network lookup.
- (c) Cellular Network (CellID, WiFi MACID)
  - i. It supports receiving locations without fixing the initial location.
  - ii. It returns coarse fixes when the GPS is not enabled.
  - iii. It provides capabilities by connecting to the specific set of hardware and telecom.

The basic location provider that is usually used for the current location of the mobile device is GPS. This signal is uncertain to estimate moving object locations under weak conditions of Satellite [66]. Sometimes, Google map is used to map the road and places of interest in the location-based system. Amit Gosavil, Vishnu [36] proposed to notify the users who are located in the possible disaster zone and guided to the nearest location of a secure or safe place on the Google map of the application. This system aims not only normal people but also blind people to save lives and move them into the nearest safe zone or shelter prior to the disaster. It supports both visual and audio warning about disaster including the evacuation guidelines. But this system is not suitable for developing countries that are poor in the ability of Google map and it becomes the main challenge in this system [92]. In fact, all of the mobile devices have their own abilities to get the location. Some devices have good features of the

cellular network and produce accurate locations for these devices. Likewise, some devices receive their fast and accurate locations with the help of network providers. The common location provider that is basically used to get location is GPS. There is an optimal or special location provider that can switch between different location providers and their functions called Google API. It supports to access facility and allows updating schedule with a combination of effective techniques.

### **2.1.3 Issues of Location Update Policies**

Along with the rapid improvements of location-based services, tracking the changing position of locations becomes a new challenge. The location of the user current position is always received at the server side whereas it would create a problem. If the mobile movements are small and frequent, unnecessary updates would be performed at the server [41]. The objective of location update strategy is to provide efficient search-updates and support location management policy in location-based mobile computing.

Mohammad Wasif [94] also proposed a distance-based method and the update is done for mobiles with different velocities using distance method and the graphs are drawn for different modes and analyzed. It may reduce the update cost since an update is needed only when the mobile moves to a new location area. It increases in traffic volumes and the bottlenecks should be removed.

Cheng, Pingzhi Fan proposed a hybrid location update strategy in which an update occurs either the movement threshold or the time threshold is reached. The cell movement is count and the timer is set to zero initially. The convex function is used for the movement threshold and the timer is increased after a particular time period. The signal cost can be reduced with the help of the movement threshold value. This system showed that the proposed scheme is better performance than MBLU scheme [21].

Vicente Casares\_Giner also proposed a hybrid location update scheme that combines two update policies such as movement and distance based location strategies [35]. This strategy avoids useless location updates and shows the results of using the distance based scheme that produces the best result compared with movement based strategy. The results obtained from the empirical analysis show that it needs little memory in the mobile terminal and the good performances can be

obtained as distance-based scheme. However, the mobile terminal has to search for the identity of the new visited cell in a cache memory after each movement.

The issues of location update on the mobile device and the results obtained after transmission of continuous location-dependent queries from the server are discussed [52]. These queries are controlled by the response time constraint. The storing and processing of moving object values have dynamic and real-time properties. Therefore, this system proposed a new method called Adaptive Monitor Method (AMM) that provide less update workload and monitor the update position of moving objects.

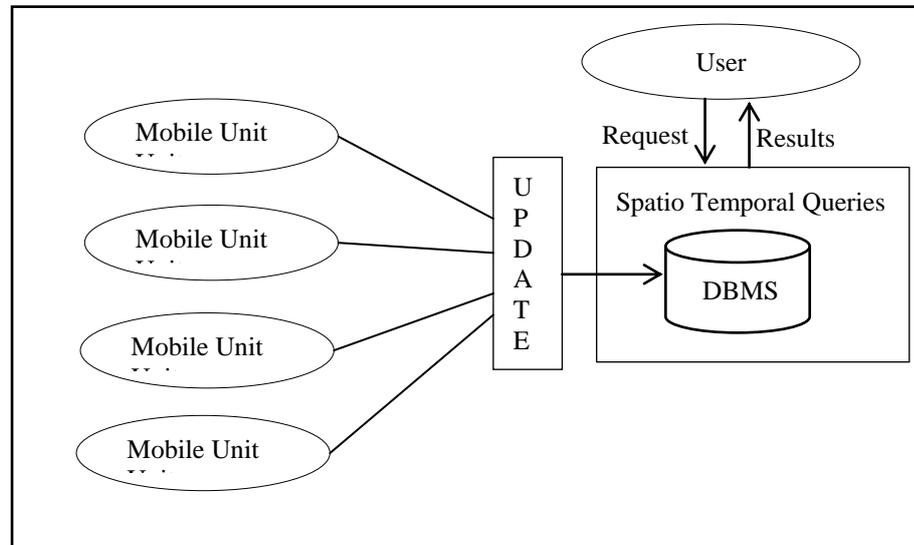
## **2.2 Structure of Moving Objects**

Since traditional database systems assume that data stored in the database remain constant unless it is explicitly modified, they are not appropriate for representing, storing, and querying dynamic attribute such as moving object because the database has to be updated continuously. Moreover, if traditional multi-dimensional index structures such as R-tree [7] are used for indexing moving objects, similar problems still remain. The modification of location may need to split or merge the nodes. This requires additional overheads. In the worst case, frequent index rebuilding may be required. For these reasons, R-tree based index structures are not proper to index dynamic attributes such as the positions of continuously moving objects. This section discusses the properties of moving object database with two perspectives and its advantages. Then the indexing based moving objects structure and query types are consulted with the applied areas and related works. Especially, indexing based range monitoring queries are studied in moving object index structures.

### **2.2.1 Moving Object Database**

Moving objects are objects whose positions are changed instantly e.g. moving cars, fighter airplane, ship etc. The normal data is usually stored as constant and less update in the database. So, there is a problem that the present database don't know how to manage and query such moving objects. To represent a moving object in a database, it needs to update very frequently the position of the moving object. Therefore, moving object database has emerged which is a database that can represent and query the moving object. It can run powerful query languages and answer any

kind of questions about movements. For example, which mobile users are closest to an imminent disaster region? Which routes are safe for them? The architecture of a moving object database system is shown in figure 2.2.



**Figure 2.2 Architecture of Mobile Object Database System**

The advantages of mobile object database are the following.

- It is a simple approach to handle location management using DBMS.
- It needs very few updates to the database
- The architecture shall not cost much to construct.
- The answer queries involve both temporal and spatial data.
- It can handle the uncertainty of the mobile unit location

The disadvantages of mobile object database are the following.

- The mobile unit should have the capability to give its location uncertainty.
- The dynamic attribute may also need to be updated frequently if the mobility pattern is not smooth.
- A function for the dynamic attribute may not be able to express in few cases.

### **2.2.1.1 Location Management Perspective**

There are two main purposes in moving object database.

- (1) To maintain the current locations of moving objects dynamically
- (2) To ask queries about the current or anticipated positions of the moving objects.

Moving object sends their current positions to the database and database performs updates. To keep and update the location of objects instantly, each of the objects sends its position very often. The update between a server and moving objects need to be balanced. If the updates occur and request to the database frequently and the error in an update or last location in the database is received small, the update cost with database workload becomes high. Conversely, if the update occurs too slowly and the stored position is wrong, the actual positions become large. If the update performs very often, the error is small and the update load is very high. If the update performs as less frequent, then the update load is low and the error is large.

### **2.2.1.2 Spatio-Temporal Data Perspective**

It is a database which stores not only the current status of traditional spatial data but also the whole development process history. This type of database is used in the following situations.

- (1) Need to reach in time to a particular period of time and to retrieve the status at that time. E.g did whales habitats move in the last 20 years?
- (2) For understanding the changing state of the things, they will be analyzed after confirming the good relationship. E.g is the Dead Sea shrinking?

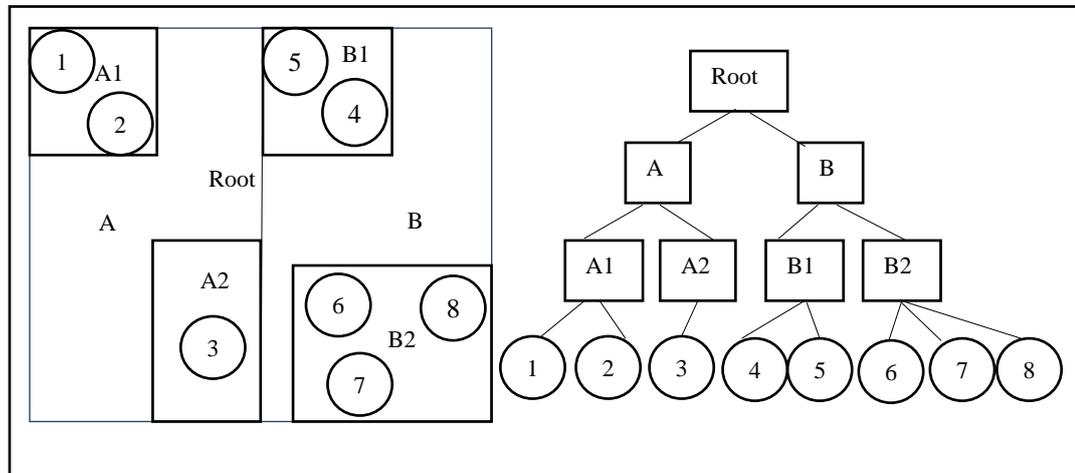
Therefore, there are two common questions.

- (1) Which data are stored in a spatio-temporal database?
  - A point, line, region, network, partition
- (2) What kind of change may occur?
  - Discrete change: Construction of a building
  - Continuous change: Vehicle

### **2.2.2 Indexing Based Moving Object Structure**

Indexing based researches are widely emerged in moving objects environment. The traditional data structures have not been designed to flexible dynamic updates of moving objects. Researchers introduced new index structures or modified the existing index structures that proved to be efficient and robust in moving objects. Most of them focus on using a single index and its related query types. Some propose a hybrid tree that combines at least two index structures and compares to single index structure. Indexing is a special data structure that allows having quick access to data or objects systematically. Indexing is usually used for users to display results that

match the desired user criteria. Normally, indexing technology has been used for moving data in recent years. It is used as an optimization technique in real time that manages and stores moving locations and query based on them. The objects and their index structures are described in figure 2.3.



**Figure 2.3 Objects and Index Structure**

### 2.2.2.1 Indexing Trajectories of Moving Objects

The index structure has two types of category. These are (1) the histories or trajectories based index structure and (2) the current and near the future position of the index structure. In the first type, the movement of an object in a k-dimensional space is transformed into a (k+1) dimensional trajectory space that has a time constraint. Some examples of these types are the Trajectory-Bundle tree (TB-tree) and Spatio-Temporal R-tree (STR-tree) which are proposed [71]. The results for both these structures are adequately supportive of moving objects and better than the traditional index type especially for queries related to moving object trajectories. Tao and Papadias [87] proposed the Multi-version 3D R-tree (MV3R-tree) that combines multi-version B-trees and 3D-Rtrees.

In the second type of category, the common approach is linear function based moving object location. In this type, the database update occurs when the mobile object or moving object changes its deviation or speed or when the parameters of the linear function change. The time-parameterized R-tree (TPR-tree) is proposed by modifying R tree Saltenis [76]. In this scheme, the moving object position is described as a reference position with the corresponding velocity vector. The TPRtree focus on both the positions and velocities of the moving points when the nodes are

splitted. An efficient index tree structure using partition trees is proposed by Tayeb [48]. PMR-Quadtree for indexing moving objects and discussed the issue of index structure for moving objects based on the query of current or near future positions [89].

Agarwal [4] discussed the main concepts of creating various schemes and proposed an efficient index-based scheme to process nearest-neighbor queries approximately and appropriately. With the extending of web service technologies, the storing and processing of moving objects based on index structure dramatically increase in a wide variety of applications. This index structure includes monitoring the continuous changes of moving objects and maintaining their accurate locations in moving object database. The new types of spatiotemporal queries are presented [99], as well as algorithms to process such queries efficiently.

Besides, the two access methods namely the Spatio-Temporal R-tree (STR-tree) which is basically come from R-tree that traces the identity of trajectory and the Trajectory-Bundle tree (TB-tree) that focus on the trajectories are introduced allowing to query moving data.

#### **2.2.2.2 Nature of Index Structures and Their Issues**

Various spatial index structures can be categorized into three main groups: the index structure group of the Grid file, the R tree [39] and the B+tree variant based on the Grid file. Normally, most of current index based algorithms and structures are built based on one of the index structure of above-mentioned groups. For example, the Grid index structure is the best index structure for main memory resident with less space and it can easily manage and process moving object queries especially supports query performance.

But, Grid index has two limitations based on data size and distribution. First, it is very fast for query performance since it is a memory-resident index structure but it cannot process for a large dataset of moving objects. This is because it has the size of memory thus it cannot handle when the size of the moving objects or static objects exceeds the memory limit. Second, the memory size and data distribution are closely related so that the grid index performance for data distribution may degrade with the low resolution of cell size. Besides, most of the cells include a very small number of moving objects in cell-based index blocks thus the storage utilization may become too slow. Moreover, the query processing and storage cost may be complex if the

resolution is too high. When the queries are overlapped in a large cell, the required objects for each query in the cell will be evaluated and retrieved. It is difficult to decide the optimal solution of each query with the great challenge for Grid-based resolution.

The second and third index structure groups are R tree and B+ tree. Both of them are disk-resident index structure. Today, many novel index structures based on R-tree variant index structures have been discussed and proposed such as PR tree, the previous R tree that always produces result by a window query [8], R<sup>\*</sup>-tree [15] that holds for various types of queries and executions for both points and rectangles and TPR<sup>\*</sup> tree which collects the unique features of moving objects from a set of data mining algorithms [88]. R tree suffers update performance for fast moving objects but it is suitable for real-time or continuous query processing. This upgraded R trees such as TPR- tree and other disk resident index structures have the same ability of R tree so that the update cost may high due to moving location update rates. The more variety of moving objects, the more frequent splitting and adding of moving nodes in the index may occur. Therefore, several research solutions and proposals are proposed to overcome the difficulty of update problems over moving objects. The disk resident index structures like the R tree are combined with the advantages of Grid file index structure which is a memory resident indexing technique. Likewise, the upgraded R tree like TPR tree incorporates with the other memory resident methods that support query performance.

Sometimes, the Grid file that has a high ability of query performance cooperates together with B+tree. The corresponding examples are Bx-tree [41], ST2B-tree [20] and Bdual-tree [98]. All of these index tree structures are firstly combined with a grid for index space then its space is divided into the same-sized cells in the grid. Moreover, the cell-based inverted index structure such as the Grid file is not used directly so that the cells are sorted based on the theory of an ordering method such as one-dimensional index method called Hilbert curve or Z-order curve [41]. In this method, the data nodes in the index structures become the cells and managed by one dimension index structure instead of using multidimensional index structures. The advantages of this index are reducing the insertion and deletion cost. The ST2B-tree is built based on the various density distributions so that it is suitable for high dimensional data values but it still limits from the suffering to decide the appropriate resolution of the grid in this index tree.

Since some traditional spatial index structures are unbalancing, they are not enough for storing moving object positions. Using an unbalance structure leads to traversing an uncertainty and an uncountable number of moving nodes and indexing pointers. Presorting before tree structure is one of the ways of construction a balanced tree. The objective of a moving index structure is both query and update to be running smoothly. Thus, moving object databases are created along with relevant structure query language for accessing [38]. The index structure that can quickly access the query is not easy to take the appropriate update. In other words, it is rarely to be perfect index structure for both updating and querying.

Therefore, moving object types and behaviors are classified and stored them separately in the indexes [83]. Thus, the duty of the update consistently undertakes at the client side and efficient and fast query processing is done on the server side by proposed index structure. In recent years, there has been a survey on indexing algorithms together with the activities of moving object databases to fast and efficient processes [73]. Normally, indexing has its own relevant structure that needs time to build it. It is saved by building a structure only for points that need to receive information and it can be used in the multicast notification. Nowadays, indexing has various types and fieldworks to apply in the real world. Some types are good in querying and some are in updating.

### **2.2.2.3 Indexing Current and Future Movement**

There have been recent research works that use indexing current and future movement of moving objects. These works can be classified into two groups: indexing the moving objects location and approximating the moving movements based on functions. The static object requires no update where fast moving object varies locations over time. Although the static objects do not need to update in the database, the moving objects regularly update their locations to the database. It causes high database workload especially for a large number of moving objects.

To overcome this issue completely, Y.Lee [55] proposed a technique that uses the hash function to hold moving objects in the bucket. The bucket data for each object is saved together with object location. An update occurs when the moving objects are out of the current bucket (i.e. objects move away from the original position). All of these update positions are triggered and locations are updated in the database. The result of this method saves time for update frequency and speed up the

process between moving objects and the database server. But, the accuracy of this method is not good for every query. This is because it is hard to distinguish the objects which are in the query range or not after passing each of the range in the bucket. Then, Kwon [51] introduced the Lazy Update R-tree (LUR-tree) which has less update concept. This index tree comes from the R tree so that it has a bounding rectangle called minimum bounding rectangle (MBR) for each query. In this tree, the deletion of every object is reduced and the objects are deleted only when the objects move out of the MBR. As a result, this proposed index may overlap with the enlargement of MBRs severely and degrade query performance with traversing the sub-trees repeatedly.

The next research work that provides to improve the update process is discussed by M.Lee [56]. The issue of the traditional R-tree is introduced and a new update strategy is proposed. When the R tree is updated, searching starts from a top-down hierarchy so that it takes more time than usual. In this R tree strategy, all of the updates are started from the root. The main idea of a new update strategy is to process the update from the bottom of the index structure. The initial idea is placing the incoming moving objects by enlarging the MBR and storing them in the leaf of this index tree. The additional Direct Access Table (DAT) is taken as an auxiliary data table structure. All of the index nodes are kept as a summary and it can be accessed directly from the R tree.

Christian S. Jensen proposed B+ tree-based index structure for moving objects. The locations of moving-object are stored together with the update time stamp as vectors. Since it is based on the B+ tree, it needs to linearize multidimensional space to one-dimensional space of moving object positions. Therefore, a novel linearization technique is also introduced for two-dimensional moving locations. The result showed the reduce insertion and deletion cost. They only focus on the current and histories of moving objects and neglect about the future position of moving objects. The proposed algorithms for both range query and nearest neighbor query provide for current or near the future position of moving objects [41].

Yuni Xia introduced a novel index structure, Q+Rtree which is the combination of Quadtree and R tree. It is a hybrid index structure that divides the moving objects into two types: fast moving objects and quasi-static objects. Each type of objects is separately stored in the R tree and Quadtree. The fast objects require more update time than quasi-static objects so that they are stored in Quadtree. The

quasi-static objects are collected and processed in the R tree. This index structure only focuses on the current positions and history of moving trajectories. In summary, this novel index structure supports not only query performance but also update performance [48].

### **2.2.3 Querying Moving Objects**

Moving objects databases need to incorporate the location of moving objects and their frequent updates that lead to support efficient queries such as nearest query, range query processing. Existing database management system is insufficient to maintain continuously changing data. Thus, there are two disadvantages to represent moving objects in the database.

1. Database management system cannot instantly keep the mobile frequent updates.
2. Continuous updates lead to a serious connection bandwidth overhead.

Therefore, the changing positions of objects are tracked to collect the continuous movements in the moving object database. The indexing techniques for moving objects develop and offer various query services based on moving positions. The point queries, exact match queries, KNN queries, and range queries are common type queries in moving objects databases. There are several queries that have been studied, e.g. simple queries, compound queries, query string queries, location queries, on-analytic queries, and special queries.

#### **2.2.3.1 Query Types**

An index structure is used to improve the speed of insertion and retrieving data in the database. It usually arranges nodes or data without searching every tuple in a database table. There are many forms of index structures that provide different types of queries. Some of the query types are the following.

1. Simple Queries: Accept words and phrases as input text, and find similar words and phrases in the text bodies in the index structure. Additionally, the highest accuracy in similar words is specified and all of the queries are searched together with respective query priorities
2. Compound Queries: Receive various queries at the same time, and give the result-sets with the conjunction of results and a disjunction result.

3. Range Queries: Search the data and objects within the range by queries and reply the results that contain values in the range.
4. Query String Queries: Input the desired query for each user, return available query results.
5. Location Queries: Accept location of the object with latitude and longitude coordinate from location provider and return accurate location as a document.
6. Non-Analytic Queries: Only produce the exact match of words and phrases without using analytical concept or method.
7. Special Queries: Produce the documents that are in the index or not according to the desired query.

### **2.2.3.2 Range Monitoring Queries**

Based on the types of moving objects, such as geographically distributed moving objects, there is a special index structure that allows generating range monitoring queries over content-match and group-aware queries [44]. Some of the indexes preserve only for a specific type such as the trajectory of moving objects [47]. The common index for different types of query special is a grid structure. With the rapid distribution of spatial moving objects, a Distributed Grid Index (DGI) was proposed [59] that accurately interrogate the objects' location by unique object's ID. Sometimes, the spatial index structure such as the R tree used the partition of grid index that became less overlap and gained information [30]. Like the grid structure, a Quadtree index offered [80] better performance regardless of the amount of moving objects. However, using a simple, uniform grid index was basically supported to query processing and it is weak in update performance [78].

In order to better help mobile objects' query was proposed [72]. Some survey paper thought uncertain data or object index structures were not applicable to all types of query processing [43]. As [29] discussed that some tree-based algorithms such as R trees are not suitable for high-dimensional data in real-life applications. Instead, they proposed Grid-index algorithm (GIR) that offers reverse rank queries with a little memory cost. On the other hand, [102] discussed that it is hard to get an optimal resolution of index based on the grid and also R tree with the skewed mobile objects. As a result, a qualified index structure depended on the structure of index, types of moving objects, and queries.

Some index for moving objects are based on queries regardless of moving positions, speed or movement nature [97]. A suitable way of indexing and querying to moving objects is having a good design, such as a spatio-temporal indexing using key-value store [32]. Due to the variation of moving objects, the concurrency problem may be occurred in their indexing so that distributed index structure that exploits in multiple machines [37].

### **2.3 Synthetic Dataset Generation**

This section consults the reason of generating synthetic dataset over moving objects and applied for works in various areas. It also discusses advantages and issues of using synthetic moving object dataset.

#### **2.3.1 Synthetic Dataset for Moving Objects**

Synthetic datasets are used to create virtual mobile locations when the thousands of mobile locations are not easy to get in reality. Mobile Objects have their own different behaviors such as moving types, bearing angles, etc. Therefore, synthetic mobile location dataset is generated based on three different behaviors. To generate virtual locations that seem realistic, random locations are created firstly and update all of them based on their behaviors of moving types [10].

Researches about generating synthetic dataset and its usage are already emerged and applied in different areas. Actually, synthetic dataset depends on the application areas. A synthetic generator for spatiotemporal data is proposed and applied in many different areas. It can be used to validate and verify data based on mining algorithms. This dataset is suitable for telecommunication companies in which positioning or location data is tracked and kept along the mobile phone cellular network. Indeed, the moving positioning data or its movement patterns are collected around the moving mobile phones in a cellular covered region. The advantage of using synthetic spatiotemporal data is free from location privacy and confidentially. Thus, it can be used for other service areas such as finding network bandwidth optimization, traffic monitoring, etc. Sometimes, the synthetic dataset is generated by tools [67] [20]. It is used to classify the optimization for ant colonies behavior based algorithm. It is aimed to create virtual dataset when real dataset is not possible to access or obtain for applications. Generating semantics-based trajectories data (GSTD) tool supports to produce virtual moving objects by virtual trajectories. For

this tool, a parameter called GSTD is added to accommodate various object variations.

It is hard to get the actual moving objects; Y.Xia and S.Prabhakar demonstrated creating of a virtual dataset using City Simulator which is introduced at IBM Almaden. This simulator is flexibility that enables to generate spatial data or location data dynamically. It can generate virtual motion up to one million. It is built as a three-dimensional index model and designed to create any types of moving location instantly. Thus, it is used for comparison of database algorithms or index structures that have a large number of the dataset. It is scalable and the region in the map can be added as an input into the City Simulator.

The synthetic dataset is generator only for the specific application such as transportation networks [62]. In this dataset, data is developed and generated by simulating moving object trajectories.

The experimental results based on this data generator are very good because of the well-defined datasets. The data types and functions can be analyzed in this network and the issues and weaknesses can be modified after checking the results. In this data generator, virtual cars or vehicles are driving on the road network and capture their speeds and positions for a given period of time. Then, all of these positions and speeds are stored in the database.

Some synthetic dataset generators are used for generating object patterns and velocities that are presented [77]. This generator allows creating spatiotemporal pattern datasets that have necessary characteristics such as cardinality and number of patterns, acceleration and velocities, spatial areas and their lifetimes. The results of dataset include different feature types of spatiotemporal objects corresponding with their spatial regions.

### **2.3.2 Advantages and Issues of Using Synthetic Data**

Nowadays, synthetic data are widely used in a variety of real world applications such as natural language processing, moving object databases, and traffic monitoring system. It has also been applied for machine learning areas. The main reasons of using synthetic data instead of real-world data are privacy, flexibility and testing cost. Many generation models produce synthetic dataset that depend on the applications. This especially applies to the motion detection of people in which real data can be not only time-consuming but also high cost. With synthetic dataset, it can

produce new motion data easily once the model is set up and the generated data avoid copyright infringement. Using synthetic data has some great advantages, too. First, it might be useful for visualization purposes and to test the scalability as well as the robustness of new algorithms. Second, the resulting indicators can be shared broadly, often as open data [61].

However, it requires a reliable framework and procedure that seem to be relevant for the specific application. Besides, the generated output from the synthetic dataset should be user-friendly and well-understand data in reality. Sometimes, synthetic models only replicate specific properties of the data. They cannot match a given dataset perfectly; only simulate general trends.

## **2.4 Notification Process**

This section explains notification techniques and types of notification together with literature reviews on various aspects of disaster notification systems. The importance of notification is also discussed along with general components and key capabilities of a complete notification system. This process is mostly associated to the location-based services. It is mainly used for emergency information such as disaster notification, fire alarm warning, gas-related fire damage etc.

### **2.4.1 Disaster Notification Techniques**

A disaster is likely to be regarded as an unpredictable and unaware condition that occurs accidentally by high consequences. It causes uncertainty conditions in a short period of time. Adequate and prior disaster information or notification saves lives and properties. It significantly reduces before and after damages of disaster. Nowadays, people use mobile phones every day and mobile applications for disaster notification are very suitable. Advancements in mobile and wireless computing technologies, all of the notifications can be sent by all mobile users in one minute. In summary, notification application for the mobile phone is one of the optimal tasks for notifying emergency and crises and it will send the message to the right people at the right moment. Many types of research have been described timely disaster prevention and mobile update policies. Most papers focus on their nations. Some discuss not only prevention but also evacuation for mobile users.

Prof. Harish Barapatre made an application to support rescue and relief activities in disaster-affected areas [14]. This application is used for sending the

location wherever disaster has taken place. It is also used by the user who can provide help in affected areas. It works with two buttons, I NEED HELP and I WANT TO HELP. So this makes interaction between the victim who is facing disaster and the volunteer who desire to help the victim. Lack of details on Google map will be the main problem.

S.Kumar and Veeramani proposed their own algorithm “Extended Polygon Match Algorithm using Quadtree” to find whether mobile is within a defined polygon shaped area using their GPS coordinates [50]. Their objectives are to build a prototype system using Android software to send alerts to mobile devices within the defined geographical area and to fix the search area and the accurate location is easy using GPS. The advantage of their system is that the disaster target region is perfectly contained. But the other regions also contain when taking the corner points of the polygon in the map. They have planned to implement this concept in all the mobiles which are equipped with GPS.

#### **2.4.2 Types of Notification**

There are three main categories of applications for notifications: (1) user-generated, (2) context generated, and (3) system generated.

1. User-generated notifications: In these notifications, human-created context and send to other humans by an application. It usually uses for any types of notification. Sometimes, it may include private context that can only know between sender and receiver. It intends to use and send message or notification for specific people from the sender. The mobile messaging system is mostly used in this type of notification.
2. Context-generated notifications: These are sent by a mobile application to the users that have to see permission. In this notification, the user can create based on contacts, locations or time that acts as an automatic message system. These notifications are fast and grow because they can send one message to all users simultaneously based on the category of notifications. Location-based notifications currently use and dominate for all users in this category.
3. System-generated notifications: These notifications are produced by an application or sometimes they are started from the service provider. This

type of system-generated notification can usually be called broadcasting or multicasting based on re-engagement.

### **2.4.3 Categories of Location Based Alerting System**

#### **1. Advance Short Message Service (Advanced SMS)**

Advanced SMS is used to send all mobile users who are subscribed in a specified area, including foreign tourists, inbound visitors and people travelling to other countries. Advanced SMS uses a powerful congestion control techniques to regulate the effects of emergency reminders on telecommunication networks.

#### **2. Location Based Application Alert**

Location based notifications or alert can be sent to all people who have the notification app installed on their mobile phones, based on their current locations. Minimal information about the device (which has registration ID) is sent to application server.

#### **3. Cell Broadcast**

Cell Broadcast depends on geo-referenced unidirectional text messages to all mobile phones within a targeted area. This will reach all mobile handsets that have been configured to receive cell broadcast messages.

### **2.4.4 Importance of Notification**

A notification is a message which includes an icon or symbol that sends when an application wants the mobile user to pay attention. The importance of notification is summarized in the following.

- It appears on users' device lock screens.
- It gets user's attention very high with its effectiveness.
- It supports easy communication between mobile users.
- It provides the users with special opportunities which can be beneficial for customer loyalty.
- It can run to rescue and remind the users for emergency events.
- It saves time and money.
- It uses mobile technologies and applies in many areas.
- It supports to combine free online services.

#### 2.4.5 General Components and Capabilities of Notification

A notification is a message that displays message information such as a reminder or timely message outside of user interface application. A notification message can be distributed from the user application to single devices, groups of devices, or subscribers. There is a protocol that provides a reliable and battery-efficient connection between the application server and devices called GCM http server protocol. Y.S.Yilmaz evaluated arrival times to elaborate how GCM performs (timing performance of GCM), Poisson distribution to the number of devices per time and conducted a chi-square goodness-of-fit test on their models [77]. They point out GCM servers on client device has reliable connection between client and server. But it does not guarantee a timely message arrival. A notification implementation includes two main components and three key capabilities for sending and receiving:

The following three components are necessary for a notification system.

1. A trusted infrastructure that has firebase cloud functions and abilities.
2. An application server that has fully web service technology to send a message.
3. An iOS, Windows, Android application for users to receive the message.

The key capabilities and their descriptions are explained with the following table 2.1.

**Table 2.1 Key Capabilities of Notifications**

<b>Key Capability</b>	<b>Definition</b>
Initiate message at application server	Starts the message process when the user makes self-query or emergence query from the application server. A message is sent to FCM firstly, it broadcasts or multicasts the message and sends to client applications.
Large-scale message targeting	Distribute messages within the desired start time and end time to the client applications. Sometimes, target to the single device, registered mobile devices or topic subscriber devices.
Send messages to the applications	Send notification messages to the application from third party server via GCM communication connection.

## **2.5 Summary**

The chapter has four review collections: location-based services (LBS), moving object structure, synthetic moving object and notification system reviews. Firstly, the chapter introduces the challenges of location-based services (LBS) and discusses location providers with location update strategies. It also describes the corresponding literature review and previous work of using update policies. Then, moving object index structures and synthetic moving object dataset generation are discussed with related works.

Finally, the chapter describes the notification system reviews. A complete notification system requires many tasks such as maintaining and monitoring of user current locations and focused range search querying in the imminent disaster area. Most disaster notification found in literature focuses on the Android-based platform with GPS and Google map. However, it is hard to get the definite area by Google map during disaster occurs. Also, only using the GPS for current locations is not enough to get location especially in indoor areas. These issues have been solved by using Google API in this system. The variety of references are illustrated such as notification system, moving object index structures and location update policies tie together background theory, technique, and algorithm.

## **CHAPTER 3**

### **THEORETICAL BACKGROUND**

#### **3.1 Trends in Mobile Technologies**

Mobile has been significantly improved like a king in digital marketing today. With the growth of the mobile, its technologies are widely applied to business areas, government and public environment. Mobile information and communication technology determines the society and behaviors since it represents an element of individual quality and a way of communicating and doing business. The potential provided by hardware can be fully used only by adapting to the users' requirements by improving the software side. Today's development of the information and communication technology has to be created with the focus on the humans and with technology transparent for the user. Regarding the mobile information and communication technology, the problem has traditionally been that the mobile applications focused too much on the technology, which means that applications had been developed for certain purposes or for special technologies. The new standpoint has been led by the idea that the application has to provide the users with what they want, anywhere they want it and in the best possible way. The user has to be able to employ the mobile device and software based on the installed applications in different environments and independent of the environment.

Developing an application that may make everyday life easier is a sufficient motive to turn an idea into reality. Mobile devices are tools used to access LBS services, send requests and correct results. Such devices can be Personal Navigation Devices (PNDs), Personal Digital Assistant (PDA), portable computers, cellular phones, etc. With the increased use of the internet over the year, the whole smartphone industry has come a long way. More and more apps are designed to allow cloud interaction paving the way for the minimal requirement of internal memory. There has been a significant improvement in mobiles which are cloud-based techniques in the modern era. The role of mobile technologies is dramatically increased and used in important application areas such as emergency response and rescue activities, prior disaster notification functions and finding nearest safe zones by tracking user current location from a mobile phone. The result data is stored on the cloud which renders the users free from any storage issues. The android is an

operating system that is developed by Google. The operating system is designed basically for the touch screen mobile devices and it is based on a simple manipulation. It uses simple touch gestures that relates to the real world situation. The Android is the first and only open source operating system; that means it has the ability to be ported to any cell phone. Android has hundreds of partners releasing new phones every year. Likewise, there are multiple Android options in mid-range or low-end price brackets. There is high demand for cloud-based mobile applications which can easily fetch the data from the cloud at any point in time.

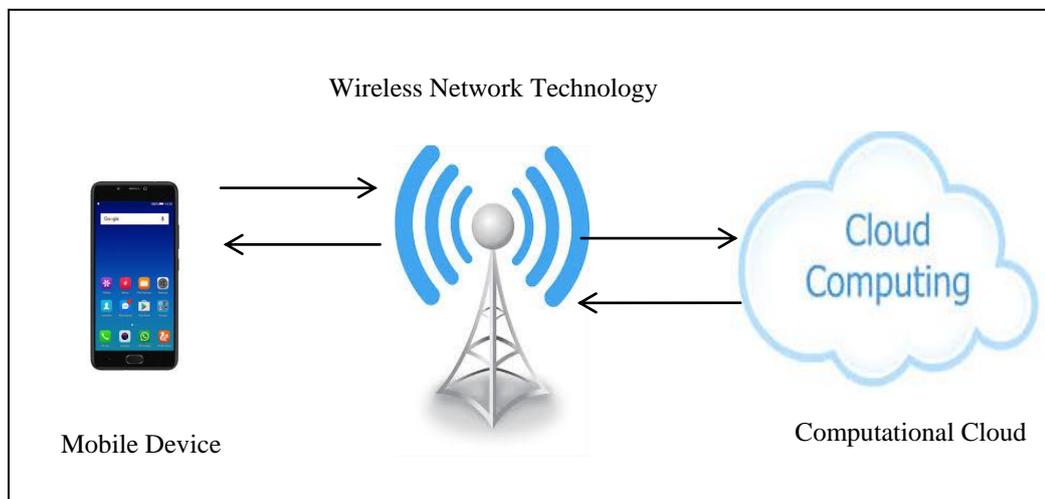
### **3.2 Cloud for Mobile**

Cloud computing is the way of accessing user desired applications from the internet. It includes online data storage, infrastructures, and cloud-based applications. There are four deployment models in cloud computing. They are public cloud, private cloud, hybrid cloud, and community cloud. The ease of maintenance, low-cost storage and accessing the data at any time and any place are major advantages of cloud computing. Mobile devices receive personal data from user self-query, automatic query via service provider and all of these data are collected in a short period of time. The personal data includes valuable knowledge and information for users. But, mobile access introduces many problems such as duplication to make data easily accessible, access to desired data, security of data, and AI techniques for quick and effective access to data. The rich mobile technology integrates cloud facilities so that it becomes elastic resources of powerful mobile cloud network technology that has fully storage services, functionalities, and mobility such as Evernote, DropBox, Google Drive, Google Cloud Server. Therefore, there has been emerged mobile cloud computing that has cloud computing facilities, mobile computing technologies, and wireless communication networks. It is also the next generation of the portable cloud computing environment. The combination of cloud-based technologies and persistent networks provide benefits for mobile users, cloud-based applications and network operators. Therefore, mobile cloud computing (MCC) has a complete mobile infrastructure where both the data storage and the data processing are provided by cloud infrastructure which is outside of the mobile device. Therefore, these applications use cloud-based data storage and computing power. Alternatively, mobile cloud computing can be defined as a cooperation of mobile devices, web service, network and cloud computing [25] [58]. Today, it is the most popular technology in a

wide variety of mobile cloud-based applications and mobile users access these applications and services on the Internet.

The main advantages of mobile cloud computing are saved battery life with the help of cloud storage, improve storage capacity that supports cloud-based database and process high power for accessing and querying required information and reliable for mobile data backup on the cloud infrastructure. Besides, the mobility feature and save time for searching information is also the key advantage of mobile cloud computing. Therefore, there are many practical applications that use mobile cloud computing such as healthcare mobile services, mobile learning, e-commerce on the mobile, mobile gaming etc.

Some issues and challenges of mobile cloud technology are service cost on the cloud, offloading from mobile to cloud, security, privacy for mobile users, network traffic cost between mobile and cloud. The relationship between mobile device and computational cloud is described in figure 3.1.



**Figure 3.1 Mobile Cloud Computing**

### **3.2.1 Context-aware Services**

It offers relevant services to users based on the user current position or desired query by using the location and context of mobile users. Context refers to real-time behaviors that have moving characteristics and positioning technologies. The samples of a context-aware service are real-time traffic monitoring with updating, vehicle tracking, route planning, and motion detection. In recent years, context-aware techniques have been widely promoted and applied in different areas of application. It still has some limitations in a small scale or single organizational environments due to

the lack of well-prepared rules, interfaces, technologies, and models for exchanging context data. In large scale or multi-organizational environments, web services protocols and technologies support and enable to exchange context information. Therefore, context-aware techniques are more powerful with the combination of web service facilities to utilize a variety of context information. As a result, context-aware behaviors and operations are adapted to dynamic changes. Several challenges are discussed by researchers that are more specific to the development of context-aware services [65].

- (1) Service adaptation: The client should be adapted by deploying on different hardware and software platforms ranging from desktop applications to IoT based mobile devices. This adaptation should be available automatically configurable services with less modification and manual settings by the user.
- (2) Context-awareness: It has the supporting infrastructure initiate from the user personal device. The clients should be aware and relocated as soon as they move out of the room or place. The immediate response for the context-awareness result is stored in the preferable device or other resources.
- (3) Resources discovery: The relocation service needs to discover the storage devices and resources through the communication network and check whether they provide the required facilities, abilities and capabilities.
- (4) Service state capture and transfer: The status of the clients should be kept while changing from one service to another especially for a seamless relocation. It needs to ensure the contact list of each user, their current conversation, activities, etc.

The common flow of context-aware services based on web service techniques are required to provide and develop with the following questions [91].

- (1) Context information and context representations: which techniques are used for modeling context information in Web services? They help us understand the context information exchanged among applications, context-aware services and supporting components.
- (2) Context sensor techniques: how context information is measured and sensed? They help us understand how context information is measured and how sensors are implemented.

- (3) Context storage techniques: how context information is stored and how the information is accessed from its storage? They are also related to context information and distribution techniques.
- (4) Context distribution techniques: how context information is distributed and propagated to different relevant supporting components? How applications and services can retrieve context information?
- (5) Security and privacy techniques: how context information is protected? Which authentication and authorization mechanisms are used for context information? Which techniques are used to ensure the privacy in connection to context sharing?
- (6) Context adaptation techniques: how context information is actually used in Web services?

### **3.2.2 Location Based Services (LBSs)**

There are various terms to define location-based services [24] defines and LBSs as services that integrate the location or position of a mobile device with other information so as to provide added value to a user. Mobile networks are usually controlled and maintained by the operators who provide connections for the mobile users and are responsible for the data and voice transfer.

The positioning of components is usually necessary in LBS applications in order to determine the location of the user's mobile device. In the majority of current LBS services the user is not required to enter the location manually, nor the input of post codes or street names. Instead, the position of the user device can be obtained by using the positioning technology, such as satellite positioning, positioning by mobile network, WLAN stations or radio connections.

One of the well-known examples of LBSs is GPS. It is freely available for all location service system around the world. GPS users belong to either military or civil environments. It lets them determine, free of direct charge, their position anywhere in the world and they use the information for several combined applications. Depending on the ensured accuracy, the GPS provides two levels of services: the Standard Positioning Service (SPS) and the Precise Positioning Service (PPS). Today, with the currently available service provides such as GPS, cellular network and WI-FI, the locations of objects include an important foundation for a variety of application areas. There are six types of LBS services.

- assistance,
- orientation,
- information,
- advertising,
- tracking,
- charging.

GPS is a complex system which combines three segments – space, control and user segment. Such distinction of segments emphasizes the main objective of the combined segments: to create a functional system that at a global level makes people aware of the possibility and potential of the services based on navigation. GPS uses the satellite constellation, where each of the satellites transmits the signal in the range which encompasses the message navigation. In the LBSs system, there are two types of services. First, user queries the desired information to the third party server (which is called pull type). Second, the user can get notification or alert by the server automatically (which is called push type). All of these two types are adequately supportive to mobile users especially for the self-querying and automatic messaging services. Generally, Location Based Services can be classified by Reactive LBSs and Proactive LBSs.

### **3.2.2.1 Reactive LBSs**

Reactive LBSs are always explicitly initiated by the client application. In this service, the activity of LBS and the client application is generally as follows: the client application query firstly to the desired service and sends a query request, either via a mobile-based application or a PC. The service provider or application server then receives for required information or query whereupon the service collects user's location coordinate, processes it, and sends back the location-based query result to the client application. This request/ respond functions will be active several times until information is requested from the client application repeatedly. Thus, reactive LBSs are regarded as synchronous request/respond activities between the client application and the application server.

### **3.2.2.2 Proactive LBSs**

Proactive LBSs start as soon as a predefined location event is automatically activated, for example, if a user reaches in certain nearest area of interest, the user will

receive related information or notification via user mobile application without making anything. In fact, proactive LBSs are not explicitly activated by the user, but the interaction between user and service provider is marked as an asynchronous. The different factor between proactive and reactive LBSs is that the user is only tracked and located once; proactive LBSs require finding user location frequently in order to get location coordinate from the user.

### **3.2.3 Categories of Location Update Policies**

Location management for mobile devices is a necessary and important factor in a mobile computing system. In a mobile network, the coverage area is formed of cells. Therefore, the mobile operators usually try to reduce the size of cellular cells for each user. To reduce cell size, location management is required and it becomes a fundamental task. The main goal of this management policy is to provide a fast and efficient update for any types of mobile. Due to different features and functions in mobile devices, it is also research challenge in the current as well as next-generation wireless networks. Various location update strategies for location management are available in mobile computing. There are two common strategies created by very simple functions. They have always update strategy and never update strategy.

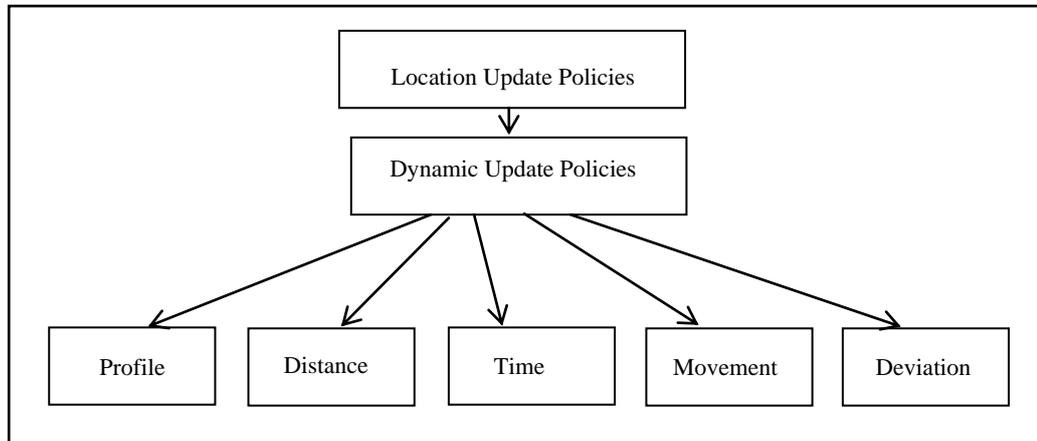
#### **3.2.3.1 Always Update Strategy**

In this type of strategy, an update occurs whenever each mobile location enters a new cell. The main benefit of this strategy is service provider or server always available the current location of each user simultaneously. Therefore, repeated searching processes are not required whenever there is a call. It would update the location whenever the mobile receives location from the service provider in every cell movement. It is totally good at one side but it creates a problem on the other side. This is because it requires more resources to get current and update location for every movement of cell update. Therefore, location update cost would be very high and the paging cost would be zero. This kind of strategy is suitable and valuable when the user is less mobility and the cell size is comparatively very large.

#### **3.2.3.2 Never Update Strategy**

The process of this update strategy is opposite to the always update strategy. In this strategy, a location update would never be performed at the server. Therefore,

the location update cost would be zero since it has no update request at the server. The main benefit of this strategy is no location update cost on one side but paging cost would be more on the other side. This kind of strategy is very much applicable and valuable if the cell size is comparatively very small and user mobility based on the behavior of the mobile user is very high.



**Figure 3.2 Dynamic Location Update Policies in Various Locations**

The other strategies in a mobile computing system that use threshold values and specific policies are distance based update strategy, time-based update strategy, movement-based update strategy, and profile based update strategy [24]. The various update policies for dynamic objects are shown in figure 3.2.

### 3.2.3.3 Distance-based Location Update

It is a simple and efficient strategy for location update. In this strategy, the mobile base station keeps each mobile terminal for distance in the number of cells by tracking moving mobile position. It takes the last update and current location. Whenever the terminal travels the number of cells starting from a predefined distance and reaches the cell size distance threshold  $D$ , at that time an update occurs. When this policy is used, the threshold value management is needed to be done. The distance counter set to 0 in the initial cell [70] [101]. Now, when the mobile object moves and crosses the cell boundary, the distance counter would increase by 1 at each time. After reaching the counter cross value greater than  $D$ , a location update occurs at the server or service provider. This strategy is suitable for users who move less, quasi-static moving objects and the users who move within a specific distance that is less than distance threshold value  $D$ . In that case, a few updates would occur and

exact location of the user would receive. The latest location is only recorded so that both the paging cost and the update cost would be very low. The main drawback of distance-based location update policy is that unnecessary location update may occur at the server when the mobile user crosses the boundary very frequently.

#### **3.2.3.4 Time-based Location Update**

This is also a simple and automatic location update strategy with the predefined timer. Here the mobile base station updates the user current location after reaching a particular time period called  $T$ . This strategy is comparatively easy to use and available because each mobile station requires keeping its internal clock only [40]. The other benefit is the value of timer to control and manage on the history of call arrival pattern in each cell or user mobility pattern. There is one more benefit due to the easy processing and nature of periodic signaling. The network is available in every mobile that the mobile terminal is powered-off or outside the coverage area if it does not perform a location updates at a predefined scheduled time.

It only focuses on the timer and it does not consider any mobile behavior or movement. Therefore, the user is stationary or no moving, at that time the unnecessary update would occur and increase the location update cost as the main drawback. Moreover, the uncertainty of mobile users' location cannot be bound when a call arrives. As a result, the searching process cannot be controlled and limited to a set of cells. It does not dependent on Location Areas (LA) and it has lower paging cost because location update definitely occurs at time  $T$ .

#### **3.2.3.5 Movement-based Location Update**

In this strategy, it needs to count the number of cell movement. Therefore, the mobile base station needs to track and keep the number of cell movements for mobile users including the number of cell cross boundary. Here, the movement counter is managed, it initially set to zero. It has an increment by 1 for each and every time as soon as the user crosses the cell boundary. Now, when the movement counter becomes greater than the predefined movement threshold called  $M$ , at that time location update is done at the server. There is the main drawback of this kind of location update policy or strategy. If the user travels around the boundary repeatedly, undesired or unnecessary updates may happen [2].

### 3.2.3.6 Profile-based Location Update

The Profile Based Location Update scheme has been proposed in [85] [68]. In this scheme, each user's profile would be maintained and from that profile, the location of the user would be traced out. The main idea behind this strategy is that the mobility pattern of the majority of subscribers could be easily predicted. This type of strategy would be useful when the user is working in the same geographical area for maximum hours of his / her daily routines. To find out the probability of the user's profile location long term statistical data would be useful. To create the profile of each user the following operations could be performed: For each time period ( $t_i, t_j$ ), the system maintains a list of location areas,  $[(a_1, p_1), (a_2, p_2) \dots (a_k, p_k)]$ . Here  $A_f$  is the location area, and  $P_f$  is the probability that the subscriber is located in  $A_f$ . It is assumed that the location areas are ordered by the probability from the highest to the lowest, that is,  $p_1 > p_2 > \dots > p_k$ .

If the subscriber moves within the recorded location areas,  $a_1, a_2, \dots, a_k$ , during the corresponding period ( $t_i, t_j$ ), the subscriber does not need to perform a location update, otherwise, the subscriber reports its current location, and the system will track the subscriber as in the classical location area strategy. Therefore location updates could be significantly reduced. In this type of scheme, the main benefit is the system knows the user's location based on its profile and if the user is in that location area only at that time the location is not required to update. Sometimes it may happen that user changes his / her daily routine due to some circumstances, at that time the service provider would need to do paging to search the latest location of the user.

### 3.2.3.7 Deviation-based Location Update

The deviation-based policy is mostly used for vehicle-based updating strategy so that it mainly concerns with moving object deviation or motion on a traffic road or road network on the map [63]. Since it usually has location update, the application server tracks moving object position related to all location attributes starting from the beginning of the step. There are various classes in moving object types (e.g. vehicles, trucks and cars). Each type of class has a specific velocity and speed. When the moving object in a specific class with an update the database at time sub-attributes the database location can be expected at any time  $t$ . Since the moving object does not travel at exact speed so that the actual location of the moving object deviates from its expected location. When the deviation reaches to a given threshold the moving object

will update the database with the actual location. Thus, in response to a query entered at the time that retrieves m location, the DBMS returns the location of m within a circle with radius equals the threshold value centered at a point of the last location update.

### **3.2.3.8 Hybrid Location Update**

This strategy is the combination of at least two location update strategies to promote the ability to update policy in mobile devices and server. The objective of this strategy is to be a valuable update policy that can switch between different policies and functions. Each of the update policies has its own advantage and limitation so that hybrid update policy overcomes each of them by combining two or three update policies in moving mobile environments. All of these updates are done after reaching the predefined threshold values or timer. The two common hybrid update policies are combining time-movement based strategy and distance-movement based strategy.

### **3.2.4 Android Operating System in Mobile Technology**

Android's mobile operating system is based on the Linux kernel and it is a software stack for mobile devices. This operating system is one of the world's best-selling Smartphone platforms. Android involves many developers writing an application that helps in extending the functionality of the devices. There are currently over 1.6 million applications available for Android. The Android open-source software stack consists of Java applications running on a Java-based, object-oriented application framework on top of Java core libraries [5].

#### **3.2.4.1 Advantages and Limitations of an Android**

An Android is an open source technology and it has some advantages which are listed as the following [54]. It is a Google-developed operating system which primarily focuses on mobile devices such as smartphones. This operating system has touch inputs to display objects and lightweight virtual keyboard. It has been improved by adding new features and fixing errors in previous versions that lead to being many upgrades operating system. Each new version has a dessert name in alphabetical order starting from Cupcake 1.5; Donut 1.6; Eclair 2.0 to the latest Oreo 8.0 version.

The advantages and limitations of an Android are the following.

- a. It is an open source technology.
- b. It is compatible with every Android device function and feature.
- c. It incorporates some hardware technologies and widely uses in the future.
- d. It allows combining other third-party development, especially for the backend server.
- e. It can be learned freely by developers.
- f. Many Android applications are easily downloaded in Google play store.

An Android has some limitations in contrast to advantages.

- a. It does not available for some Windows-based applications such as Firefox.
- b. It does not have many big companies except only HTC.
- c. It has poor security in each application.
- d. It has some issues such as in Bluetooth technology.

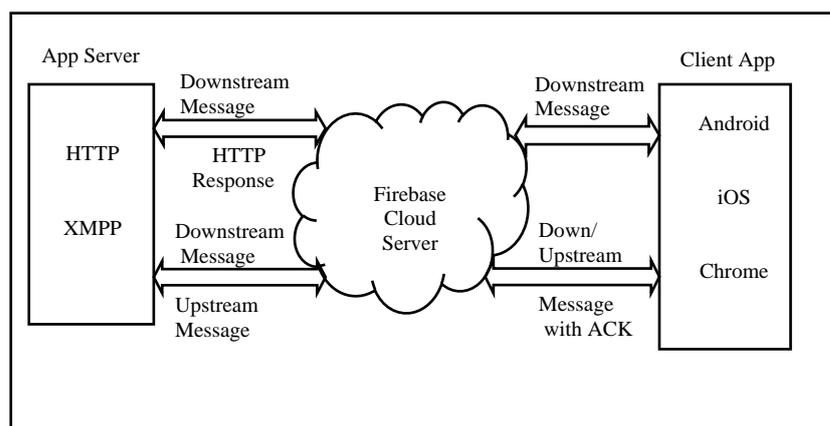
### **3.2.5 Cloud to Device Messaging**

Many mobile applications use remote resources and services especially on the cloud because of the limitations on their built-in memories and spaces. Some resources are freely provided by cloud infrastructure and some have fees to use them. An open source and free cost service are Google cloud messaging (GCM). It is one of the services that exist between developers and Android devices to send push messages. GCM acts as a proxy server or communication server and supports multicast messages between the android uses and server [86]. It is used for emergency response and rescue actions such as disaster notifications, gas related fire damage and other sensitive applications for Android devices with the high performance of GCM.

It provides persistence connection and retains messages when the Android devices are in offline. Sometimes, GCM message arrival time has latency so that it is unpredictable message arrival time. Although Google's GCM servers have a reliable connection between Android devices and application server, it does not guarantee the time to receive message [77]. Recently, Google upgraded FCM by removing the issues of GCM. Google introduced a backend service called Firebase that helps developers to build reliable and real-time mobile applications for Windows, iOS, Android and other web applications [53]. Using firebase cloud messaging (FCM), messages can send to all mobile users without using an operator or application server [103].

Firebase is a cloud service supplier and backend as a service. A special platform is provided by firebase that mostly supports to create mobile applications. It has no cost to build an application and update it in real time. It is a platform and a tool that is known for its speed and reliability in terms of the time. It takes for building applications that are real-time with a highly simpler platform, many of the Google features are carried forward along with other advanced features like crash reporting and thereby allowing the developers to create critical and more functional applications providing a wide variety of services. It has a great scope in future because of the variety of advanced features that are added in firebase that provide a variety of services.

Firebase cloud messaging is upgraded from Google cloud messaging. The JSON configuration files, libraries, dependencies, and plugins are needed to use FCM [6]. There is no need to configure the server for using firebase. Instead, firebase supports these functions automatically. So server-side implementation is not required. It provides reliability and queuing of messages thus it saves time. It offers the wide range of messaging solutions with advanced options and newer capabilities. It has two types of services-Message type and Notification message. The Hypertext Transfer Protocol (HTTP) is an application layer protocol which is used for the secure communication over the internet and between sender and receiver. Extensible Messaging and Presence Protocol (XMPP) is also the open source communication technology that acts as a protocol between sender and receiver based on XML techniques [28]. The flow of activities from App server and the client application is conveniently carried out by Firebase Cloud Messaging system. The following figure 3.3 simply explains the activities of firebase cloud messaging.



**Figure 3.3 Activities of FCM between App Server and Client App**

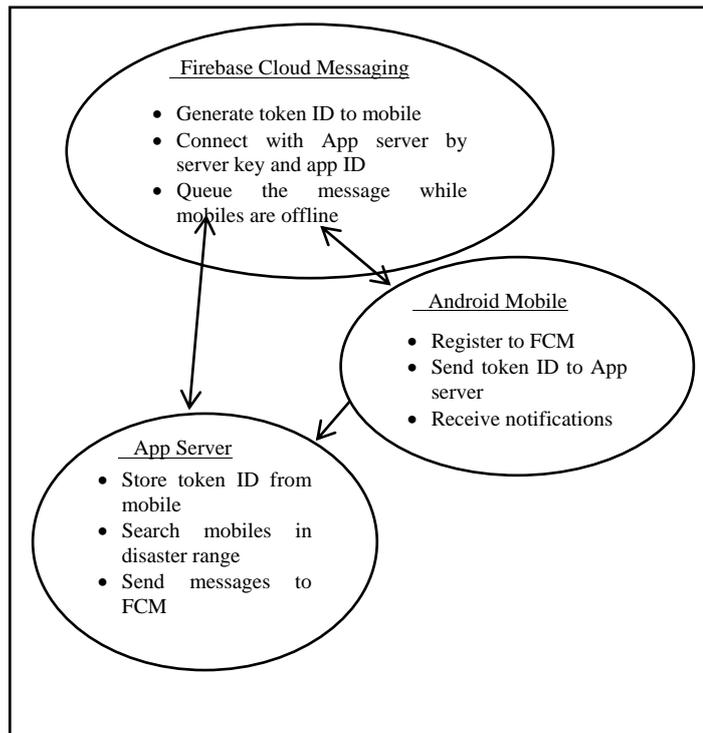
### 3.2.5.1 Firebase Cloud Messaging

“Push” is an important function on mobile devices, as it allows users to receive updates or messages without actively launching an app. Firebase cloud messaging (FCM) is an internet-based communication message service system. It delivers messages from the third-party server to client applications. Sometimes, it queues messages when the client applications are offline. Using FCM, a server can notify or send both messages and voices to client applications. Thus, it is used for sending the notification message, instant message and other short messaging services. It can transfer message size up to 4kb between server and client applications. An implementation using FCM has three major components between sending and receiving messages [57].

It is a free service for delivering cross-platform messaging by multicast or broadcasting techniques. It supports push notification well and queuing of messages. It inherits from Google cloud messaging. The message requests have not time constraint; FCM will maintain it until the period of four weeks. If the application has not connected to FCM for more than one month, FCM discards this application and it will not receive any message from service provider or application server. There are some requirements to access FCM service. Firstly, Google play services maintain a connection to the cloud, firebase-instance-ID-service that receive instance ID, and firebase-message-service to receive notifications are required on the mobile device. Next, the application server to store data and to manage the message and firebase that routes messages to the application server and devices are also needed in the cloud. The connections between Application Server, Android Mobile and FCM are shown in figure 3.4.

Normally, there are three basic segments for sending notification message to the mobile users.

1. User – represents a person who uses the possibilities provided by the mobile device and the alert application installed on the mobile device;
2. Mobile Terminal Device – hardware-equipped terminal which enables the usage of alert or notification application;
3. Location Provider System – system of satellites and receivers intended for positioning.
4. Application Server – service providers that automatically send all of the notification to the users or subscribers.



**Figure 3.4 App Server, Android Mobile and FCM relations**

The goal of FCM is sending a notification message to the correct place, providing reliable network connection between thousands of client applications and service providers or server, and queuing of this message when the client applications of mobile users are offline. To get FCM service, a project is created in Google Developer Console firstly that releases project number and API Key. The application server communicates FCM by server key and sender ID. The client application needs register ID or token ID from FCM. Besides, the required JSON configuration files, plugins, libraries and dependencies are added to the client application. The required credentials are shown with explanations in table 3.1.

**Table 3.1 Credential and Parameter Explanation**

Credential	Explanation
Sender ID	unique numerical value API project (Google Developer Console)
API Key	save on app server (header post)
Application ID	client app register to receive message
Token ID	FCM connection servers issue ID that uses the client app and receives the notification message

### 3.3 Scope of Moving Object Database

Database support for the modeling and querying of time-dependent geometries & moving objects is known as Moving Object Database. Currently, the applications of moving objects database are being developed in the market. Database Management System (DBMS) technology constitutes a fundamental foundation to develop these applications [11]. The powerful moving object database can answer any kind of query the moving object. Generally, physical objects that have position and extent which may change over time, e.g.

- Countries, rivers, roads, pollution areas, land parcels.
- Taxis, airplanes, oil tankers, criminals, polar bears, hurricanes, flood areas, oil spills.
- Users with location-aware portable wireless networked devices such as mobile phones, PDAs.

Since the data of millions of moving objects changes incessantly, it has become inevitable to store and manage the voluminous and by devising scalable data management system. The DBMS for moving objects would deal with data mining, location propagation, privacy, and synchronization, efficiently.

Some databases such as spatio-temporal databases have roots in spatial and in temporal databases and moving object database (MOB) includes spatio-temporal features. Besides, MOB supports for continuously changing geometries called movement. It considers moving point objects and regions that may move and change their shape. It also thinks about current and near future movement/history of movement by time-dependent linear functions.

Moving object database is widely used for motion tracking and monitoring in broad application scope. It includes:

- Geo-fencing Applications
- Digital-based Technology Fields
- Traffic Monitoring and Control Applications
- Airline Control Systems and Applications
- Location Awareness and Tracking
- Web Mining
- Machine Learning Applications
- Communication System
- Real-time weather forecasting

- Moving Object Management, etc.

It is a quite popular database because it has efficient and effective support for those kinds of sectors with rapid development.

### 3.4 Indexing and Its Factors

The index allows people to reach information rapidly. It is about direct inquiry performance. At the part where the searches are concentrating, the main logic of index is, building itself at that related part. Therefore the performances of the results are higher and more accurate [9]. If the index defines a column, as shown in the below inquiry, as an index domain at the same time database will create a directory inside of database which will use later for the search. Searching in an indexed domain means searching through the directory, thus receiving feedback will be much faster. In order to make select inquiries faster, if the index is done for every column then this will be a mistake. As the number of indexes increases add on and updates, then features might slow down and system performance might be affected badly in general.

```
CREATE INDEX system.plane_idx
ON system.plane (plane location)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

The main factor in Spatial/temporal data indexing is creating groups which will have any sort of relationship with the groups that are formed out of these objects and the inquiry sentence when there is a query [3].

```
SELECT *
FROM plane, landing
WHERE plane.mail code ='A01' AND
distance (plane.geodata, landing.geodata) < 200000 ;
```

The above query will group the plane data with landing field data within the same time interval and will be configured by the database to form an answer. These methods have the fundamental main idea of moving object database approaches. With the help of this, the spatial objects which have geometrical features can be abstracted from their complexities so they can be viewed in more simplified shapes. Thanks to this abstraction the substances which take place in space (point, line, and region) can be structurally modeled [22]. With these attempts, depending on the variety of indexing algorithms, time factor can take place. The relation between the modeling

objects (intersection, tangent etc.) in terms of their features like (distance < 200000) and the operations defined on the objects will involve in the problem.

### **3.4.1 Range Searching Process and Types**

There are two types of range searching: static range searching and dynamic range searching. The difference between two of them are that the data in the dataset has no significant change in static but the data insertion and deletion occurs within specific queries in dynamic range search. The sample query for range searching has the following process. Input Description: A set  $S$  of  $n$  points and a query region  $Q$ . Problem description: Which points from  $S$  lie within  $Q$ ?

Discussion: Range search problems arise in the database and location-based system applications. Any data object with  $d$  numerical fields, such as a person with height, weight, and income, can be modeled as a point in  $d$ -dimensional space. A range query describes a region in space and asks for all points or the number of points in the region. For example, asking for all people with income between \$0 and \$10,000, with a height between 6'0" and 7'0", and weight between 50 and 140 lbs. defines a box containing people whose body and wallets are both thin.

The difficulty of a range search problem depends on several factors: number of range queries, query shape, number of dimension and result and count of points in the region.

The location coordinates are stored as the two-dimensional objects and three main queries normally ask for high dimensional storage in the database.

1. Exact match query: The objects whose coordinates in the range are exactly matched query coordinates.
2. Partial match query: Some features are the same but not all coordinates exactly match [49].
3. Range query: The objects whose coordinates are in a specified query range or service area.

#### **3.4.1.1 One Dimensional Range Query**

There are two types of range query in index tree based range structure. They are one dimensional range query and two or multidimensional range query. A range tree on a set of 1-dimensional points is a balanced binary search tree on those points. The points stored in the tree are stored in the leaves of the tree; each internal node

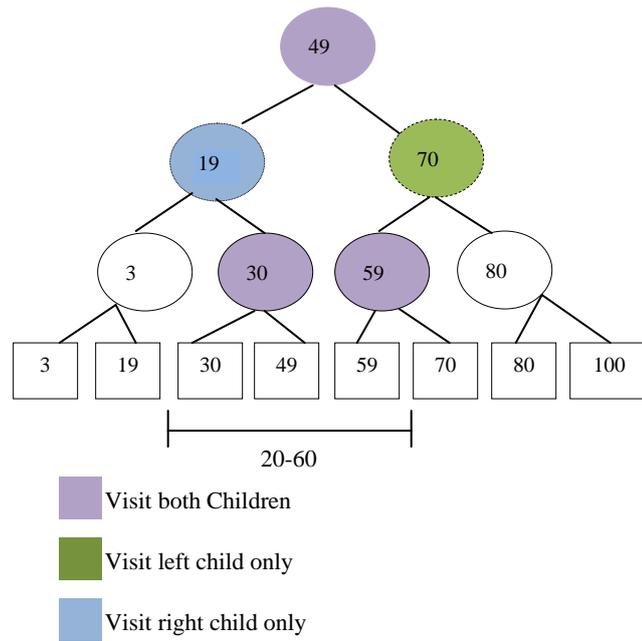
stores the largest value contained in its left subtree. The simplest structure for one-dimensional range searching is a sorted array. The preprocessing sorts the  $N$  elements to be in ascending order by key. To answer a range query, two binary searches are done to find the positions of the low and high end of the range in the array. After these two positions have been found all the points in that part of the array are listed as the answer to the range query [46].

Given one dimensional space, the query asks for the points inside an interval  $[x:x']$ . Let  $P = \{p_1, p_2 \dots p_n\}$  be the given set of points on the real line. The leaves of  $T$  store the point  $P$  and internal nodes of  $T$  store splitting values to guide the search. Let the value at node  $v$  is  $x_v$ . Left subtree of a node  $v$  contains all the points smaller than or equal to  $x_v$  and right subtree contains all the points strictly greater than  $x_v$ . To report the points in the range query  $[x: x']$  and searching with  $x$  and  $x'$  in the tree  $T$ . let  $u$  and  $u'$  be the leaves where the searches end. Then the points in the interval  $[x:x']$  are ones stored between the leaves  $u$  and  $u'$ . The procedure for one-dimensional range tree is the following. There is a tree ( $T$ ) and two range values ( $x_1, x_2$ ). All of the nodes that are in the range of the tree will be displayed as the results of one-dimensional range search.

Procedure: OneDRange( $T, x_1, x_2$ )

- i. while not isLeaf( $T$ ) and ( $x_2 \leq T.data$  or  $x_1 > T.data$ ):
- ii. if  $x_2 \leq T.data$ :  $T = T.left$
- iii. else:  $T = T.right$
- iv. if isLeaf( $T$ ):
- v. if  $x_1 \leq T.data \leq x_2$ : output( $T.data$ )
- vi. else:  $v = T$
- vii. while not isLeaf( $v$ ):
- viii. if  $x_1 \leq v.data$ :
- ix. utput\_subtree( $v.right$ )
- x.  $v = v.left$
- xi. else:  $v = v.right$

This range searching is used for both static and dynamic index structure. This range query allows speeding up the processing power of searching time between an arbitrary number of nodes and pointers. The sample one dimensional range search with the interval  $[20:60]$  is shown in figure 3.5.



**Figure 3.5 Sample one dimensional Range Search between 20 and 60**

### 3.4.1.2 Two Dimensional Range Query

The structure of this tree is built where the top level structure is a range tree based on the x coordinate value and each leaf node is also built a range tree based on the y coordinate values of all the objects in its subtree. It treats range query as two nested one-dimensional queries:  $[x_1, x_2]$  by  $[y_1, y_2]$ . Let  $[x: x'] * [y: y']$  be the range query. First ask for the points with x coordinates in the given range  $[x_1, x_2] \Rightarrow$  a set of subtrees instead of all points in these subtrees, only want those that fall in  $[y_1, y_2]$ . The main tree T on the x coordinate points at any node stores a pointer to a tree  $T_y(v)$  (associated structure of v), which stores canonical subset  $P(v)$  organized on the y-coordinate, and then 2D points are stored in all leaves [8]. The time complexity of a balanced two-dimensional tree is  $O(n \log n)$  space. The query time for all allocation nodes in two-dimensional range query is  $O(\log n)$ . Generally, the two-dimensional range tree is built by two steps: preprocessing step and range searching step. The procedures of preprocessing and range searching are explained the following.

Preprocessing: A building of the 2D-Tree.

Input the N points (array) and initialize the empty tree

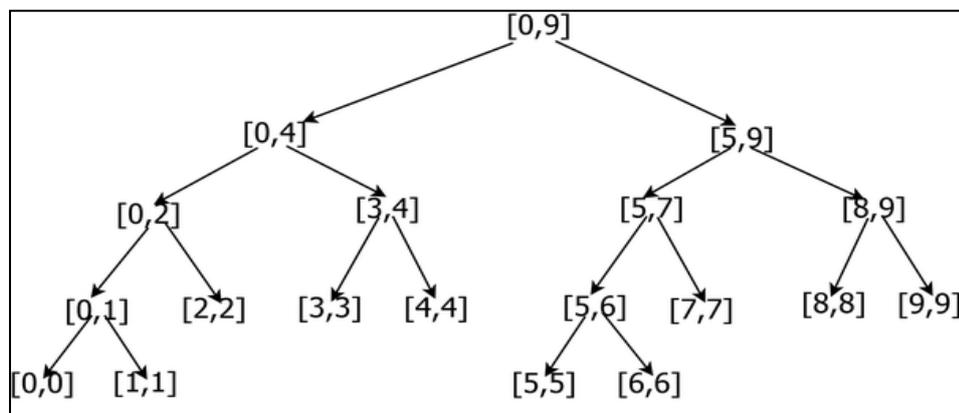
- i. Pick one (randomly) as the root of the empty tree
- ii. For each point  $p_i (i=1, 2, \dots, N-1)$ , insert it into the tree by

- iii. Going down the tree by comparing its keys (two coordinates) to the root nodes and other nodes in the tree
- iv. Alternating the x and y coordinates as the comparing keys while going down the tree level by level
- v. For every comparison, if smaller, go to left sub-tree; if bigger, go to right sub-tree. Repeat until reaching an external node
- vi. Insert point  $p_i$  into the current position
- vii. Loop until all points are inserted

Range searching: Read in the Range  $R(x_1, x_2, y_1, y_2)$

- i. Travel through the 2D-Tree by comparing the key of the node point to the corresponding interval  $(x_1, x_2)$  or  $(y_1, y_2)$ :
- ii. Go down to the left sub-tree, if  $x_2 \leq p.x$  ( or  $y_2 \leq p.y$  )
- iii. Go down to the right sub-tree, if  $x_1 \geq p.x$  ( or  $y_1 \geq p.y$  )
- iv. Check both sub-trees, if  $x_1 < p.x < x_2$  ( or  $y_1 < p.y < y_2$  )
- v. Check the other coordinate of those node points which are cut through the interval or on the edge lines
- vi. Repeat until reaching external nodes, output the points within  $R$ .

The sample two-dimensional range tree is shown in figure 3.6. It is built by input two dimensional values for such as  $[0,0], [0,1], [2,2], [0,2], [3,4], [3,3], [0,4], [0,9], [4,4], [5,5], [7,7], [5,6], [8,8], [9,9], [5,9], [8,9], [1,1], [6,6], [5,7]$ .



**Figure 3.6 Sample Structure for Two-dimensional Range Tree**

### 3.5 Presort Range Tree

Presort Range tree come from Range tree thus this index structure is a two-dimensional relationship that recursively builds based on one dimensional Range tree structure. The only difference is that the data is added by sorted lists and it is taken as

input parameters. Thus, the creation of an index is faster than the usual because it does not have to search for the correct space to store the new value, and it saves both I/O and CPU costs. The new value that will always be joined adjacent to the last value that was stored. The index tree would be fast as it is built sequentially and range search easily.

Normally, a tree data structure is a useful tool for organizing data objects based on index. It is equally useful for organizing multiple mobile objects in terms of two dimensional relationships. In the traditional index structure, it is built by unsorted two-dimensional data. When it is used by unsorted data, its structure might be bigger than the index structure of the sorted data. Especially it is going to be hard to build and store in a large amount of data size. Ordering or sorting the data may take additional extra time, but it will be faster than any other unsorted structure and thus it will bring to be a fast and compact index including range searching. To construct any tree structure, the first thing is pre-processing the data into the data structure. Then, queries and updates on the data structure are performed. There is an assumption for mobile locations that no two points have the same x- coordinate and also y-coordinate in this index structure.

### 3.5.1 Procedure of Presort Range Tree (PRTree)

The presort range tree procedure is the following;

Input: Lats =Array of two dimensional points sort on latitudes

Longs =Array of two dimensional points sort on longitudes

Output: Two dimensional locations index structure by ordered queries

**Procedure: PRTree (Lats, Longs)**

1. if Lats.length==1 then return new LeafNode(Lats[1]);
2. medium= [Lats.length/2];
- 3 copy Lats[1....medium] to Lats<sub>L</sub> and Lats[medium+1..... Lats.length] to Lats<sub>R</sub> ;
- 4 for i=1 to Longs.length do
- 5 if Longs[i].x <= Lats[medium].x then append Longs[i] to Longs<sub>L</sub> ;
- 6 else append Longs[i] to Longs<sub>R</sub> ;
- 7 root= new Node((Lats[medium].x),One D Range(Y));
- 8 root.left= PRTree(Lats<sub>L</sub> , Longs<sub>L</sub>);

```

9  root.right= PRTree(LatsR , LongsR);
10 return root;

```

### 3.5.2 PRTree with Circular Range Searching

After preprocessing of tree construction is done, the structure allows searching circular range query for mobile objects. To determine whether registered mobiles are in service area or not so that this system has to **get bounding coordinates** with center and service distance: (centerLat, centerLong, bearing, distance).

```

bearingRadians = Radians(bearing);
lonRads        = Radians(centerLong);
latRads        = Radians(centerLat);
maxLatRads = asin((sin(latRads) * cos(distance / 6371) + cos(latRads)
                  sin(distance / 6371) * cos(bearingRadians)));
maxLonRads = lonRads + atan2((sin(bearingRadians) * sin(distance / 6371)
                             cos(latRads)),(cos(distance / 6371) - sin(latRads) * sin(maxLatRads)));

```

### 3.6 Example : Calculating Proposed Tree with center and service distance

Firstly, sort the mobile locations by latitudes and longitudes. Then the presort range tree is built and shows as the following;

```

25.40319 98.11739
LEFT:    16.80958 96.12909
LEFT:    16.35099 96.44281
RIGHT:   16.77923 96.03917
RIGHT:   24.99183 96.53019
LEFT:    24.77906 96.3732
RIGHT:   25.38048 97.87883
RIGHT:   25.88635 98.12976
LEFT:    25.59866 98.37863
RIGHT:   25.82991 97.72671
RIGHT:   26.35797 96.71655
LEFT:    26.15312 98.27074
RIGHT:   26.69478 96.2094

```

The results of sample range search in centerLat, centerLng, distance: 26.693, 96.208, 100 km that are registered mobile locations to send notification as follows:

node (25.40319, 98.11739)  
RIGHT: node (24.99183, 96.53019)  
LEFT: node (24.77906, 96.3732)  
RIGHT: node (25.38048, 97.87883)  
RIGHT: node (25.88635, 98.12976)  
LEFT: node (25.59866, 98.37863)  
RIGHT: node (25.82991, 97.72671)  
RIGHT: node (26.35797, 96.71655)  
LEFT: node (26.15312, 98.27074)  
RIGHT: node (26.69478, 96.2094)

### **3.7 Nearest Neighbor Techniques**

The nearest neighbor (NN) rule basically comes from the supervised learning technique in which class is already known and identifies the nearest neighbor points from the category of an unknown data point. One reason for the use of this rule is its conceptual simplicity, which leads to straightforward, if not necessarily the most efficient, programming. Different techniques are used for nearest neighbor search by many researchers. The two basic techniques are structure-based technique and structureless technique. The first technique consists of suitable index-based structure and stores all of the objects appropriately. Some of the two-dimensional index structures that can be used for nearest neighbor search are Ball tree, Cover tree, and KD tree. The second technique basically comes from calculating the distance between the query position  $p$  and every point position in the training set of positions  $P$ . Examples of this technique are Brute Force Method,  $k$  Nearest Neighbor (kNN),  $k$  Nearest Neighbor (kNN), weighted  $k$  Nearest Neighbor (wkNN), Condensed Nearest Neighbor (CNN), clustered Nearest Neighbor, reduced Nearest Neighbor (RNN) and rank  $k$  Nearest Neighbor (rkNN) [27].

#### **3.7.1 Structure Based Techniques**

Structure based techniques allow to retrieve certain information in the database. An improvement of this technique over kNN is speed. In this structure, the relevant data is kept at the leaf nodes of the tree and the rest of the internal nodes provide to be efficient search through leaves. It provides fast access and improves the search query time such as range search, nearest neighbor search and circular search

but the expense of the space for the data structure must be maintained for the search. In order to obtain the data structures, it takes the time complexity of preprocessing the data. It can be used not only static structure with stationary data but also dynamic structure with moving objects [27].

### **3.7.1.1 Ball Tree k Nearest Neighbor (KNS1)**

It is very similar to KD trees, spatially organizing points in multiple dimensions. However, unlike KD trees, which split the points parallel to the axis, ball trees split such that points closer to each other go to one child while the other set of nearby points go to the other child. As it is also a two-dimensional balance tree, it provides to improve the speed of KNN. The data is displayed by level order in the hierarchy. It deals perfectly with a large dataset and high dimensional entities. Its structure is balanced and saves the searching time for both range search and nearest neighbor search [60]. However, it may be costly by insertion algorithms. Moreover, KNS1 degrades as distance increases.

### **3.7.1.2 KD Tree Nearest neighbor (kdNN)**

KD tree data structure is most often used to speed up fast nearest neighbor queries. It divides the training data exactly into half plane as soon as the input data is allocated and produces a two-dimensional balanced tree. It supports not only the entire nearest query from the query point in the tree but also all nearest query results in the range [64]. It is appropriate to use for nearest neighbor searching in a large dataset. So, it is fast and simple for both building tree and searching in queries. However, it needs more computation and intensive search by adding an extra neighbor algorithm or procedure. The tree construction time is needed to use this index structure that can reduce the time by eliminating duplicate tuples or data. Besides, it slices points blindly into half which may miss data structure. The procedures of KD tree structure and nearest neighbor search are described in the following.

#### **Procedure: BuildKDTree (P,l,r,odd)**

- a. if odd then Sort P on x  
    else Sort P on y
- b.  $v = \lfloor (l+r)/2 \rfloor$
- c. if  $l=r$  then root=new LeafNode(P[v])

- d. else
- e. if odd then root=new Node(P[v].x)  
          else root=new Node(P[v].y)
- f. root.left=BuildKDTree(P,l,v,not odd)
- g. root.right=BuildKDTree(P,v+1,r,not odd)
- h. return root;

**Procedure: Nearest Neighbor Search (q: point, n: node, p: ref point w: ref distance)**

```
//initial call (q.root,p,infinity)
if n.left=n.right=null then {leaf case}
w' :=||q-n.point||;
if w' < w then w:=w' ; p:n.point;
else
if q(n.axis)= n.value then
search_first:=left;
else
search_first:=right;
if(search_first= =left)
if q(n.axis)-w <= n.value then Nearest Neighbor Search (q, n, n.left, p, w);
if q(n.axis)+w > n.value then Nearest Neighbor Search (q, n, n.right, p, w);
else // search_first==right
if q(n.axis)+w > n.value then Nearest Neighbor Search (q, n, n.right, p, w);
if q(n.axis)-w <= n.value then Nearest Neighbor Search (q, n, n.left, p, w);
```

**3.7.1.3 Cover Tree**

A cover tree T is a level tree in which each level on a data set is a cover for the level beneath it. It is a type of tree-based data structure that is specially designed to save the searching time of the nearest neighbor approach. It is a modification of the Navigating Net data structure, and it is associated with a variety of other index-based data structures developed for low-dimensional data [26].

By storing data in this index based tree, it can easily retrieve and add data whenever the user wishes a query. The tree structure has built the hierarchy of levels that contain the root point at the top of the tree and the rest of the levels are in the metric form of space [16].

### **3.7.2 Structure Less Techniques**

The main concept of the structure less technique is based on k-nearest neighbor which is the most popular supervised learning method. It lies in the first category in which the whole data is classified into training data and testing data point. Then, distance is calculated from all training data to the testing point and collected the point that has the lowest distance. After being a calculation, k nearest neighbors is produced. This technique is very simple and it is only based on the distance to implement this technique. However, the value of k affects the result so that it needs to carefully determine k value in some cases. This method needs various calculations according to the number of the dataset but it overcomes memory requirements. On the other hand, structure-based techniques are not suitable for the small dataset and it can reduce the complexity of calculation.

#### **3.7.2.1 Brute Force Method**

The most straightforward exact nearest neighbor search method is by brute-force, the structure less-search method. Brute force method first calculates the distance between the query location  $l$  and every location in the set of location points  $P$ , then orders all calculated distances, and last identifies the shortest distance(s) and the corresponding point(s) in  $P$  nearest to  $q$  [1] [79]. It has a time complexity of  $O(nd)$  where  $n$  is the number of the points in the point set  $P$  and  $d$  dimensionality of the space  $S$ , and complexity of space is  $O(1)$  because no data structure is required to be maintained for the search. It usually contains problem-solving statement based on computation and the definition of its relations. Therefore, it is an inefficient method related to the problem that has high complexity or hierarchy. It is only suitable for solving small problem or modules of a large concept.

#### **3.7.2.2 k Nearest Neighbor (kNN)**

It uses the nearest neighbor rule such that it assigns the class of the nearest neighbor for a specific query node  $n$ , computes the k nearest neighbors and assigns the class by majority vote. It is a lazy supervised learning algorithm and classifies the objects based on similarity measurement. The advantage of this technique is that training time is very fast for relevant attributes, simple to learn, robust to noisy data in the training dataset. However, it may bias by neighbor value or predefine value  $k$ , heavy in computational complexity for large dataset and overloading in memory

requirements. As it is a lazy algorithm in supervised learning, it takes time too much for the running process and easily false by irrelevant training attributes [26]. The variations of nearest neighbor distance measures are the following.

- City-block distance (Manhattan dist)  
Add absolute value of differences
- Cosine similarity  
Measure angle formed by the two samples (with the origin)
- Jaccard distance  
Determine percentage of exact matches between the samples (not including unavailable data)

### **3.7.2.3 Weighted k Nearest Neighbor (WkNN)**

It assigns weights to neighbors as per distance calculated. The advantages of this algorithm are the following: (1) it reduces some issues of kNN by assigning equal weight to k nearest neighbors implicitly. (2) it carries out all of the numbers of k training data. (3) It provides as the global algorithm based on weight. However, this algorithm runs slow and computation complexity increases in calculating weights [13].

### **3.7.2.4 Condensed Nearest Neighbor (CNN)**

It reduces the data in the training set that has similar data or it does not require as valuable information. It cannot be picked up points on boundary because CNN is an order dependent technique [7]. The way of this work is to classify the object into three different types.

1. Outliers: objects which would not be included in the predefined type if inserted to the database.
2. Prototypes: the minimum set of objects required in the training set for any other non-outlier objects to be correctly included.
3. Absorbed objects: objects that are not outliers, and would be correctly included in the object group.

### **3.7.2.5 Reduced Nearest Neighbor**

It is an extension of the CNN rule. It removes duplicate tuples and patterns which do not impact the output of training dataset. It also scales the size of data in the

training set and eliminates extra or unnecessary templates. It can improve processing time for each query and fewer memory requirements by reducing the execution rate. It suffers computational complexity and time-consuming [33].

#### **3.7.2.6 Clustered k Nearest Neighbor**

This cluster simply refers to collecting the nearest neighbors of a specific point by getting all the points in a service area with the radius and then it returns the cluster. It is built by clusters that are formed to retrieve nearest neighbor object efficiently. It overcomes the incomplete sharing of training datasets with robust cluster features. But, the threshold value is difficult to determine at the initial state of clustering. Besides, it can bias by the value of k for clustering.

#### **3.7.2.7 Rank Nearest Neighbor**

It can be assigned the ranks of data in the training tables or databases for each category [12]. It is suitable for the dataset which has many features variations. As it is based on rank, it easily searches for any data and robust under various datasets. But, the multivariate rank nearest neighbor (RNN) depends on the distribution of the data.

### **3.8 Summary**

Firstly, this chapter presents the relationship between cloud technologies and mobile devices with location update policies and then discuss the cloud to devices messaging services. The chapter explains the scope of moving object databases. The procedures of KD tree and presort range tree are explained and discussed that procedures are used to compare the proposed index tree for performance evaluation. The index approaches especially for the nearest neighbor techniques and two types of range dimension are also described. And then the chapter presents the detail process of the different nearest neighbor techniques for structure less and structure-based methods in indexing technologies. Finally, this chapter also discusses the nature and workflow of the two-dimensional range search.



## **CHAPTER 4**

### **THE PROPOSED SYSTEM ARCHITECTURE**

This chapter presents a complete notification system for mobile users by proposing methods and architectures. This system contains three main approaches: getting and updating mobile user locations, storing and processing them by indexing and sending notification by firebase cloud messaging.

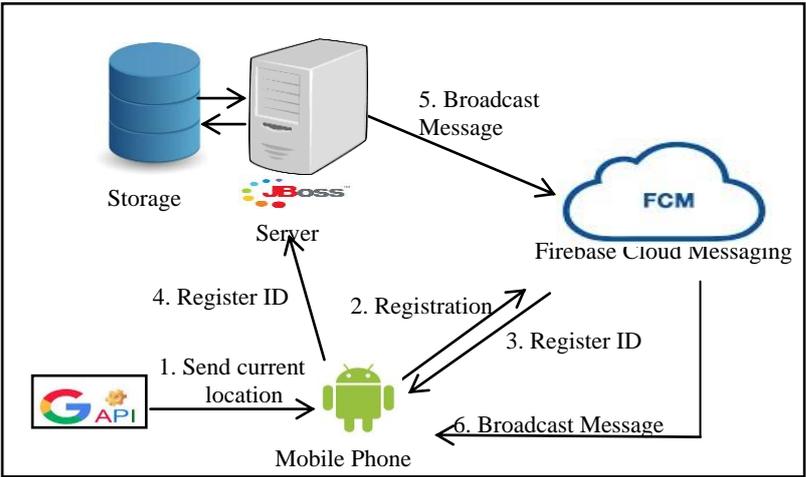
In the first approach, Google API is used to support accurate location for mobile users that are applied to the server. It is a collection of location APIs. It includes three different providers to get the location. (1) GPS provider: This provider determines location using satellites. Depending on conditions, this provider may take a while to return a location fix. (2) NETWORK provider: This provider determines location based on the availability of cell tower and Wi-Fi access points. Results are retrieved by means of a network lookup. (3) PASSIVE provider: This provider will return locations generated by other providers. The users passively receive location updates when other applications or services request them without actually requesting the locations themselves. For updating the locations, the mobile users send their locations regularly and continuously in traditional update strategy. However, in fact, it is not necessary. If the mobile movements are small and frequent, thus unnecessary update would be performed at the server. Hybrid Update Algorithm is proposed to update locations appropriately for both client side as well as server side.

At the second approach, presort-nearest location index tree based index structure is proposed that provides two-dimensional range queries and nearest queries. Normally, the index is built by unsorted two-dimensional data. When it is built by unsorted data, it might be bigger than the index structure of the sorted data. Especially it is going to be hard to build and store in a large amount of data size. The proposed index structure is a two-dimensional relationship that recursively builds based on a one-dimensional tree structure. The data is added by sorted lists and it is taken as input parameters. Ordering or sorting the data may take additional extra time, but it will be faster than any other unsorted structure and thus it will bring to be a fast and compact index including range searching. Thus, the creation of an index is faster than the usual because it does not need to search for the correct space to store the new value, and it saves both I/O and CPU costs. The new value that will always

be joined adjacent to the last value that was stored. The index tree would be fast as it is built sequentially and range search easily.

The third approach is sending a notification to mobile users. In this system, mobile users who are necessary for notification or they are in the imminent disaster zones will be sent notification. To send the notification, firebase cloud messaging intermediates between mobile users and third-party server or application server. As FCM is a free service offered by Google cloud server, it supports to get persistent connections. FCM solves the following two requirements. (1) It needs to have a persistent network between client and server communication. (2) Many connections may occur as each client continuously communicates to the server along with sending locations and updating them thus it takes the high cost. It also queues the messages while the mobiles are offline.

The system analyzes the proposed index tree structure on moving mobile objects. Besides, a mobile object generator is also proposed that apply to the performance evaluation of proposed system. And the system analyses range searching queries based on index structure with comparing distance-based approach. It is described in detail in chapter 6. Figure 4.1 describes the overall system architecture for notification.



**Figure 4.1 Architecture of the Proposed System**

This figure 4.1 shows the architecture of a complete notification system for mobile users. The server sends all of the notifications automatically to the users. In this architecture, the following steps are carried out to send notification by push type of service.

- (1) Android application from the mobile phone receives the current position through a special location provider called Google API.
- (2) The application registers to FCM with application ID.
- (3) FCM generates Token ID for each Android application.
- (4) Then, the application communicates with the server not only giving token ID but also sending the location of the mobile current position. After that, the server keeps mobile's latitude and longitude to the database which has built index structure.
- (5) For users who are located in an imminent emergency area, the server searches all of the users who are in the service area by indexing tree based range query. Then, the server sends the message to FCM with the lists of mobile users in the emergency area.
- (6) Afterwards, the application fetches the message from FCM push technology by multicasting.

The detail process of system architecture cooperates with the following four modules.

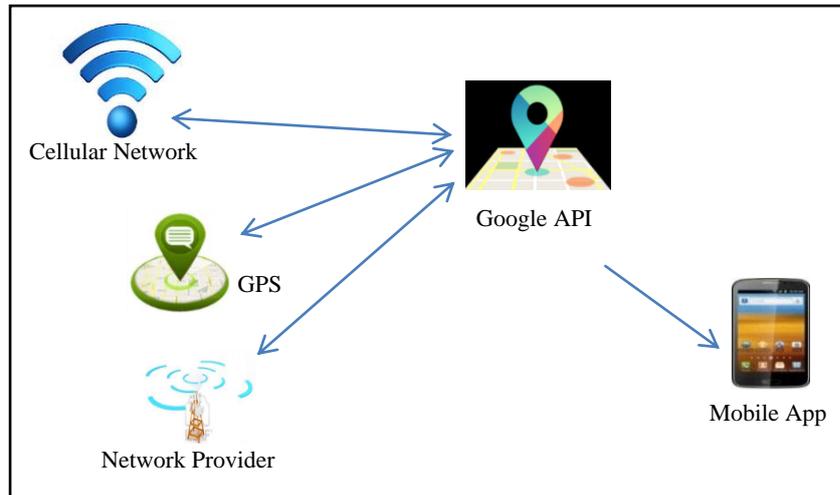
- i. Getting mobile locations
- ii. Updating Client application's location
- iii. Index based structure
- iv. Circular range searching

#### **4.1 Getting Mobile Locations**

In this module, Google API is applied to get the current location of mobile users. It is a special set of communication tools. But it is hard to get the accurate locations as the delivery is limited at some places and measurement of the results is impossible. Instead, Google API platform has succeeded in linking mobile devices and location providers. Thus, accurate and reliable current locations are provided by Google API. It is used for the following two objectives.

- To provide a more powerful framework that automatically switches between location providers and location accuracy.
- To handle location update schedule.

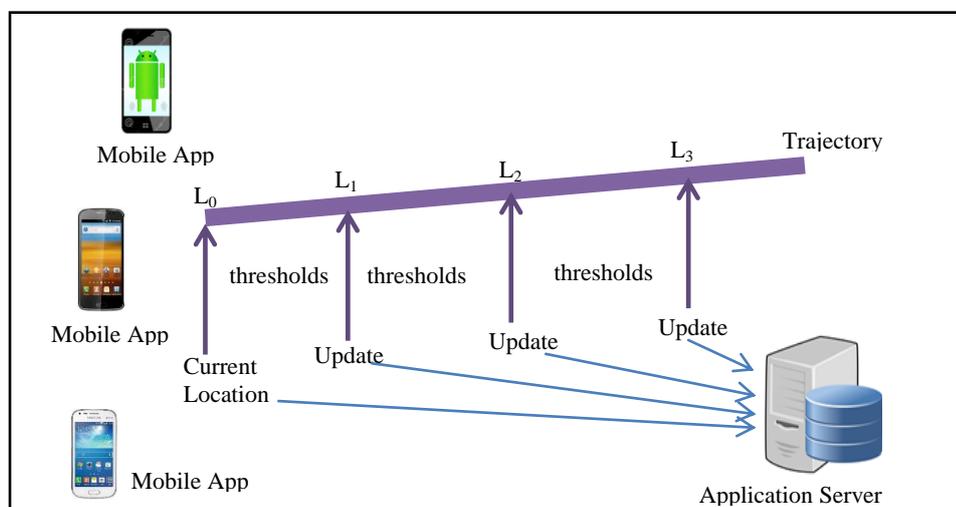
The following figure 4.2 shows the relationship between Google API that combines three location providers and Mobile App.



**Figure 4.2 Getting Location of Client Application**

#### 4.2 Updating Client Application's Location

In general, tracking mobile positions are one of the most common requirements for many location management services. In recent years, mobile location-based services are definitely growing with updating policies and strategies. Since the location of mobile positions changes continuously but the database location of the moving object cannot be updated continuously; therefore, an updating strategy for moving object is required [95]. There is the trade-off between the number of update messages and information accuracy in designing moving object database systems [96]. In this system, the appropriate update is done by taking distance and time predefined thresholds at the client side. The flow of Hybrid location update for mobile application is shown in figure 4.3.



**Figure 4.3 Hybrid Location Update for Mobile Application**

**Algorithm:** Hybrid Location Update

**Input:** Database of mobile locations contains the locations of registered mobile with time.

**Output:** current registered mobile location

1.  $i=0$ ;  $dis\_threshold$ ;  $time\_threshold$ ;  $time\_scheduler$ ;
2. Read the current location  $(L_{xi}, L_{yi}, t_i)$  and previous location  $(L_{xi-1}, L_{yi-1}, t_{i-1})$  of registered mobile location.
3. If the  $time\_scheduler > time\_threshold$  &&

$$\sqrt{(L_{xi} - L_{xi-1})^2 + (L_{yi} - L_{yi-1})^2} > dis\_threshold \quad (4.1)$$

3.1. Update the database with current location  $(L_{xi-1}, L_{yi-1}, t_{i-1}) =$  current location  $(L_{xi}, L_{yi}, t_i)$ ,  $i+1$ .

3.2. Total number of update=Total number of update+1;

3. Else current location  $(L_{xi}, L_{yi}, t_i) =$  previous location  $(L_{xi-1}, L_{yi-1}, t_{i-1})$ ,  $i=i+1$ ;

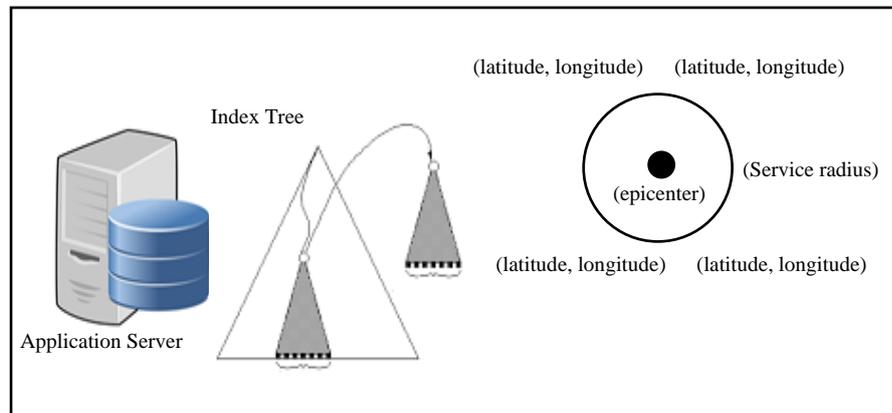
Assume that every ( $t= 2$  seconds) for example both position and speed supposed to be known as it comes from the GPS. These values inserted in the local database of the mobile locations. When these values are inserted and connected to a trigger which computes the current location  $(L_{xi}, L_{yi})$  and previous location  $(L_{xi-1}, L_{yi-1})$  of the position. Since where  $x$  and  $y$  are the  $x$  coordinate and  $y$  coordinate at the location,  $t$  is the time elapsed from the last update. Then compute the current position and previous position of each location. If the result value is greater than the value of  $(th)$  and  $time\_scheduler$  is greater than the value of  $(th)$  the local database of the moving object is updated with the actual position and actual speed. Also, update the central database with the actual position and time. Otherwise if one of the result values is less than the  $(th)$  no update occurs.

### 4.3 Index Based Structure

Indexing is a special data structure that allows having quick access to data or objects systematically. Indexing is usually used for users to display results that match the desired user criteria. In fact, a static data indexing is enough to use with traditional structures. However, most of them are unbalancing construction it as it is hard to store moving object positions or mobile locations. Generally, the purpose of an index structure is a fast and easy query within the desired time. A good moving data indexing needs not the only faster query but also ability to update regularly. In order

to manage moving objects' location, update their positions appropriately, presort-nearest location index structure is proposed in this system.

Due to the rigorous requirements of dynamic range searching, the proposed algorithm allowed getting the desired range queries with faster response. All of the incoming objects or points are ordered before building index structure. The sample two-dimensional index tree structure and its circular range searching are shown in figure 4.4.



**Figure 4.4 Sample Index Tree with Circular Range**

#### **4.3.1 Presort-Nearest Location Based Application Server**

Today, the number of researches based on indexing is known as mobile objects indexing came out from the traditional static one. There are some indexing approaches to handle the complicated moving positions. One of the suitable ideas is pre-ordering or presorting these objects before building index structure. Moreover, an efficient index structure usually promotes application server performance and reduces database workload. In this system, an index structure, presort-nearest location index tree is proposed that allowed maintaining, updating, and range querying mobile objects within the desired period. The main objective of using this index tree is speed data access.

The mobile locations that have latitudes and longitudes are input as a two-dimensional object to this index tree. In this system, the server sends a notification message to the users in the service range area. Messages will be sent starting from the nearest locations in the range. Therefore, the proposed index tree focuses on retrieving the nearest location from the center location in the range. As a result, all of these nearest locations are displayed by level order from the proposed index tree.

#### 4.3.1.1 Proposed Presort-Nearest Location Tree

This index structure is used to find the nearest positions from arbitrary mobile locations in the range by level order.

##### **Algorithm: Presort-Nearest Location Tree**

Input: c: center of location query

R[m]: mobile locations in range lists that are sorted by latitude and longitude of each location.

m: mobile locations

Output : Nearest locations index structure from center points by level order

Nearest Location Tree(c, R (m))

- i. if m != null then Tree->root := c;
- ii. minIndex = 0;
- iii. Tree->leftKey:=FindNearest(c);
- iv. Tree->rightKey:=FindNearest(c);
- v. GenerateTree(c);

FindNearest(c)

- i. nearest := Node[0]; mindist=0;
- ii. for i=1 to R(m)
- iii. d= distance(c,nearest);
- iv. if(d[i]<d[mindist])
- v. nearest=d[i];
- vi. return nearest;

GenerateTree(c)

- i. for d = 1 to height(Tree)
- ii. printGivenLevel(Tree, d);
- iii. if level is 1 then
- iv. print (Tree->c)
- v. else if level>1 then
- vi. printGivenLevel (Tree->leftKey,level-1);
- vii. printGivenLevel (Tree->rightKey,level-1);

The proposed presort-nearest index tree is an ordered tree data structure to hold a list of points. It allows all points within a given range to be reported efficiently starting from the nearest points and is typically used in two-dimensional moving or

static points. This procedure starts with range list arrays that are sorted by latitude and longitude of each location. Both of these arrays are arrays of locations which have the same location points but one is ordered by latitude and another one is ordered by longitude. Firstly, the center point is initialized as a root, the left subtree and right subtree are built according to the finding nearest distances from the center point. Here initially the mindist would be set to 0 in the nearest distance. Now, the mobile locations in the range list are calculated by distance. Each of these distances is compared to the array [mindist] and nearest node would less than array [mindist] that returns the nearest in the range list. Then, the index tree structure is generated by the level order.

#### 4.3.2 Presort-Nearest Location with Circular Range Searching

The circular range search is easy to calculate with center locations and circular distance. As this range search is in the dynamic setting of mobile locations, the results of range queries may vary based on inserted or deleted of mobile positions. This system takes a circular range search with the building of presort–nearest location index structure. To determine whether registered mobiles are in the service area or not so that this system has to get bounding coordinates based on center location and service distance: (center\_latitude, center\_longitude, and radius). A range query retrieves all objects whose location falls within the circular range. The four maximum and minimum coordinates of a circular range query are calculated in the following formula.

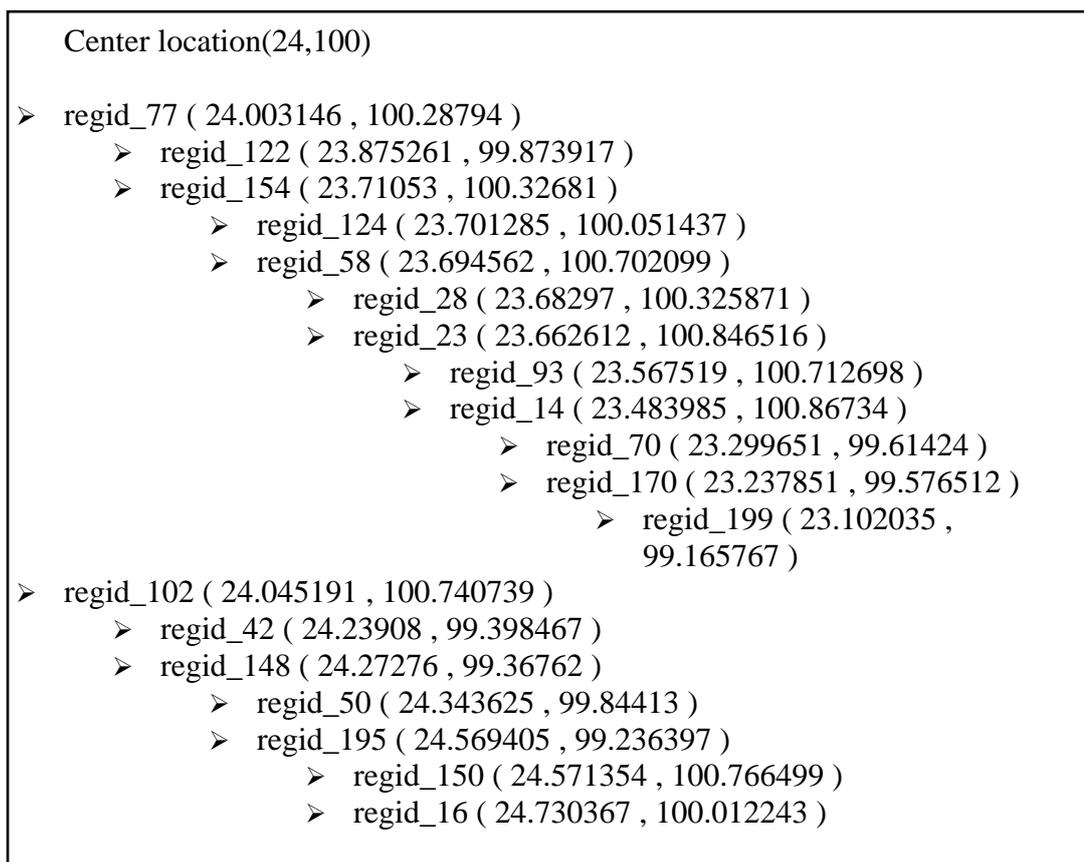
- latitude= center\_latitude;
- longitude = center\_longitude;
- radius = service radius; // km
- R = 6371; // earth radius in km

Then the maximum and minimum latitudes and longitudes with service radius for epicenter is calculated.

- minLongitude=longitude-Math.toDegrees(radius/R/Math.cos(Math.toRadians (lat)));
- maxLongitude=longitude+Math.toDegrees(radius/R/Math.cos(Math.toRadians (lat)));
- maxLatitude = latitude + Math.toDegrees(radius/R);
- minLatitude = latitude - Math.toDegrees(radius/R);

### 4.3.3 Example: Calculating Proposed Index Tree

The sample index structure of presort-nearest location tree is shown in figure 4.5. For this calculation, center location which includes latitude and longitude is input as 24 and 100 with service radius 100 km. The service range location that has maximum and minimum latitudes and longitudes are produced the value of 23.100, 24.899 (minimum latitude, maximum latitude) and 99.0155, 100.9844 (minimum longitude, maximum longitude). Then, the proposed tree is built according to mobile locations in the service range. All of these locations are generated from the nearest positions by level order.



**Figure 4.5 Presort-Nearest Location Index Structure by Level Order**

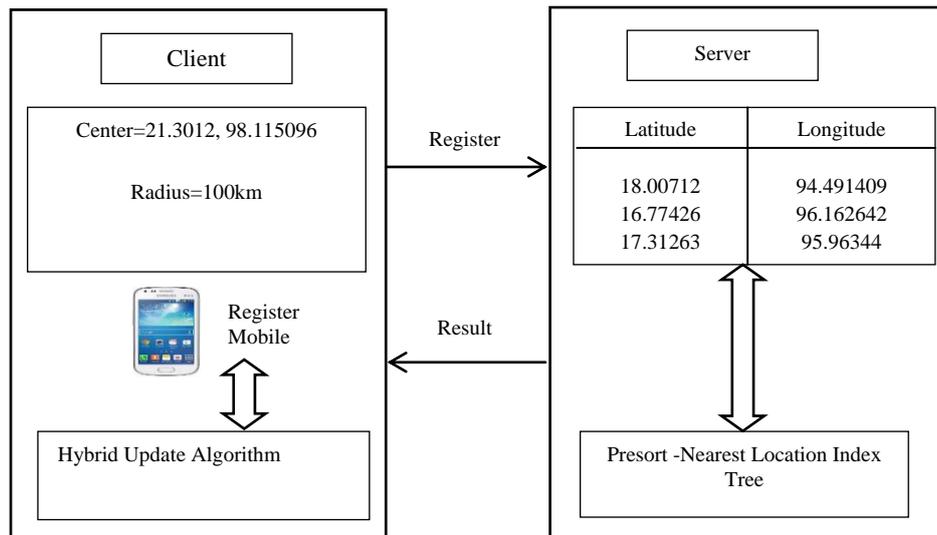
After generating the nearest mobile locations by level order, the message will be sent to the mobile users depending on the status of nearest locations. For sending message appropriately, there are three message statuses exist such as "Success", "Already Sent" and "Fail". A message will be sent only once and then "Success" status is shown together with the message if it is successfully delivered to the mobile users. If not, Fail status will be seen with the message type. It may have some users who are already received the message but they are still in the range, the message will

not need to send again. In this situation, "Already Sent" status is controlled and duplicate messages will not be received by each of the mobile users. Therefore, the notification message can easily be sent from the nearest mobile locations in the range. The figure 4.6 shows the delivery of message by level order with message arrival time.

```
-----LEVEL (2) -----
<< Success >>: Good Weather to regid_77 (24.003146, 100.28794)
<< Success >> Good Weather to regid_102 (24.045191, 100.740739) [21:42:04]
-----LEVEL (3) -----
<< Success >>: Good Weather to regid_122 (23.875261, 99.873917)
<< Success >>: Good Weather to regid_154 (23.71053, 100.32681)
<< Success >>: Good Weather to regid_42 (24.23908, 99.398467)
<< Success >> Good Weather to regid_148 (24.27276, 99.36762) [21:42:08]
-----LEVEL (4) -----
<< Success >>: Good Weather to regid_124 (23.701285, 100.051437)
<< Success >>: Good Weather to regid_58 (23.694562, 100.702099)
<< Success >> Good Weather to regid_50 (24.343625, 99.84413) [21:42:11]
<< Success >>: Good Weather to regid_195 (24.569405, 99.236397)
-----LEVEL (5) -----
<< Success >> Good Weather to regid_28 (23.692355, 99.875067) [21:42:13]
<< Success>>: Good Weather to regid_23 (23.68297, 100.325871)
<< Success >>: Good Weather to regid_150 (24.571354, 100.766499)
<< Already Sent >>: Good Weather to regid_16 (24.730367, 100.012243)
-----LEVEL (6) -----
<< Success >>: Good Weather to regid_93 (23.567519, 100.712698)
<< Success >>: Good Weather to regid_14 (23.483985, 100.86734)
-----LEVEL (7) -----
<< Success >> Good Weather to regid_70 (23.299651, 99.61424) [21:42:20]
<< Success >> Good Weather to regid_170 (23.237851, 99.576512) [21:42:21]
-----LEVEL (8) -----
<< Success >> Good Weather to regid_199 (23.102035, 99.165767) [21:42:21]
```

**Figure 4.6 Sending Message by Level Order**

#### 4.4 System Model



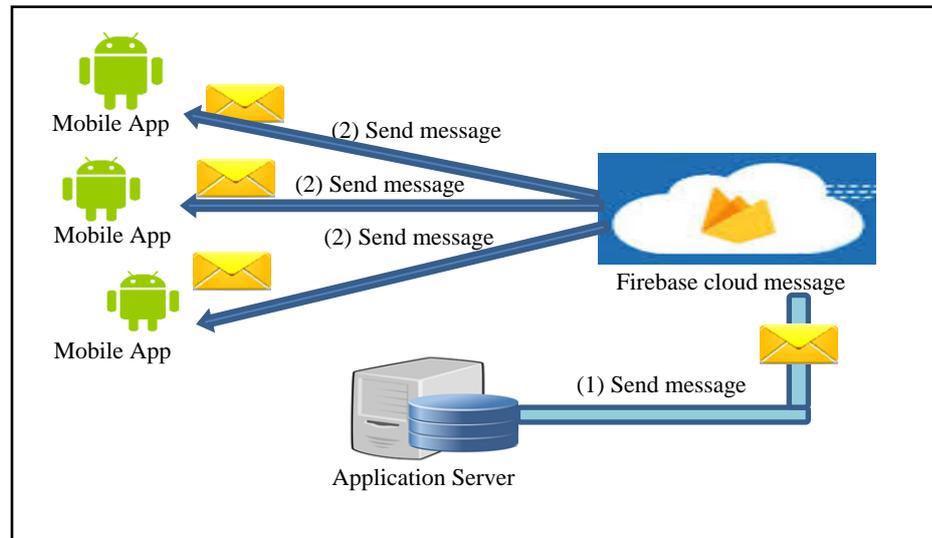
**Figure 4.7 Client-Server System Model**

A model, searchable model is built to incorporate dynamic attributes in presort- nearest location index tree and query processing. The flow of this client-server system model is shown in figure 4.7. This includes a server and a collection of registered mobile objects. In order to keep the location information up to date, these objects regularly send their updated positions to the server. Unnecessary updates wouldn't be performed at the server because Hybrid Update Algorithm is applied to the client side. The required information query the server with range queries like "which mobiles are currently located within a disaster area?" To process such queries efficiently, the server maintains an index tree that, in addition to speeding up the query processing, is also able to absorb all of the incoming updates.

#### 4.5 Multicast Messaging System

Multicast messaging is a "one-to-many" technique which sends messages from a single source to as many destinations as express a specific interest in receiving it. It should be used instead of broadcast traffic if messages want to send only to the mobile users who actually need to receive them. Multicast doesn't consume CPU and bandwidth resources than broadcast. Therefore, multicasting prevents unwanted message transmission and avoids clogging of the network. firebase server is suitable to use multicast notification between mobile device and service provider. The

multicast communication between mobile phone, firebase message server, and the application server is shown in figure 4.8.



**Figure 4.8 Communications between Mobile Phone, FCM and Server**

In this figure 4.8, there are three components that are incorporate each other to send the message. The application server sends a message to the firebase cloud messaging server. Then the Firebase server sends this message to the mobile users who are in the defined service range by multicast type. The relationship between these components is available based on the following three points.

1. Mobile Application registers to firebase cloud messaging (FCM) that generates a token ID.
2. The server authenticates with FCM by server key.
3. It communicates with FCM by token ID for sending notification.

#### **4.6 Notification System Steps**

The notification system is processed by mobile locations and connected to the firebase cloud messaging service that provides persistent connection for multicast notification process. The detail steps of this proposed system has the following phases:

- a. Information Acquiring and sending: The application server acquires disaster information from disaster server and sends a message through a

firebase cloud server to the intended mobiles. The cloud server sends all of the messages by multicasting with a push type of service.

- b. Location Tracking and Updating: Android Application gets the user's mobile position with latitude and longitude of location by Google API. It connects to the application server and sends the current and update positions by update policy conveniently.
- c. Range Searching: The disaster area is defined by a range, and then the mobile locations which are within the disaster region search by circular range queries.
- d. Index Tree Building: It is used for handling of two-dimensional mobile locations systematically. It stores mobile locations structurally and supports dynamic and continuous range queries and nearest queries.
- e. Device Messaging via Cloud: Firebase cloud server sends a message to the application on the android devices which are in the disaster area. Devices receive these messages with pop up notifications.
- f. Notification: Whenever a disaster occurs, users in the disaster region receive notification that shows outside of the Android applications' UI. A notification message consists of disaster information and emergency guideline.

#### **4.7 Scheduler Process Module**

Scheduler process is so important requirement for the moving object applications. It allows building customized Web service at the server. Especially in continuous range search, working with a lot of mobiles that have both getting and updating locations to the database can be a problem. Spring boot provides for the solution of this problem. Besides, Spring boot has persistent trigger and it can retain persistent information in its own repository. In this system, Spring scheduler is used that supports consistency scheduling different in JDKs like J2SE and J2EE environments. To schedule tasks without re-compiling and re-deploying the entire system, spring scheduler is used that supports as the abstraction layer with flexibility and loose coupling.

In this system, each execution of the range search is independent so that the beginning of the range search execution doesn't wait for the completion of the previous range search. As a result, multiple range searches are allowed in this system.

In this system, the number of milliseconds to delay with a string value at the initially scheduled task is used. To run every 25 seconds, the fixed initial delay string is added to the server side program such as @Scheduled (fixed Rate String = "25000"). By using Spring scheduler annotation @Scheduled, it runs automatically as soon as the server starts.

#### 4.8 Process Flow Using Scheduler

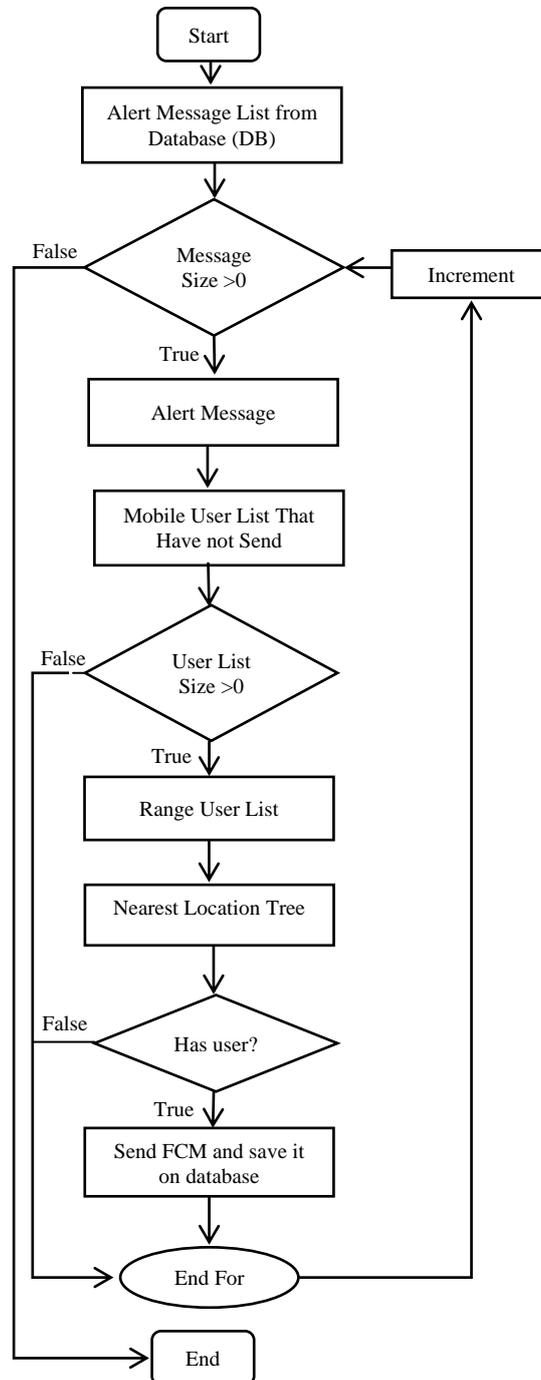


Figure 4.9 Process flowchart of System with Scheduler Process

The process flow of a system that uses scheduler is shown in figure 4.9. This flow explains the detail processes and steps of the server-side module and its database. The server intends to send a notification message to the firebase cloud server. To send this notification message, the system does the following steps. Firstly, message lists query from the database and checklists that have to send. Based on the message lists, mobile users lists those are not yet received by notification message query from the database and check user lists that have to receive. At the same time, the mobile user lists are checked whether they are in range or not. If they exist, presort-nearest location index tree is built and continuous range query is executed. Then, the nearest locations in the range are produced by level order. The notification message is sent to a firebase cloud server with the nearest mobile lists. Moreover, the users who received the message are saved in the database thus the users will not receive the duplicate message from this server. All of these processes are automatically run by Spring scheduler at a fixed period. The system does not have to send any message or the mobile users are not in range, the scheduler is stopped and finished.

#### **4.9 Virtual Mobile Dataset Generation**

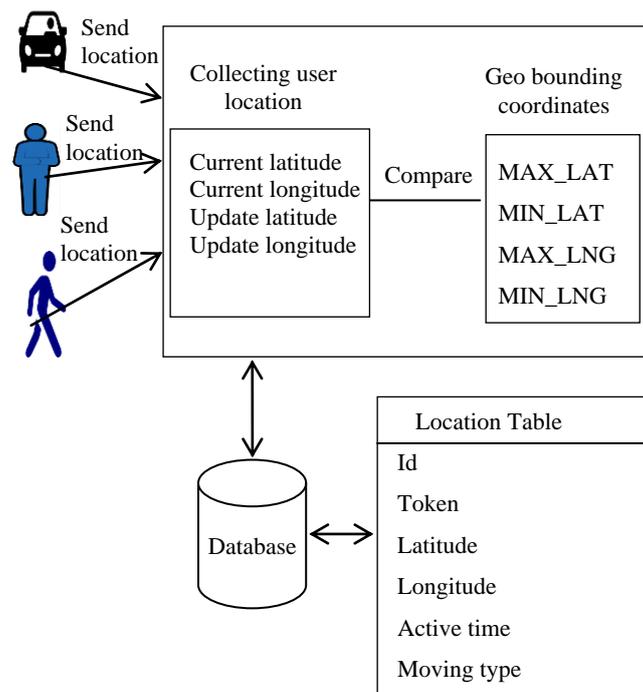
Synthetic data is information that's artificially manufactured rather than generated by real-world events. Synthetic data is created algorithmically, and it is used as a stand-in for test datasets of production or operational data, to validate mathematical models and, increasingly, to train machine learning models. The benefits of using synthetic data include reducing constraints when using sensitive or regulated data, tailoring the data needs to certain conditions that cannot be obtained with authentic data. However, the virtual mobile dataset has some drawbacks that occur the original dataset includes inconsistencies and difficult to replicate instantly. This inability may bias to produce an accurate dataset that nearly the same as real data.

Generation of synthetic datasets is a common practice in many research areas. Such data is often generated to meet specific needs or certain conditions that may not be easily found in the original, real data. The nature of the data varies according to the application area and includes text, graphs, social or weather data, among many others. The common process to create such synthetic datasets is to implement small scripts or programs, restricted to small problems or to a specific application. In this system, a

generator is proposed designed to generate high dimensional mobile datasets. The data creation is driven by three parameters which are drive, walk, and stationary based on mobile user behaviors. First, a grounding dataset is created according to given inputs, and then updating is done according to predefined time and distance thresholds by three behaviors. It can successfully be used to create synthetic datasets for moving mobile users.

#### 4.10 Moving Object Generator

Many datasets are generated synthetically according to environmental needs such as animal, mobile user, bus, Hurricane dataset, and so on. Also, a synthetic dataset usually comes from application-dependent generation. Different datasets have different applications and usage. Therefore, all of the researchers should think about their domain of interest before making any dataset synthetically. It has to compare the behaviors and functions of real data so that it seems realistic. A suitable way that can generate synthetic dataset is deriving a model or architecture with valid properties of real data. An appropriate and valid architecture, a moving objects generator is proposed. The architecture of proposed moving objects generator is shown in figure 4.10.



**Figure 4.10 Architecture of Moving Object Generator**

In each of the mobile user, different action and characteristic take different location update. This generator automatically runs and processes by taking as a thread function with `run()`, `wait()` and `sleep()`. In this architecture, synthetic mobile users' dataset is generated along with their behaviors and activities. Practically, mobile users are generally located at stationary or no moving type, sometimes they drive, and some are walking. Especially for updating moving data, one of the suitable forms is updating their process based on their motions. As an example, a user who is walking and a user who is driving will not be the same motion as the required update.

#### 4.10.1 Mobile Location Generator

It is used to create virtual mobile location dataset that seems real mobile locations. There are two main steps included in implementing this generator.

- (a) Generate locations which include latitude and longitude between maximum and minimum location coordinates.
- (b) Update latitude and longitude consistency.

Both of these above two steps are done alternately to produce synthetic mobile locations. The procedure of mobile location generator is the following.

##### Procedure: Synthetic Mobile Location Generator

**Input:** generate locations Lat and Lng with latitudes and longitudes within  $\epsilon$

//  $\epsilon$  is (maxlat minlat 28.5, 9.6 and maxlon minlon 101.17, 92.2)

// Lat and Lng is randomly generate latitudes and longitudes

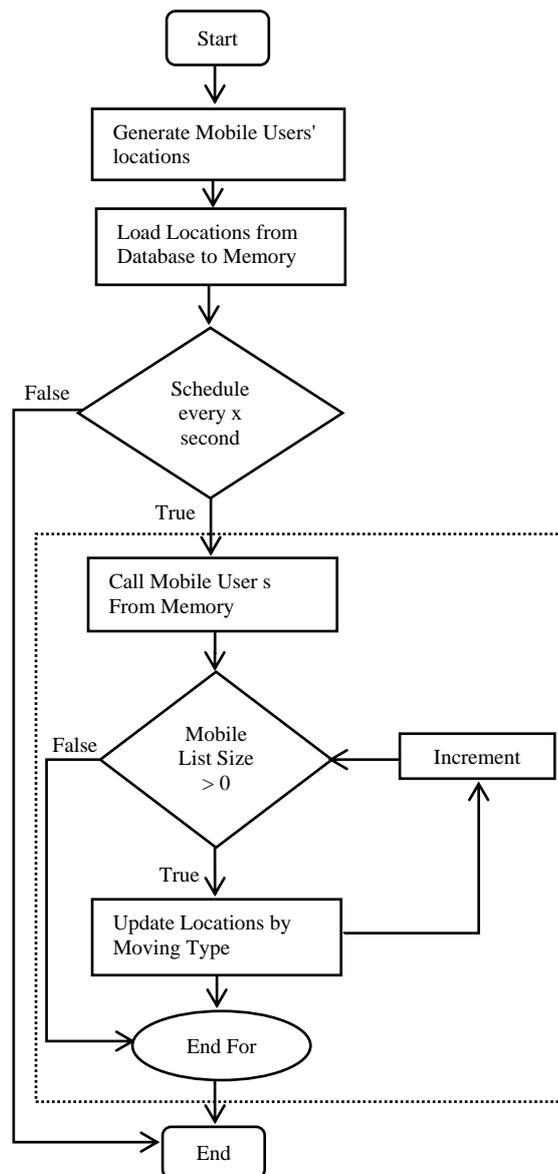
**Output:** Synthetic mobile locations

- i. for each `regId=1` to number of users;  $R=6378.1\text{km}$ 
  - a. `latitude=minLat+(Math.random()*((maxLat-minLat)+1));`
  - b. `longitude=minLon+(Math.random()*((maxLon-minLon)+1));`
  - c. `bearingAngle = Math.random() * ((360 - 0) + 1);`
- ii. update latitude and longitude consistency
  - a. `updateLat = latitude+ asin(sin(latitude) * cos(distance / R) + cos(latitude) * sin(distance / R) * cos(BearingAngle));`
  - b. `updateLon = longitude + atan2(sin(BearingAngle) * sin(distance/ R) * cos(latitude), cos(distance / R) - sin(latitude) * sin(updateLat));`

Since the mobile user location has its own latitude and longitude, each of them is calculated separately. After getting random locations, all of these are divided by

three types of mobile users who are walking, driving and stationary according to the time and distance threshold values along with their updates are done for each. At the second step, a location update policy is drawn as mobile users' behaviors and actions are quite different. In this step, time and distance based location update strategies are used to separate mobile users' types. The output will be a synthetic dataset that includes finding locations and updating them appropriately.

#### 4.10.2 Process Flow of Generating Mobile Dataset



**Figure 4.11 Process Flow of Moving Object Generator**

The process flow of the proposed architecture on moving objects generator is clearly explained in figure 4.11. The first phase is to find the location of mobile users

synthetically because it is hard to get the millions of mobile locations in reality and it is also a way of privacy protection for the real world. After getting these locations of mobile users, all of these locations are loaded from database to memory. Then, the following two steps are done by spring scheduler that runs with the help of Spring Boot at the specific time. Firstly, mobile users are taken out of memory before doing any process. Secondly, mobile users' locations are updated by moving types after checking mobile user lists which are available for processing.

#### 4.11 Distance-based Range Search

This distance-based search determines whether registered mobiles are in the service area or not so that this system compares the distance between the mobile device and center with circle's radius. If the radius of the circle is greater than or equal to the distance, the mobile is inside the service area. The following Mobile Region Check (MRC) procedure can be used to decide the registered mobile region.

Procedure: MRC ( $m\_latitude$ ,  $m\_longitude$ ,  $c\_latitude$ ,  $c\_longitude$ , radius)

1. Initialization:  $m\_latitude$ =mobile's latitude;  $m\_longitude$ =mobile's longitude;  $c\_latitude$ =epicenter's latitude;  $c\_longitude$  =center's longitude; radius=circle's radius,  $d$ = distance between mobile device and center;

$$2. d = \sqrt{\frac{(c\_latitude - m\_latitude) * (c\_latitude - m\_latitude)}{+} (c\_longitude - m\_longitude) - (c\_longitude - m\_longitude)} \quad (4.2)$$

3. If ( $d \leq$  radius) then print: Given point is inside the service area;
4. Else print: Given point is outside the service area;

#### 4.12 Summary

This chapter presents the architectures of the proposed methods, algorithms which are applied in the multicast notification steps. The proposed system is implemented to send disaster notification to the mobile users within the disaster area. The system takes current locations of mobile users by Google API and updates them by hybrid update algorithm. The system proposes to presort-nearest location index tree structure with the availability of dynamic range queries. The required performance of this proposed index structure is evaluated by proposing a mobile object generator. The detail explanations of this generating process bring a good idea

for making any other moving objects generator. The more dataset varies, the more result different, and all of these outcomes will be used in part for incoming research. In order to better findings, virtual mobile users are created with different forms and versions and tested by using indexing in this system. The implementation of proposed system is described in chapter 5 with program demonstration, detail system process and implementation. The performance evaluations over range search queries are discussed in chapter 6.

## **CHAPTER 5**

### **IMPLEMENTATION OF THE PROPOSED SYSTEM**

Disaster Notification is the process of sending notification into registered mobiles which are in the imminent disaster area. This chapter describes the proposed approach that builds presort-nearest location index tree with the dynamic range queries. The storing of mobile locations and processing them by presorting the nearest location index structure is done thus it can speed up the query performance especially for both nearest and range searching. This index structure mainly supports not only continuous range query but also nearest neighbor query by level order in the range. In reality, millions of mobile user locations cannot get easily so that synthetic dataset for mobile objects are generated by proposing Synthetic mobile dataset procedure.

The program demonstration includes Synthetic mobile location dataset and the proposed index tree structure is firstly demonstrated in this chapter. The detail process of program demonstration steps is briefly discussed starting from the synthetic moving objects creation to sending a message to the objects in the range. The program demonstration provides a clear explanation with visual support. Then, the system implementation is explained step by step.

In this implementation includes two main parts: client-side application and server-side implementation. At the client side, it first defines two types of client application: getting location and updating location by Hybrid Update Algorithm. It then sends the current location to the application server with register ID that received it from Google cloud server that supports firebase cloud messaging (FCM). It exists as a persistent connection between thousands of Android applications and application server. At the server side, there are collecting of registered mobile locations with latitudes and longitudes, indexing user locations in the service range by proposed index tree, displaying all of these locations by level order and sending notification messages to mobile users which are in the imminent disaster area with the help of FCM. All of the mobile users in the range will receive notification message according to the nearest location level order. The continuous and dynamic range queries are available at the server side. All of the steps in the implementation are controlled by Spring scheduler that supports as the abstraction layer with flexibility and loose coupling.

## **5.1 Program Demonstration**

Program demonstration has used the ability to see the flow and features of this system with the detail tested results. It provides visual support to enhance the quality of system presentation. In this demonstration, multithreading is used to perform many operations without blocking in parallel tasks and to update the virtual mobile locations independently in the range.

The process of program demonstration includes the following four steps.

- (1) Creating synthetic moving objects and updating them
- (2) Finding moving objects within the service range search
- (3) Applying the presort-nearest location index tree based on synthetic moving objects
- (4) Sending a message to the objects in the range by level order

In this demonstration, it shows in practice how a particular step is cooperated together to be a complete mobile location indexing based on synthetic moving objects.

The detailed flow of the demonstration is the following. First, a center location which includes latitude and longitude is input and range bounding coordinates such as minimum and maximum latitudes and longitudes are calculated. Then, the numbers of mobile users in the range are output together with each register\_Id, latitude, and longitude. After that, presort-nearest location index tree is built and the location coordinates in the range are released starting from the closest locations of center location by level order. Moreover, messages will be able to send simultaneously with that output. In this system, continuous range search is supported as messages will be delivered within the specified time automatically.

### **5.1.1 Main View of Program Demonstration**

The first page of Mobile Location-based Indexing based on the Synthetic dataset by program demonstration includes three main parameters. They are the center location (latitude and longitude), and a service radius which is regarded by kilometer. For this demonstration, the synthetic dataset is generated based on the proposed procedure (moving objects generator) that produces synthetic mobile locations. In this generator, location objects or nodes are generated using random functions, categorized by different behaviors that seem realistic. The update locations are done according to the type of mobile objects. In this system, three types of mobile objects

are specified. As it intends to send notification within the disaster area, service area or range area is calculated and maximum and minimum bounding coordinates are produced. A generate and send button is provided to send notification messages by proposed nearest index structure.

### **5.1.2 Building Range Users Index Tree**

The operation of the nearest location index tree provides to send fast and appropriate notification messages. In this system, the nearest location index structure is proposed to allow maintaining and updating user locations with continuous range queries. It also supports generating nearest points by index structure from the desired query point. It places the location nodes by level order thus nodes at any distance can easily find without traveling the whole tree and the searching may reduce greatly. It gives the advantage of an index structure to easy data access and fast query along with the retrieving nearest locations from a location point. Moreover, it is harmonized to solve continuous range queries together with the nearest neighbor location queries. In addition, all mobile locations in the range will be available by the distance at one time.

### **5.1.3 Sending Level Order Messages**

The sending message to all mobile users in the range starts as soon as the emerging of nearest location index tree. As the users' positions are ordered by closest to the center point in the index structure, the message will be sent according to the index tree. It provides the users who are the nearest positions to receive the message first. The delivery of the message to the mobile users occurs when the range query starts and searches for users in the service area. The message will only be received by users who are in the service area. This process is based on a multicast messaging system and the continuous range query is provided by multithread functions.

## **5.2 System Implementation**

This system has developed a client-server communication that supports disaster notification message of the upcoming disaster. At the client side, an Android-based mobile application is provided by using a special location provider called Google API and firebase cloud messaging (FCM). This application will run on mobile

phones supported by Android version 4.0 or high upgraded versions. The mobile phone must have a location supported identification facility. This application has two major parts:

- (1) Getting current and update location of the mobile device.
- (2) Registration to get a token ID and connecting with firebase cloud messaging.

Through a special location provider called Google API, mobile phone catches the current location of its user and sends it to the server. Using the current position of the user, this system will determine whether the user is in probable disaster exposed area or not. The application registers to FCM that generates Token ID. Then the application communicates with the server not only giving token ID but also sending the location of the mobile current position. The required update position is done after reaching the predefined threshold values or constraints that are provided by the combination of distance and time-based location update policies.

At the server side, there are three parts in the Microsoft SQL Database: mobile users, alert message and mobile user message. At the first part of mobile user, there is holding mobile latitudes, longitudes, creates register dates and active times of registered mobile users. The second part is alert message where service region radius, messages for client, message start date, end date, epicenter latitude and longitude is collected. The third part is mobile user message where it uses presort-nearest location index structure and filters mobile users which are within the disaster service area. If the mobile users are within the service area by procedure and message is not expired, mobile users will get multicast message from server.

In this system, an application server is taken as a third-party server. This system has to mark the notification service region based on the center location and the service distance called circular range search. For registered users are in the imminent disaster area, the application server needs to push a message to users' Android application. It requests to send a multicast message to FCM server via an HTTP POST. Afterward, the registered android application fetches the message from the FCM server. The general work of server-side implementation includes the following two things.

- (1) Marking service range area by circular searching
- (2) Sending notification to real mobile devices in the range by FCM

### **5.2.1 Prerequisites for using Firebase Cloud Messaging**

The following documents are required for firebase cloud messaging client.

- i. Android Studio 2.2 or later
- ii. The device that has Google Play Store app
- iii. Android version 4.0 or higher
- iv. Google APIs configurations
- v. The Android SDK Manager with Google Repository
- vi. Push Android SDK 1.7 or later

### **5.2.2 Client Side Application**

The UI design of this application is very simple and user-friendly design. The main contents of the client application are location log and messages button. The location log provides to see the summary of the history and current locations of this application. The messages button is the collection of messages that are sent by the application server.

The status of location for both update and current location with respect to their date and time are shown in the location log. All of the update locations are easily seen by scrolling down the surface of the location log. The location is provided by Google API that helps to get the fast and accurate position of mobile phones by switching between available location providers such as GPS, Network Provider, and Cellular Network.

When the application server sends the required messages to the registered mobile phones, the notification message is pop up at the top of the mobile phones and save them in the application messages. In this application, the messages are available and supported to send and read for not only Myanmar language but also English language.

#### **5.2.2.1 Prerequisites for Client Application**

It is an Android based application and it work with firebase cloud messaging, local server database and push notification service. It includes the free services and configurations supported by mobile computing technologies. It is also a location-based communication service with Android technologies. The required features and their explanations for client application are explained in table 5.1.

**Table 5.1 Features and Their Explanations**

<b>Features</b>	<b>Explanations</b>
Google-map-key	To use special location provider (Google API)
Base-URL="http://172.20.10.3.8080/disaster-server/"	To connect application server and client application
Notification-sound	To send notification message with sound
Token	To save and send FCM registration service
Message adapter	To send message with title, body, and date in message list
Position adapter	To receive and update latitude and longitude of mobile location
Retrofit Request	To retrieve and upload JSON via REST based web service
Access_Find_Location	To request and find client application's location
Request_Google_play_service	To available Google API services in client application
REST	To use HTTP requests that have GET, PUT, POST and DELETE data
'com.google.firebase:firebase-plugins:1.0.5'	To add firebase configuration options
'com.google.gms:google-services:3.1.0'	To add Google service configuration options

### **5.2.3 Server Side Implementation**

All of the activations in this server are done by Spring boot [90]. It is used to get the important concepts of automatic configuration, starter dependencies, and the command line interpreter. Spring boot provides fast and wide accesses for all development and provides non-functional features. In this system, most of the processes are dynamic such as getting update locations of mobile users, searching continuous range query, dynamic index tree for mobile users in the range and sending a message to the target client applications. Therefore, Spring scheduler exists the main controller of this system.

The main menus for this system are "Message", "Mobile User" and "FCM". "FCM" menu supports sending a different message to each of the mobile users. To send multicast or broadcast message, it can be sent from the "Message" menu on the

application server. The stored user information in the database is shown in the "Mobile User" menu. The message title is available up to 80 words including spaces and message body can be sent up to 200 words with spaces. The application server delivers a message to the FCM server firstly, and then the FCM server multicasts this message to all mobile users which are in the service area. For users who are in the range but they are offline, this message is maintained by FCM server before the message is expired.

The collection of required information for mobile users is stored in the database. As soon as the client application is installed and its location is available, token ID and its current location are received from the application server with the help of FCM and special location provider called Google API. For each mobile user, there is a user token that receives from firebase server, its current location that has latitude and longitude, the create date of this application, and its active time. The current location is updated as soon as the server gets user locations from the mobile application. The places for all of the registered mobile locations are available and connected to the Google Map on the mobile user Web page.

To send a message from the application server, the message number, message status, center location of latitude and longitude, service radius with kilometer are provided on the Web page. Besides, the important message title, message text, message date and expiry date that has detail hour, minute and second are described and the direction of center location is also available by the Google Map on the notification message Web page. The system provides both Myanmar language and English language to send the messages.

### **5.3 Summary**

This chapter describes the design and implementation of the proposed system by displaying the output results. Program demonstration is added so that it can clearly understand the flow of this system and the proposed methods. The structure of the proposed index tree for mobile locations in the range incorporates in this demonstration. It displays a more understandable form of sending the message to mobile users. Besides, the step by step explanation is clearly described not only at the client side but also server side. The places for both mobile users and the service area with center location are displayed on the server side Web page. Moreover, this chapter explains the cooperation of Spring boot with a scheduler and the proposed

methods in Web services. The detail process starts from getting the current location to sending the notification to the real mobile devices in the range by FCM. The evaluation results of the proposed system are discussed in chapter 7.

## **CHAPTER 6**

### **EXPERIMENTAL RESULTS**

In this chapter, the experimental study is discussed on performance evaluations of the proposed presort-nearest location index tree with the comparison KD tree, presort range tree and distance-based range searching. The results are attained by the required preprocessing, range query, nearest query, and total execution. The goal of this chapter is to evaluate the performance of presort-nearest location index tree for moving mobile objects. There are two main parts in this chapter. The first part is generating synthetic mobile objects' dataset. The second part is the evaluation of performance based on proposed index tree construction, range searching and nearest neighbor searching along with the comparison of KD tree and presort range tree. Moreover, the range searching over proposed index tree is also compared to the distance-based range searching.

An experimental setting is managed and discussed on a computer with an Intel Core i7-4590U CPU, 8G RAM, and 1-terabyte hard disk storage. The proposed synthetic location dataset generator is applied to generate mobile locations that use to simulate moving objects. This dataset basically comes from the random function of three different behaviors. In order to be a realistic mobile location dataset, it has two properties: generating mobile locations that include latitude and longitude coordinates and updating all of them consistently. This system consists of finding the nearest positions of mobile locations that change the positions in a continuous range query. Therefore, multithreading has been used in this system that can work without delay for parallel tasks.

For performance evaluations, this system tests about proposed index tree construction, its range queries and nearest neighbor queries with the comparison of using KD tree and presort range tree. Moreover, CPU time is calculated and noted with the increasing number of mobiles. To improve performance evaluations, this chapter clearly showed the tested results by different range values and number of mobiles. Experimental results indicate the efficiency of using the proposed index tree is more than using the KD tree structure for moving objects. Besides, the range query result indicates that the proposed index tree has faster processing time than the normal distance-based calculation. This is because the variation of mobile objects caused the delay of normal distance-based calculation. The large volume of mobile locations can

handle by presorting in the proposed tree structure. Moreover, the proposed structure which is presorted by input queries resist the skewed distributions of mobile objects. Therefore, the proposed structure performed the desired range queries and nearest neighbor search within the short execution time, especially in the large mobile dataset. The required parameters and their values for performance evaluations are described in table 6.1.

**Table 6.1. Parameters and Values**

Parameters	Values
Synthetic Mobile Generation Range	Min Latitude 9.6, Max Latitude 28.5 Min Longitude 92.2, Max Longitude 101.17
Number of Mobile Objects	1000 – 100,000
Object Types	Car, Walking, Stationary
Bearing Angles	0° – 360°
Update Distance	0.00832km,0.01112km,0.0102km
Schedule Fixed Rate (for Random Mobile Users)	2000msec
Schedule Fixed Rate (for Building Tree)	1000msec
Initial Delay	5000msec

### 6.1 Means for Evaluation Environment

In order to measure the performance of presort-nearest location index tree, one must secure the following four means:

1. Dataset(s): A collection of mobile object locations. Each location consists of latitude and longitude together with the registered id (reg\_id). It is generated synthetically and required not only to create mobile object locations but also to update them that seem realistic.
2. JUnit: It is an open source testing framework which is used to write and run repeatable automated tests. The experiment was performed for computing preprocessing, query and execution time for range search, nearest neighbor search and building tree construction time with a number of data set points that are organized in two dimensions.

3. **Spring Scheduler:** It is a schedule annotation in Spring that is used for task scheduling. The trigger information needs to be provided along with this annotation. The `fixedDelay`, `fixedRate` and `cron` are common services to provide the triggering information and each of these has the following property.
  - `fixedRate`- makes Spring run the task on periodic intervals even if the last invocation may be still running.
  - `fixedDelay`- specifically controls the next execution time when the last execution finishes.
  - `cron`- is a feature originating from Unix cron utility and has various options based on your requirements.
4. **JBoss Server** –It is the open source implementation that has high flexibility and powerful architecture for Java EE suite of services. It is developed as Java application server JBoss, a division of Red Hat Inc. It is simple and easy with preconfigured enterprise procedures and components. It supports the server-side implementation of Java and Web-based applications and software.
5. **Multithreading**– It is used to allow multiple thread execution at a particular time. It performs many operations without blocking in parallel tasks. It can update the virtual mobile locations independently in the range. The advantages of using multithreading are reducing development time and sharing the same memory.

### **6.1.1 Synthetic Dataset**

Synthetic mobile location data is created procedurally, and it is used as a stand-in for test datasets of moving mobile locations, to validate performance evaluation of proposed index tree and, increasingly, to train comparison of other index structure and another similar approach. These datasets are needed to simulate mobile objects when they are unavailable for millions of mobile positioning data in reality. To generate synthetic datasets appropriately, moving mobile objects are created dynamically by classifying their behaviors. In fact, different mobile objects have different velocities and movements. Therefore, location objects are generated by using random functions, which can be categorized by different behaviors in such a way that they seem realistic. It is aimed to get realistic mobile locations for performance

evaluation of presort-nearest location tree with indexing and storing dynamic location objects.

### **6.1.2 Experiments and Test Cases**

This system involves communication as well as the connection of the firebase cloud server, application server, and client application. The required location, sending and receiving the message is tested with scenarios. All of the possible positive and negative test cases and their results are lists the following.

Scenario 1: App is installed and the location is available. But it does not take any connection to the application server and FCM.

Results: Success case. It automatically connects to the FCM as soon as the application is installed. Application server gets Token ID that generates from FCM, a location of a mobile phone and saves them in the mobile user lists.

Scenario 2: The registered mobile phone is not in the range while sending a message from the application server. It reaches in the range after sending the message from the server.

Results: Success case. This system supports continuous range search with the help of Spring Scheduler so the message will be delivered as long as the message expires.

Scenario 3: The registered mobile phone is offline. But, it is in the service range area.

Results: Success case. Since it is in the range, it is included in the nearest location index structure. It will receive notification message as soon as it is online.

Scenario 4: Registered mobile phone received the first message in the range. Application server sent the second message in the same range.

Results: Success case. Messages are defined by message id so that the message will be received by mobile users if it is not the same as the previous messages. Only the duplicate message will not send to the mobile phone.

Scenario 5: Application is installed and only GPS or WiFi is available.

Results: Success case. Google API takes location for Application with the help of any location provider. Application server gets a mobile location and then the message will be received by mobile Application.

Scenario 6: The registered mobile phone is power off. The application server sends a message for it.

Results: Success case. Messages are retained by firebase cloud server until the message is not expired and the mobile phone is power on.

Scenario 7: Application is installed. It does not query any message in the range.

Results: Success case. This system is based on the push technology. Therefore, Application will receive message automatically by the service provider without requesting it at that instant.

Scenario 8: Application is uninstalled and reinstalls it again. The application server sends the message to this application.

Results: Success case. FCM generates new Token ID and Application will receive a message which is not duplicated from the server.

Scenario 9: Application is installed and GPS or WiFi is off.

Results: Partial success case. There is a background process to get a location from a mobile phone called cellular network. It takes time and needs to be in the network coverage area (i.e. using the cell network is accurate to basically 500m from cell tower).

Scenario 10: Many registered applications are in the same service range. They need a notification message instantly.

Results: Success case. FCM handles thousands of network connections between application server and application. It can send a multicast message to all mobile users in the range.

Scenario 11: The original application is shared from Zappya or file transfer applications and installed in another phone. The application server sends a message to the original application that is in the service range.

Results: Success case. The application can be shared by file transfer application or any types of transfer software. FCM generates a token ID for each application in the mobile phone. Therefore, only the original application that is in the service range will receive the message.

Scenario 12: Message has no expired date and time. Application arrives in the message service range after one month.

Results: Failure case. Messages are retained in FCM storage for a maximum of four weeks. Therefore, FCM automatically deletes this message and Application will not receive the message.

Scenario 13: Application is offline until the message is expired.

Results: Failure case. The message is sent by firebase cloud server so that it can only connect to the application with the help of network connection.

## 6.2 Evaluation of Response Time by Location Providers

In this system, a special set of communication tools, Google API is used to access the current location which is the key feature of the client application. It supports efficient communication between client and application server with the availability of Google service and other important services for messaging. One of the Google API capabilities is providing the best service based on location providers. It supports the most accurate and fastest current and updates location with the help of location providers. The accessing location analysis of basic location providers such as GPS and Wi-Fi enabled Network Provider for a client application is evaluated and described in figure 6.1 and 6.2. This experiment focuses on the results of response time available from the initial state to the difference update results states of GPS and Network provider based mobile phones. Both of these evaluations are performed in the indoor test and location updates are recorded within three minutes. The comparison of each provider is discussed and the results are explained in each provider.

### 6.2.1 Evaluation of Response Time by GPS

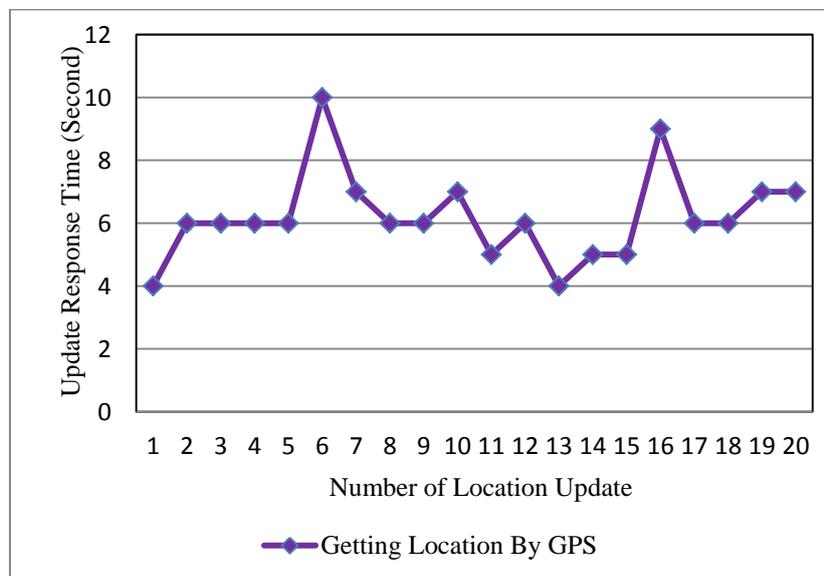
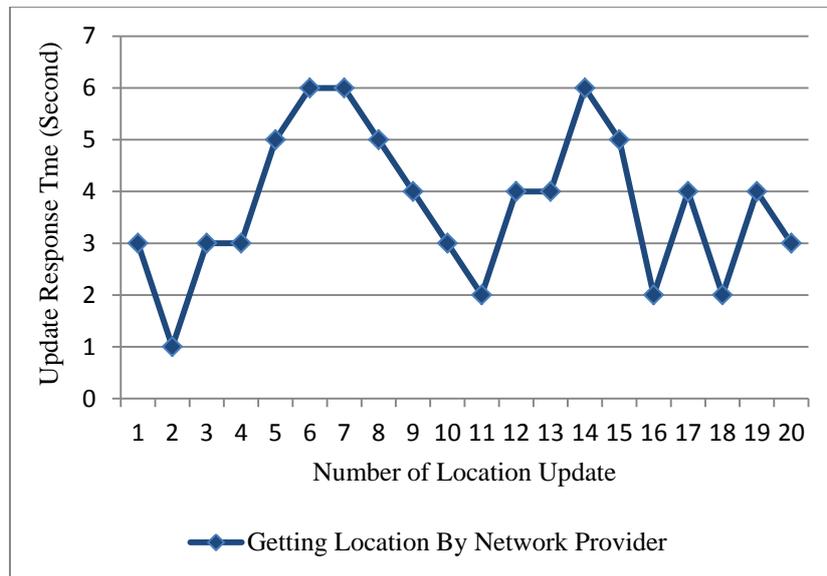


Figure 6.1 Client Application's Location Update by GPS

The status of getting location by basic location provider called GPS with the response time (second) is shown in figure 6.1. In this experiment, the client application takes GPS location once every 2 seconds as the default step. It is the indoor test and location update has been marked in 3 minutes. The result shows the response time of the availability of location update by GPS. Generally, the normal application usually takes the location from GPS more than 3 minutes to get a fix. Subsequent updates are reasonably fast (like 5-10 seconds). According to the experiment, the obtaining precise location at the start takes 30 seconds to 40 seconds. The variation of location update occurs from 4 seconds to 8 seconds. By combining the features and functions of Google API, it provides location over GPS in less than 1 minute at the first time. Besides, it gets location three times faster than normal GPS based application at the first time. Moreover, the subsequent updates are reasonably fast like 2 to 3 seconds.

### 6.2.2 Evaluation of Response Time by Network Provider



**Figure 6.2 Client Application's Location Update by Network Provider**

The result of the availability of location updates response time by Network Provider is described in figure 6.2. In this experiment, accessing the location in the client application based on the supportive of Network provider. It also tests the indoor and location updates are marked in 3 minutes. To get the more accuracy with Network provider's location, this experiment has enabled the WiFi in the setting. As the results,

the precise location at the start time is obtained within the 10 seconds to 15 seconds. The variation of location update has been found from 2 seconds to 6 seconds. According to the experiment, the starting state of getting a location from network provider is faster than the GPS.

As a result, the mobile network should be used if getting location is the initial positioning or there is low satellite visibility. Besides, it is more compatible with Google API features than GPS.

### 6.3 Evaluation of Message Arrival Time for Users

The required feature of sending notification message is receiving the right users in a suitable period of time. Message arrival time is very important especially for a limited-time alert or notification message. In this system, multicast notification is provided to all of the client applications in the service range area. Therefore, message arrival time is emphasized and prepared by mobile push services.

Then, the experimental results are tested based on the status of mobile users. The evaluations of message arrival time are discussed in figure6.3 and figure 6.4. These figures show the tested results of message arrival times to mobile phones. This experiment is based on the two potential mobile phone positions such as Online and offline mode.

#### 6.3.1 Evaluation of Message Arrival Time for Online Users

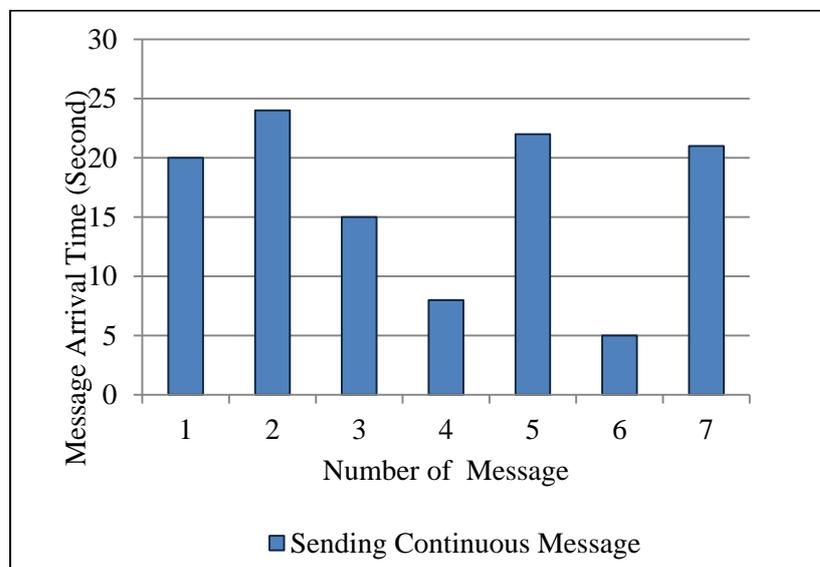
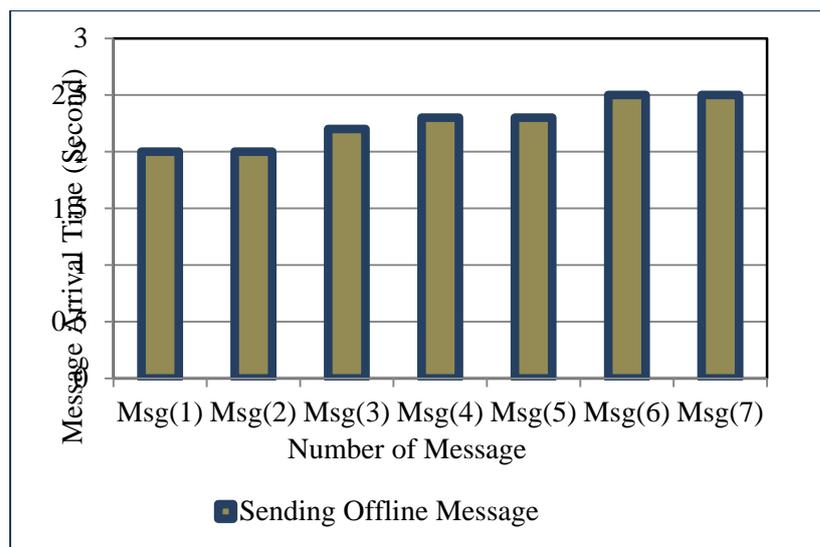


Figure 6.3 Sending Continuous Messages to Mobile Users in the Range

The test for the application server to send a continuous message is shown in figure 6.3. In this experiment, the client application is online and the server sends the messages continuously with the message title and message text. The message arrival time (second) is marked during a series of sending the message. The variation of message arrival time is from 5second to 24second. It is because sending a message based on the delay time of Scheduler and a database update message so that the message arrival time is found a few changes.

### 6.3.2 Evaluation of Message Arrival Time for Offline Users



**Figure 6.4 Sending Messages to Mobile Users Who are Offline**

Figure 6.4 shows the sending messages to mobile users who are offline. This experiment is tested while mobile phone offline and the receiving messages and their arrival times are listed as soon as the mobile phone is online. The number of test message has 7 letters and all of these were sent by the terms of the time in a range. When a mobile phone online and the message arrival time has noted before the messages expire. According to the experiment, the messages are received accurately, as well as duration is also within the 2 seconds to 3 seconds after being online.

### 6.4 Performance Evaluation Metrics

In this chapter, performance evaluations are done as the following objectives.

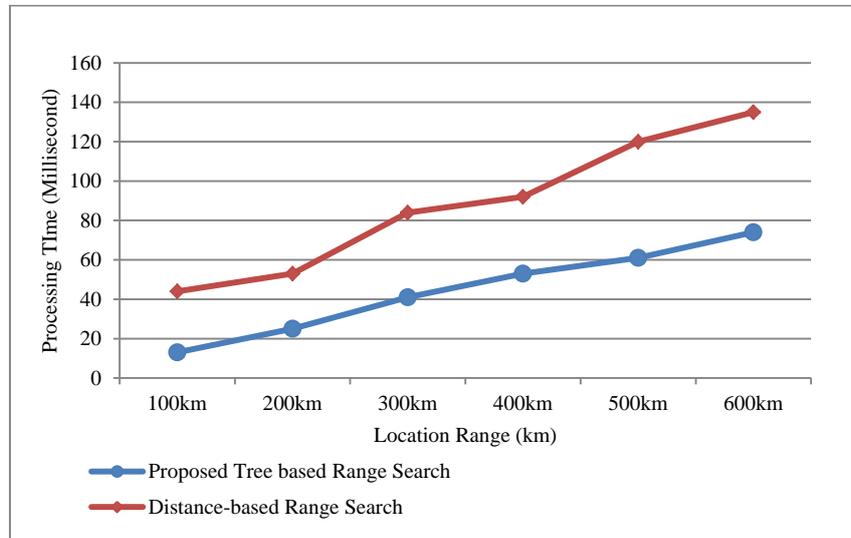
- to analyze the ability of the proposed system

- to verify the purpose of the proposed methods
- to compare the advantages and requirements with other similar methods

The important requirement for the proposed index structure is to access search queries such as range query, nearest neighbor query efficiently. As the evaluation for the above query searching time is measured, the other evaluations are done the following criteria.

- Tree Construction Time
- Processing Time
- Execution Time
- Response Time
- CPU Time

### 6.5 Evaluation of Processing Time



**Figure 6.5 Comparison of Processing Time**

In figure 6.5 shows the evaluation of the processing time of the index structure with the range searching implemented by this system. It has been conducted with various location ranges with the number of mobile locations over index structure. The processing time is tested and determined starting to the end of the computing time for index-based range query and distance-based range query. The comparison of processing time is calculated according to the following equation 6.1.

$$T_p = T_{end} - T_{start} \quad (6.1)$$

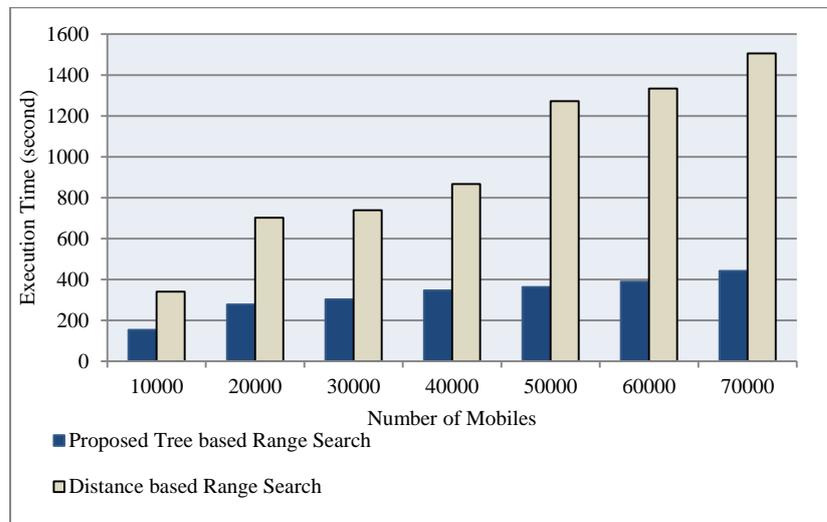
Where,  $T_p$ = indexing with range search query/processing time

$T_{start}$ = Computation start time

$T_{end}$  = Computation end time

The processing time comparison between presort-nearest location range search and distance-based range search approach is displayed in figure 6.5. The experiment is conducted upon 1million mobile locations in the database server. Each query was done 25 times and the average was recorded. The results show that both of them gradually increase in range search time along with the increasing number of location range. But the processing time for index-based range search is about two times speedy than the distance-based range searching.

## 6.6 Continuous Range Search Time



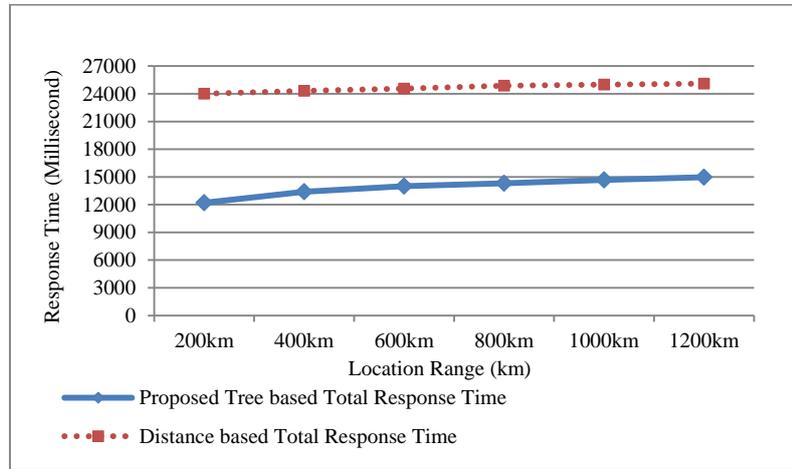
**Figure 6.6 Comparison of Continuous Range Search Time**

The execution time of range searches between proposed nearest-location index tree and distance-based method are compared in figure 6.6. In this comparison, the range queries are continuously searching in a period of time along with the number of mobile locations. There is no significant difference in the number of mobile objects in index-based range searching. Besides, all of the mobile locations are already ordered before tree structure thus it saves time and support to query performance. The more mobile variation occurs, the more execution time takes in distance-based range searching. The results of range search for the number of mobiles less than 10000 are not described in this figure 6.6. In fact, the distance-based range search execution time

with the number of mobiles less than 10000 is nearly the same time as the proposed tree based range searching.

The distance-based approach requires a repeated calculation for each mobile location because this experiment is based on not only changing the mobile locations but also searching continuous range query for these locations. Therefore, it is not difficult for small mobile location dataset but it is very time-consuming for large moving mobile location dataset. To conclude this figure 6.6, distance-based range search is slower than the proposed tree based range search especially for a large number of a dataset.

### 6.7 Evaluation of Computational Responsiveness Time



**Figure 6.7 Comparison of response time**

This system is evaluated by response time that starts message list query from the database to indexing with range query calculation. This experiment consists of various location ranges, comparing index-based range and distance-based range. It takes the time between marking a region for multicasting and sending notification by searching range query. The total response time is calculated based on the following equation 6.2.

$$T_{resp} = T_{mreg} + T_p + T_{noti} \quad (6.2)$$

Where,  $T_{resp}$  = Response Time of the notification service

$T_{mreg}$  = acquiring message by admin and marking the service region

$T_p$  = indexing with range search query/processing time

$T_{\text{noti}}$ =sending notification by multicasting

The comparison of total response time based on the proposed presort-nearest location index tree and the distance-based method is shown in figure 6.7. In this experiment, different location ranges and range search times are marked on millisecond. It conducts one million mobile locations in the database server. In this experiment, the processing time of the two approaches is slightly increased when the requested location range is greater. In summary, the total response time of the index-based approach is about two times faster than the distance-based range search approach.

## 6.8 Discussion

The experiment basically performs the comparison for range searching and total execution along with processing time and response time of proposed presort-nearest location index tree and distance-based method. The proposed index tree has three activities such as preprocessing, querying, and updating. It is compared to distance-based range searching for processing time, responsiveness time and execution time on mobile locations. The range searching time of the proposed index structure is about two times faster than the distance-based range searching. The index-based range search takes better performance when it is used for both a larger range area and the number of datasets. Especially, it needs the less number of seconds when it is used for a large number of moving dataset.

## 6.9 Comparison of Tree Construction Time

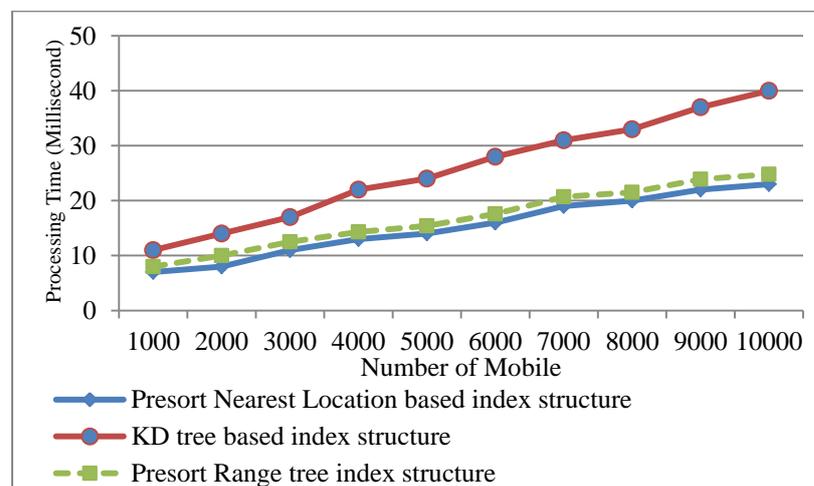
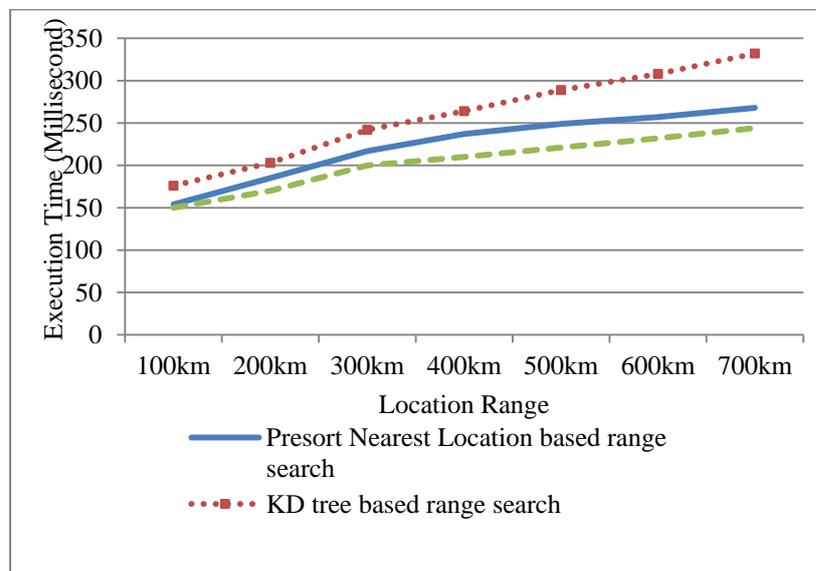


Figure 6.8 Comparison of Tree Constructing Time

Figure 6.8 describes the results of the experiment on presort-nearest location index tree, presort range tree and KD tree over tree construction time. To construct presort tree structure; the first thing is preprocessing the data into the data structure. In this process, presorting is done by each dimension and then it takes into the input parameter for building tree. This sort operation is time consuming especially for a large result set. According to values in one dimension is defined as ORDER BY clause of SQL statement such as order by latitude and order by longitude. After taking preprocessing, queries and updates on the data structure are performed. For this comparison, the results are confirmed after testing an average of 15 times with the dataset from 1000 to 10,000 mobile locations. In this experiment, the number of mobiles is generated by a proposed generator which is initialized by adding the same number of moving object types. It can be concluded that the proposed tree and presort range tree is nearly the same in tree construction. Besides, the proposed tree construction is one third faster than KD tree. This is because both comparative trees are unbalancing two-dimensional trees but the proposed tree and presort range tree input are being included the order queries that provide to be fast in tree construction. In addition, the form of the proposed index tree provides nearest neighbor locations in the range by level order in tree construction.

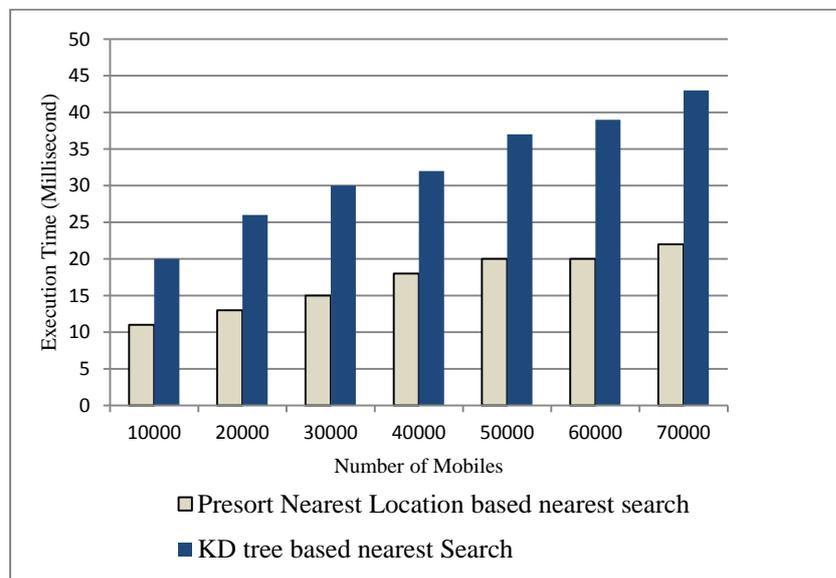
### 6.10 Comparison of Range searching Time



**Figure 6.9 Evaluation of Range Searching Time**

The figure 6.9 shows the experimental results of range searching time for presort-nearest location based range search, presort range tree based range search and KD tree based range search. It is the test for the duration of searching time within a distance of 100 kilometers to 600 kilometers ranges search queries. This experiment includes a dataset 10000 and a center latitude and longitude to take as University of Computer Studies, Yangon GPS coordinate. It was not described the tested results within a distance of less than 100 km range searching time because it was taken nearly the same in (1 millisecond). In this experiment, the larger the range size, the more increase the number of mobile locations. According to figure 6.5, all of these three index structure are adequately supportive to range searching. Especially, presort range tree supports to get fast range search than other two methods. The results describe a fast access query which is regarded as one of the features of the index tree.

### 6.11 Comparison of Nearest Neighbor Search within a Range

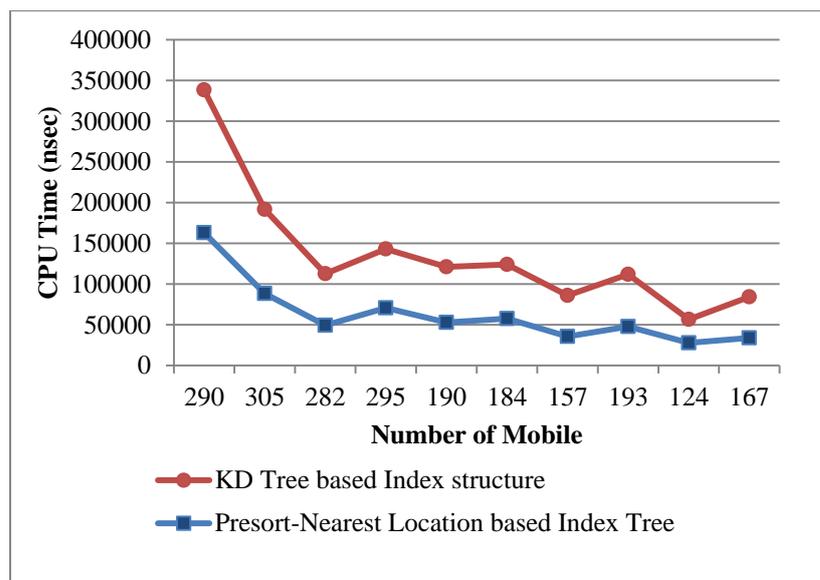


**Figure 6.10 Nearest Searching Time between proposed tree and KD tree**

The figure 6.10 shows the comparison results of the presort-nearest location index tree and KD tree that acts within a range of mobile locations from the nearest. In this experiment, the circular range is used for finding range search. To undertake the duration of the experiment, the mobile locations are firstly checked within the range or not, which has marked the time. Then, the displaying the mobile user lists by nearest called nearest neighbor search is tested and also marked the time. Such a

query can be used for not only static or stationary locations but also moving the location query that repeatedly undertakes as a continuous range query. According to the experimental results, the presort-nearest location index tree execution takes a half time in the execution of KD tree. This is because although the searching time over two comparison trees is nearly the same, the structure of presort-nearest location index tree is already supported to the nearest neighbor since the tree construction thus it can search for nearest neighbor within one millisecond.

## 6.12 CPU Time of Continuous Range Query



**Figure 6.11 CPU Time for Continuous Range Query Over Moving Objects**

The CPU time of proposed index tree with the continuous range queries is discussed in figure 6.11. As this system is based on moving mobile objects, it has used multi-threaded functions. There has been to compare the CPU time analysis because multiple operations such as continuous range search operation, update the mobile location operation and dynamic index tree operation are simultaneously executed in this system with some application active threads. To get an average timing for each range query, this system uses the Nano time function which is suitable for the exact calculation of programmed delays and actively working on a certain task. This process is started from searching maximum and minimum coordinate points and continuous range queries of the predefined service area. The new mobiles are allowed

during this process and updating is done in the mobile use lists. The results show the processing time is highest at the first time and half reduce at the second time. It has to load classes and call static blocks the first time. The variation of processing times occurs when the insertion of new mobiles is updated in the mobile lists.

### **6.13 Discussion**

The calculation of the performance evaluation over three non-balanced trees (presort-nearest index tree, presort range tree and KD tree) is based on the tree construction time, execution time during range searches and nearest neighbor searches. In this calculation, the generating of the latitude and longitude which is the source of the synthetic mobile dataset is taken by Myanmar country boundaries within the maximum and minimum latitudes and longitudes along with the proposed formula to retrieve and update. One prominent characteristic within tests found that the proposed tree (presort-nearest location index tree) and presort range tree are faster in tree building due to ordered data by queries as the input data. It can be found that the comparative trees (presort-nearest location index tree, presort range tree and KD tree) are conveniently and fast in the range search without delay. Besides, the proposed tree provided the nearest neighbor by level order since the tree building so that it does not need to give private time in the nearest neighbor searching.

### **6.14 Summary**

This chapter focuses on the experimental results of range queries and nearest query execution time over mobile objects. Normally, these queries are easily retrieved by simple calculation such as distance-based formula. But, it may take too much time for large mobile location data because it requires searching sequentially by each object in the dataset. Presorting before search queries are appropriate for unbalanced index structure along with the large and skewed mobile objects. Thus, the proposed presort-nearest location index structure supports a powerful range search queries and nearest queries. According to the experiments, the proposed structure, presort-nearest location index tree is very suitable for continuous range queries in a fixed time. Especially, it is more relevant to use skewed mobile objects indexing and range searching. Besides, it supports to get nearest locations by level order thus notification message can easily be sent to mobile users that exist from the nearest location in the desired service range. According to the experimental results, the proposed presort-

nearest location tree structure provides a better outcome: taking less execution time and processing time. The KD tree, presort range tree and proposed index tree are relevant for mobile location range searching. Besides, the proposed index structure offers more impressive range and nearest neighbor search than the distance-based method. In addition, the system selects suitable index structure and procedures that are jointly optimized to achieve availability and efficiency of delivering messages.

## **CHAPTER 7**

### **CONCLUSION AND FUTURE WORKS**

The use of location-based services is significantly increased day by day with the numerous growths of mobile technologies. This system proposes a location-based disaster notification system that has a client application, an application server and Google cloud server for firebase cloud messaging.

This system provides the solution to find the location of mobile users in a defined disaster region in a faster and efficient manner. Then, notification is sent by FCM with multicast technology. As an imminent disaster notification is valuable for mobile users so that people aware that the accurate disaster notification system is very important. Disaster takes a loss of social and economic progress. The number of dead and injured people is affected by lack of notification about the natural disaster. Thus, the integration of mobile and location-based technology application is needed for disaster information.

For this purpose, this system proposes the development of a disaster notification system based on LBS technology by using FCM and Google API. Then, the web-based prototype and system architecture are designed and developed as the application server to provide current locations within the range of the disaster area. Furthermore, an Android application is designed and created that automatically received disaster information as the message with notification. The application communicates with the server not only to send token ID but also the latitude and longitude of user's update position. For registered users who are located in the disaster area, the server sends a notification to FCM. Afterward, the application fetches the notification from FCM and store a message in the application.

The mapping of input locations from the application to the web-based notification system has been tested. It shows that the current locations can be gained by Google API timely and accurately and can be accessed directly by the application server. This system takes not to be clients who receive the same message multiple times and not to message expire before being accepted during the time-to-live period. If the target mobile device is offline, the messages will be queued by FCM and then delivered as soon as the device becomes online.

This system includes monitoring of mobile locations and updating them to the server. In getting location, when there is low satellite visibility, or it is the initial positioning, the mobile network should be used. This system explains how to incorporate dynamic attributes in presort-nearest location index tree and a model is added to deal with the overall system. There is a finding that better performance was achieved when the presorting index structure was used for a larger number of data sets for range search. The more volumes of data tests, the less number of seconds needs in the presort-nearest location index tree. It makes less computational complexity because of balancing structure. It removes the duplicate tuples thereby avoids the requirement for any further sorting when building a tree structure.

### **7.1 Advantages and Limitation of the Proposed System**

The advantages of this system are the following. First, the message can send to all without using an operator. Second, it remains notification as a message and users can get it without missing any alert. Third, the system mostly works on the server that is reducing and managing the information overload. Besides, the system will be compromised frequency of update due to the locations of mobile objects. It supports the nearest query in the range with dynamic object locations.

This system is cheap, less amount of memory needed, save time to get current locations and query processing. The cost is reduced since this system uses most of the open source software like Android and MySQL. This system sends notification by firebase cloud messaging. It is a service provided by Google and inherited by Google cloud messaging that allows sending or receiving the message between the server and mobile devices equipped with Android. FCM retains offline messages for mobile users who are in the imminent disaster area. The notification appears whether mobile users are using an application or not.

According to the cloud-based messaging system, it needs the internet connection for both of updating location to the server and sending the notification. This system supports only for the mobile users that are already registered to get notifications. The system has a program demonstration together with range query evaluation and message sending by virtual mobile locations. The testing of range query is suitable for mobile users up to 1000 service range. To test for the service range 1000 and above, the system needs to re-configure multithread functions with delay time.

## **7.2 Conclusion**

This system provides the solution to find the location of mobile users in a defined disaster region. Then, notifications or alerts will be delivered to users that save lives and properties. Therefore, the system is done for the monitoring of mobile objects, to be able to efficiently locate and answer queries related to the position of these objects in the desired time. Firstly, the system maintains the moving mobile locations and then the circular range query is available from the server. Besides, it included storing mobile locations based on the index tree structure and updating location appropriately. The system will help to be tradeoff unnecessary update due to the locations of mobile objects.

According to the difficulties of getting millions of mobile locations, a synthetic mobile dataset model is proposed along with the required framework and its process flowchart.

This system also focuses on range queries over mobile objects. Normally, these queries are easily retrieved by simple calculation such as normal distance-based formula. But, it is enough for only small data because it requires searching sequentially by each object in the dataset. It takes more time along with the large and skewed mobile objects. In fact, tree-based indexing has its own building time but it can neglect with a powerful range search queries. The presort-nearest location index tree is proposed that systematically to address and update the locations of mobile and to support desired range queries and nearest neighbor queries over mobile objects.

According to the experiments, the proposed structure is very suitable for continuous range queries in a fixed time. Besides, it is more relevant to use skewed mobile objects indexing and range searching.

The processing time of proposed index tree is faster than normal distance-based calculation. Then the response time is calculated that starts message list query from the database to range query calculation over various location ranges. The result shows that the response time for both approaches is gradually increased depending on the size of the location range; the proposed index tree is about two times faster than distance-based range searching.

## **7.3 Future Works**

This research is concerned with dynamic attributes that represent mobile location coordinates, but it can be used for other hybrid systems, in which dynamic

attributes present, for example, moving cars and temperature. The system is implemented for push-based location-based services that automatically send a message by the application server. This proposed application has a future plan to add for rescue and relief operation in the application. The system also prefers a better server-side application that can detect automatically disaster-prone area based on geographical analytic. Moreover, this system will do authentication that confirms the message sends by an unadulterated server. The system is composed of firebase cloud messaging service thus it exists as a persistent network maintaining thousands of connections for each client app between the server and user's device. As a further extension, all of the web services at the server side can be implemented using cloud computing technology.

This system focuses on an android phone for now but this system will do for windows and ios phones about push notifications as further research areas. The required security for a location-based application will be added to the client app. The proposed Hybrid Update approach at the client app will be carried out other index structures (e.g. the Quadtree, the K-D-B tree) by finding relevant threshold values. Moreover, this proposed index structure can be used to apply other moving objects such as temperature, vehicle location and so on. The results obtained from the other index tree structure can be compared to this system's results. This system will not send the duplicate message to the clients and the server keeps the message not to be during a time-to-live period.

## AUTHOR'S PUBLICATIONS

- [P1] Thu Thu Zan, Sabai Phyu “Mobile Location Based Indexing of Notification System”, International Multi Conference of Engineers and Computer Scientists IMECS (IAENG conferences), HONG KONG, ISBN: 978-988-14047-8-7 ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online), March 2018. **Lecture Notes in Engineering and Computer Science (Scimago index) Page [88-93]**
- [P2] Thu Thu Zan, Sabai Phyu “Mobile Location Based Indexing for Range Searching”, First International Conference on Big Data Analysis and Deep Learning Applications (ICBDL), Miyazaki, JAPAN, May 2018. **Advances in Intelligent Systems and Computing (Book Series Volume: 744), Springer. (Scopus index) Page [240-249]**
- [P3] Thu Thu Zan, Sabai Phyu “Range Tree based Indexing of Mobile Tracking System”, The 1<sup>st</sup> International Conference on Advanced Information Technologies (ICAIT, 2017), Yangon, MYANMAR, February 2017. **Page [145-150]**
- [P4] Thu Thu Zan, Sabai Phyu “Implementing Mobile Tracking System on Disaster Notification”, 15<sup>th</sup> International Conference on Computer Applications (ICCA, 2017), Yangon, MYANMAR February 2017. **Page [190-194]**
- [P5] Thu Thu Zan, Sabai Phyu “Query and Update Efficient Presort Range Tree based Indexing of Mobile Tracking System”, The World Congress on Engineering 2018 (WCE 2018), (IAENG conferences), LONDON, U.K, ISSN: 2078-0958 (Print), July 2018. **Lecture Notes in Engineering and Computer Science (Scimago index) Page [201-204]**
- [P6] Thu Thu Zan, “Location-based Disaster Notification System”, NAM S&T Centre Publication on The Impact of Extreme Natural Events: Science and Technology for Mitigation (IRENE), COLOMBO, SRI LANKA, July 2018.

- [P7] Thu Thu Zan, Sabai Phyu “Mobile Location Indexing Based On Synthetic Moving Objects” , **International Journal of Electrical and Computer Engineering (IJECE)** , Indonesia, December 2018. (**Scimago index –Q2**)
- [P8] Thu Thu Zan, Sabai Phyu “Reliable Multicast Notification System on Mobile Location Indexing”, **International Journal of Computer Science and Information Security (IJCSIS)**, ISSN: 1947-5500. Pittsburgh, PA, 15213, USA, July 2018. (**Scopus index**)

## BIBLIOGRAPHY

- [1] R.A.Abdeen, "An Algorithm for String Searching Based on Brute-Force Algorithm". IJCSNS International Journal of Computer Science and Network Security, Vol 11 (7), July 2011.
- [2] H.M.Abdul-Kader, "Location Updating Strategies in Moving Object Databases". International Journal of Computer Theory and Engineering, Vol 1 (1), 1999.
- [3] F.B.Adamu, A.Habbal, S.Hassan, I.Abdullahi, "A Survey On Big Data Indexing Strategies". 4th International Conference on Internet Applications, Protocol and Services (NETAPPS2015), Cyberjaya, Malaysia, December 2015.
- [4] P.K.Agarwal, L.Arge, O.Procopiuc, and J.S.Vitter. "A framework for index bulk loading and dynamization". Proceeding of 28<sup>th</sup> International Collaboration Automative Language and Program, 2001.
- [5] K.N.Ahire, "Revolutionary mobile operating system: Android". International Research Journal of Engineering and Technology (IRJET), Vol 3 (7), July 2016.
- [6] M.Ahmadi, B.Biggio, "Detecting Misuse of Google Cloud Messaging in Android Badware". 6th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), October 2016.
- [7] E.Alpaydin, "Voting Over Multiple Condensed Nearest Neighbors". Artificial Intelligent Review, 1997.
- [8] L.Arge, M.d.Berg, H.J.Haverkort,"The priority R-Tree: A Practically Efficient and Worst-Case Optimal R-tree". SIGMOD, June 2004.
- [9] İ.Ayabakan, P.Kilimci, "Moving Object Databases-Indexing Algorithms". International Journal of Computer Theory and Engineering, Vol 6 (6), December 2014.
- [10] B.Aydin, R.A.Angryk, "ERMO-DG: Evolving Region Moving Object Dataset Generator". Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference, 2014.
- [11] S.Azri, U.Ujang, F.Anton, A.A.Rahman, "Review of Spatial Indexing Techniques for Large Urban Data Management". International Symposium

& Exhibition on Geoinformation (ISG), January 2013.

- [12] S.C.Bagui, K.Pal, "Breast Cancer Detection using Nearest Neighbor Classification Rules". Pattern Recognition, Vol (36), 2003.
- [13] T.Bailey, A.K.Jain, "A Note on Distance weighted k-Nearest Neighbor Rules". IEEE Transactions Systems, Vol.8, 1978.
- [14] Prof.H.Barapatre, "Disaster Management Using Android Technology". IJRIT International Journal of Research in Information Technology, Vol 2 (4), April 2014.
- [15] N.Beckmann, H.P.Kriegel, R.Schneider, B.Seeger, "The R\*-tree: An Efficient and Robust Access Method for Points and Rectangles". Proceeding of ACM SIGMOD intl. conf. management of Data, 1990.
- [16] A.Beygelzimer, S.Kakade, J.Langford, "Cover Trees For Nearest Neighbor". Proceedings of the 23<sup>rd</sup> International Conference on Machine Learning, Pittsburgh, PA, 2006.
- [17] K.C.Brata, D.Liang, S.H.Pramono, "Location-based Augmented Reality Information for Bus Route Planning System". International Journal of Electrical and Computer Engineering (IJECE), Vol 5(1), 2015.
- [18] D.Burgstahler, U.Lampe, N.Richerzhagen, R.Steinmetz, "Push vs. Pull: An Energy Perspective". Proceedings of the 2013 6<sup>th</sup> IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013), Institute of Electrical and Electronics Engineers (IEEE), December 2013.
- [19] N.Chan, "Introduction to Location-Based Service". Geo Information Systems, August 2003.
- [20] S.Chen, B.C.Ooi, K.L.Tan, M.A.Nascimento. "ST 2 B-tree: A Self-Tunable Spatiotemporal B+-Tree Index For Moving Objects". Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM New York, USA, 2008.
- [21] M.Cheng, P.Fan, X.Lei, R.Q.Hu, "Cost Analysis of A Hybrid-Movement-Based and Time-Based Location Update Scheme in Cellular Networks". IEEE Transactions on Vehicular Technology, Vol 64(11), November 2015.
- [22] H.H.Chern, H.K.Hwang, "Partial Match Queries in Random k-d Trees". Society for Industrial and Applied Mathematics SIAM Journal on Computing, Vol 35 (6), 2006.

- [23] H.J.Cho, "Efficient Shared Execution Processing of k-Nearest Neighbour Joins in Road Networks. Mobile Information Systems, Article ID 1243289, April 2018.
- [24] Ch.Y.Chow, M.F.Mokbel, "Privacy in Location-Based Services: A System Architecture Perspective". ACM New York, USA, Vol 1(2), July 2009.
- [25] J.H.Christensen, "Using RESTful Web-Services and Cloud Computing to Create Next Generation Mobile Applications." Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA), October 2009.
- [26] T.M.Cover, P.E.Hart, "Nearest Neighbor Pattern Classification". IEEE Transaction of Information Theory, Vol. IT-13, January 1967.
- [27] S.Dhanabal, S.Chandramathi, "A Review of various k-Nearest Neighbor Query Processing Techniques". International Journal of Computer Applications, Vol 31(7), October 2011.
- [28] J.Dizdarevic, F.Carpio, A.Jukan, X.Masip-Bruin, "A Survey of Communication Protocols for Internet-of Things and Related Challenges of Fog and Cloud Computing Integration". International Joint Conference on Advances in Signal Processing and Information Technology, ACM Computing Surveys, Vol 1 (1), April 2018.
- [29] Y.Dong, H.Chen, "Grid-Index Algorithm for Reverse Rank Queries". 20<sup>th</sup> International Conference on Extending Database Technology EDBT, 2017.
- [30] A.Elashry, A.Shehab, A.Riad, A.Aboelfetouh, "2DPR-Tree: Two-Dimensional Priority R-Tree Algorithm for Spatial Partitioning in SpatialHadoop". International Journal of Geo-Information, Vol 7(5), May 2018.
- [31] A.Elazab, B.Shababa, H.Hefny, "Location Based Approach for Messaging Services". Egyptian Computer Science Journal, Vol 42 (2), May 2018.
- [32] A.Fax, C.E.Berger, J.Hughes, S.Lyon, "Spatio-temporal Indexing in Non-relational Distributed Databases". IEEE International Conference on Big Data, December 2013.
- [33] G.W.Gates, "Reduced Nearest Neighbor Rule". IEEE Transactions of Information Theory, Vol 18 (3), September 1971.
- [34] S.Ghorbani, M.H.Mobini, B.M.Bidgoli, "Continuous Mutual Nearest

Neighbour Processing on Moving Objects in Spatiotemporal Datasets". International Journal of Information and Education Technology, Vol. 7(5), May 2017.

- [35] V.C.Giner, P.G.Escalle, "An Hybrid Movement-Distance-Based Location Update strategy for Mobility Tracking". CICYT (Spain) for financial support under project number TIC2001-0956-C04-04.
- [36] A.Gosavil, S.S.Vishnu, "Disaster Alert and Notification System via Android Mobile Phone by Using Google Map". India International Journal of Emerging Technology and Advanced Engineering, Vol 4 (11), November 2014.
- [37] G.Graefe, F.Halim, S.Idreos, H.Kuno, S.Manegold, "Concurrency Control for Adaptive Indexing". Proceedings of the VLDB Endowment, Vol 5(7), August 2012.
- [38] R.H.Guting, M.Schneider, "Moving Objects Databases", Theory Book, 2005.
- [39] A.Gutmann, "R-trees: A Dynamic Index Structure For Spatial Searching". Proceeding ACM SIGMOD international Conference Management of Data, 1984.
- [40] M.Ilyas, I.Mahgoub, Mobile Computing Handbook. 2013.
- [41] C.S.Jensen, D.Lin, B.C.Ooi, "Query and Update Efficient B+-Tree Based Indexing of Moving Objects". Proceedings of the Thirtieth international conference on Very large databases, Vol 30, September 2004.
- [42] Z. Ji, I. Ganchev, M.O.Droma, Q.Zhao, "A Push-Notification Service for Use in the UCWW". 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, IEEE Computer Society, Shanghai, China, 2014.
- [43] A.John, M.Sugumaran, R.S.Rajesh, "Indexing And Query Processing Techniques In Spatio-Temporal Data". ICTACT Journal On Soft Computing, Vol 6 (3), April 2016.
- [44] H.R.Jung, M.B.Song, H.Y.Youn, U.M.Kim, "Evaluation of Content-Matched Range Monitoring Queries over Moving Objects in Mobile Computing Environments". Sensors (Basel), Vol 15 (9), September 2015.
- [45] S.Kamur, M.A.Qadeer, A.Gupta," Location based services using android

- (LBSOID)". 2009 IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA), January 2010.
- [46] T.Kanda, K.Sugihara, "Two-Dimensional Range Search Based on the Voronoi Diagram". Lecture Notes in Computer Science Vol 15, January 2005.
- [47] H.Y.Kang, J.S.Kim, K.J.Li, "Similarity measures for trajectory of moving objects in cellular space". Proceedings of the 2009 ACM Symposium on Applied Computing (SAC), January 2009.
- [48] G.Kollios, D.Gunopulos, V.J.Tsotras, "On Indexing Mobile Objects". Proceeding of ACM, 1999.
- [49] M.Kratky, V.Snasel, J.Pokorny, P.Zezula, "Efficient Processing of Narrow Range Queries in Multi-dimensional Data Structures". International Database Engineering & Applications Symposium, December 2006.
- [50] S.Kumar, Veeramani, "GPS Location Alert System". IOSR Journal of Computer Engineering (IOSR-JCE), Vol 16 (2), April 2014.
- [51] D.Kwon, S.Lee, S.Lee, "Indexing the Current Positions of Moving Objects Using the Lazy Update R-Tree". Proceedings Third International Conference on Mobile Data Management MDM, February 2002.
- [52] K.Y.Lam, Ö.Ulusoy, T.S.H. Lee, E.Chan and G.Li, "Efficient Method for Generating Location Updates for Processing of Location-Dependent Continuous Queries". Proceedings Seventh International Conference on Database Systems for Advanced Applications, Hong Kong, China, April 2001.
- [53] F.Lardinois, "Google Acquires Firebase to Help Developers Build Better Real-Time Apps". October, 2014.
- [54] L.Lazareska, K.Jakimoski, "Analysis of the Advantages and Disadvantages of Android and iOS Systems and Converting Applications from Android to iOS Platform and Vice Versa". American Journal of Software Engineering and Applications, Vol 6 (5), 2017.
- [55] Y.Lee, S.Song. "Distributed Indexing Methods for Moving Objects based on Spark Stream". International Journal of Contents Vol 11(1), 2015.
- [56] M.Li.Lee, W.Hsu, C.S.Jensen, B.Cui, K.L.Teo, "Supporting Frequent Updates in R-Trees: A Bottom-Up Approach". Proceedings of the 29th

VLDB Conference, Berlin, Germany, 2003.

- [57] J.A.C.Lema, L.Forlizzi, R.H.Gauting, E.Nardelli, M.Schneider, "Algorithms for Moving Objects Databases". The Computer Journal, 2003.
- [58] L.Liu, R. Moulic, D. Shea, "Cloud Service Portal for Mobile Device Management." Proceedings of IEEE 7th International Conference on e-Business Engineering (ICEBE), January 2011
- [59] F.Meng, R.Akella, M.L.Crow, B.McMillin, "Distributed Grid Intelligence For Future Microgrid With Renewable Sources And Storage". North American Power Symposium 2010, September 2010.
- [60] H.Munaga, V.Jarugumalli, "Performance Evaluation: Ball-Tree and KD-Tree In The Context of MST". International Joint Conference on Advances in Signal Processing and Information Technology, October 2012.
- [61] M.A.Nascimento, "Synthetic and Real Spatiotemporal Datasets". Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2003.
- [62] D.F.Nettleton, "A Synthetic Data Generator for Online Social Network Graphs". Research Gate, July 2016.
- [63] S.J.Oh, "Mobile Locality based Location Management Scheme". International Journal of Software Engineering and Its Applications, Vol 8 (2), 2014.
- [64] R.Panigrahy, "An Improved Algorithm Finding Nearest Neighbor Using Kd-trees". Lecture Notes in Computer Science, Springer, 2008.
- [65] J.Pauty, D.Preuveneers, P.Rigole, Y.Berbers, "Research Challenges in Mobile and Context-Aware Service Development". Workshop on Research Challenges in Mobile and Context-Aware Service Development FRCSS, 2006.
- [66] V.A.Paz-Soldan, R.C.Reiner, A.C.Morrison, S.T.Stoddard, U.Kitron, T. W.Scott, J.P.Elder, E.S.Halsey, "Strengths and Weaknesses of Global Positioning System (GPS) Data-Loggers and Semi-structured Interviews for Capturing Fine-scale", HumanMobility: Findings from Iquitos, Peru, June 2014.
- [67] N.Pelekis, C.Ntrigkogias, P.Tampakis, "Hermoupolis: A Trajectory Generator for Simulating Generalized Mobility Patterns". Part of the

Lecture Notes in Computer Science book series (LNCS, volume 8190), 2015

- [68] G.P.Pollini, C.Lin, "A Profile-Based Location Strategy and Its Performance". IEEE Journal on Selected Areas in Communications, Vol 15 (8), October 1997.
- [69] K.A.Popat, P.Sharma, "Various Location Update Strategies in Mobile Computing". International Journal of Computer Applications (IJCA) (0975 – 8887) Proceedings on National Conference on Emerging Trends in Information & Communication Technology (NCETICT 2013).
- [70] K.Popat, "Analysis, Design and Comparative study on Location Updating Strategies in Mobile Computing". Ph.D. Thesis, Gujarat Technological University, 2015.
- [71] Prof.Dieter, C.S.Jensen, Y.Theodoridis, "Novel Approaches to the Indexing of Moving Object Trajectories". Proceedings of the 26<sup>th</sup> International Conference on Very Large Databases, Egypt, 2000
- [72] Prof.Dieter, C.S.Jensen, "Querying the trajectories of on-line mobile objects". Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access, USA, 2001.
- [73] K.Raptopoulou, Y.Manolopoulos, A.N.Papadopoulos, "Fast Nearest-Neighbor Query Processing in Moving-Object Databases". Kluwer Academic Publishers, 2003.
- [74] S.Razzaq, M.N.Abd, E.G.Muhssan, "Object Tracking based on GPS Technology". International Journal of Advanced Research in Science Engineering and Technology, Vol 4(5), May 2017.
- [75] M.Rundle, M.Huffington, "Future of Technology". Whitepaper, 2015.
- [76] S.Saltenis, C.Jensen, S.Leutenegger, M.Lopez, "Indexing the Position of Continuously Moving Objects". Proceedings of ACM SIGMOD Conference, 2000.
- [77] Y.Selim, B.Aydin, M.Demirbas, "Google Cloud Messaging (GCM): An Evaluation". Symposium on Selected Areas in Communications: GC14SAC Internet of Things, Globecom, 2014.
- [78] J.Selke, W.T.Balke, "SkyMap: A Trie-Based Index Structure for High-Performance Skyline Query Processing". Part II of Lecture Note in

Computer Science, 2011.

- [79] L.Sharma, B.Sharma, D.P.Sharma, "Implementation of Compressed Brute-Force Pattern Search Algorithm Using VHDL". Advanced Computing, Networking and Informatics , Part of the Smart Innovation, Systems and Technologies book series, Vol (28), Springer, 2014.
- [80] D.Sidlauskas, S.Saltenis, C.W.Christiansen, J. M.Johansen, D.Saulys, "Trees or Grids? Indexing Moving Objects in Main Memory". A Database Technical Report, December 2009.
- [81] M.Singhal, A.Shukla, "Implementation of Location based Services in Android using GPS and Web Services". IJCSI International Journal of Computer Science Issues, Vol 9(1), January 2012.
- [82] N.Srivastava, U.Shree, N.R.Chauhan, D.K.Tiwari "Firebase Cloud Messaging (Android)". International Journal of Innovative Research in Science, Engineering and Technology, Vol. 6 (9), May 2017.
- [83] Y.X.Sunil, S.Prabhakar, "Q+Rtree: Efficient Indexing for Moving Object Database", Eighth International Conference on Database Systems for Advanced Applications IEEE, April 2003.
- [84] N.Syafie, Y.Nurdin, R.Roslidar, "The Development of Online Disaster Information System Using Location Based Service LBS Technology". International Journal of Informatics and Communication Technology (IJ-ICT), Vol 3(1), 2014.
- [85] S.Tabbane, "An Alternative Strategy for Location Tracking". IEEE Journal on Selected Areas in Communications, Vol 13 (5), June 1995.
- [86] C.Tamilselvi, B. Kumar," Cloud to Device Messaging with Voice Notification Using GCM". Proceedings of the World Congress on Engineering and Computer Science 2015 WCECS 2015, Vol I, October, 2015.
- [87] Y.Tao, D.Papadias, "Mv3r-tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries". Proceeding of 27<sup>th</sup> International Conference on Very Large Data Bases, 2001.
- [88] Y.Tao, D.Papadias, J.Sun, "The TPR\*-tree: An Optimized Spatio-Temporal Access Method for Predictive Queries". Proceedings of the international conference on very Large databases, 2003.

- [89] J.Tayeb, O.Ulusoy, O.Wolfson," A Quadtree-Based Dynamic Attribute Indexing Method". The Computer Journal, 1998.
- [90] G.L.Turnquist, Learning Spring Boot 2.0, Second Edition, Theory Book. November 2017.
- [91] H.L.Truong, S.Dustdar, "A Survey on Context-aware Web Service Systems". The International Journal of Web Information Systems, Vol 5(1), August 2009.
- [92] M.Varsha, S.Sonwane, "Disaster Management System on Mobile Phones Using Google Map". (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5), 2014.
- [93] B.Wang, "Mobile Location-Based Services in New Zealand". Auckland University of Technology, Master of Computer and Information Sciences, February 2008.
- [94] M.Wasif, "Mobile Location Update using Distance Method", HCL Technologies Ltd, Chennai, Tamil Nadu, India, 2000.
- [95] O.Wolfson, L.Jiang, S.Chamberlain, B.Xu, "Moving Object Databases: Issues and Solutions". Statistical and Scientific Database Management (SSDBM), 1998.
- [96] O.Wolfson, H.Yin, "Accuracy and resource consumption in tracking and location prediction". Proceedings of 8th International Symposium on Spatial And Temporal Databases", July 2003.
- [97] K.L.Wu, P.S.Yu, L.Liu, "Processing Moving Queries over Moving Objects Using Motion Adaptive Indexes". IEEE Transactions on Knowledge and Data Engineering, Vol 18 (5), May 2006.
- [98] M.L.Yiu, Y. Tao, and N. Mamoulis, "The Bdual-tree: Indexing Moving Objects by Space Filling Curves in the Dual Space VLDB", 2008.
- [99] M.Yuan, "A Typology of Spatiotemporal Information Queries". Article, January 2002.
- [100] V.Zalud, "Wireless Cellular Mobile Communications Radio engineering". Wireless Cellular Mobile Communications, Vol 11(4), December 2002.
- [101] J.H.Zhand, J.W.Mark, "A Local VLR Cluster Approach To Location Management For PCS Networks". Wireless Communications and Networking Conference, Vol 1, 1999.

- [102] W.Zhang, W.Wu, X.Yang, G.Xiang, "An Optimized Query Index Method Based on R-Tree". Computational Sciences and Optimization (CSO), 2011 Fourth International Joint Conference, May 2011.
- [103] "Firebase Cloud Messaging", <https://firebase.google.com/docs/cloud-messaging/>

## LIST OF ACRONYMS

2D	Two-Dimensional
AMM	Adaptive Monitor Method
API	Application Programming Interface
ATM	Automatic Teller Machine
CPU	Central Processing Unit
CNN	Condensed Nearest Neighbor
DBMS	Database Management System
DAT	Direct Access Table
DGI	Distributed Grid Index
FCM	Firebase Cloud Messaging
GSTD	Generating Spatio-Temporal Datasets
GPS	Global Positioning System
GCM	Google Cloud Messaging
HTTP	Hypertext Transfer Protocol
ID	Identification
J2SE	Java 2 Standard Edition
J2EE	Java 2 Enterprise Edition
JDK	Java Development Kit
JSON	Javascript Object Notation
KD	K Dimensional
KNN	K Nearest Neighbor
KDNN	K Dimensional Nearest Neighbor
KDB	K Dimensional Ball
LUR	Lazy Update R

LA	Location Areas
LBS	Location-Based Service
MACID	Media Access Control Identification
MBR	Minimum Bounding Rectangle
MCC	Mobile Cloud Computing
MOD	Moving Object Database
MV3R	Multi Version 3 Dimensional R
NN	Nearest Neighbor
PMR	Polygonal Map Region
RAM	Random Access Memory
RF	Radio Frequency
RNN	Rank Nearest Neighbor
RNN	Reduced Nearest Neighbor
SQL	Structured Query Language
STR	Spatio Temporal R
TPR	Time Parameterized R
TB	Trajectory-Bundle
WKNN	Weighted K Nearest Neighbor
WIFI	Wireless Fidelity
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol