

A DATA COLLECTION FRAMEWORK FOR SENSING PROXIMITY

***Faculty of Engineering & Technology,
Jadavpur University
In the partial fulfillment of the requirements for
the degree of
Master of Computer Science and Engineering***

Submitted by

**VRIJENDRA PRATAP SINGH
REGISTRATION NUMBER: 128992 OF 2014-2015
CLASS ROLL NUMBER: 001410502006
EXAMINATION ROLL NUMBER: M4CSE1606**

**Under the Supervision of
Dr. Sarbani Roy
Associate Professor**

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2016

Department of Computer Science and Engineering

Faculty of Engineering and Technology

Jadavpur University

Kolkata - 700032, India

C E R T I F I C A T E

This is to certify that the P.G Thesis entitled "**A DATA COLLECTION FRAMEWORK FOR SENSING PROXIMITY**" has been carried out by Vrijendra Pratap Singh (University Registration No.: 128992 of 2014-15, Examination Roll No.: M4CSE1606) as the record of the work carried out by him, is accepted as the P.G Thesis Work Report submitted in partial fulfillment of the requirements for the Degree of Master of Computer Science and Engineering. under Department of Computer Science and Engineering, Faculty of Engineering and Technology, Jadavpur University, Kolkata-700032.

.....

Dr. Sarbani Roy(Thesis Supervisor)

Department of Computer Science and Engineering

Jadavpur University, Kolkata-32

Countersigned

.....

Prof. Debesh Kumar Das

Head, Department of Computer Science and Engineering,

Jadavpur University, Kolkata-32.

.....

Prof. Sivaji Bandyopadhyay

Dean, Faculty of Engineering and Technology,

Jadavpur University, Kolkata-32.

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Jadavpur University
Kolkata - 700032, India

CERTIFICATE OF APPROVAL

The foregoing Thesis is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as pre-requisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made, opinion expressed or conclusion therein but approve this thesis only for the purpose for which it is submitted.

1. ----- 2. -----

Signature of Examiners

ACKNOWLEDGEMENT

I take this opportunity to express my deepest gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of this thesis. Foremost, I would like to express my sincere gratitude to my respected guide and teacher Dr. Sarbani Roy (Associate Professor, Department of Computer Science & Engineering, J.U.) for her exemplary guidance, monitoring and constant encouragement throughout the course of this Project. Her guidance helped me in all the time of research and writing of this thesis. Without her valuable advice and support the completion of this project could not have been accomplished.

I would like to thank our respected teacher Dr. Chandreyee Chowdhury (Assistant Professor, Department of Computer Science & Engineering, J.U.) for her valuable suggestions throughout my work. I would also express my profound gratitude and deep regards to the Head of the Department and other respected teachers of Computer Science & Engineering Department, JU, for their kind support, valuable information, advice and encouragement in completion of this thesis.

Besides my guides, I would like to thank Joy Dutta, Research Scholar, Department of Computer Science & Engineering, J.U. and entire teaching and non-teaching staff in the Department of Computer Sc. Engineering and J.U. laboratory for all their help during my tenure at J.U. I am grateful to all my friends, seniors, for their help, encouragement and invaluable suggestions. I am obliged to all staff members of Department of Computer Science & Engineering, J.U. for the valuable information provided by them in their respective fields. I am grateful for their cooperation and support during the period of my assignment.

.....

VRIJENDRA PRATAP SINGH

REGISTRATION NUMBER: 128992 OF 2014-2015

CLASS ROLL NUMBER: 001410502006

EXAMINATION ROLL NUMBER: M4CSE1606

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his Master of Computer Science & Engineering studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

NAME: VRIJENDRA PRATAP SINGH

REGISTRATION NUMBER: 128992 OF 2014-2015

CLASS ROLL NUMBER: 001410502006

EXAMINATION ROLL NUMBER: M4CSE1606

***Thesis Title:* "A DATA COLLECTION FRAMEWORK FOR SENSING PROXIMITY".**

Signature with Date: _____

DETAILED TABLE OF CONTENTS

ACKNOWLEDGEMENT

CHAPTERS

LIST OF FIGURES

CHAPTERS:

1. Introduction

1.1 Overview

1.2 Motivation

1.3 Objective

1.4 Contribution

1.5 Organisation of the thesis

2. Literature Review

2.1. Mobile Crowd Sensing and Sensing Proximity

2.2. Big Data

2.3. Sensing Proximity Based Services

3. Data Collection Framework

3.1 Our Reference data Collection Framework

3.2 Application Scenarios

3.2.1 Public safety or emergency services

3.2.2 Consumer services

3.2.3 Tourism

3.2.4 Enterprise services

4. Implementation

4.1 Android

4.2. Enabling the GPS in the mobile device

4.3. Android general concepts

4.4 Evaluation

5. Conclusion and Future Work

References

LIST OF FIGURES

Figure 1: Mobile Crowd Sensing-----	11
Figure 2: Generation of Big Data from Various Sensors-----	13
Figure 3: SPS and MCS Control Flow-----	18
Figure 4: Generation of Big Data and it's analysis in cloud-----	21
Figure 5: Reference architecture for Mobile Cloud Sensing-----	23
Figure 6: Reference architecture of Sensing Proximity Based Services-----	25
Figure 7: Location based services-----	29
Figure 8: Mobile Location Based Gaming in Real-time-----	31
Figure 9: Use of Sensing Proximity in Sports-----	31
Figure 10: Location based Tourism-----	33
Figure 12: Location based Augmented Reality-----	34
Figure 13: Relationship between the android environment, Linux-Kernel and the built in applications-----	36
Figure 13: The major components of the Android operating system-----	37
Figure 14: Current Position of the user in terms of latitude and longitude-----	39
Figure 15: Activity Lifecycle-----	41
Figure 16: Fragment Lifecycle-----	43
Figure 17: Service Lifecycle-----	44
Figure 18: Available Languages-----	46
Figure 18: Installing Google Play Services-----	53
Figure 19 The Bluetooth app-----	55
Figure 20 The client server app using Wifi-----	56
Figure 21 The accelerometer app-----	58
Figure 22 Screenshot of the client app on AVD-----	59
Figure 23 Screenshot of the server app on AVD-----	60
Figure 24 App showing places of interest in proximity-----	61
Figure 25 Screenshot Of the Code when type is cafe only in the code-----	61
Figure 26 The Corresponding Screenshot Of the App showing cafe within 1.5 km-----	62

Figure 27 Screenshot Of the Code when type specified is hospital only in the code.-----63

Figure 28 Screenshot Of the App showing hospitals within 1.5 km.-----63

Figure 29 Screenshot Of the App showing police stations within 1.5 km.-----64

Figure 30 Running of the app which shows places of interest in the vicinity.-----65

Figure 31 Screenshot of file log while running server Application-----65

Figure 32 Screenshot of memory usage over time while running application-----66

Figure 6.10. Screenshot of CPU load over time while running Application-----66

List Of Tables

Table 1 Significance of Three Axis Measurement-----57

Chapter 1

Introduction

Introduction:

Providing a reliable technology and architecture for determining the location of real world objects and people has undoubtedly enabled applications, customization, and inference. Sensing Proximity based services (**SPS**) provide the ability to find the geographical location of a mobile device and then provide services based on that location. E.g., Yahoo/Google Maps, MapPoint, MapQuest etc. This sensing gets a new dimension with the arrival of Smartphone. Mobile Crowd Sensing (**MCS**) arises as a new sensing paradigm based on the **power of the crowd** jointly with the sensing capabilities of various mobile devices, such as smart phones or wearable devices. The increasing popularity of smart phones, already equipped with multiple sensors from GPS to microphone and camera, paired with the inherent mobility of their owners enables the ability to acquire local knowledge from the individual's surrounding environment through the mobile device's sensing properties or even from the individual itself. This local knowledge ranges from location information to more specialized data such as pollution levels going through a longer list of personal and surrounding context, noise levels or traffic awareness among others.

1.1 Overview

In today's scenario, mobile computing has advanced to such an extent where the user has access to all the information on a single device. Today people are always moving with mobile devices like laptops, Smart phones, tablets etc. Using the user's geographic location, a lot of information related to the user of the mobile device can be collected. The knowledge of mobile user's location can improve the class of services and applications that can be provided to the mobile device user. These classes of applications and services are termed as location based services. Sensing Proximity Service (**SPS**) is a kind of service that helps in getting the geographical location of the user and more useful information near to the user location.

This work focuses on SPS in detail and identifies key components of SPS for providing this service to the user on the Android platform. It also explains the use and implementation of Google Maps and its APIs in getting various location based information on Android.

Application of Sensor Proximity Service (SPS)

Public safety or emergency services:

The location of the client can be determined by the mobile carrier hence it finds great use during Emergency since it can be used during the emergency/health hazard to locate the mobile clients.

Consumer services:

- a. Navigation – navigating to a particular destination using route maps, routing real time traffic etc.
- b. Location based advertising – sending advertisements to a user that is within the vicinity of a store
- c. Family and friend finder – helps in keeping track of the location of family and friends.
- d. Location based search – allows users to query on the information about its surrounding like nearby restaurant, hospital, theatre etc.
- e. Location based mobile gaming.

f. Tourism- This service will provide real-time and correct information to the tourists while they are on their trip. This service also reduces potential risks to tourists in such a way that they provide quick and easy access to information on services such as hospitals and policing. It can also alert them of any activity as well as for transaction and telebanking, destinations can make themselves ready for this by providing the necessary infrastructure needed for them to be available by sensing proximity app in real time.

Enterprise services:

By enterprise services we mean the services listed below -

- a. Firms in fleet and asset tracking
- b. Field service dispatching
- c. Route and delivery optimization
- d. Mobile workforce management

Mobile crowd sensing (MCS)

Current mobile phones are becoming multi-purpose computers for which voice transmission is just one of the functions provided to the user. Smartphones are powerful measuring devices, equipped with multiple sensors and a large data storage capacity. They also allow one to wirelessly transmit the measurement results to any location on the Internet. Smartphones are carried by the users to virtually any location they visit. Therefore, a smartphone seems to be a natural device for mobile



Fig 1: Mobile Crowd Sensing
(image courtesy: www.google.com)

measurements in order to realize the idea of crowdsensing. The mobile crowd sensing is a new sensing paradigm based on the power of various mobile devices. It is used to acquire local knowledge through sensor enhanced mobile devices, where individuals use them to collectively share data and to extract information to measure and map phenomena of common interest. Mobile crowdsensing, where individuals with sensing and computing devices collectively share data and extract information to measure and map phenomena of common interest.

Multiple applications have been proposed to use a mobile phone as a platform for the measurements and gathering of information about, for instance, human mobility, locations of

interest, air quality or wireless network connectivity. Crowdsensing enables carrying on extensive measurements covering a large area with very limited costs, but requires user participation.

Mobile Crowd Sensing Applications

Smart Cities: Worldwide, cities with high population density and a very large number of interconnected issues make effective city management a challenging task. Crowd sensing can reduce the costs associated with large scale sensing and, at the same time, provide additional human related data. For example, people can actively participate in sensing campaigns to make their cities safer and cleaner.

Road Transportation: Departments of transportation can collect fine grain and large scale data about traffic patterns in the country/state using location and speed data provided by GPS sensors embedded in cars. These data can then be used for traffic engineering, construction of new roads, etc. Drivers can receive real-time traffic information based on the same type of data collected from smart phones. Drivers can also benefit from real-time parking data collected from cars equipped with ultrasonic sensors. Transportation agencies or municipalities can efficiently collect pothole data using GPS and accelerometer sensors in order to quickly repair the roads.

Healthcare & Wellbeing: Wireless sensors worn by people for heart rate monitoring and blood pressure monitoring can communicate their information to the owners' smart phones. Typically, this is done for both real-time and long-term health monitoring of individuals. Mobile sensing can leverage these existing data into large scale healthcare studies that seamlessly collect data from various groups of people, which can be selected based on location, age, etc. A specific example involves collecting data from people who eat regularly fast food. The phones can perform activity recognition and determine the level of physical exercise done by people, which was proven to directly influence people's health. As a result of such a study in a city, the municipality may decide to create more bike lanes to encourage people to do more physical activities. Similarly, the phones can determine the level of social interaction of certain groups of people (e.g., using Bluetooth scanning, GPS, or audio sensor). For example, a university may discover that students (or students from certain departments) are not interacting with each other enough; consequently, it may decide to organize more social events on campus. The same mechanism coupled with information from "human sensors" can be used to monitor the spreading of epidemic diseases.

Marketing/Advertising: Real-time location or mobility traces/patterns can be used by vendors/advertisers to target certain categories of people. Similarly, they can run context-aware surveys (function of location, time, etc.). For example, one question in such a survey could ask people attending a concert what artists they would like to see in the future.

MCS & BIG DATA

This term is also unavoidable in this work perspective. As we are working on location based services like SRS, we can't neglect the buzzword Big Data. Basically, Big data is a term for data sets that are

so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, querying, updating and information privacy[1]. In other words, Big Data refers to datasets that grow so large that it is difficult to capture, store, manage, share, analyze and visualize using the typical database software tools. Basically, when we are working with sensors, all the sensors are generating data. We can imagine if each individual is using a Smartphone and also other smart gadgets then obviously those things will sense the environment and eventually they will generate data. Those data will be of heterogeneous in nature. Data generation rate will also be different and it is obvious that some time some data will be missed since its real time. We need to tackle this entire thing. So, we have to keep in mind that the data is collected from each citizen, then it will be obviously Big Data & we have to handle that data carefully.

Big data can be described by the following characteristics:

Volume: The quantity of generated and stored data. The size of the data determines the value and potential insight and whether it can actually be considered big data or not.

Variety: The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.

Velocity: In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.

Variability: Inconsistency of the data set can hamper processes to handle and manage it.

Veracity: The quality of captured data can vary greatly, affecting accurate analysis.

Digitized World

Drowning in Vast Amount of Data

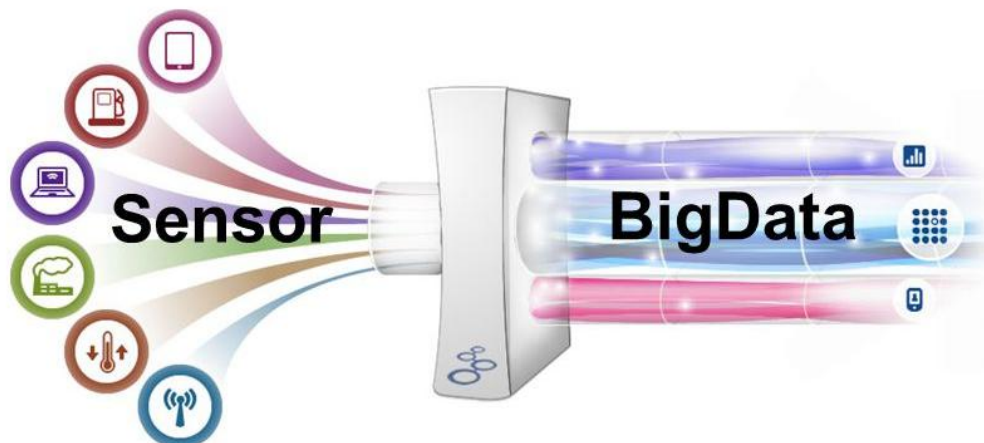


Fig 2: Generation of Big Data from Various Sensors

(image courtesy: www.google.com)

So, it is quite evident that when we want to scale up the application in City level, the data set we need to tackle will be nothing but big data set. Not only we have to gather data, but we need to

analyse also using big data analytics which will give the work a whole new dimension. So, the potential of this work is huge with big data analytics.

1.2MOTIVATION

The motivation for this sensing proximity based information system is “To assist with the exact information, at right place in real time with personalized setup and location sensitiveness”. In this era we are dealing with palmtops, Notebooks, tablets and Smart Phones, which are actually replacing the bulky desktops even for computational purposes. We have vast number of applications and usage where a person sitting in a roadside cafe needs to get information about relevant area data depending upon his current location. Such needs can only be catered with the help of SPS. These applications include general survey regarding traffic patterns, decision based on vehicular information for validity of registration and license numbers etc. A very appealing application includes surveillance where instant information is needed to decide if the people being monitored are any real threat or an erroneous target. We have been able to create a number of different applications where we provide the user with information regarding a place he or she wants to visit. But these applications are limited to desktops only. We need to import them on mobile devices. We must ensure that a person when visiting places need not carry the travel guides with him. All the information must be available in his mobile device and also in user customized format.

1.3OBJECTIVE

In this work, our main objective is to design and develop a suitable framework to sense important places of our interest from smart-phone in list form with address and contact information. This designed framework provides answers to queries like what is around, Retrieve and display all places of interest located in the surrounding area (according to user’s location e.g., list of all the gas-stations and ATMs within a distance of 1km, the fast food restaurants within 3 miles from my location. Let me know if I am near to a restaurant while any of my friends are there using client server paradigm with the help of WIFI. Retrieve and display the nearest POI (restaurants, museums, gas stations, hospitals, etc.) with respect to current location e.g., find the two restaurants that are closest to my current location.

1.4 CONTRIBUTION

In this research work, we have designed and implemented a framework for Sensing Proximity Based Services for Android Devices (e.g. smart phones/Tablets). We also present a complete framework design for client server paradigm. It is targeted at the Android platform, since Android provides a simple application model and also has a huge user base. This work provides an implementation to view and find details (e.g. address and contact) of the places of interest in a list form for a user. By places of interest we mean emergency places like Police station/Hospital or can be used for entertainment purpose like shopping mall / Café etc. within vicinity of one’s current location. With the help of our reference framework we also explored an accelerometer app which senses the X, Y and Z co-ordinate which can find bumps in a road and can communicate with other smartphones using its Bluetooth connection to inform nearby devices, if any.

1.5 ORGANISATION OF THE THESIS

In the first chapter we present the introduction about sensing proximity, its applications, mobile crowd sensing and big data, motivation for sensing proximity service, and our objective and contribution towards the research work. In the next chapter, we discuss about the Mobile Crowd Sensing Architecture, existing techniques etc. and then existing Sensing proximity based Services systems and their descriptions.

In Chapter 3, we present a suitable framework for Sensing proximity & its Architecture and various application Scenarios like in Entertainment (Restaurants, Malls, Cinema Halls), Tourism (Heritage buildings, Parks, Museums), Emergency (Hospital, Transits) next, in Chapter 4 we describe elaborately about the implementation of the framework in android app and its evaluation. Finally we conclude this work along with some important hints towards future directions.

Chapter 2

Literature Review

Literature Review

People-centric sensing with smart phones can be used for large scale sensing of the physical world at low cost by leveraging the available sensors on the phones. Despite its benefits, mobile people-centric sensing has two main issues: (i) incentivizing the participants, and (ii) reliability of the sensed data. Unfortunately, the existing solutions to solve this issue either require infrastructure support or add significant overhead on user phones. We believe that mobile crowd sensing will become a widespread method for collecting sensing data from the physical world once the data reliability issues are properly addressed. We present the concept of mobile crowd sensing and its applications to everyday life. There are several popular application domains (some of them already prototyped) that can be linked with our work, also. Some of them are explored here.

Smart Cities: Worldwide, cities with high population density and a very large number of interconnected issues make effective city management a challenging task. Therefore, several significant government and industrial research efforts are currently underway to exploit the full potential of the sensing data by initiating smart city systems to improve city efficiency by deploying smarter grids, water management systems and ultimately the social progress. For example the government of South Korea is building the Songdo Business District, a green low-carbon area that aims at becoming the first full-scale realization of a smart city. Despite their potential benefits, many of these efforts could be costly. Crowd sensing can reduce the costs associated with large scale sensing and, at the same time, provide additional human-related data. For example, a recent work on ParticipAction proposes to leverage crowd sensing to directly engage citizens in the management of smart cities; people can actively participate in sensing campaigns to make their cities safer and cleaner. Also, Barcelona has been named 'Global Smart City 2015', **prevailing over New York (USA), London (UK), Nice (France) and Singapore**. Juniper Research, analysts in the mobile and digital technology sector, considered that the Spanish city performed well across subjects like **smart grids and smart traffic management**. The company also took into account information dealing with **smart street lighting** and each city's **technological capability and social cohesion**, among other items. Moreover, while Barcelona was recognized for implementing **environmentally sustainable projects**, the research found that New York and London needed to improve their performance on this issue[2].

Road Transportation: Departments of transportation can collect fine grain and large scale data about traffic patterns in the country/state using location and speed data provided by GPS sensors embedded in cars. These data can then be used for traffic engineering, construction of new roads, etc. Drivers can receive real-time traffic information based on the same type of data collected from smart phones. Transportation agencies or municipalities can efficiently collect pothole data using GPS and accelerometer sensors in order to quickly repair the roads. Similarly, photos (i.e., camera sensor data) taken by people during/after snowstorms can be analysed automatically to prioritize snow cleaning and removal.

Smart Parking: Drivers can also benefit from real-time parking data collected from cars equipped with ultrasonic sensors. Parking issues can be resolved in real time using sensors. There are many works that is using sensors to make things smarter. Reservation-based Smart Parking System (RSPS) that allows drivers to effectively find and reserve the vacant parking spaces. By periodically learning about the parking status from the sensor networks deployed in parking lots, the reservation service is affected by the change of physical parking status. The drivers are allowed to access this cyber-physical system with their personal communication devices[3].

Healthcare:

MCS and Sensing proximity has a huge impact in healthcare perspective. So many wireless devices are there to monitor our health. Wireless sensors worn by people for heart rate monitoring and blood pressure monitoring can communicate their information to the owners' smart phones. Typically, this is done for both real-time and long-term health monitoring of individuals. Mobile sensing can leverage these existing data into large scale healthcare studies that seamlessly collect data from various groups of people, which can be selected based on location, age, etc. A specific example involves collecting data from people who eat regularly fast food. The phones can perform activity recognition and determine the level of physical exercise done by people, which was proven to directly influence people's health. As a result of such a study in a city, the municipality may decide to create more bike lanes to encourage people to do more physical activities.

Social Activity:

The phones can determine the level of social interaction of certain groups of people (e.g., using Bluetooth scanning, GPS, or audio sensor). For example, a university may discover that students (or students from certain departments) are not interacting with each other enough; consequently, it may decide to organize more social events on campus. The same mechanism coupled with information from "human sensors" can be used to monitor the spreading of epidemic diseases. Marketing/Advertising: Real-time location or mobility traces/patterns can be used by vendors/advertisers to target certain categories of people [31], [32]. Similarly, they can run context-aware surveys (function of location, time, etc.). For example, one question in such a survey could ask people attending a concert what artists they would like to see in the future. The figure below explains how SPS and MCS are related to our daily life and how they are processed, analysed and finally form an Application of our use.

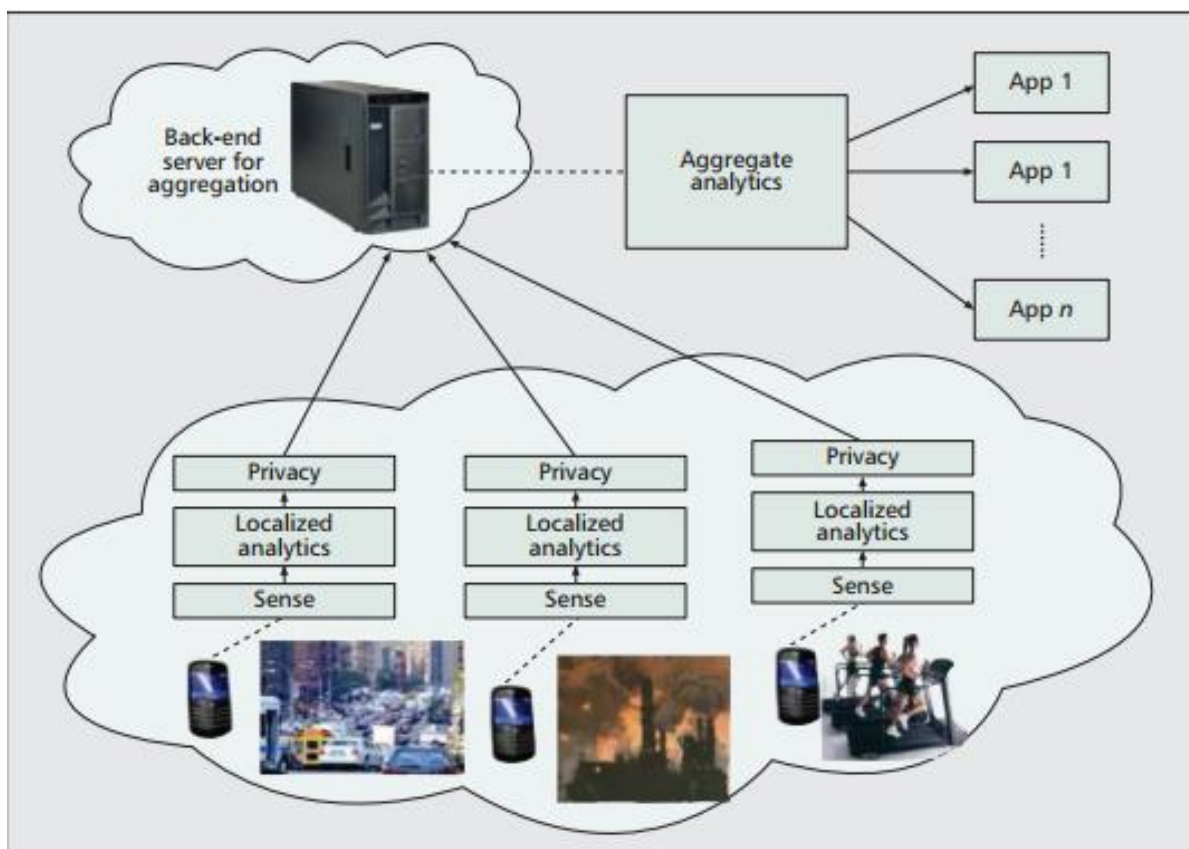


Fig 3: SPS and MCS Control Flow{image courtesy:taken from[4]}

2.1. Mobile Crowd Sensing and Sensing Proximity:

The aggregation of data or information from a group of people often results in better decisions than those made by a single person from the group. So, Crowdsourcing is a practice for obtaining needed services or content by soliciting contributions from a crowd of people, especially from an online community. Participatory sensing from companioned mobile devices is used to form participatory sensor networks for local knowledge gathering and sharing.

MCS /SPS existingArchitecture:

Mobile crowd sensing and computing (MCSC) an extension to the participatory sensing concept to have user participation in the whole computing lifecycle: (1) leveraging both offline mobile sensing and online social media data; (2) addressing the fusion of human and machine intelligence in both the sensing and computing

In the Fig 3, sensor data are collected on devices and processed by local analytic algorithms to produce consumable data for applications. The data may then be modified to preserve privacy and is sent to the backend for aggregation and mining. The architecture is followed as described below.

We illustrate the current state of the architecture of existing MCS applications and point out its drawbacks. Currently, a typical MCS application has two application specific components, one on the device (for sensor data collection and propagation) and the second in the backend (or cloud) for the analysis of the sensor data to drive the MCS application. This architecture is depicted in Fig. 3. We refer to this as each application is built ground-up and independent from each other. There is no common component even though each application faces a number of common challenges in data collection, resource allocation and energy conservation. Such architecture hinders the development and deployment of MCS applications in several ways. First, it is hard to program an application. To write a new application, the developer has to address challenges in energy, privacy, and data quality in an ad hoc manner, reinventing the wheel all the time. Further, he may need to develop different variants of local analytics if he wants to run the application on heterogeneous devices using different OSes . Second, this approach is inefficient. Applications performing sensing and processing activities independently without understanding the consequences on each other will result in low efficiency on an already resource constrained platform. There is a high likelihood of duplicating sensing and processing across multiple applications. For example, traffic sensing, air and noise pollution all require location information, but these applications would each do its own sampling without reusing the same data samples. Furthermore, there is no collaboration or coordination across devices. Devices may not all be needed (e.g., traffic sensing in a given location), especially when the device population is dense. Finally, the current architecture is not scalable. Only a small number of applications can be accommodated on each device (e.g., limitations imposed by the device operating system, human capacity to keep track of a large number of applications). Also, the data gathered from societal-scale sensing may overwhelm network and back-end server capacities, thus making the current architecture non scalable. We envision that a unifying architecture could address the current limitations of how MCS applications are developed and deployed. It will satisfy the common needs for multiple different applications. First, it should allow application developers to specify their data needs in a high-level language. It should identify common data needs across applications to avoid duplicate sensing and processing activities on devices. Second, it should automatically identify the set of devices that can provide the desired data, and produce instructions to configure the sensing activities on devices properly. When dynamic changes happen, it should adapt the set of chosen devices and sensing instructions to ensure the desired data quality. Finally, to avoid writing different versions of local analytics on heterogeneous devices, a layer that can shield the differences in physical sensor access application programming interfaces (APIs) and provide the same API upward is necessary. This makes it possible to reuse the same local analytics across different device platforms, assuming these platforms all support a common programming language such as Java.

MCS existing Techniques:

There is an exhaustive research work is actively going on this topic, out of them we will highlight here a few. MCSSENSE , CAMTON is some of the established MCS techniques.

MCSSENSE: It is a mobile crowd sensing platform that allows clients to collect many types of sensing data from smart phones carried by mobile users. McSense Android Application showing tabs and task screen for a photo task. The McSense application on the smart phones shows a registration screen for first time users. New sensing tasks can be posted by clients using a web interface running on the McSense server.

CAMTON: This system includes the tourist navigation system, exploration of the cultural heritage information and recording the on-site information and update, intelligent personalized tourist assistant and navigation services to users anywhere and anytime, a context-aware framework convert primary context to useful knowledge by means of an ontology modelling and reasoning system. The Context-Aware Mobile Touring and Navigation (CAMTON) system proposed in this study, investigates the development of a system that is able to adapt navigation and touring services to different contexts such as location, time and user profile. This system provides relevant information in each status in a computationally efficient manner. The experiments conducted using the system show that the context model and the application adaptation strategies provide promising tourism services[5].

2.2. BIG DATA:

Connecting a large number of physical objects like humans, animals, plants, smart phones, PCs, etc. equipped with sensors to the Internet generates what is called "big data." Big data needs smart and efficient storage. Obviously, connected devices need mechanisms to store, process, and retrieve data. But bigdata is so huge such that it exceeds the capability of commonly used hardware environments and software tools to capture, manage, and process them within an acceptable slot of time. The emerging and developing technology of cloud computing is defined by the US National Institute of Standards and Technology(NIST) as an access model to an on-demand network of shared configurable computing sources such as networks, servers, warehouses, applications, and services. Cloud services allow individuals and companies to use remote third-party software and hardware components. Cloud computing enables researchers and businesses to use and maintain many resources remotely, reliably and at a low cost. The MCS / SPS employs a large number of embedded devices, like sensors and actuators that generate big data which in turn requires complex computations to extract knowledge. Therefore, the storage and computing resources of the cloud present the best choice for the MCS / SPS to store and process big data. In the following subsections, we discuss the relation between the MCS/SPS and big data analytics, cloud and fog computing.

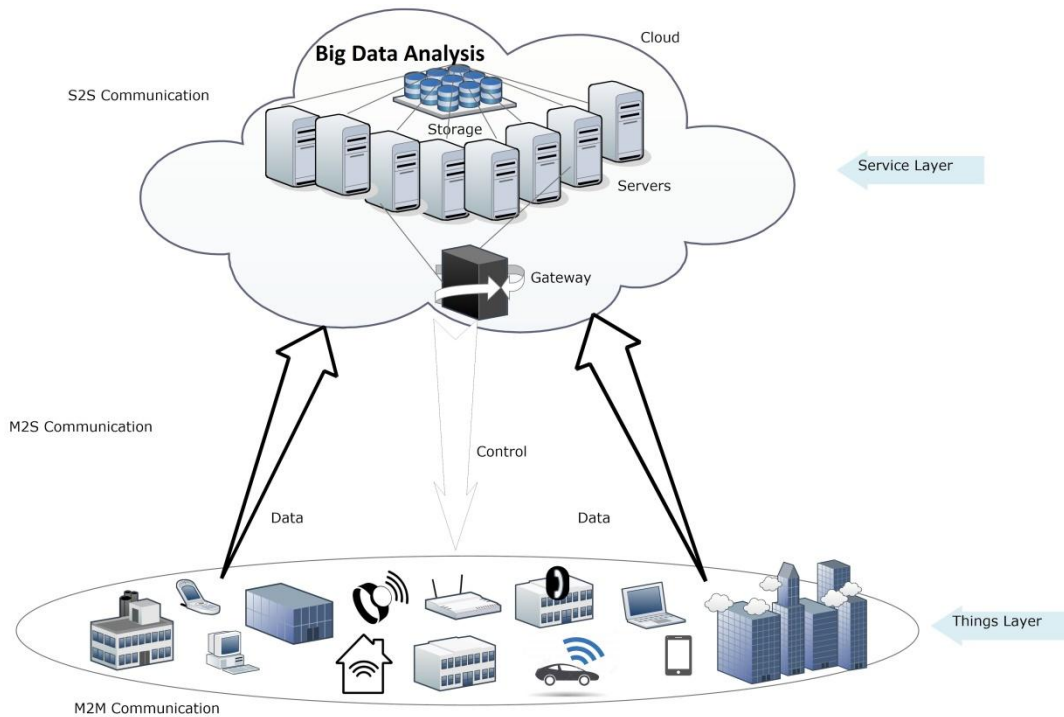


Fig 4: Generation of Big Data and it's analysis in cloud

2.3. SENSING PROXIMITY BASED SERVICES:

Sensing Proximity based services (SPBS) are a general class of computer program level services that use location data to control features. As such SPBS is an information service and has a number of uses in social networking today as an entertainment service, which is accessible with mobile devices through the mobile network and which uses information on the geographical position of the mobile device. This has become more and more important with the expansion of the smart phone and tablet markets as well.

mTrigger: mTrigger is an event-based framework for scalable processing of location-based mobile triggers (location triggers for short). A location trigger is a standing spatial trigger specified with the spatial region over which the trigger is set, the actions to be taken when the trigger conditions are met, and the list of recipients to whom the notification will be sent upon the firing of the location trigger[6].

MoveMine: The MoveMine project investigates effective and scalable methods for mining various kinds of complex patterns from dynamic and noisy moving object data, finding multiple interleaved periodic patterns, and performing in-depth multidimensional analysis of moving object data[7].

GeoPKDD: Geographic Privacy-aware Knowledge Discovery is the GeoPKDDproject ,to develop theory, techniques and systems for geographic knowledge discovery and delivery, based on new privacy preserving methods for extracting knowledge from large amounts of raw data referenced in space and time. More precisely, we aim at devising knowledge discovery and analysis methods for trajectories of moving objects; such methods will be designed to preserve the privacy of the source sensitive data[8].

Chapter 3

Data Collection Framework

3. 1 OUR REFERENCE DATA COLLECTION FRAMEWORK

Based on the elaboration of MCSC characters and applications, we are trying to work with reference architecture to illustrate the key functional blocks and explain the key techniques of MCSC. It is intended to be the starting point that advances this new research area. Figure 5 shows the reference architecture which we followed to build this application. It consists of five layers: crowd sensing, data transmission, data collection, crowd data processing, and applications.

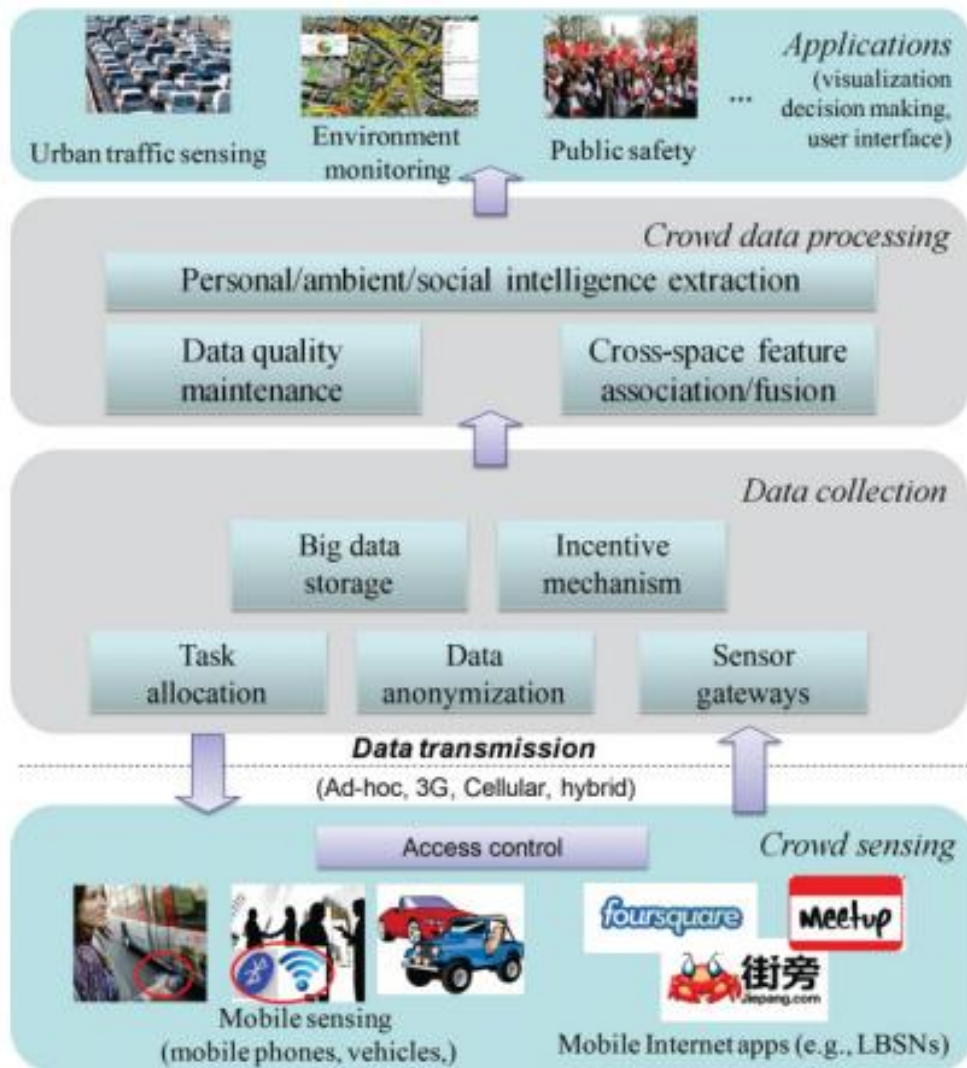


Figure 5: Reference architecture for Mobile Cloud Sensing

{image courtesy: taken from[4]}

(1) Crowd sensing: This layer involves heterogeneous data sources for mobile crowd sensing, including the sensory information from mobile devices and user contributed data from mobile Internet applications. Considering the significance of security and privacy issues in MCSC, it is important that participants be able to determine what kind of information to publish and whom to share with. To this end, access control becomes an important function of the participatory sensing clients.

(2) Data transmission: In MCSC, sensed data from mobile devices should be shipped to the backend server for further processing. MCSC applications should make data uploading tolerant of

transient networking environments and inevitable network interruptions. Therefore, data forwarding and routing protocols become significant for data transmission. If the network infrastructure is not available, the data should be transmitted to the destination in a pure opportunistic networking manner [Conti and Kumar 2010]. The cooperation of heterogeneous networking nodes to enhance the performance of data transmission is also important to this layer [Guo et al. 2013].

(3) Data collection infrastructure: This layer gathers data from selected sensor nodes and provides privacy-preserving mechanisms for data contributors. The following components are involved: (a) Task allocation. It can analyze a sensing task from an application requester and assign it to a selected number of human nodes in terms of specified requirements, such as sampling contexts (e.g., time, location), device capability, user willingness, and the given budget (e.g., monetary cost, time limit). (b) Sensor gateways. It provides a standard approach (e.g., the web service techniques used in the SenseWeb [Kansal et al. 2007]) to facilitate data collection from various crowd sensing sources; that is, sensor gateways serve all the top level components (e.g., data processing and applications) by supplying a unified interface. (c) Data anonymization. One major concern when sharing personal data is privacy. In addition to the access control function at local stakeholders, this component also supports privacy protection by providing anonymization mechanisms before data is published. (d) Incentive mechanism. This component provides strategies for incentives and reputation to data contributors. (e) Big data storage. The collected data in MCSC systems has two characters: large scale and multimodality. First, the volume of data to be stored and managed is so large and complex (e.g., collecting pollution information at the city scale) that it becomes more and more inefficient to process using existing data management and processing approaches and tools. Second, the characteristics and attributes of different types of sensors usually vary a lot, leading to big differences in the accuracy of crowd sensing. Therefore, in order to boost further processing (e.g., learning and reasoning), the raw data collected from different sensors must be first transformed and represented in a unified manner, for example, based on the same vocabulary or ontology [Zhang et al. 2011].

(4) Crowd data processing: It aims to extract high-level intelligence from the raw sensory data by leveraging a wide variety of machine-learning and logic-based inference techniques. In other words, the focus of crowd data processing is to discover frequent data patterns to obtain the three dimensions of crowd intelligence at an integrated level. (a) Data processing architecture. Instead of a purely centralized or self-supported method (see Section 2.5), we advocate the hybrid data processing architecture and propose the Hybrid Data Processing (HDP) solution. In HDP, some of the data processing tasks are allowed to execute on mobile devices to fulfil local perception (e.g., analysing individual behaviour on a smartphone), and after that, local results will be transmitted to servers for further processing. By adopting such a hybrid data processing approach, the communication cost between clients and servers can be notably cut down, and the resilience of the entire network will be enhanced. (b) Data quality maintenance. The data from different contributors has distinct quality and credibility and is often redundant. Therefore, quality measurement and data selection metrics are needed to pre-process the data and eliminate the data with low quality. (c) Cross-space feature association/fusion. The mobile crowd data can be collected from both offline and online spaces. This component studies approaches for cross-space data association and complementary feature fusion [Guo et al. 2014b]. (d) Crowd intelligence extraction. It aims to extract the three types of crowd intelligence, that is, user awareness, ambient awareness, and social awareness, by applying various data processing techniques. (5) Applications. This layer consists of different types of applications and services that could be enabled by MCSC. Associated key functions include data visualization, user interface, and so on. Visualization (e.g., mapping, animation, graphing) displays the crowd computing results in a legible format to the users. User interface is designed for the interaction between humans and machines. Both of them facilitate decision making and knowledge sharing to users (e.g., government officials, citizens, service providers).

Now when we start working with sensing proximity based services (SPS) , we found the below architecture to be useful. So, for our location based application we followed the architecture shown below. The architecture of SPBS can be generalized as shown in Figure 6.

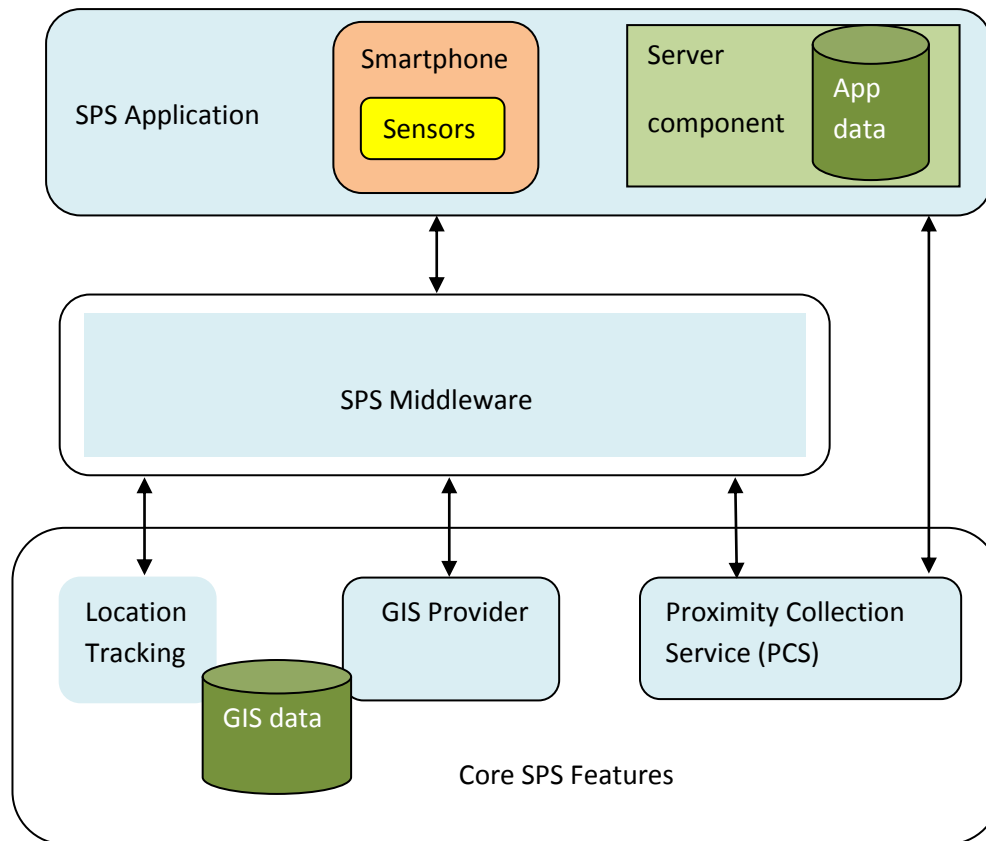


Fig 6: Reference architecture of Sensing Proximity Based Services

As it is shown in the Fig 5, LBS consist of three layers as follows and is shown in figure 2. They are designed in such a fashion such that each layer will do their dedicated work allocated to those layers. Those layers are: SPBS Application Layer, LBS Middleware Layer and Core LBS Features Layer.

(1) SPBS(Sensing Proximity Based Service) Application layer

SPBS Application layer comprises of applications such as locating people on a map, getting nearby places, find my friends, and so on. This layer also consists of Smart phones with number of sensors and server components such as application server and spatial databases. Application server, which is the processing center for a SPBS platform that handles user interface functions and communicates with the spatial database.

(2) SPBS Middleware Layer

SPBS middleware bridges protocols and network technology with wireless and Internet technology. It is either deployed within the network operator’s network or hosted by an application service

provider. SPBS middleware gives access to core features which include: a. Location tracking b. GIS Provider c. Location Collection services

(3) Core SPBS Features Layer

In this layer we can talk about GPS, GIS and Location Collection service. Location tracking is a service for Global positioning System (**GPS**) phones that can be used by users to monitor the movement of the person caring the mobile device. This component stores traced location of users and it is the fundamental component in SPBS as it contains the data that allows a user's route to be determined and potentially predicted. Whereas the **GIS** Provider facilitates with geo-specific functionality for SPBS application including map information, map visualization and directory services. For example, Google maps with its API can be considered as a GIS Provider. Now, **location Collection Service** is a component that basically focuses on getting latitude and longitude for a specific user. Location Collection Service can be accessed with the help of SPBS middleware, For example, mobile network triangulation via a service provider or directly via GPS receiver in the smart phone.

Location Tracking:

This component stores the location trace of individual users. This represents a fundamental component in next generation SPBS as it contains the data that allows a user's route to be determined and potentially predicted. In particular, this component would typically support the following functionality:

1. Keep records on user's current and past locations.
2. Notify other components when a specific user has moved, or when they move in or out of an area. This supports proximity based notifications being sent to users.
3. Determine which users are within a defined location this supports geo-casting features.
4. Queries of location trace to generate user movement models GIS Provider -This component provides geospatial functionality for many SPBSs including map information, map visualization and directory services. Google Maps with its API can be considered a GIS provider.

Location Collection Service

This component performs location collection to get a latitude and longitude for a specific user. Depending on the technology, this component may be accessed via the SPBS Middleware (e.g., mobile network triangulation via a service provider) or directly (e.g., via GPS receiver in the Smartphone). Android provides access to the above components to facilitate the implementation of SPBS services through the help of following classes;

1. Location Manager
2. Location Provider
3. Geo-coding

4. Google-Map Location Manager - Location Manager Class of android is present to manage all other components needed to establish a SPBS system.

Location provider

Location provider represents the technology to determine the physical location i.e. to handle GIS. Location Provider component of Android application is present to facilitate the determination of available provider and selection of suitable one. There are two methodologies to implement SPBS : (a) To process location data in a server and to forward the generated response to the clients and (b) To find location data for a mobile device based application that can use it directly.

To discover the position of the mobile, SPBS must use positioning methods in real time. The accuracy of the methodology depends on the approach used. Locations can be represented in spatial terms or as text descriptions. A spatial location can be represented in the used latitude longitude-altitude coordinate system. Latitude is defined as 0-90 degrees north or south of the equator and longitude as 0-180 degrees east or west of the prime meridian, that passes through the Greenwich, England. Altitude is represented in meters above sea level. A text description is usually defined as a street location, including city, pin code. The location of the device can be retrieved by:

i) Mobile Phone Service Provider Network: The current cell ID is used to locate the Base Transceiver Station (BTS) that the mobile phone is interacting with and the location of that BTS. It is the most basic and cheapest method for this purpose as it uses the location of the radio base station that the cell phone is connected to. A GSM cell may be anywhere from 2 to 20 kilometres in diameter. Other approaches used along with cell ID can achieve location granularity within 150 meters. The granularity of location information is poor due to Wide Cell Range. The advantage is that no additional cost is attached to the handset or to the network to enable this service.

ii) Satellites: The Global Positioning System (GPS) uses a constellation of 24 satellites orbiting the earth. GPS finds the user position by calculating differences in the times the signals, from different satellites, take to reach the receiver. GPS signals are decoded, so the smart phone must have in-built GPS receiver. Assisted GPS (A-GPS) is the new technology for smart phones that integrates the mobile network with the GPS to give a better accuracy of 5 to 10 meters. This fixes the position within seconds, has better coverage and can, in some cases, be used inside the buildings, consumes less battery power and requires fewer satellites. The granularity of location information is most accurate (Latitudes and Longitudes). The disadvantage is cost of AGPS enabled handsets for the user.

Now, in order to make SPBS services possible, some infrastructure elements are necessary including mobile devices, applications, communication network, positioning component, and service providers. Mobile devices have inbuilt tools that are used by the users to access SPBS services, to send requests and to retrieve the results. Such devices can be portable navigation devices (PNDs), Personal Data Assistants (PDAs), laptops, mobile phones, and so on. Application is the interface for users to access the SPBS service. It is usually a software developed by an application provider, downloaded and installed on user's mobile device. A specific application is usually developed for a specific SPBS service. Due to the restrictions of mobile devices (small screen size, limited processor power and memory, battery capacity), SPBS applications need to be lightweight and battery saving.

In general, communication network refers to the mobile network which transfers service request from user to service provider, and provide requested information back to the user. A positioning component is usually needed in a SPBS application to determine the location of user's mobile device. Service providers maintain service which offer different kinds of SPBS services to users and are responsible for processing service requests and sending back request results. Servers calculate positions, search for a route, or search specific information based on user's position. Service providers usually do not store and maintain all the information requested by users. Instead, content providers are responsible for collecting and storing geographic data, location-based information, and other related data. These data will be requested and processed by service servers and then returned to users. Figure 5 shows how the interactions among these components, and the process of a SPBS service. First, user sends a service request using the application running on mobile device (Step 1). The service request, with user's current location information obtained from the positioning component (in this example, GPS data), is sent to service server via the mobile communication network (Step 2). The service server requests geographic database and other related database to get required information (Step 3, 4). At last, the requested information is sent back to user's mobile phone via mobile communication network. Every SPBS's contain a number of components including maps and Geographic Information System (GIS) information, location collection services, and SPBS application specific subcomponents.

3.2 Application Scenarios

There are so many types of applications which is possible by only using SPBS alone. In the recent trend, it is noticed that majority of the websites/apps sense location data to improve Quality of Services. Some of those use only location to serve some specific purpose. There are numerous applications of Sensing proximity and mobile crowd sensing. Following are the few Sensing proximity application domains: Marketing, Emergency, Information Services, Navigation, Location Based Social Media, Mobile Location-Based Gaming, Sports, Billing, Geo-tagging, Tracking, Augmented Reality etc.

3.2.1 Public safety or emergency services

The location of the client can be determined by the mobile carrier hence it finds great use during Emergency since it can be used during the emergency/health hazard to locate the mobile clients. One of the fundamental application of sensing proximity is utilizing the ability to locate an individual calling to emergency response agency (911 in US, 112 in EU) who is either unaware of his/her exact location or is not able to reveal it because of an emergency situation. Based on this spatial information emergency response agency (e.g. ambulance, police, fire fighters) can provide help in a quick and efficient way. Also, data from proximity based Services can be used as well for disaster management.

3.2.2 Consumer services

Navigation: Navigation services allow locating the exact geographical position of a mobile device using one of available positioning systems and get direction or navigate user to the required destination including vehicles, crafts, and pedestrians. This service is often linked with 'Information Services' (e.g. get direction to point of interest).



Fig 7: Location based services (image courtesy:<http://geoawesomeness.com>)

Location Based Social Media

Social media have been widespread on the Internet and have become a craved research topic. Social networks like Myspace, Facebook and Twitter changed the way how people communicate and maintain relations among friends, colleagues, peers or even a family. The development and ubiquity of proximity aware mobile devices gave social media possibility to integrate location with content created users. There are different models that the networks are based on. What most of the networks have in common is 'checking-in' – the act of sharing one's location (verified by positioning system) with social network, public, or individuals and have access to location broadcasted by their friends.

Geotagging

Geotagging is defined as adding geospatial metadata to digital media such as photographs, videos, messages, blogs, web pages and GeoRSS. Significant amount of the social media content is created by users through location-aware mobile devices. It allows browsing the content of the Web with geographic filtering. It is as well possible to visualize some of the content. Especially photo-sharing sites enabling geotagging are popular among users of LBS. There is an on going research about using geotagged images to determine location of touristic service points.

Goods/Human Tracking

Real-time tracking is one of the most useful applications of Sensing proximity. It can be used for people tracking: children, patients with dementia, prisoners with arrest ankle bracelets, employers to track their workers. Proximity tracking solutions are used as well for animal tracking.

Vehicle tracking is another broad application. This can concern single vehicle tracking (e.g. car security systems) but as well control and coordination of entire fleets of vehicles. UPS one of the World's biggest shipping companies uses own proximity based systems for management and logistics thousands of tracks. With fleet over 60 000 vehicles, even one saved mile by every track per day means millions of dollars savings.

Find my friend application:

One of the most popular application in today's smart phone enabled world is "Find my friend " application. This consists of a Smartphone component, which has a number of sensors, and potentially a server component that includes application specific data (such as location-tagged information to SPBS Middleware .This wraps access to Core SPBS Features (Location Tracking, GIS Provider and Location Collection Services) to provide a consistent interface to SPBS applications. This type of application is pretty helpful for not only the new generation but also for the parents to find whether their children reached school in proper time or reached home safely on time or not.

Mobile Location-Based Gaming

Mobile Location Based Gaming (MLBG) is a growing trend among sensing proximity application. MLBG is linking elements of traditional open-air field games (e.g. Hide-and-peek, Paper Chase) with new technologies available on mobile devices including positioning technologies, wireless fast speed internet, image recognition and augmented reality among others. MLBG can be defined as a location-based game that can run on a mobile device and by using a communication channel the game exchange information with a game server or other players. In most of MLBG there are several scenarios and themes used:

- Treasure hunts (e.g. [Geocaching](#), [GeoSocial](#))
- Territory defense and claiming (e.g. [Please Stay Calm](#), [Shadow Cities](#), [Geo Wars](#), [QONQR](#), [Bunker Busters](#))
- Persistent-World Construction Games (e.g. [Fleck](#), [MyTown](#))
- Scavenger hunts (e.g. [City Secrets](#), [Code Crackers](#), [Bounty Island](#), [iSpy](#), [SCVNGR](#))
- Role playing game (e.g. [GeoHunters](#), [Dusk Falling](#), [Code Runner](#))
- Movement Games (e.g. [Seek and Spell](#), [Map Attack](#), [Zombies! Run!](#), [Meatspace Invasion](#))
- Mixed themes (e.g. [Gbang](#), [Parallel Kingdom](#), [Dokobots](#))



Fig 8: Mobile Location Based Gaming in Real-time{image courtesy:www.google.com\locationbasedGaming}

Sensing Proximity & Sports

The potential of Sensing proximity and modern mobile devices can be used as well to monitor sports activities. Location-based application including Nokia Sports Tracker, Nike +, Run Keeper and Endomondo has millions of users. Functionality of those applications allows user to automatically collect his/her workout data, such as location, distance, speed, duration, or burned calories and store them on the server. Endomondo allows to visualized real-time route of outdoor sport activity via smart phone using Google Maps and sharing that data with a social networks. Another application called Zombie Run is integrating jogging with a territory defence location based game.



Fig 9: Use of Sensing Proximity in Sports

{image courtesy: taken from google.com\sensingProximityinsports-images}

3.2.3 Tourism

This service will provide real-time and correct information to the tourists while they are on their trip. This service also reduces potential risks to tourists in such a way that they provide quick and easy access to information on services such as hospitals and policing. It can also alert them of any activity as well as for transaction and telebanking, destinations can make themselves ready for this by providing the necessary infrastructure needed for them to be available by sensing proximity app in real time.



Fig 10: Location based Tourism

{image courtesy:taken from googlehttp://www.bizmoby.com/#!geomoby/c1kbb }

3.2.4 Enterprise services

Marketing: Proximity Based Marketing is one of the coolest things and is the driver for new innovations. Basically, people's taste is similar in location. For example, if there is a festive season in a particular state then obviously the demand of dresses will be high. Companies will stock their local vendors on basis of this information. Their strategies will also be changed accordingly. Online Shopping giant like Flipkart, Amazon are using this technique to better understand their customer.

Information Services: Proximity based information services refer mostly to the digital distribution of information based on device location, time specificity and user behaviour. This is one of the most widespread and earliest implemented types of sensing proximity utilizing both pull and/or push services.

The user could query the server with simple questions concerning for example the address of the closest cinema or automated teller machines. The services were usually provided by wireless network operators. Nowadays the information is commonly delivered by external providers with wireless internet via smart phone apps (e.g. Yelp, Yellow Pages, Gas Buddy). The scope of data and information offered by service provider is very comprehensive and it includes local street map, wide variety of points of interests (restaurants, gas stations, cafes, stores, pharmacies, hospitals, services, touristic attractions etc.), weather forecast, real-time traffic information etc.



Fig 11: Location based Information Services

{image courtesy:taken from google.com\locationBasedInformationServices-images}

In the era of information society and social web, users are not only producing content of World Wide Web but also geographically localize it. Sensing Proximity can give users access to value-adding information published by other users of the Web in the geographical proximity, for example recommendations of restaurants or particular items from a menu of a restaurant, spontaneous public event etc.

Billing

Location based billing refers to ability to dynamically charge users of a particular service depending on their location when using or accessing the service. Although the concept could be applied to many businesses, two major industries use this option.

The primary industry that is utilizing this application of sensing proximity is cellular network. Proximity based billing allows a mobile operator to charge different rates to mobile subscribers based on their physical location. It gives a mobile operators possibility to directly compete with wire line providers by charging clients at home or at work with rates comparable to wire line and with standard rate when they leave. This possibility is utilized only by a small percent of mobile operators. Another industry is Online shopping where it is decided whether your delivery is chargeable or not on basis of your location.

Sensing Proximity & Augmented Reality

Augmented Reality is a growing trend in Proximity based Services. It combines real and virtual world by combining camera view with virtual overlaid augmented graphic or information. Due to its character it is interactive and registered in 3D. The location dimension is crucial in order to deliver relevant information to user.

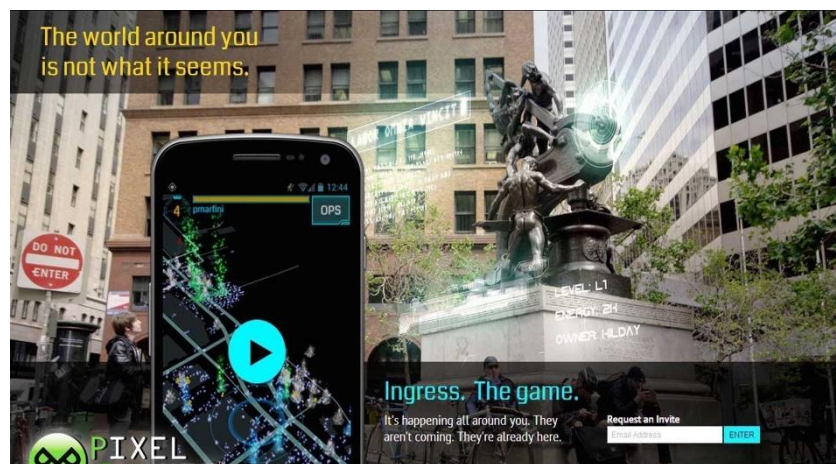


Fig 12: Location based Augmented Reality

{image courtesy:taken from google.com\augmentedReality-images}

Chapter 4

Implementation

4.1 Android

ANDROID is a mobile operating system, it delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications. Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It was built to be truly open. Android is built on the open Linux Kernel. Android support SPBS Application Programming Interfaces (APIs). Location service allows finding out the device current location. The application can request for periodic update of the device location information. The application can also register a intent receiver for proximity alerts like when the device is entering and exiting from an area of given longitude, latitude and radius. On a basic level, android is a distribution of Linux that includes a Java Virtual Machine (JVM), with Java being the preferred programming language for most Android applications. The Android Software Development Kit (SDK) includes a debugger, libraries, a handset emulator, documentation, sample code and tutorials. The ADT plugin includes an Android emulator that allows for the simulation of GPS and Wi-Fi.

Android Architecture

Android is not only a mobile operating system that uses a modified version of the Linuxkernel, but also a software stack for mobile devices that includes an operating system,middleware and key applications .The components of Android's underlyingoperating system are written in C or C++, but user applications are built for Android usingthe Java programming language .Figure 13 below shows the relationship between theandroid environment, Linux-Kernel and the built in applications.

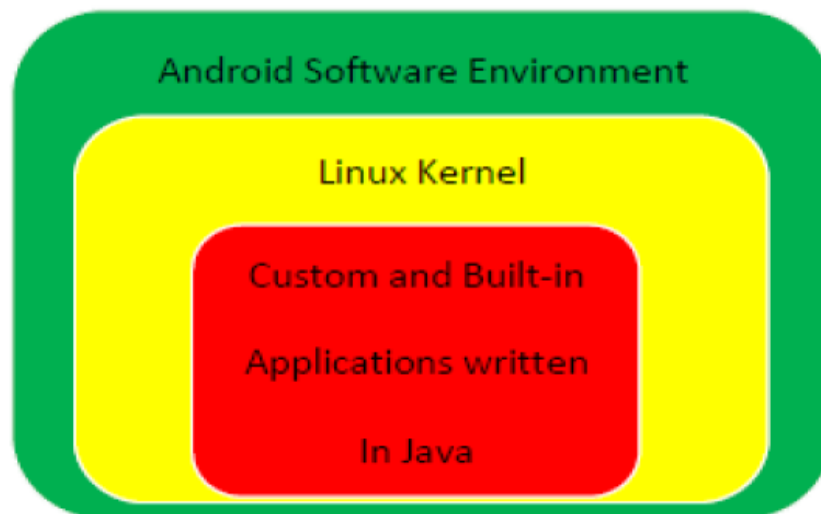


Figure 13: Relationship between the android environment, Linux-Kernel and the built in applications
{image courtesy: taken from [9]}

The major components of the Android operating system are show in Figure 14 and then we describe about each component.

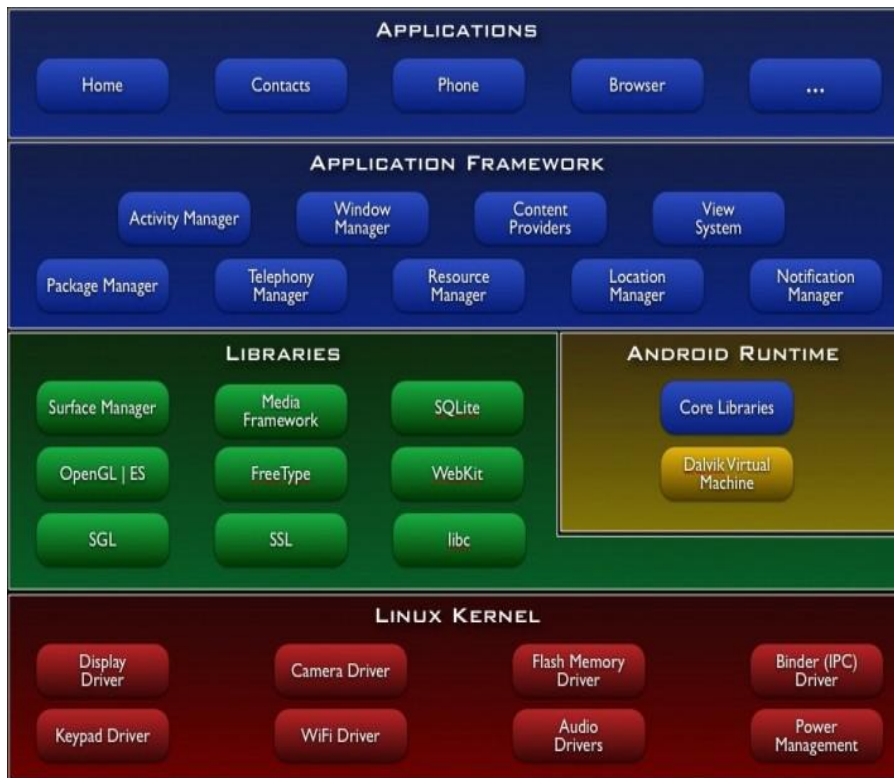


Figure 13: The major components of the Android operating system {image courtesy: taken from [9]}

4.1.1 Applications

Android has a set of applications that consists of an email client, calendar, maps, browser, SMS program and contacts. All the applications are written in Java.

4.1.2 Application Framework

Android provides an open development platform. This characteristic favours the development of extremely rich and creative applications. The architecture of the android application component is designed such that the reuse of components is greatly simplified. A set of services and systems underlies all the android applications. They include:

- (a) A extensible and a rich set of *Views* that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser;
- (b) An *Activity Manager* that provides a common navigation back-stack and mainly manages the lifecycle of applications;
- (c) There are *Content Providers* that enable applications to connect with and access data from other applications (Ex: Contacts) or also share their own data;
- (d) A *Resource Manager* that provides access to non-code resources such as localized strings, layout files and graphics, and
- (e) A *Notification Manager* that enables all applications to display custom alert messages in the status bar .

4.1.3 Libraries

Apart from Java, Android includes a set of C/C++ libraries that is used by various components of the Android system. The Android application framework makes these available to developers.

4.1.4 Android Runtime

Most of the functionality in the core libraries of the Java programming language is made available in Android with its own set of Core libraries. Android applications are developed using java as the programming language. However they are executed by a custom virtual machine called the “**Dalvik**” rather than the traditional java virtual machine (JVM). During execution, each Android application would run in its own process, and would have its own instance of the **Dalvik** virtual machine. One important characteristic of **Dalvik** is that it has been written so that a device can run multiple VMs efficiently. The **Dalvik** VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

4.1.5 Linux Kernel

Android relies on Linux version 2.6 for core system services and it also acts as an abstraction layer between the hardware and the rest of the software stack. Each Android application runs in a separate process within its own Dalvik instance and hence hands over all responsibility for memory and process management to the android runtime. The android runtime then stops and kills processes according to the priority assigned in order to manage resources.

4.2. Enabling the GPS in the mobile device

Android provides many technologies to determine the location of the user which generally depends on the device being used. In most scenarios it’s unlikely that the user will want to explicitly choose the *LocationProvider* to use. More commonly, the user will specify the requirements that a provider must meet and let Android determine the best technology to use. This is done by using the *Criteria* class with which we can state the requirements of a provider in terms of accuracy (fine or coarse), power use (low, medium, high), financial cost, and the ability to return values for altitude, speed, and bearing. Once the user has defined the required criteria, he or she can either get the best provider which would match our criteria or he or she can get a list of all the possible matches. One with the greatest accuracy is returned when the result obtained consists of more than one provider. I have used the Global Positioning System as the location provider for my application.

The next task is to find the physical location of the user/device. This can be achieved by gaining access to the *LocationManager* class, which is responsible for the location based services in android. One or more permissions have to be acquired for supporting access to the LBS hardware before one can use the *LocationManager* class. The GPS provider requires *fine* permission, while the Network (Cell ID/Wi-Fi) provider requires only *coarse*. Figure 5.3 shows the result of what we have obtained by using the *LocationProvider* and the *LocationManager* system service. It shows the current position of the user in terms of latitude and longitude.

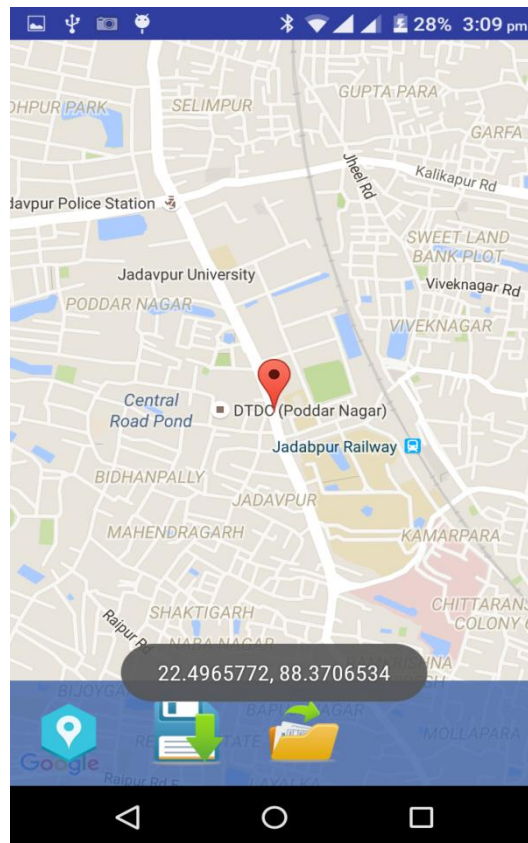


Figure 14: Current Position of the user in terms of latitude and longitude

Most of the location-sensitive applications are expected to be responsive to user movement. We use the ***requestLocationUpdates*** method to get updates whenever the current location changes. The ***requestLocationUpdates*** method accepts either a specific ***LocationProvider*** name or a set of Criteria to determine the provider to use. To optimize efficiency and minimize cost and power use, we also specify the minimum time and the minimum distance between location change updates. Therefore the new location is traced whenever the minimum time and distance values are exceeded.

4.3. Android general concepts

Here, we present the tools used to spread the use of the application: the design of a web page, the implementation of Google Cloud Messaging to establish a communication channel between the server and all the users of the app, and the publication of the app to the official Android Store.

4.3.1. Activity

An activity is an application component providing a view users can interact with. An application consists of multiple activities. The “main activity” is presented to the user when the application is launched for the first time. In our case the App chooses between two “main” activities: Login Activity and the Main Activity. The first one is launched the first time App is opened and the second one the following times. This Main Activity will start Location service and Activity Recognition service. Each activity can start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (i.e. the “back stack”). When a new activity starts, it is pushed onto the back stack and takes user focus. When the user is done with the current activity and presses the Back button, it is popped from the stack and destroyed and the previous activity resumes. Activities must be declared in the manifest file in order

to be accessible to the system. This manifest file presents essential information about the app to the Android system. Information like: minimum SDK (Software Development Kit) version, permissions, activities, service[10].

4.3.2. Activity Lifecycle

It is essential to manage the lifecycle of our activities in order to develop a strong and flexible application. An activity has three states:

- **Resumed:** The activity is in the foreground of the screen and has user focus.
- **Paused:** Another activity is in the foreground (visible) and has focus, but this one is still visible. Paused activities are completely alive but can be killed by the system in low memory situations.
- **Stopped:** The activity is running in the background and is no longer visible by the user. Stopped activities are also alive and can be killed by the system.

In order to manage the lifecycle of our activity, we need to implement the callback methods. Callback methods can be overridden to do the appropriate work when the state of the activity changes. The most important callback methods are:

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```

onCreate() method is called when the activity is created for the first time. This is where all the set up needs to be done such as create views, bind data to lists, and so on.

onResume() method is called just before the activity starts interacting with the user.

onPause() method is called when the system is about to start resuming another activity.

onDestroy() method is called before the activity is destroyed.

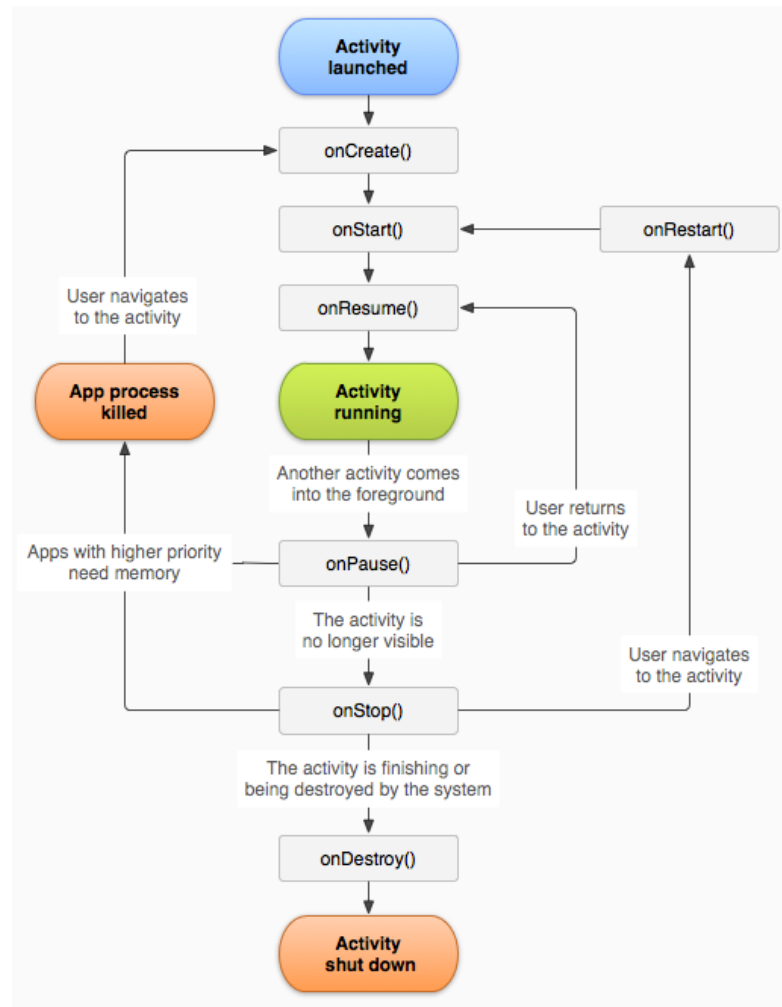


Figure 15: Activity Lifecycle
(image courtesy: taken from [11])

4.3.3. Fragment

A Fragment represents a portion of the user interface in an Activity. Multiple fragments can be combined in a single activity. Each fragment has its own lifecycle and can be added or removed while the activity is running. This App uses fragments in all the activities because allows more dynamic and flexible designs.

There are two ways we can add a fragment to the activity layout:

First way is, declaring the fragment inside the activity's layout file. In this case, we can specify layout properties for the fragment as if it were a view. This is how we add the map on the activity.

```
<fragment
android:id="@+id/map"
```



```
android:layout_width="match_parent"  
android:layout_height="match_parent"  
class="com.google.android.gms.maps.MapFragment" />
```

Second way is programmatically add the fragment to an existing ViewGroup .At any time the activity is running, we can add fragments to our activity layout. We simply need to specify a ViewGroup in which to place the fragment. Adding, replacing or removing actions are called fragments transactions and we need an instance of *FragmentManager*[6] to use methods such as: add, remove or replace.

```
getSupportFragmentManager().beginTransaction().add(R.id.containerhist, new  
HistoryFragment()).commit();
```

In this case, the *ViewGroup* is a *FrameLayout* declared in the activity's layout file and we use the "id" as a reference.

```
<FrameLayout  
android:id="@+id/containerhist"  
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:layout_below="@id/toolbar"  
android:visibility="gone" >  
</FrameLayout>
```

The fragment can access to its activity instance (and use activity methods) using *getActivity()*

4.3.4. Fragment Lifecycle

The most important callback methods are the same as with Activity plus one more: *onCreateView()* method is called when it is time for the fragment to draw its user interface for the first time.

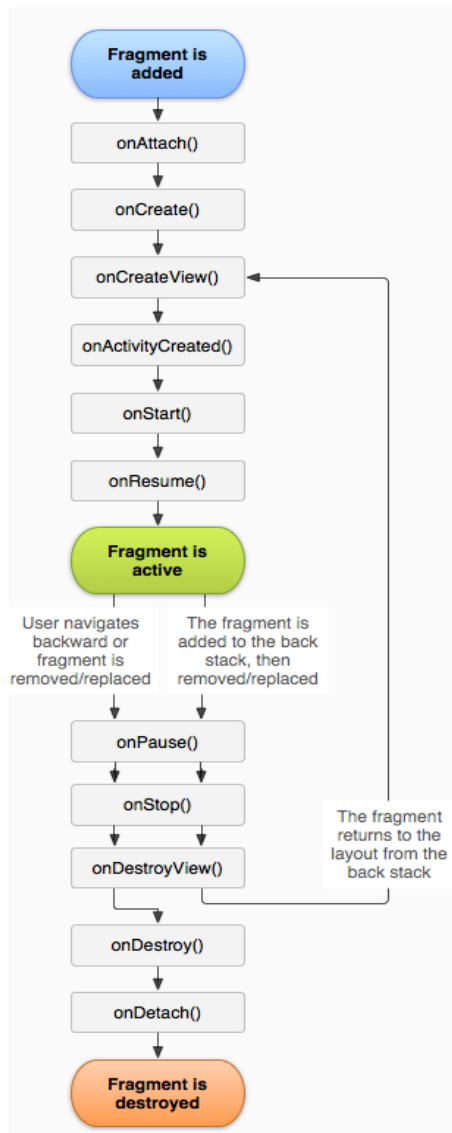


Figure 16: Fragment Lifecycle
 {image courtesy: taken from [12]}

4.3.5 Service

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. A service can take two forms:

- a) **Started:** A service is started when an activity or a fragment starts it by calling *startService()* method. Once started, a service can run in the background indefinitely, even if the activity or fragment is destroyed. This is the form used by this App for Location Service and Activity Recognition Service.
- b) **Bound:** A service is “bound” when an activity or a fragment binds to it by calling *bindService()* method. A bound service offers a client-server interface that allows activities/fragments to interact with the service. This form is not used by this App. There are two types of service: the normal (i.e. *Service*) and the *IntentService*.

The main difference between the two is that *Service* uses the application's main thread so it could slow the performance of any running activity. *IntentService* uses a worker thread to handle all start requests, one at a time. This work queue passes on intent at a time to our *onHandleIntent()* callback method.

4.3.6. Service Lifecycle

The most important callback methods are:

- a) *onStartCommand()* method is called every time the service is started by calling *startService()*
- b) *onHandleIntent()* method is called to process intents in the worker thread. When all requests have been handled stops itself.



Figure 17: Service Lifecycle
{image courtesy: taken from [13]}

4.3.7. Location Sources in Android devices

The selection of location sources is directly related to meet, on the one side, the low power consumption criterion, and on the other side, the specified requirements for mobility data. There are at least four location sources available in almost every Android device: GPS, WPS (Wi-Fi-based Positioning System), Cell ID and sensors. Each one of these sources has its own features regarding average power consumption, accuracy and coverage which basically depends on how location data is obtained as well as the minimum optimal update intervals to achieve with an acceptable level in quality of data. Unfortunately, Android platform is not well documented regarding detailed profiles of average power consumption of each one of the location sources. For that reason, we are going to use approximations of their power usage profiles in order to select those that could meet our low power consumption criterion.

4.3.7.1. Global Positioning System

GPS consists of up to 24 or more satellites broadcasting radio signals providing their locations, status and timestamp. The GPS receiver in the Android device can calculate the time difference between broadcast time and the time radio signal is received. When the device knows its distance from at least four different satellite signals, it can calculate its geographical position.

- Average Power consumption: GPS receiver works as an independent-powered component in the Android platform with the unique objective to obtain location samples. Therefore, all the battery power consumed by GPS is consequence of a user positioning operation. According to several references, we can estimate that the average power consumption of an active GPS with a location update interval between 5 and 15 seconds is approximately 125 to 145 mA.

- Coverage: GPS location coverage is probably the most limited of all four available sources. Although is a source that cover the whole globe, it only provides good performance in outdoor scenarios, because it requires satellites' visibility. GPS does not work in indoor locations (may provide some location data in very few of them if it has some kind of visibility of satellites, but with very poor performance) like buildings or undergrounds. Even in outdoor environments, GPS may suffer from poor performance, especially in cities, because objects like buildings, trees and other obstacles can overshadow visible satellites, decreasing its performance. Therefore, GPS range of coverage can be quite limited in a metropolitan region, compared to other sources, specifically if we want a continuous position tracking of the citizens.

- Accuracy: In outdoor scenarios with good satellite visibility, GPS is the positioning system that can provide the most accurate location of all four existing sources. In perfect conditions GPS can achieve a precision higher than 3 meters, but in optimal conditions the range is between 3 and 10 meters. In non-optimal conditions, i.e. lower visibility on the satellites, GPS accuracy range is typically between 30 and 100 meters. GPS is also capable to provide an estimation of altitude and speed along with location.

4.3.7.2. Wi-Fi-based Positioning System

WPS sends a location request to Google location server with a list of MAC that are currently visible by the device, then the server compares this list with a list of known MAC addresses of the device itself, and identifies associated geo coded locations. After that, Google server uses these locations to triangulate the approximate location of the user.

- Average Power consumption: Wi-Fi receiver works as an independent-powered component in the Android platform but multiple-purpose that is not restricted only to obtain location samples, as happens with GPS receiver. Average power consumption of WPS represents only a small part of power usage of Wi-Fi services.

According to multiple references, we are going to estimate that the average power consumption of an active WPS with a location update interval around 20 seconds is approximately 25 mA.

- Coverage: WPS location coverage is less broad than GPS but is more versatile. Its coverage it is not global but block level, because it has to be in the range of a wireless signal to be able to provide

location updates. However, within this range, it can provide good performance both outdoor and indoor scenarios. Therefore, WPS range of coverage can be very useful when tracking continuous locations within cities and towns.

- Accuracy: In indoor scenarios, WPS is the positioning system that can provide the most accurate location of all four existing sources. In normal conditions, WPS can achieve a precision between 3 and 10 meters. In outdoors, the accuracy of GPS decreases significantly being able to provide a range between 25 and 150 meters, depending on the external conditions of the environment and the signal strength.

4.3.8. Languages

The default language is set (i.e. English). To make this possible we need to create locale directories and string files. Within *res/* directory are subdirectories for various resource types (e.g. Layout resources are saved in *res/layout*, Drawable resources are saved in *res/drawable...*). String Resources define strings and string arrays and are saved in *res/values*. This is the default directory, so if we don't have any additional *values* directory this will be the one Android would use. An example of a bad practice is the following: all the strings in *values* directory are written in Spanish but user's mobile locale are set in English. In this situation user does not understand anything because MobileApp dialogs and text are in Spanish and user only understands English. To solve this, we need to create additional *values* directories. These additional *values* directories are followed by a hyphen and the ISO language code. If the language is spoken in different countries we add another hyphen and the region (preceded by lowercase "r").

- Spanish *values* directory: *values-es-rES*
- Catalan *values* directory: *values-ca*

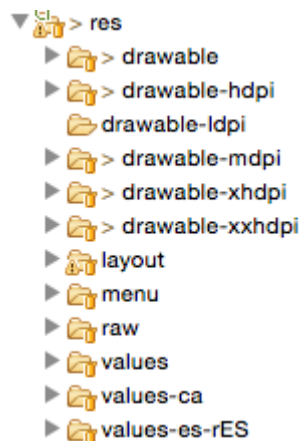


Figure 18: AvailableLanguages

Now we can create a *strings.xml* file in each directory with its respective language and then add string values for each locale into the appropriate file. At runtime, the Android system uses the appropriate set of string resources based on the locale currently set for the user's device. Example, the following is a string resource to be defined in the three languages.

```
<string name="acep">Aceptar</string>
```

<string name="cancel">Cancelar</string>

In this research, the Wi-Fi-direct platform is based on a sample project from the android SDK 4.3 named WiFiDirectDiscovery. The main feature of this program is that it can create a socket connection directly between 2 peers without any intermediate access point or router. This feature is the reason for using this sample program, as in real life traffic scenario the access point range may not cover enough area to connect all the peers. Although it is possible to cover more area by adding more routers, it is not good to the financial aspect, as more routers will inflict higher budget. The Wi-Fi-direct program runs as follows, first it will scan a service discovery of available peer nearby, the user can choose which peer he wants to connect and then start a chat with that peer. They can connect to each other without first needing to be connected to the internet, but simply by turning on each Wi-Fi and connect with each other like using Bluetooth.

4.3.9. Development Environment & Pre requisites:

4.3.9.1. Google Maps API

Google maps is the desktop/mobile web mapping service that has been developed by Google. It is the primary maps application for Android operation system and also available for usage in all other major mobile operating systems. It offers satellite imagery, street maps, 360° panoramic views of streets (Street View), Google Traffic (country specific) and information regarding public transportation for a few selective countries. Maps API facilitate location and location related operations like accessing cell phone GPS, getting location specific information, determining routes etc. Thus numerous applications exists both for android and other smart phone operating systems that utilizes Google maps API to make their application location aware.

Android Location API

These are the different classes present under Location API package to retrieve the Location information of the user. Android contains the android.location package which provides the API to determine the current geo position. Android Location API methods require ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION permissions to be included in the manifest file which provides essential information about an application to the android system. ACCESS_COARSE_LOCATION allows an app to access approximate location derived from network location sources such as cell towers and Wi-Fi. ACCESS_FINE_LOCATION allows an app to access precise location from location sources such as GPS, cell towers, and Wi-Fi.

LocationManager- The class provides access to the location service. It also provides facility to get the best Location Provider as per the criteria.

LocationProvider- It's an abstract super class for location providers. A location provider provides periodic reports on the geographical location of the device.

LocationListener- This class provides callback methods which are called when location gets changed. The listener object has to be registered with the location manager - The class provides the application to choose suitable Location Provider by providing access to set of required properties of the Location Provider. Android also provide an API to access the Google maps. So with the help of the Google maps and the location APIs the application can show required places to the user on the map.

The Google Places API is a service that returns data about Places — defined within this Web Service as, spatial locations, or preferred points of interest — using HTTP requests. Place response specifies locations as latitude/longitude coordinates. The four types of requests are available with the Google Places API. There are 4 fundamental Place services available:

Place Searches - It returns an array of nearby Places based on a location defined by the user. A Place Search request is an HTTP URL defined in the following way : <https://maps.googleapis.com/maps/api/place/search/output> ?arguments Where output may be either of the following values json shows the response in JavaScript Object Notation (JSON) xml shows output as XML.

Place Details - It returns more specific data about a user defined Place. Place Details A Place Details request returns more detailed information about the user defined place such as its address, contact number, user rating, etc. Once we have a Reference Number of Particular Place from Place Search Request, we can initiate the search about that place details. A Place Details request is an HTTP URL of the following form: <https://maps.googleapis.com/maps/api/place/details/output> ?arguments json (recommended) shows the output in JSON xml gives output as XML.

Place Check-ins - It allows the request that a person has checked in to a Place. Check-ins is used to gauge a Place's popularity; frequent check-ins will boost a Place's priority in application's Place Search responses.

Place Reports - It allows the users to add new locations to the Place service, and to delete Places that the application has added to the database. The Google Places API has the following limitations on the query processing: Users are allowed only 1000 requests per 24 hour period which are having the API Key. Clients who have also validated their identity through the APIs console are allowed 100 000 requests for 24 hours period. A credit card is required for authentication, for enabling billing.

System Testing

We developed the mobile application on Android covering all the mentioned APIs and the application was tested using Micromax E313 handset (which is A-GPS enabled handset) with Android Version-5.1(Lollipop).

Android Permissions Required For Operation

- android.permission.INTERNET
- android.permission.ACCESS_FINE_LOCATION
- android.permission.ACCESS_COARSE_LOCATION

Google Map in Android provides a number of objects to handle maps in SPS system like MapView which displays the map. To handle this MapActivity class is there. To annotate map it provides the overlays class. Even it provides canvas by which one can easily create and display multiple layers over the map. Moreover, sufficient provisions are there to zoom the map, localize the map by means of MapController. Following code-line shows the Map Handling in Android:

```
// map controller
MapController mapController = myMapView.getController();
mapController.setCenter(point);
mapController.setZoom(1);
//List of present overlays
List overlays = mapView.getOverlays();
// adding a new overlays
MyOverlay myOverlay = new MyOverlay();
overlays.add(myOverlay);
mapView.postInvalidate()
```

4.3.9.2 Compiler & IDE

For our particular project, since we chose Android as our primary OS to build our software for, we had to use JAVA as our programming language. Henceforth, we set up JAVA development kit and configured our java run-time environment. For building android application, we chose Android Studio as our IDE even though we could use Eclipse with ADT but Google as announced before will no longer support Eclipse due to proprietary issues. Android Studio comes with a compiler named Gradle. Gradle is the next evolutionary step in JVM-based build tools. It draws on lessons learned from established tools such as Ant and Maven and takes their best ideas to the next level. Because Gradle is a JVM native, it allows us to write custom logic in the language we're most comfortable with. Dependency management is employed to automatically download these artifacts from a repository and make them available to our application. Gradle's ability to manage dependencies isn't limited to external libraries. As our project grows in size and complexity, we'll want to organize the code into modules with clearly defined responsibilities. Gradle provides powerful support for defining and organizing multiproject builds, as well as modelling dependencies between projects.

API Keys for Google services

API(Application programming interface keys are procedure of how authentication, authorization, and accounting are accomplished. For all API calls, our application needs to be authenticated. When an API accesses a user's private data, our application must also be authorized by the user to access the data. For instance, accessing a public Google+ post would not require user authorization, but accessing a user's private calendar would. Also, for all sorts of purposes, all API calls involve accounting. We will summarize the protocols used by Google APIs for our particular project.

Access Types

It is important to understand the basics of how API authentication and authorization are handled. All API calls must use either simple or authorized access (defined below). Many API methods require authorized access, but some can use either. Some API methods that can use either behave differently, depending on whether we use simple or authorized access.

1. Simple API access (API keys)

These API calls do not access any private user data. Your application must authenticate itself as an application belonging to your Google Developers Console project. This is needed to measure project usage for accounting purposes.

API key: To authenticate our application, we had to use an API key from our Developers Console project. Every simple access call from our application makes must include this key.

2. Authorized API access (OAuth 2.0)

These API calls access private user data. Before we can call them, the user that has access to the private data must grant our application access. Therefore, our application has to be authenticated,

the user must grant access for your application, and the user must be authenticated in order to grant that access. All of this is accomplished with OAuth 2.0 and libraries written for it.

Scope: Each API defines one or more scopes that declare a set of operations permitted. For example, an API might have read-only and read-write scopes. When our application requests access to user data, the request must include one or more scopes. The user needs to approve the scope of access your application is requesting.

Refresh and access tokens: When a user grants our application access, the OAuth 2.0 authorization server provides our application with refresh and access tokens. These tokens are only valid for the scope requested. Our application uses access tokens to authorize API calls. Access tokens expire, but refresh tokens do not. Our application can use a refresh token to acquire a new access token.

4.3.10. Flow for obtaining user location

Here's the typical flow of procedures we followed for best location update: 1. Start application. 2. Sometime later, start listening for updates from desired location providers. 3. Maintain a "current best estimate" of location by filtering out new, but less accurate fixes. 4. Stop listening for location updates. 5. Take advantage of the last best location estimate. On each update of location, we save the time of the current update. So, we will have the previous and current location, and time update. Then we calculate the distance in meters between those two locations (the old and new one). A code snippet from our application to calculate distance as follows:

```
private static long calculateDistance(double lat1, double lng1, double lat2, double lng2) {
    double dLat = Math.toRadians(lat2 - lat1);

    double dLon = Math.toRadians(lng2 - lng1); double a = Math.sin(dLat / 2) * Math.sin(dLat /
    2) + Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) * Math.sin(dLon / 2)
    * Math.sin(dLon / 2);
    double c = 2 * Math.asin(Math.sqrt(a));
    long distanceInMeters = Math.round(6371000 * c);
    return distanceInMeters; }
```

Then dividing it by the time differences, we measure speed of a particular car. To access GPS sensor, we had to explicitly include permission in androidmanifest.xml file to read the data.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" /><uses-
permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
```

Repeating this process, we can create an effective database with sets of co-ordinates that will later allow us to take location reports and associate them with a specific map fragment for a specific time.

4.3.11 Client side implementation of Google Maps API

With the Google Maps Android API, we can add maps based on Google Maps data to our application. The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. We can also use API calls to add markers, polygons, and overlays to a

basic map, and to change the user's view of a particular map area. These objects provide additional information for map locations, and allow user interaction with the map. The API allows you to add these graphics to a map:

- >Icons anchored to specific positions on the map (Markers).
- >Sets of line segments (Polylines).
- >Enclosed segments (Polygons).
- >Bitmap graphics anchored to specific positions on the map (Ground Overlays).
- >Sets of images which are displayed on top of the base map tiles (Tile Overlays).

MapView

Google provides via Google play framework a library for using Google Maps in our application. The following description is based on the Google Maps Android API v2 which provides significant improvements to the older API version.

The library provides the `com.google.android.gms.maps.MapFragment` class and the `MapView` class for displaying the map component. We need to add additional information to your `AndroidManifest.xml` file to use Google Maps.

```
<meta-data
android:name="com.google.android.maps.v2.API_KEY"
android:value="AlzaSyBuGHO9IEStgSUCKPxVfWrGTZh31FCHP9Y" />
```

MapFragment

The `MapFragment` class extends the `Fragment` class and provides the life cycle management and the services for displaying a `GoogleMap` widget. `GoogleMap` is the class which shows the map. The `MapFragment` has the `getMap()` method to access this class. The `LatLng` class can be used to interact with the `GoogleView` class.

Markers

We can create markers on the map via the `Marker` class. This class can be highly customized. On the `GoogleMap` we register a listener for the markers in your map via the `setOnMarkerClickListener(OnMarkerClickListener)` method. The `OnMarkerClickListener` class defines the `onMarkerClicked(Marker)` method which is called if a marker is clicked. We can also listen to drag events and info window clicks.

Changing the GoogleView

The `GoogleMap` can be highly customized. The following example code demonstrates the purpose.

```
static final LatLngShahbag = new LatLng(53.558, 9.927);
static final LatLngFarmgate = new LatLng(53.551, 9.993);
```

```

privateGoogleMap map;

... // Obtain the map from a MapFragment or MapView.

//Move the camera instantly to hamburg with a zoom of 15.

map.moveCamera(CameraUpdateFactory.newLatLngZoom(Farmgate, 15));

// Zoom in, animating the camera.

map.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);

```

4.3.12 Install Google Play services

The client library contains the interfaces to the individual Google services and allows us to obtain authorization from users to gain access to these services with their credentials. It also contains APIs that allow us to resolve any issues at runtime, such as a missing, disabled, or out-of-date Google Play services APK. The Google Play services APK contains the individual Google services and runs as a background service in the Android OS. We interact with the background service through the client library and the service carries out the actions on your behalf. An easy-to-use authorization flow is also provided to gain access to the each Google service, which provides consistency for both us and our users. We have to download and install Google play Services framework to access GooglemapsAPI. For this we need to configure this particular framework.

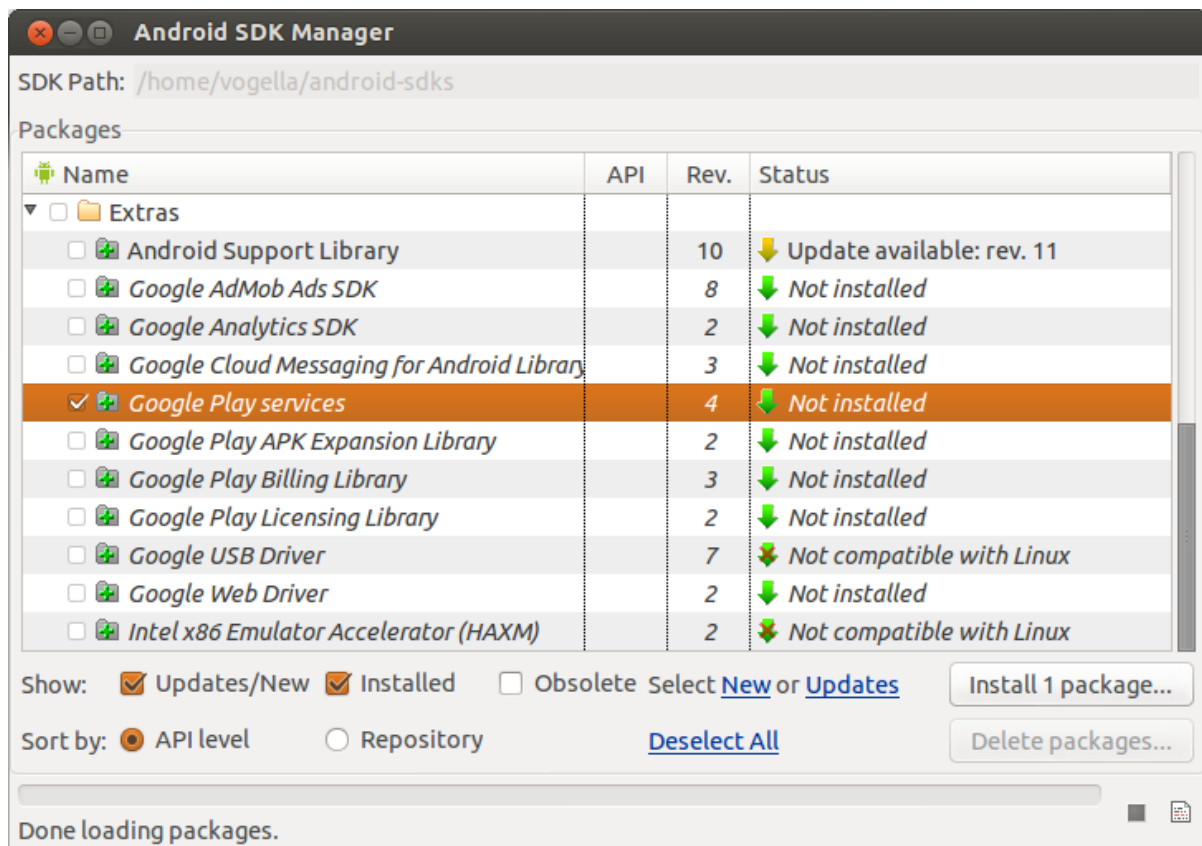


Figure 18: Installing Google Play Services {image courtesy:taken from [14]}

4.4 Evaluation

Implementation and Methodology

Sensing Proximity based service is another key functionality that gets used in smart phone applications. It is often combined with maps to give a good experience to the user about their location. Android support SPS Application Programming Interfaces (APIs).

Location service allows finding out the device current location. The application can request for periodic update of the device location information. The application can also register a intent receiver for proximity alerts like when the device is entering and existing from an area of given longitude, latitude and radius. Service as, spatial locations, or preferred points of interest using HTTP requests. Place response specifies locations as latitude/longitude coordinates.

Establishing Wireless Link

Here main challenge is to identify the other devices in the vicinity which are also running our application. When connected, each device exchanges a file with the other device, which contains the traces of the readings of accelerometer or GPS in the past few seconds. The two ways of establishing communication among devices is via Bluetooth or via WiFi.

Bluetooth

In case of bluetooth, we can enable the visibility of the device from our application, and other devices can see this device in their device list and then pair with it and send a file which contains the data as discussed above. However, the visibility of the other visible Bluetooth device may not be set by the application, and in fact may not even be running the app. In this case we need to handle the following behaviour

- The pairing request or the file reception may not be accepted by the other device.
- The other device may receive the file, and do nothing.

In case of using bluetooth, there is less chances that two devices which are in the communication range of each other are in two different vehicles since the range of bluetooth is only around 10 meters. But this small range may also be a disadvantage that, two devices which are in the same large vehicle (like a bus) may not be in the communication range of each other. Another problem in using bluetooth is that it is not possible to suppress the dialogue boxes asking user permission to enable bluetooth, pair with other devices or communicate with other devices. This is designed so as to avoid the phone becoming vulnerable when the user forgets to disable the bluetooth . However, this makes automatic detection of devices in the vicinity using bluetooth difficult.

Scanning and connectivity via Bluetooth

By scanning we get the list of all the available devices in our range. Connectivity between two devices may be set up by Bluetooth. This connection process creates a socket that our smart phone as well as our service provider will use to transmit and receive data. Smart phones today have the capability to communicate using Bluetooth. This is useful to mobile application developers whose

applications require a wireless communication protocol to send and receive data to and from another device wirelessly.

There are several issues that must be overcome before the Android device can successfully transmit and receive data via Bluetooth. First, the Android device must determine if it supports Bluetooth, and if it does, Bluetooth is turned on. After opening our application in our smart-phone we have a “discoverable” button to do this. We must have a scanning option available. By using “connect” button we scan to get the list of all the devices in our range. Then, it must pair and connect with the Bluetooth module of the targeted resourceful device. Finally, the Android device must actually send and receive data.

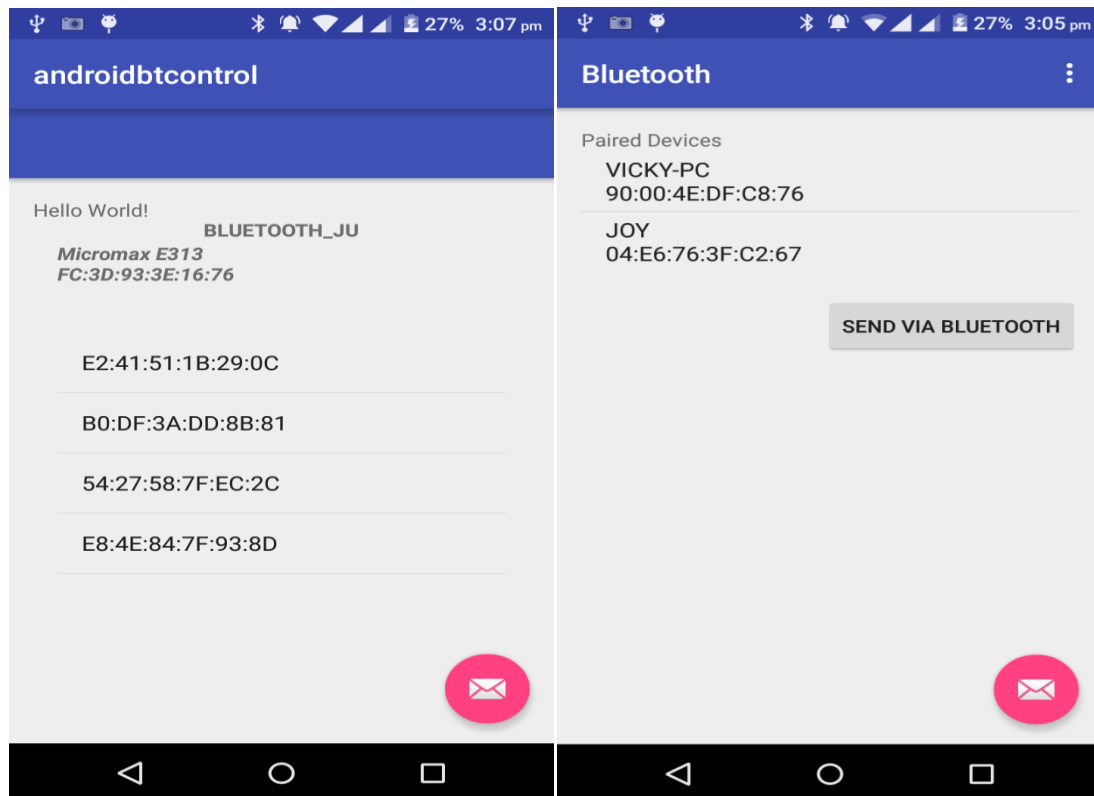


Figure 19 The Bluetooth app

WiFi

In case of WiFi, our application can set up a hot-spot (or an adhoc network) with a pre-decided network name and security pin. Any device, which also runs our application will be aware of this network name and the security pin, connect to the application on the other device, and then exchange the necessary information. We still need to determine whether this connection is to be made in Infrastructure mode, or AdHoc mode. Since the range of wifi is high, the devices which are far away in different vehicles may also be able to connect each other. Hence it becomes crucial to devise a mechanism that can determine whether two vehicles are in the same vehicle or not.

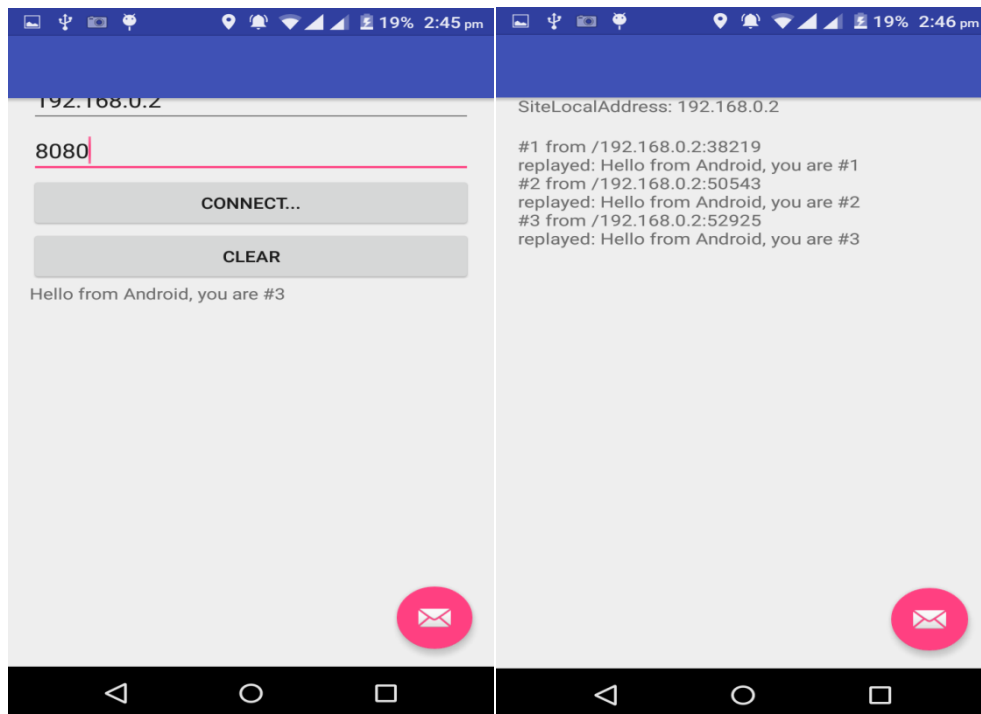


Figure 20 The client server app using Wifi

Accelerometer

The acceleration of the Android smart phone is determined by the phone's position in space, the x, y and z axis. The left and right or lateral movement is defined by the x-axis. The upward and downward movement is defined by y-axis, where the z-axis is responsible for the movement in and out of the plane defined by x and y axis. (See Table. 2.1)

<u>Axis</u>	<u>Direction</u>	<u>Typical Driving</u>
x-axis	Left/Right	Turning/ Lane change
y-axis	Front/Rear	Acceleration/ Braking
z-axis	UP/Down	Vibrations/Road Anomalies

Table 1 Significance of Three Axis Measurement

Accelerometer Orientation Measurement

The accelerometer is very sensitive even to a steady movement. It will give a result about several times every second at selected rate. To minimize the very detailed result of very small change in acceleration, there will be a filter to neglect any small value measurement between 2 m/s^2 to -2 m/s^2 . Another important factor is that the gravity will also be measured by the accelerometer. The constant value of gravity will give 9.8 m/s^2 to the y-axis (0 m/s^2 minus the force of gravity, which is -9.81 m/s^2) and 9.8 m/s^2 to the z-axis when the phone is laid down. The gravity measurement is not a good practice in determining certain events, therefore to remove the gravity, a low pass filter and high pass filter is used. The low pass filter is used to isolate the force of gravity, whereas the high pass filter is used to remove the gravity's contribution. To enhance the user experience a real time graph is implemented to the user interface. This graph will show all the three axis accelerations the device has experienced. It will provide both the positive and negative values as well as the absolute acceleration, which is the combination of all 3 axis acceleration. The graph value will be generated against time, in seconds. This graph is also used in the testing to acquire the appropriate data in determining the events. Back to the accident prevention method, the Android built-in accelerometer is utilized in fulfilling the accident prevention objectives. It will trigger certain warning event when the measurement exceeds the threshold value. The threshold value is acquired by averaging the acceleration data after several test on the same event location. This will ensure the best threshold value to trigger the warning event accurately.

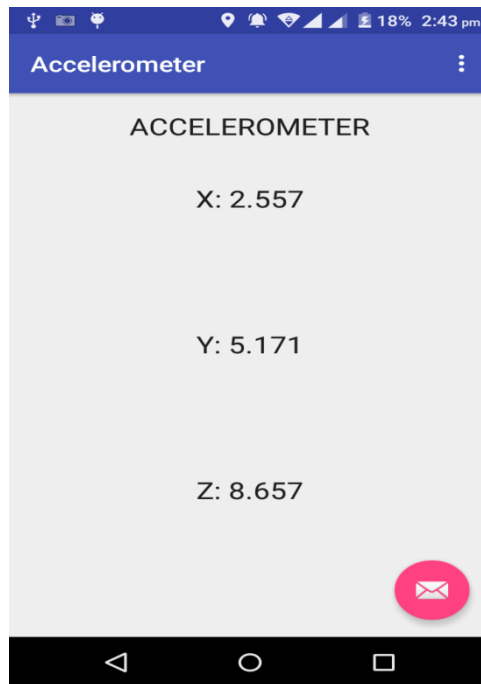


Figure 21 The accelerometer app

GPS

The GPS feature in Android phones is integrated with Google Maps. This will enhance the positioning, as a Google map is more accurate and detailed than other normal GPS maps. It can show current location, as well as a marker to an important location. In order to use Google maps API v2 the application need to be sign up to get the map API key. The API request is sent directly from the Android phones to Google server. Google verifies the API key to check whether it matches with one of their SHA1 fingerprint and the package names data. When it has been verified, the Google map server can be accessed. Each program package registered will generate an unique API key.

Client side algorithm & workflow

On client side, we try to implement a reiterative service which constantly runs in the background and feed data to the server.

Server side algorithm & workflow

On server side, we implement a multi-threaded listener. The server keeps listening for either location reports made by devices or a device who is requesting current map view of a particular area. If a device on a particular road is reporting its location, we take the co-ordinate and speed. Then we check if it overwrote the timer, which would indicate it is exiting a particular road and would no longer be able to report conditions of that road. In that case, we pop it from the corresponding road's device stack and register its value. Otherwise, we just treat the report as routine entry and just update the speed cost of the road accordingly. Then we go back to listening for further requests. In case a device is requesting for a real time traffic view of a particular road.

Form of Communication between Server side and Client side

The client and server side are going to communicate with formatted data. When a device will report its location it will include co-ordinates, speed and any sentinels in a JSON object and send them to the server. The server will have a graph data structure representing current location and store necessary information about a place in the object reference that represents it. These values will include places around like cafe, restaurant and hospitals and mark them by color marker . When the server will send map data to a requesting device, it will likewise format them in JSON that can be interpreted in the map view directly by the application.

The application first finds out the current location of the phone using the Android Location API. The Location API allows us to get a coarse location based on the vicinity phone using the GPS. Next, the application sends over the current location to the web service running on the cloud. It finds its current location and sends them over to the server .The web service keeps track of the events of significance, and it computes the ones in the vicinity of the smart phone. It then responds to the phone application with a JSON array of these events, giving the location, type of event and severity of the event. The phone receives this JSON array, parses it, and then displays the locations of these events in different colours. Thus, the user can choose to find friends while in his vicinity.

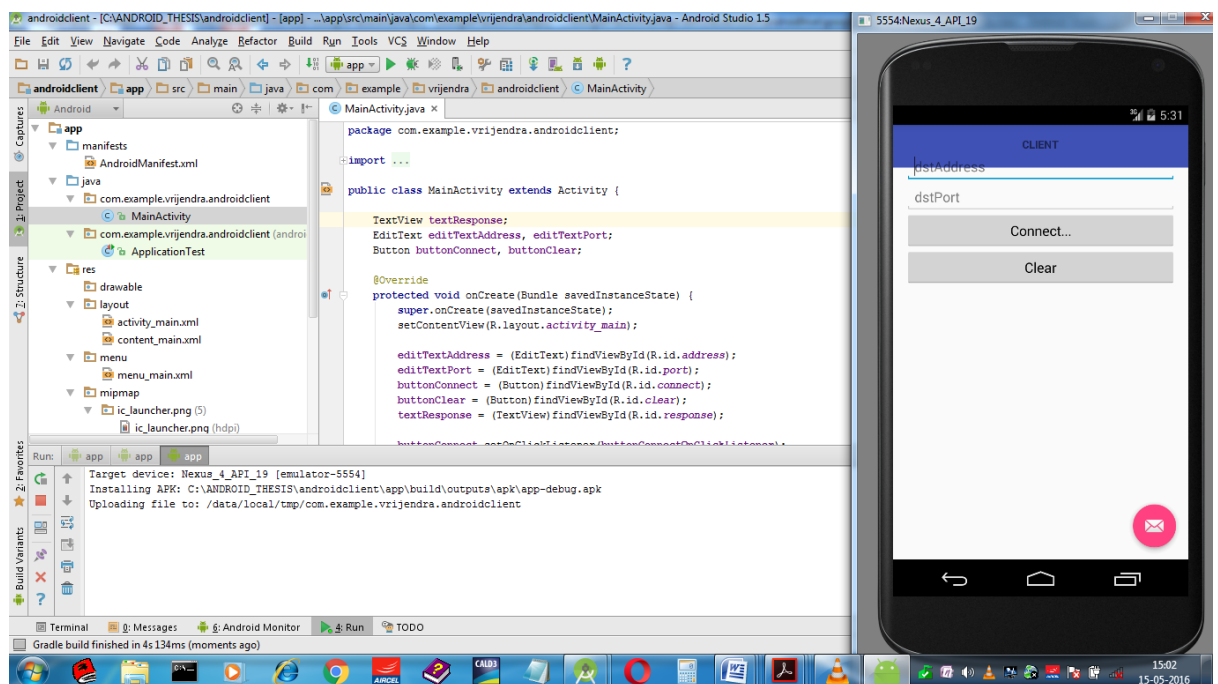


Figure 22 Screenshot of the client app on AVD

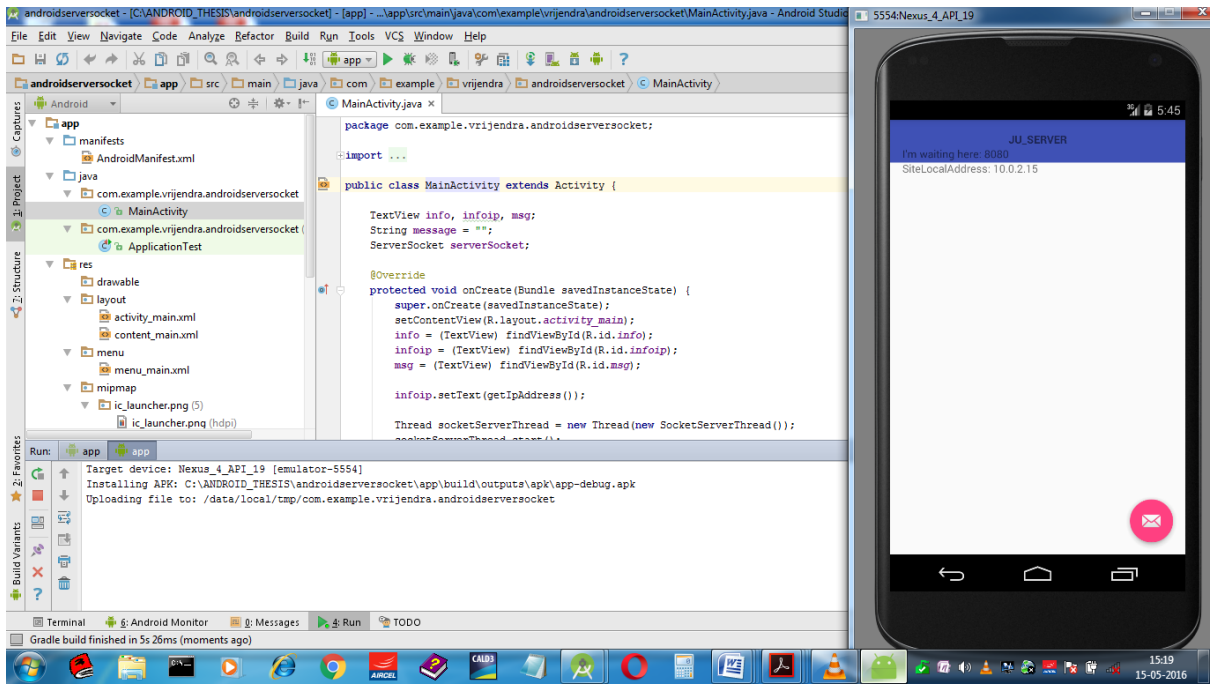


Figure 23 Screenshot of the server app on AVD

SCREENSHOTS OF APPLICATION

The application running on the smart phone acts as both the lowest layer (to collect raw sensor data), as well as the highest layer, it is used to display places of interest after sensing current location of the device . An Android Application using Google Android Maps APIs that achieves this goal. Figure 24 shows a screenshot of the application. The red pin shows the current location of the smart phone. The blue pins show locations of important places in desired proximity.

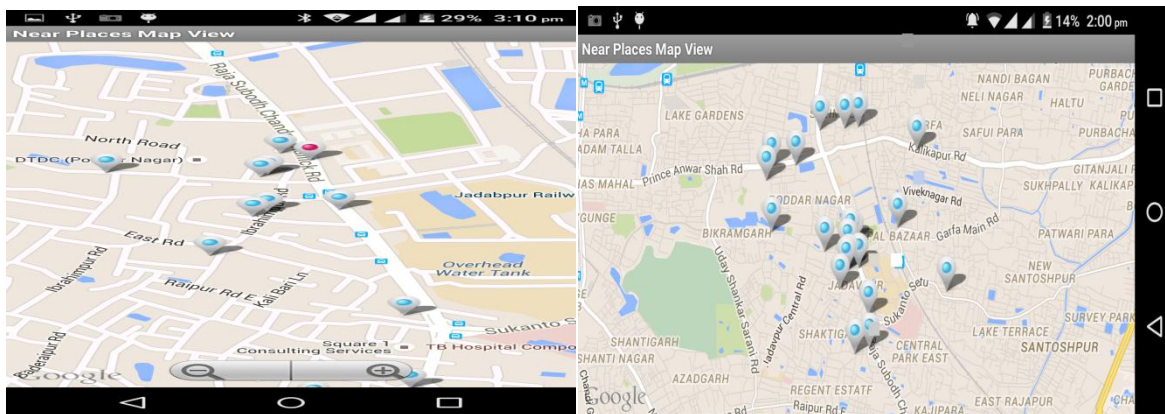
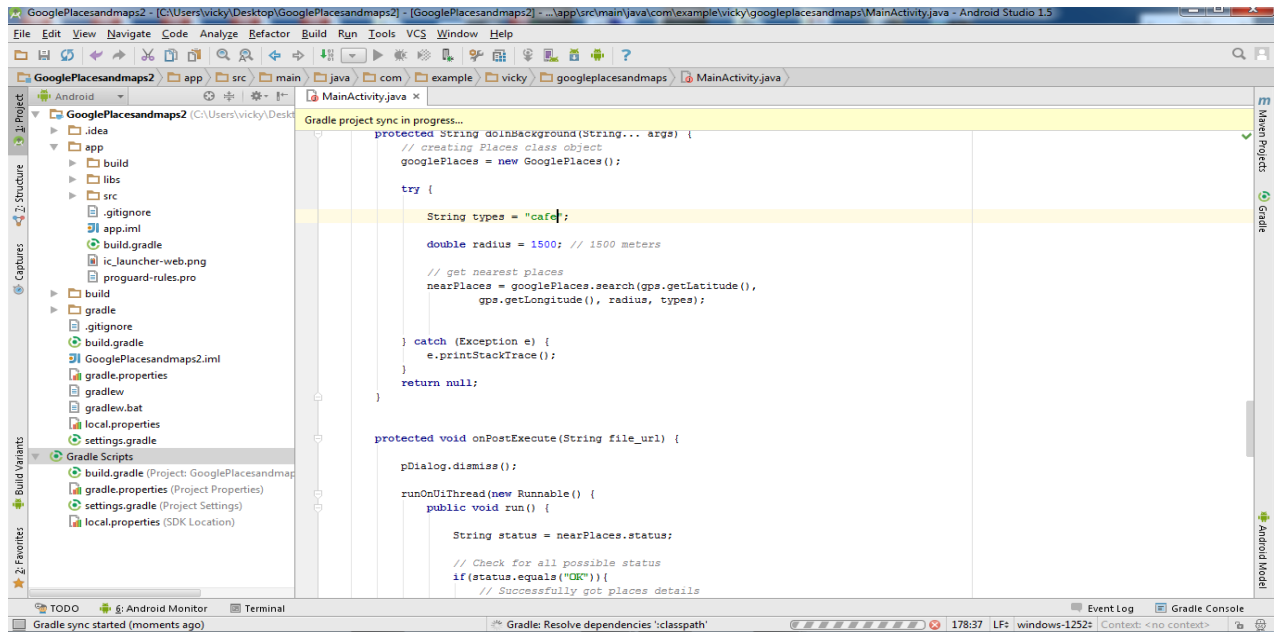


Figure 24 App showing places of interest in proximity

ENTERTAINMENT

When a user wants to view entertainment places like café, restaurant etc. He/she can view the list of all those places of interest in a list form. e.g. the following screenshot shows list of all cafes within user's vicinity. After selecting a café the contact details and address is shown.



```
protected String doInBackground(String... args) {
    // creating Places class object
    googlePlaces = new GooglePlaces();

    try {

        String types = "cafe";

        double radius = 1500; // 1500 meters

        // get nearest places
        nearPlaces = googlePlaces.search(gps.getLatitude(),
            gps.getLongitude(), radius, types);

    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

protected void onPostExecute(String file_url) {
    progressDialog.dismiss();

    runOnUiThread(new Runnable() {
        public void run() {

            String status = nearPlaces.status;

            // Check for all possible status
            if(status.equals("OK")){
                // Successfully got places details
            }
        }
    });
}
```

Figure 25 Screenshot Of the Code when type is cafe only in the code.

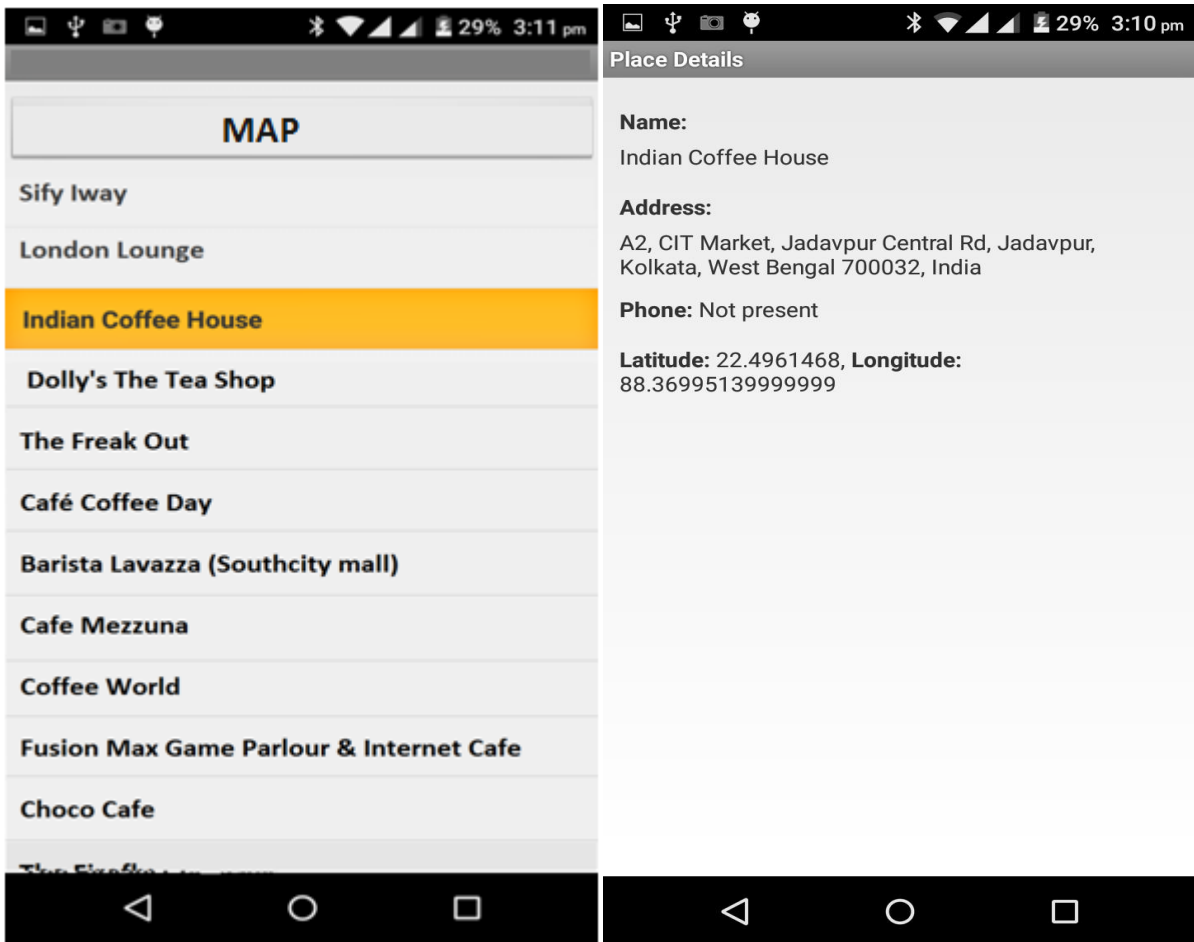


Figure 26 The Corresponding Screenshot Of the App showing cafe within 1.5 km.

EMERGENCY

In case of any emergency like hospital and police station, this app shows the list of hospitals or police stations in the vicinity of current location of user in a list along with contact details and

address.

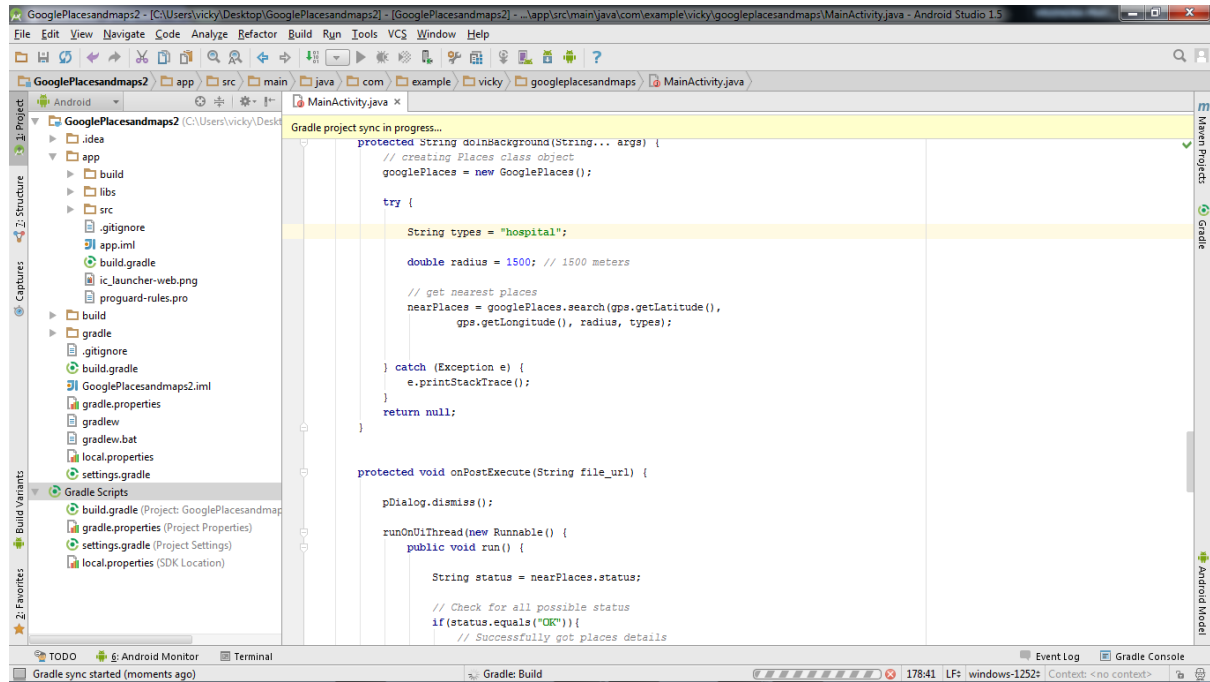


Figure 27 Screenshot Of the Code when type specified is hospital only in the code.

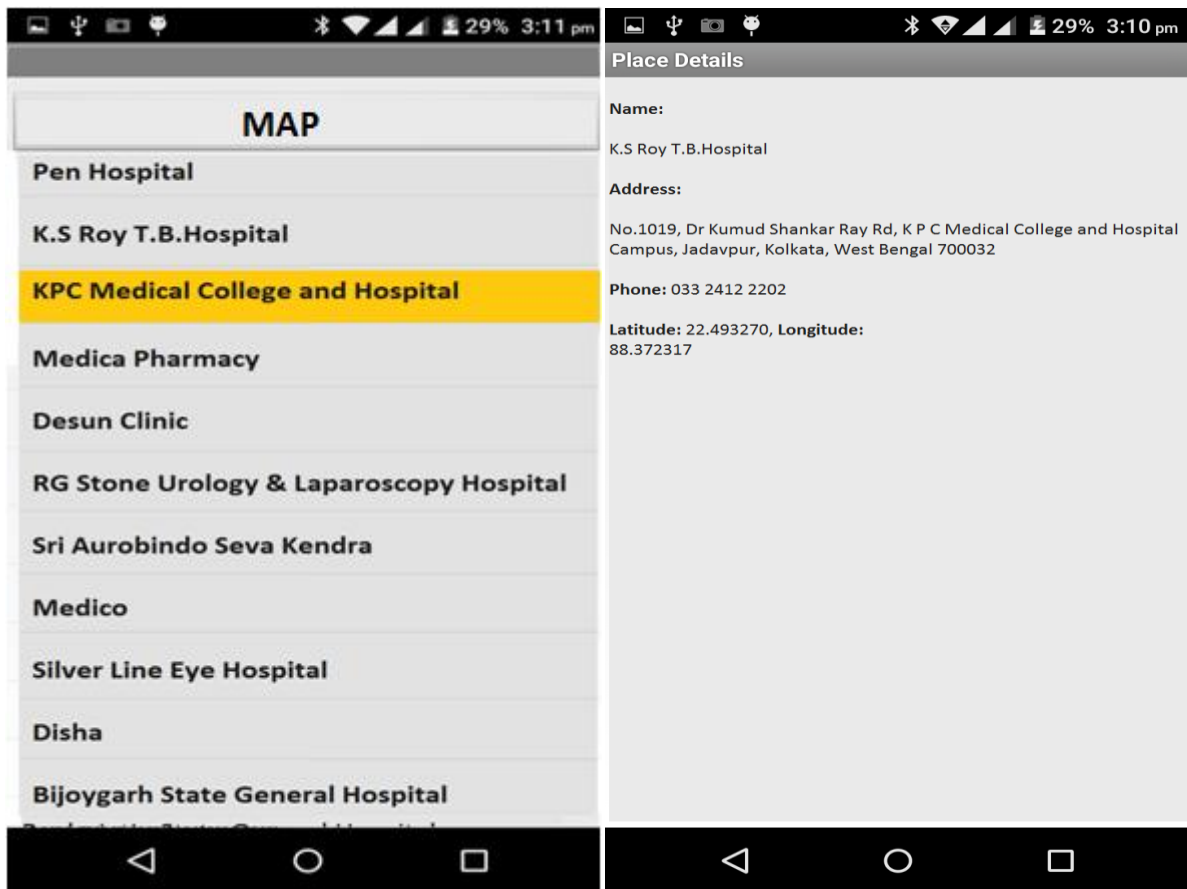


Figure 28 Screenshot Of the App showing hospitals within 1.5 km.

POLICE STATIONS

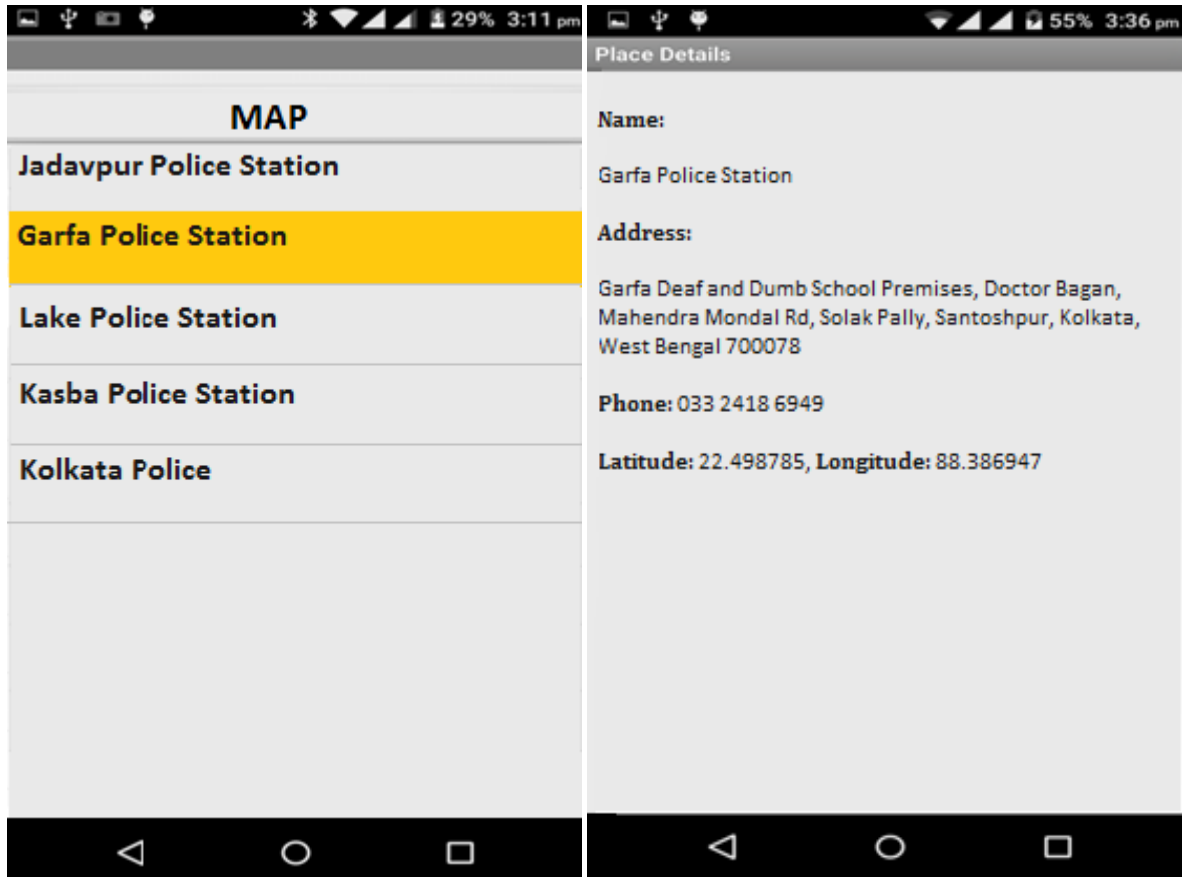


Figure 29 Screenshot Of the App showing police stations within 1.5 km.

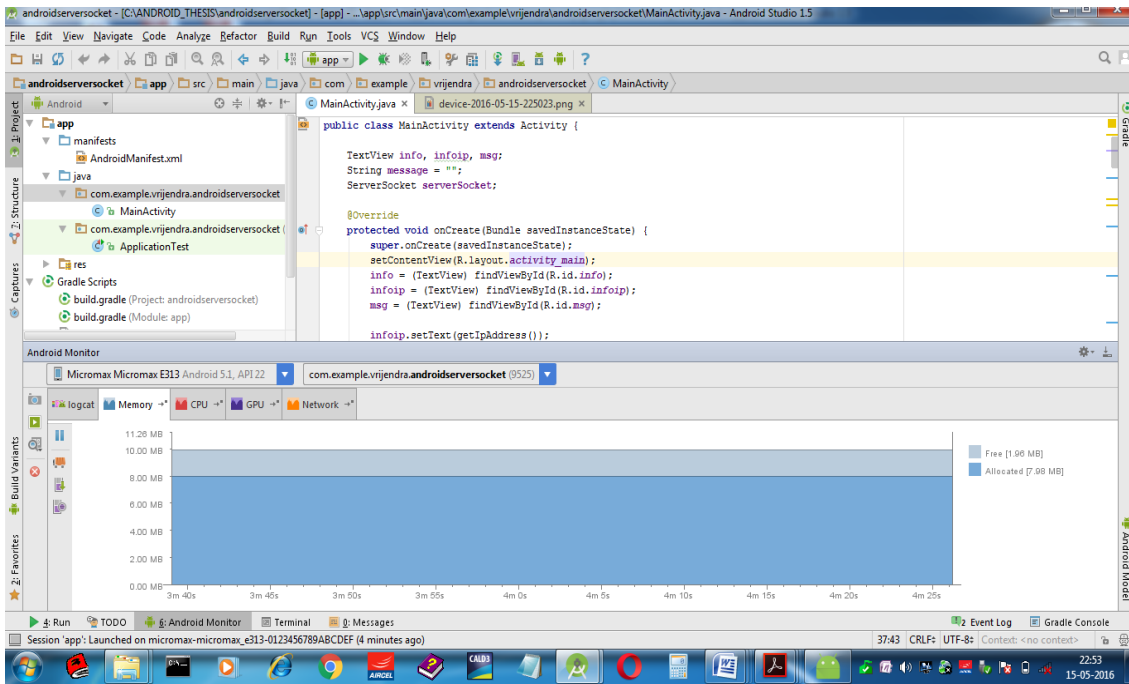


Figure 30 Running of the app which shows places of interest in the vicinity.

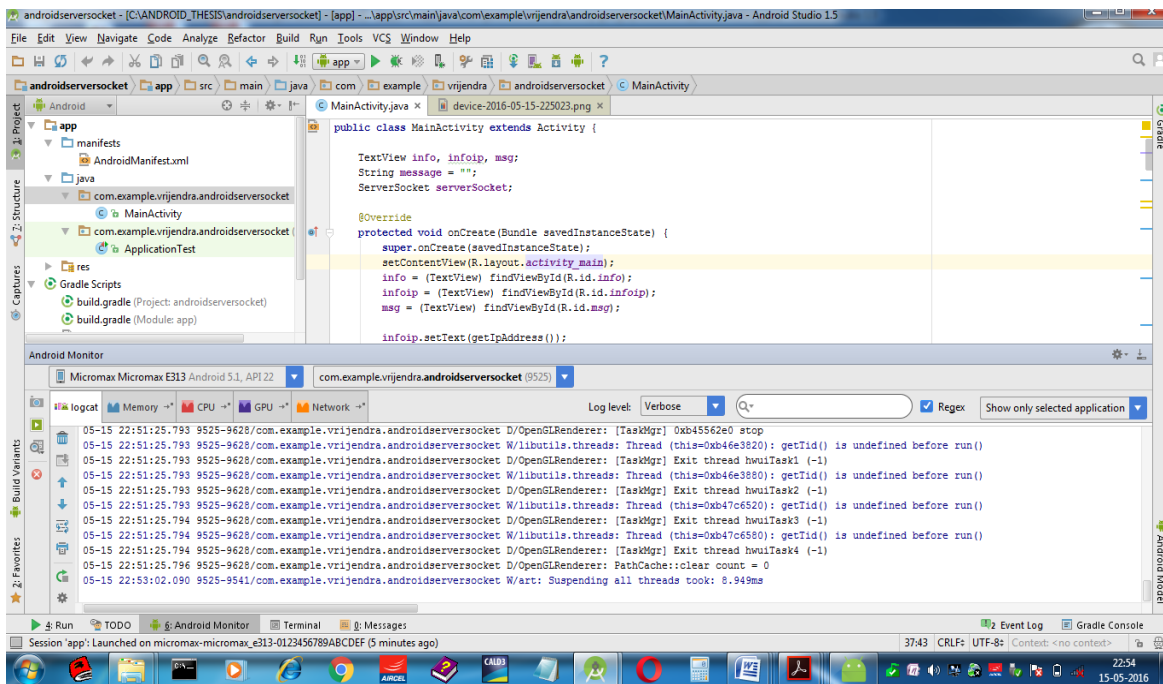


Figure 31 Screenshot of file log while running server Application

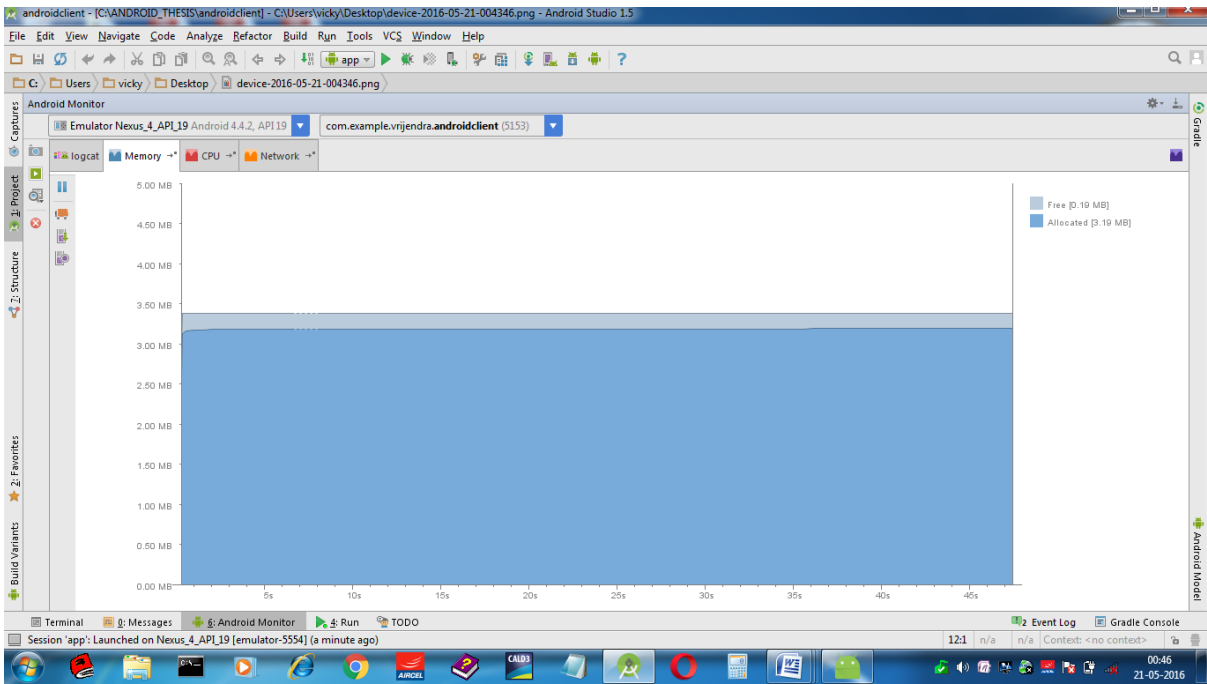


Figure 32 Screenshot of memory usage over time while running application

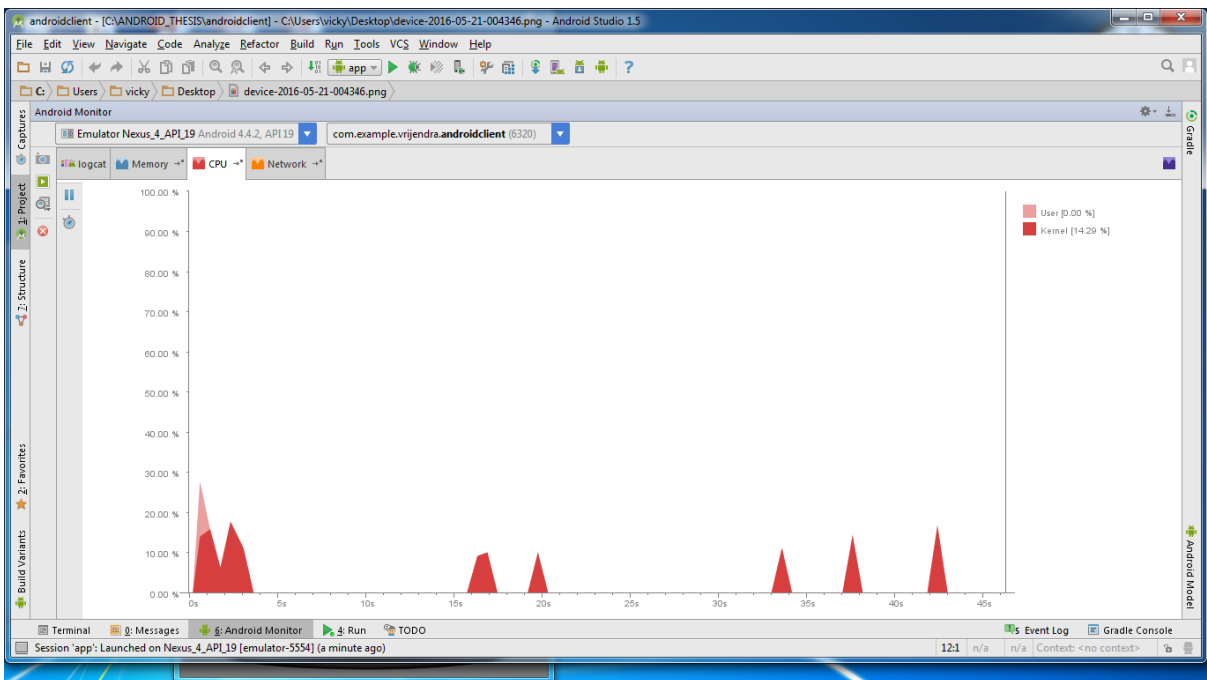


Figure 6.10. Screenshot of CPU load over time while running Application

Chapter 5

Conclusion & Future Work

Conclusion:

This research has introduced an approach to use Google map's api in a more efficient way from a user's view point. Also, we build an app that can give our user a more friendly solution for location based services in a more effective way using sensing proximity. We also aimed to outline the possible way of implementing an effective real time bump monitoring system that address all the aspects of such system and develop a proof of concept application in the field of SPBS to facilitate the use of public transportation for people. Although a lot of effort has been done to detect road conditions, a lot of methods/systems have been developed using Smartphone sensors but a highly reliable method is yet to be built. Real time road anomalies detection is so challenging that none of the methods can address it alone completely. On the basis of works in this paper, we believe that implementing such a system is quite feasible.

Future Work

We can show the existing paths to visit the required destination graphically using Google Map's api. The feasible routes between any one of the places that is listed by our app and our present location determined by GPS , can be a useful work for a user because in that case the user can get all the related information from a single app. Sensing Proximity based services has numerous potential in smart city. The route between any of the lists of places and user's present location generated in our application can give our app user a very good solution from all aspects.

Another important thing that can be determined on a particular route is that road can be sensed with an app that senses the variation in x- axis, y-axis and z-axis which in turn can determine whether bumps are there in the roads and if present there then how much. This information has a very high significance in our Smart city concept. While the system can work even if there is only one user at any road at any given time, the more users on the road the better the system will work. As road condition deterioration is a continuous process, monitoring it in an effective way has been challenge to researchers. Since smart phones are penetrating into common people's lives very fast, utilizing the sensors available in them for roadway monitoring is good idea. The data processed by the mobile can be sent to a central server, which can use the information received to annotate maps accessed by the users through this application. This annotation can contain lot of information like the bumpy nature of the road. Some filtering techniques and advanced machine learning methods are to be applied to further improvement. If the concept is successfully implemented then this will make the lives of people more comfortable and secure. So, it is quite evident that there is huge potential lying behind our present work.

References

[1] https://en.wikipedia.org/wiki/Big_data

[2] <https://eu-smartcities.eu/content/barcelona-world%E2%80%99s-smartest-city-2015>

[3] "A Reservation-based Smart Parking System" by Hongwei Wang and Wenbo Hey

[4] Ganti, R.K.; Ye, F.; Lei, H. Mobile crowdsensing: current state and future challenges. IEEE Commun. Mag. 2011, 49, 32–39.

[5] AN ONTOLOGY BASED CONTEXT MODELLING APPROACH FOR MOBILE TOURING AND NAVIGATION SYSTEM

S. Saeedi, Dr. N. El-Sheimy, Dr. M.R. Malek, N. Neisany Samany

[6] Ling Liu (Georgia Institute of Technology, USA) mTrigger: Location-based Triggers— Scalable and Location-Privacy Preserving Framework for Large Scale Location— Based Services

[7] Jiawei Han (University of Illinois, Urbana-Champaign, USA) MoveMine: Mining Sophisticated Patterns and Actionable Knowledge from— Massive Moving Object Data

[8] Stefano Spaccapietra (Swiss Federal Institute of Technology -Lausanne, Switzerland) GeoPKDD: Geographic Privacy-aware Knowledge Discovery and Delivery.

[9] Android Architecture. Google web site. [Online]
<http://developer.android.com/guide/basics/what-is-android.html>

[10] Android Developers, [Online]. Available: <http://developer.android.com/guide/components/fragments.html>.

[11] Android Revolution HD | Mobile Device Technologies, "Do we really need S-OFF?," 14 June 2013. [Online]. Available: <http://android-revolution-hd.blogspot.de/2013/06/do-we-really-need-s-off.html>. [Accessed 2014 January 16].

[12] Android Developers, [Online]. Available: <http://developer.android.com/reference/android/app/FragmentManager.html>.

[13] Android Developers, [Online]. Available: <http://developer.android.com/guide/components/services.html>.

[14] Android Architecture. Google web site. [Online]
<http://developer.android.com/guide/basics/install-google-play-services>.

