AN EFFICIENT METHOD OF AVOIDING TRAFFIC CONGESTION AND OPTIMAL ROUTING (ATCOR) SYSTEM IN YANGON

NYEIN CHAN SOE

UNIVERSITY OF COMPUTER STUDIES, YANGON

JANUARY, 2020

An Efficient Method of Avoiding Traffic Congestion and Optimal Routing (ATCOR) System in Yangon

Nyein Chan Soe

University of Computer Studies, Yangon

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

Doctor of Philosophy

January, 2020

Statement of Originality

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

……………… ……………………..

Date Nyein Chan Soe

ACKNOWLEDGEMENTS

First of all, I would like to thank the Minister, the Ministry of Education for giving opportunity to attend this course leading to the doctoral degree courses in the University of Computer Studies, Yangon and giving provision to finish this research.

I especially thank to Dr. Mie Mie Thet Thwin, Rector, University of Computer Studies, Yangon, for overall supporting during my thesis. I greatly appreciate the help I have received from her.

My heartfelt thanks and respect go to Dr. Khine Khine Oo, Professor, Dean of the Ph.D. 10th batch, University of Computer Studies, Yangon, for her excellent guidance, caring, and providing me during the Ph.D. study. She also gives us patient teaching during my Ph. D course work.

I would like to express special thanks to my supervisor, Dr. Thin Lai Lai Thein, Professor, University of Computer Studies, Yangon, for her kind support, patience, and guidance over the thesis period. She was a strong influence in instilling inspiration and motivation drive to tide me over the arduous tasks of research activities and her support has been essential for the success of my work.

I wish to extend special thanks to Dr. Myint Myint Sein, Professor, University of Computer Studies, Yangon, for their general guidance, benevolence and encouragement. Moreover, I would like to thank a lot to all my teachers and the board of examiners for making precious suggestions and corrections for mentoring, encouraging, and recommending the thesis. I also would like to express my gratitude to Daw Aye Aye Khine, Associate Professor, Head of English Department, University of Computer Studies, Yangon, for her overall supporting throughout my Ph.D. course work and doing research.

During I have been doing my research, I am thankful to all my friends for their motivating encouragement, for the stimulating discussions about research and many other topics, for being there when I needed them.

Finally, I am grateful to my family and my friends, who specifically offered strong moral and physical support, care and kindness, during the year of my Ph.D. study. Without their full support, my dissertation would not have been possible.

ABSTRACT

Road traffic congestion or jam is the main problem in urban area of both developing and developed countries. In order to solve this problem, traffic congestion states of road networks are estimated so that the best optimal route can be set to avoid the congestions. In this system, the real time traffic congestion states of users` desired between source and destination are estimated and the results presented in Google Map. The historical GPS data of each road network on each time using the collected data are utilized in this research. The proposed system aims to analyze and avoid traffic congestions and get optimal route with modified A* algorithm, also called Routing Pattern Algorithm. This includes the run-time traffic data from mobile with GPS and historical data in the database.

This is the android application to solve the traffic complexing on the roads in Yangon. It helps the people driving and travelling easily on the routes through the mobile phones. Location Based Service is implemented in the system firstly and Global Positioning System is recommended in this work. The system is to analyze and monitor traffic congestion with Geographical Information System and using GPS data for Public Transport Planning in Yangon, Myanmar. This system provides the accurate map of more efficient estimation results for traffic state from GPS data and saving more time other.

The establishment of centralized GPS Server database infrastructure gives any kinds of analysis requiring the stored GPS traffic data in a distributed client-server environment. In this system, the efficient traffic congestion statement for users` desired between source and destination are estimated and the results presented with Map. It takes the GPS data (current location) and searching area of user by K-d tree and Haversine algorithm. Secondly, search the traffic congestion data by Google Traffic layer and Routing Pattern Algorithm. Finally, Analysis the traffic by Routing Pattern Algorithm (modified-A*) and then show the result of traffic congestion statement and best optimal route. In the case, there are three main components: Data Collection, Data Extraction and Implementation. And this is Client-Server database system that stores the data and server in the cloud Virtual Machine (VM) that also called droplet. The droplet is a type of cloud database and this can be ordered and used for any projects.

These estimated traffic probabilities states are presented by coloring (traffic jam for red, traffic normal for orange and traffic light for green) users' desired source and destination road segments on Google Map. The estimating system has been evaluated by using dataset generated by collect data from mobile phone-equipped vehicles over a period of one year in Yangon.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1
INTRODUCTION

Yangon City is the main center of Myanmar, an old capital and economic zone. The city has a population of about 6 million people, accounting for 14 percent of the country. Such city with a population of more than 6 million, Yangon's traffic situation is very similar to that of other great cities, such as developing countries around the world. In Yangon city, the most common forms of transport generally for citizens and people are buses lines such as Yangon Bus Service (YBS). But, due to the traffic congestion in any places, most of the buses and private cars are travelling very slowly during rush hours.

The survey used GPS data from vehicle phones to get real-time traffic conditions. The system uses the proposed KD tree and the improved modified A* algorithm to reduce data complexity or latency, estimate current traffic congestion conditions and resolve the best route for users in crowded cities. The purpose of this proposed system is to show the traffic congestion area of the route in order to get accurate location map results in a short time. The last motivation is to provide mobile users with the best route with accurate data results.

1.1 Problem Identification

Global Positioning Service system or GPS is a relatively new technology. Although it was mainly used for military purposes at the time of invention, this technology was later used in many consumer applications. In recent years, each smartphone has been primarily equipped with Google GPS and location information services to complement and manage applications such as Google Maps [42]. Therefore, using GPS data, people activities can be represented more accurately than ever before. As a result, recent years have seen a huge number of applications and usage services based on the location of the smartphone. Considering that many drivers and travelers are smartphone users, road traffic scenarios can also be proved using GPS data [1]. And this is consistent with the fact that even in developing countries, more and more growing people are now using GPS smartphones, which are traffic checking and avoiding system based on the concept of collecting and classifying location data. Using a specific highway to be transmitted and using this data to represent the statement of traffic condition present on a particular highway, a very practical and versatile traffic avoiding system can be provided [42]. Therefore, the purpose of this system is to improve a real-time traffic detecting, monitoring and avoiding framework consisting of mobile applications and back-end Web services for fast and efficient deployment in some ASEAN countries. This article also examines various traffic detect and checking systems that use similar concepts. The system will also research and develop the algorithms and workflows required for data transfer between end users and servers [42]. In addition, the system investigated possible failures and the scope of the system improvement [1].

One of the ways to solve this congestion problem is to let travelers make quick and informed decisions through Sha Tao's real-time or up to time traffic information [41]. The Traffic Information System (TIS) is one of the traffic analysis systems that provides people and travelers with useful traffic data and information to help them make decisions about the route. TIS takes full advantage of the rapid development of sensors, computers, electronics and communication technologies. The latest TIS takes two general forms such as cellular and sensor-based network management system. The Sensor-based TIS is costly to distribute and maintain; it covers only a little number of roads. Traffic analysis of the TIS cellular network can solve problems associated with high cost and limited coverage; however, it experiences significant precision changes. In the present, a new method has been considered which uses a mobile phone (with GPS function) as a device for collecting traffic data [41]. Traffic congestion sometimes appear if the traffic capacity depended by the transportation facility is close to or lower than the current traffic demand (such as high vehicle value, other obstacles (such as accidents, double parking), insufficient green traffic lights, etc.). Regardless of the cause, traffic congestion increases travel time, resulting in delays in transportation systems and inefficient operations. Over the years, many researchers have proposed different solutions to solve traffic congestion problems. One of the solutions to avoid traffic congestion is to detect traffic congestion on the road. Therefore, the traffic jams status and the best optimal route of the road segment required by the user in real time have been estimated and avoided.

For vehicle positioning, mobile GPS positioning performance will be affected by actual conditions in other outdoor environments. Therefore, in order to evaluate performance, a reliable method is to perform field testing. However, evaluating testing systems in this way is costly in all environments [15] [41]. The location positioning simulator should be able to provide reasonable positioning accuracy estimates under realistic conditions [19] [41]. This chapter introduces a location simulator that can help analyze OTDOA positioning provided by UMTS. The first part makes the models of the global wireless network at the system level. Includes this network topology, mobile mobility, and radio propagation model. The second part of this research is a WCDMA FDD (link level) simulator. This describes the transmission code generation module, Rayleigh fading channel mode, and related reception modules. The third part is location estimation that consists of time measurement algorithm and location implementation algorithm. The system level models and positioning algorithms were developed by MATLAB language. The link-level simulator is implemented in Simulink, which uses the Communications Tools system and mobile WCDMA network end-to-end physical layer [41].

There are many processes in estimating or detecting urban traffic, such as data fusion, fuzzy control theory, routing pattern method and microscopic route traffic simulation. However, all these methods required expensive traffic data. Therefore, the choice of process to take to estimate, detect and avoid traffic is related to the nature of the available data.

In our day-to-day operations, it is necessary to integrate GPS to capture spatial and temporal data. There is a growing need for practical methods involving data mining and mapping of raw GPS data from GPS trackers for a variety of purposes and operational purposes. Activity-based analysis using GPS devices as data collectors has been a major issue in recent decades. As the demand for data grows, more sophisticated methods of data collection have been developed, initially represented by the transition from travel diaries to activity diaries, and the development of GPS-enabled activity tracking continues.

Traffic analysis is one of the key aspects of demanding better and effective avoiding in developed countries. In order to control and manage the traffic on Yangon streets, the government relies heavily on transportation policies that are strategically located on major roads, bypasses and intersections. Due to the intelligence system in the traffic and lighting system, Traffic-lights at the main roundabout on the road leading to the Yangon Central Business District are often replaced by the police.

1.2 Motivations of Thesis

Traffic Control and Management in Yangon has been and still is a major concern to the Government, road users and the Traffic Department of the Yangon Police. In the recent years, Yangon has been experiencing extremely heavy traffic especially towards the City downtown and Central Business District (CBD). This is as a result of the upgraded in the number of mobile in vehicles especially privately-owned cars and Public Service Vehicles (PSV) hence heavy load than the roads can accommodate. Furthermore, the manual systems of traffic control and analysis together with lack of intelligent traffic lighting systems on major roundabouts also contribute to traffic jams and congestions all over the Yangon region.

It is as a result of the above stated problem that the research project focused on the use of GPS technology incorporated with a road mapping concept to help in the analysis of traffic on Yangon roads, for instance, speed of traffic and road usage pattern analysis. The project was also geared towards the establishment of a centralized GPS data bank through the development of a GPS Server.

. Congestion is directly related to the dynamic interaction between demand and traffic volume. If traffic demand is high and road capacity is low, congestion may occur. Variables such as speed, strength or density can be used to measure traffic conditions. In order to successfully implement real-time traffic congestion, information about past and current network status is required. However, the road network does not have enough sensors to get complete and accurate information.

Today, the information that can be obtained from GPS is very rich for development. It can be used through city network floating cars. You can get information about the state of the network from the collected data. This means traffic conditions. In this thesis, GPS data from used mobile on vehicles has been applied to get the update time and historical traffic status data. Then, the current traffic condition is got by Google Traffic layer and Distance Matrix Routing Pattern Algorithm to show the road traffic congestion condition results of user's desired road network to user's mobile phone.

Google Maps and some Navigation system does not have up-to-the-minute information, some important locations are missing, less accurate at another times, and limitations of satellites. Besides there are not locations that Traffic-light points in these applications. City needs application to reduce traffic jams on the roads and streets quickly by users or human. In this way, mobile GPS applications and web-based traffic analysis systems were developed that collect GPS traffic data and provide the possibility to avoid Yangon traffic.

1.3 Objectives of Thesis

The major objectives of thesis are five objectives. The first objective is to analyze and avoid the traffic congestions with GPS data and map. And then to assist the people by using mobile system with taking the supplies of the transportation that leads to be wastes a lot of gas and fuel. The third one is to find nearest bus points and traffic points of the user and to support the time management.

The fourth objective is to show efficient traffic avoiding system with more accurate and performance that avoid the traffic congestions. The last objective is to implement the process to estimate and avoid traffic jams of a route for a user to choose the best optimal route to the target destination.

1.4 Contributions of Thesis

The contributions of thesis for avoiding traffic and optimal routing are in the following ways:

It avoids current traffic conditions of user’s desired source and destination using mobiles GPS data and present result to users’ mobile phones to avoid congested roads in Yangon. The one is creating the Android Application for Location and Navigation system and viewing the routing statements.

Many people handling mobile phones use the application to check and avoid the traffic jams easily in Yangon Division. Data storing and extracting by using KD tree indexing method for reducing data complexing and bottleneck for data in server. Data Analysis by using Routing Pattern Algorithm with loading traffic data is taken from Google Map Layer. It reduces mismatch with wrong road segments from ambiguous GPS trajectories data. Modified A* algorithm is used to get the best optimal route using both historical and real time traffic data. It uses Cloud database (Virtual Machine) as backend server to reduce processing time and better performance.

1.5 System Overview

In this thesis, traffic avoiding and optimal routing system using GPS data is developed in a cloud database. General architecture of the system is presented in Figure 1.1.

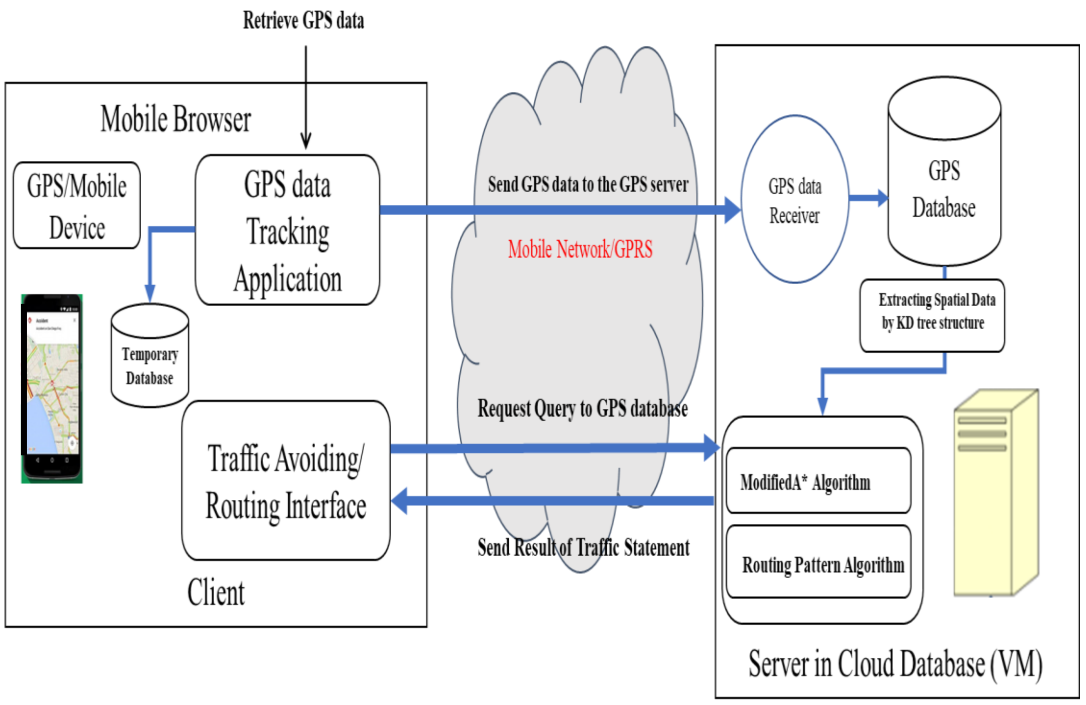

Retrieve GPS data
Mobile Browser
GPS/Mobile Device
GPS data Tracking Application
Temporary Database
Traffic Avoiding/ Routing Interface
Client
Send GPS data to the GPS server
Mobile Network/GPRS
Request Query to GPS database
Send Result of Traffic Statement
GPS data Receiver
GPS Database
Extracting Spatial Data by KD tree structure
ModifiedA* Algorithm
Routing Pattern Algorithm
Server in Cloud Database (VM)

Figure 1.1 General Architecture of The ATCOR System

The system displays the results of avoided traffic congestion conditions for the source and destination desired by the user. Car GPS Android devices are used to capture GPS location data, start and end points, and then transfer this data to the data center. Use a map matching algorithm to merge this data with the road network to see which vehicles are on which road networks. However, if the GPS data is not clear, the match of the map will be inaccurate. When the system corresponds to the road segment of the wrong vehicle, the estimation of the traffic congestion state result will be incorrect because the road conditions of each road segment are different.

The Google Maps data layer provides a function of container for storing data in any geospatial space. It can use the data layer to collect the custom data and represent Geo JSON data in Google Maps [52]. Using the Google Maps JavaScript API, this can mark maps with multiple overlays, including markers, polylines, and polygons. Each of these functions combines the embedded style information/data with location data. If the data contains geometry such as polygons, lines, and points, by default, the API displays them as markers, polylines, and polygons. This can create these features as normally overlap and apply style rules based on other dataset properties.

1.6 Organization of Thesis

This thesis is organized with seven chapters. The motivations, contributions, system overview and the objectives of the thesis are described in this chapter 1.

Chapter 2 contains the literature reviews and related work with some existing methods, which are also surveyed the prior studies that are dealing with the thesis.

Chapter 3 describes background theory. This chapter also presents an overview of the background theory using in traffic avoiding system.

Chapter 4 elaborates the detailed design and implementation of efficient traffic avoiding and optimal routing system.

Chapter 5 presents the evaluation and experimental result of the optimal routing and road traffic congestion system in detail.

Finally, Chapter 6 concludes the thesis with a summary, and suggestions for future extension research.

CHAPTER 2
LITERATURE REVIEW AND RELATED WORK

Traffic congestion remains a major social problem worldwide. Therefore, urban traffic management, such as traffic congestion avoiding systems, includes the management of public transportation and the processing of specific information sources, such as priority, real-time management of cities and traffic signals.

Traffic management measures have increased over time, so from an architectural point of view, data, applications and control are extensive (traffic speed, heavy vehicles, excellent, etc.), including the use of vehicles for effective traffic analyzing VMS (variable Message signal), etc.). Cycling, use of public transport, vehicle occupancy and traffic congestion. This change in research methods improves the accuracy and effectiveness of management strategies and policies. More information about the information acceptable on the route pattern network, such as the speed limit and the passage of time, traffic management will be more accurate and efficient. The choosing of right data is an important step in traffic management, depending on the city [29]. The complexity of urban transport depends on the participants that may lead to conflict. Therefore, priority systems such as Automatic Vehicle Identification and Positioning (AVI / AVL) are critical to improving traffic management strategies.

Travel time is an important information to help investigate traffic conditions. Several methods have been improved to gather information on traffic speed and trip time, including probe vehicles, moving objects, loop detectors, motion sensors and GPS service on vehicle. Some researchers have discovered a variety of travel time estimation methods based on predictive or estimation models with historical data.

2.1 Case Studies Comparing

As the introduction mentioned in the previous chapter, in the last few times, there have been many difference types of iterations and processing of both simple traffic detecting systems, but more recently, mass supply and location data. To analyze the feasible ways to solve, existing systems and how they work are needed to investigate.

2.1.1 Android System & Google Maps API

Google Maps is a desktop/mobile web map service developed by Google. This is the main mapping application for the Android operating system and can be used with all other mobile operating systems. Provides information about satellite imagery, street maps, panoramic street views, Google traffic, and public bus transportation in certain countries. There are other map services, but Google Maps is one of the best navigation services on the mobile and web for the people. In August 2013, the mobile version of Google Maps was the world's most useful smartphone app, with more than 56% of smartphone users worldwide using it at least once [45].

For developers and researchers, Google released the Google Maps API since 2005. This allows Google Maps to integrate with mobile websites and applications. The Google Maps API services and location-related operations such as mobile phone GPS access, location-specific information retrieval, route determination, and more. [46] As a result, many applications on Android, Java, IOS and other smartphone (OSs) operating systems use the Google Maps API to identify location information.

In the proposed traffic processing application, this is the best choice for the app because this can use the Google Maps API to access all the necessary data. Android is also a Google product and is fully compatible with the Google Maps API, so choose Android for your mobile operating system. In addition, Android's market penetration rate is much better than any other mobile operating system in the world, apparently in Bangladesh [1] [47].

2.1.2 Google Map Traffic

Google Traffic Detecting service is a function of Google Maps features that displays real-time traffic statements for major roads. Google traffic have been detected on the Google Earth or on the Google Maps app on handheld devices [46].

The first version of Google Maps provides users with information based on historical data how long it takes to travel on a particular road, streetway. This information is not real-time and is very inaccurate. In 2005 year, Google acquired Zip-Dash, which specializes in real-time traffic analysis process. In 2007-2008, Google integrated Zip-Dash technology into Google Maps to provide traffic data based on anonymous information collected from mobile users as the Figure 2.1[1].

Google Maps is also committed to the concept of crowdsourcing. Crowdsourcing is the process of requesting power information from a huge number of people. Google said: "If you combine this speed with the speed of other mobile phones on the go, you can always get real-time traffic flow views through thousands of mobile phones all over the city." [1]

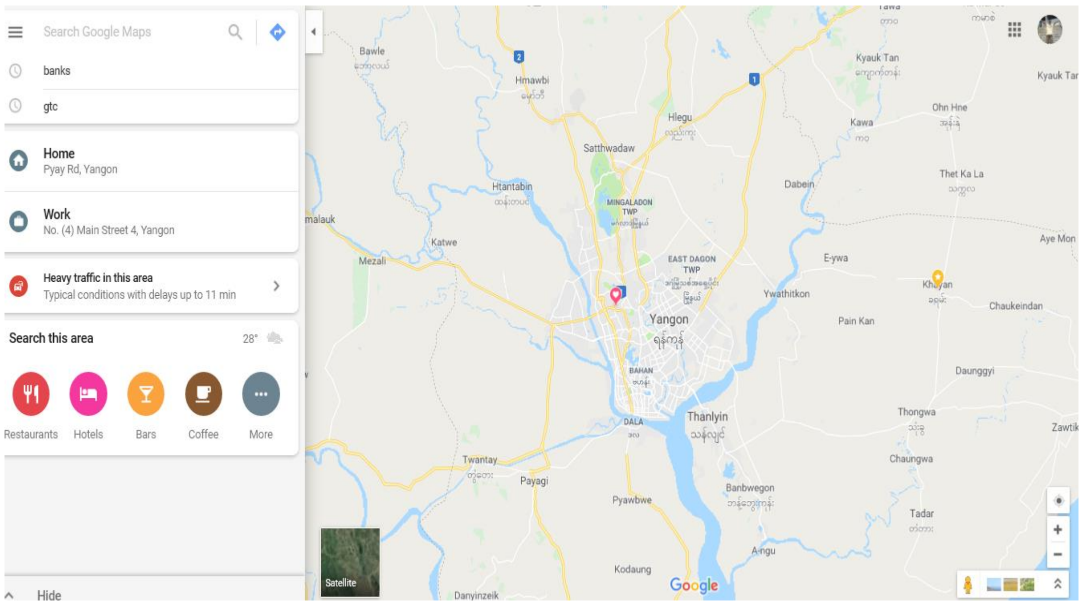


Figure 2.1 Google Map Desktop Version

2.1.3 Analyzing Traffic Flows from GPS Signals

Alfio Costanzo [7] The main purpose of the Mobile Information System (MIS) is to support both driver and pedestrian mobility and logistics activities using both specialized navigators (Garmin, Tom-Tom, etc.) Explained that it is to be a smartphone. MIS typically helps mobile users find the best service for their business and the best route to reach them from their current location. However, the recommended route may not match the best route due to lack of data on current vehicle traffic. In fact, all market surfers use information about average traffic volume and have not taken into account accidents, car congestion, or weather conditions that significantly change driver behavior and road travel time involved.

Creating the City Road Graph mentioned above is a fairly complex and time-consuming calculation. But it runs very rarely. That is, at the start of the information service and when there is a change in the territory infrastructure, for example, creating new roads or deleting old roads [6]. Therefore, the temporary performance of this process does not interfere with the temporary performance of the proposed service. After you create a city street chart, you can easily track mobile users using GPS data that is regularly sent from your mobile device to the MIS. By processing this data, the MIS can locate the driver as shown in Figure 2.2 and monitor the traffic flow as shown in the next section [7].

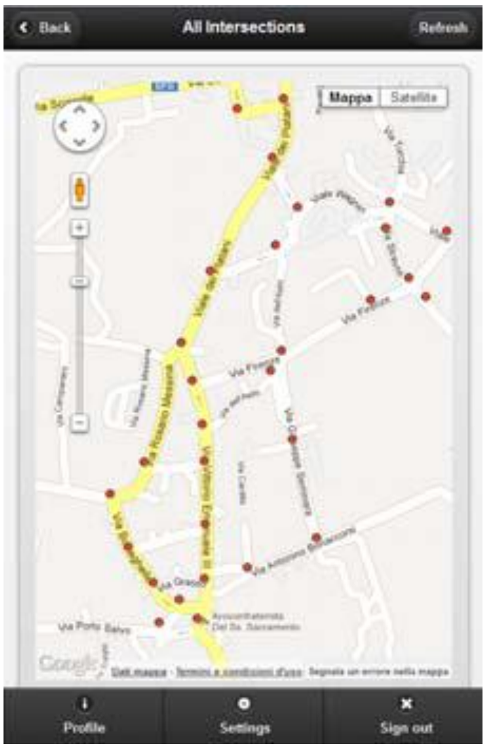


Figure 2.2 Urban Road Graph

The GPS of the driver's mobile phone is faster than the method proposed by S. Tao [40], etc., so that the server can calculate the travel time of each section of the road in real time. In this way, using the appropriate color scale, you can visualize the current traffic flow on the driver's mobile device map and provide the ultimate support system for decisions made on the server. It uses relevant information to coordinate user-scheduled activities. In the proposed methodology, the GPS of a user traveling on a city road map is periodically sent to the server, and the user. The time when these coordinates were detected. This allows MIS to locate the driver on the city road graph [6].

To build a traffic checking system, an enough number of drivers need to send GPS data to the server and calculate traffic flow in real time while traveling. In order to encourage users to do so, this will offer discounts to receive traffic information and offer some additional features [5]. Because it is not easy to get traffic information from a driver's GPS across the entire traffic network to be suitable to maintain the all of network. For example, webcams can be used in busy streets where traffic is low, or where video surveillance is useful and traffic can be calculated as the following Figure 2.3. The proposed GPS system may be suitable for the surrounding and busy city centers [7].

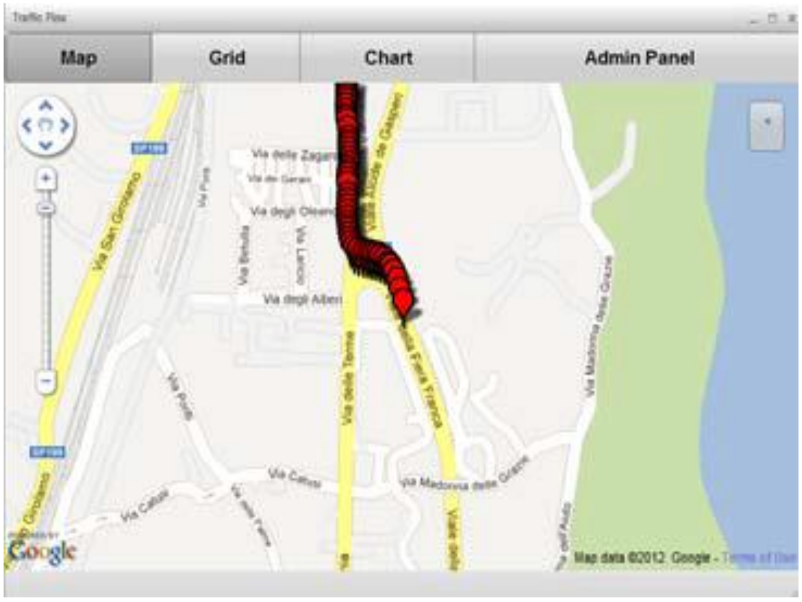


Figure 2.3 Positions with Traffic Checking Software

2.2 Traffic Estimation Using Traffic Information Systems

Some research related to trip time estimation was based on the fact that traffic flow was not interrupted. However, traffic flow is so dynamic that these studies cannot be applied directly to the archived real world. There are many different states to estimating and detecting urban traffic, such as data fusion, diffusion control theory, and microscopic traffic simulation, to solve this problem in the literature. However, all of these techniques require expensive traffic data. Therefore, the choice of steps to take to estimate traffic is referenced to the nature of the available data.

Different types of sensors have been developed to collect different types of traffic data. General traffic data can be summarized as speed data (distance per unit time), occupancy data (percentage of time at a point on a road occupied by a point) and traffic data (amount of vehicles per unit time). Density data and travel time (time required to trip between two locations). Another type of data is vehicle route data stored from GPS enabled vehicles. The latter type of data has the advantage of directly calculating going time, distance and speed.

Sensors used for collecting traffic information are:

1. GPS Sensors
2. RFID Transponders
3. Wireless Sensors

2.2.1 Global Positioning System (GPS) Sensors

The two types of GPS probe data that can be collected from GPS sensors probe GPS data and high-frequency GPS data sparingly. Probe GPS data rarely indicates when the probe sends GPS information at a fixed frequency. In addition, it may not be enough to accurately measure changes in speed or travel time (frequency may exceed 10 seconds). There are many challenges to estimating travel time using this type of data. The first is related to GPS location mapping or digital maps on the road network. This means that the correct location must match the correct road section and the correct itinerary of the probe trip. The second challenge relates to the situation where the probe travels many roads in the road network before sending GPS data.

Therefore, a route estimate is required to calculate the travel time estimate. High-frequency GPS indicates that the probe vehicle can send current GPS information every few seconds or every second (within 10 seconds). Based on this data, the speed and travel time can be derived directly from a short distance. Since the intersection may be ambiguous, the map matching problem still exists, but it is usually easy to determine the path when examining the entire trajectory. Moreover, receiving this type of traffic data each second through the means for data communication into the server is not cheap.

These GPS data are collected using vehicles' GPS sensors in most system. The GPS cell phone as a sensor is used to collect data from the GPS probe of this system because of its low cost, high penetration rate and high accuracy. In the era of multimedia convergence, communication and detection platforms, GPS-enabled mobile phones are becoming an essential contributor to location-based services. These devices combine the advantages of the aforementioned mobile sensors (low investment cost, high penetration rate, high accuracy with GPS receiver). In addition, GPS-equipped mobile phones can accurately provide not only location but also speed and direction of movement. The following Figure 2.4 shows the collected GPS data using GPS-enabled mobile phone in selected area.

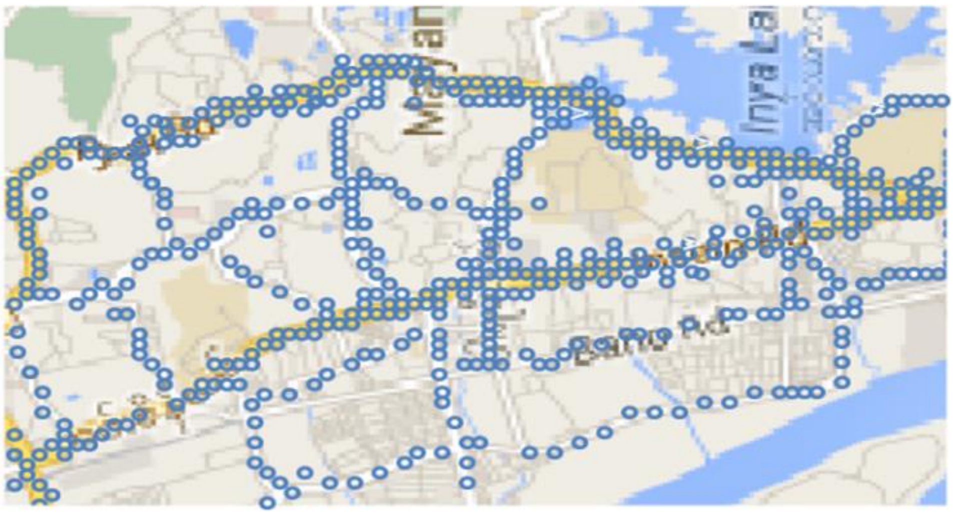


Figure 2.4 GPS Data from Mobile Phones on Vehicles

2.2.2 RFID Transponders

Radio frequency identification (RFID) is a ubiquitous technology in many fields. Transport agencies use RFID in a variety of ways. One of the original uses of this technology was the collection of tools by the user (client) as he crossed the bridge or entered/out of the toll road. The vehicles have an RFID radio transponder that is detected and take by a reader located at the entrance or exit of the toll road. Typical RFID systems include RFID tags, this readers and middleware, as shown in Figure 2.5 below [22].

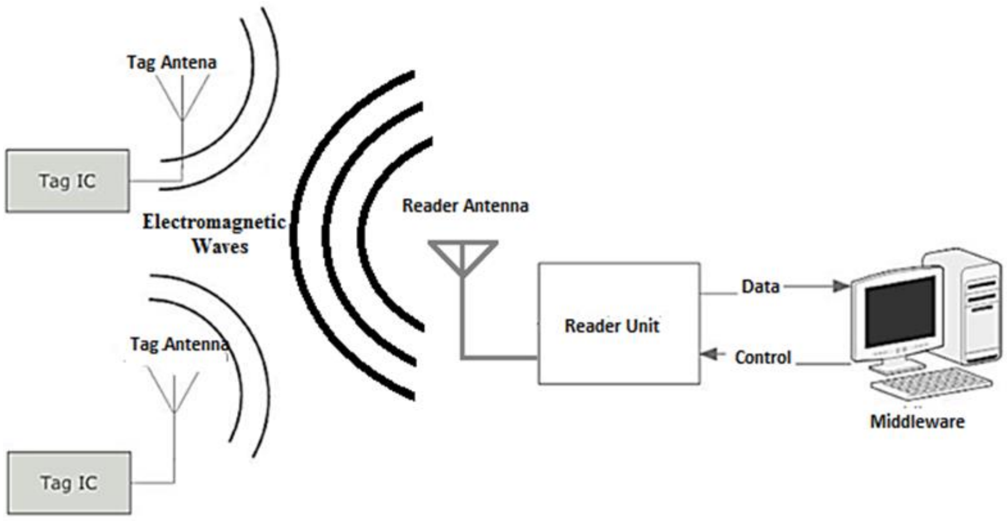


Figure 2.5 Components of an RFID System

The same technique can be used to store traffic data by placing the readers in various locations on the road. Travel(trip) time can be stored between each of points and processed in the same manner as the license plate reader can handle travel time. The accuracy of RFID transponders depends on the signal strength. In general, it is sufficient to provide an estimate of travel time for long distances, but it may not be possible to provide accurate travel time for short distances. RFID readers are usually far from the current distribution; therefore, they can help collect information about long travel times, but do not provide input data for detailed traffic estimation algorithms. In Ryan Jay Herring [22], an RFID-based simulation system was proposed to estimate traffic congestion. The RFID reader reads the vehicle tag and transmits the necessary information to a database in the central computer system.

2.2.3 Wireless Sensors

Wireless sensors are small devices (similar to loop detectors) that are integrated on the road, but the detection mechanisms in these sensors allow the vehicle to be re-identified with up to 80% accuracy in successive sensor locations of a particular system. Therefore, these sensors provide travel time for most traffic flows. The main advantage of these sensors over other travel time measurement sensors is their lower production costs, making it possible to deploy on a large scale on a trunk road [22].

When noise affects the position of the car, problems arise, making it difficult to assign travel time to the relevant area. However, as shown, this problem can be partially resolved by inserting an algorithm that estimates the nearest points. The authors compared two methods: probability classification and proximity. The results show that the best accuracy can be obtained using a simple brute force method (recent neighbor K).

Another problem is that the access point may not be detected correctly due to WiFi data interruption. As a result, travel time information for certain sections is not sufficient.

2.3 Methods for Traffic Estimation

The mapping is the process of matching the original GPS data with a road network on a digital map. Map correspondence is an essential step for many applications and researchers, such as traffic flow analysis, traffic jam estimation, and driver behavior research, as it can determine which road a vehicle is located on. However, since the GPS sensor reading has positioning errors and sampling errors, it is difficult to avoid starting GPS tracking data from the actual trajectory. In order to solve the problem of correctly determining which vehicle is driving on which road, the researchers proposed different mapping methods for traffic management.

2.3.1 Using Map API Key by Google Map Data Layer

This section shows how to load data from the same domain as the JavaScript MAP API application or from a different as like. There are loading data from the same domain. The Google Maps data layer supported as a database for arbitrary geospatial data (GeoJSON). If the data is in an index located on the same domain as the JavaScript Maps API service, it have load the data using the map.data.loadGeoJson () method. Then the index file must be in the same domain, but can be hosted on a different subdomain [52]. For implementation, the system can send a request from www.example.com to files.example.com.

```
map.data.loadGeoJson('data.json');
```

If the system is loading information from multiple domains, it has request data from a domain other than our domain configuration allows such requests. This authorization standard is called Cross-Origin Resource Exchange (CORS). If our domain allows this domain requests, the response header must include the following sentence:

```
Access-Control-Allow-Origin:
```

2.3.2 Fuzzy Logic Map-matching

Fuzzy logic is the effective way to be sample with tasks involving qualitative terms and concepts, ambiguity and human activities intervention. Expert knowledge and experience employed by fuzzy algorithms based on logic-based concept is represented as the rules for determining vehicle location.

Carola A. Blazquez [2] implemented the map matching algorithm based on simple fuzzy rule system. The algorithm estimates the probability that the candidate road is the actual driving road. To do this, it uses fuzzy tree rules, including route comparison, road similarity, one-way or free lane and off-road verification. The test results of the simulation data show that the algorithm has higher precision.

2.3.3 Personal Navigation Assistants and Map-Matching

The researcher, White et al. discussed the solutions to map matching problems for Personal Navigation Assistant (PNA). Four different map matching algorithms have been implemented and tested:

1. Take the minimum(shortest) distance (point to point),

2. Compare route information with arcs and tracks,

3. Take topology to select roads that can be reached through different routes

4. Use these points to compare them to the curve of the centerline (curve-curve corresponds).

These algorithms are most effective when the distance between the GPS point and the correct and tight match to get at higher speeds on the highway. Carola A [2] describes a method for updating digital maps using GPS points to identify map inconsistencies.

2.3.4 Topological Network-Based Algorithms

A map matching algorithm using connectivity and continuity information is called a topological map matching algorithm. The correspondence of the topological maps of the decision rules determines the midline of the correct road traveled by the vehicle to obtain the shortest route between the GPS data location points acquired in the post-processing mode. Greenfield proposed a map matching process consisting of two algorithms. An algorithm estimates and evaluates the similarity between the types of the road network and the vehicle located positioning model. The second algorithm performs a topological implementation and it applies weighting values to match each GPS data point like parameters to the road network. The most valuable weighted parameters score determines the most likely families for the correct match [44]. The authors point out that further research for extension is needed to determine accuracy data that the correct location of the vehicle along the road segment and to prove the accuracy of the algorithm.

2.3.5 K-D Tree

The k-d tree is in which each leaf node is a k-dimensional point. It can be considered that each node does not generate a genre by implicitly generating a segmentation hyperplane that divides two parts space (called a half space) [50]. The point to the left of this hyper-plane is represented by the left substructure of the node, and the point to the right of the hyper-plane is represented by the substructure of the right. The direction of the hyperplane is selected as follows: each tree node is associated with one of the dimensions k, and the hyperplane is perpendicular to the axis of the dimension. For example, if the "x" axis is selected for a particular partition, then all points in the substructure whose "x" values are lower than the node will appear in the left substructure, and all points with larger "x" values will Appears on the left. In the correct substructure. In this case, the hyperplane will be set by the x value of the point, and its normal will be the x axis of the unit [50].

Henry Stern has briefly reviewed some basic methods of communicating with the nearest neighbor and partially meets the requirements of a computer science degree [38]. A basic discussion of the compromises that arise from spatial and temporal complexity occurs for each method and describes the KD tree in detail. This describes techniques for optimizing and searching query times as shown in Figure 2.6.

The discussion on tree optimization is centered on the strategy called “split strategy”, and the compression of the area and data is also briefly discussed. The department strategy described here is a simple strategy that selects the widest dimension of the department and the most disparate dimension of the department. Search algorithms based on the information described are detailed survey, first search, and A * survey. These algorithms make use of various heuristics that are affected by tree construction [38].

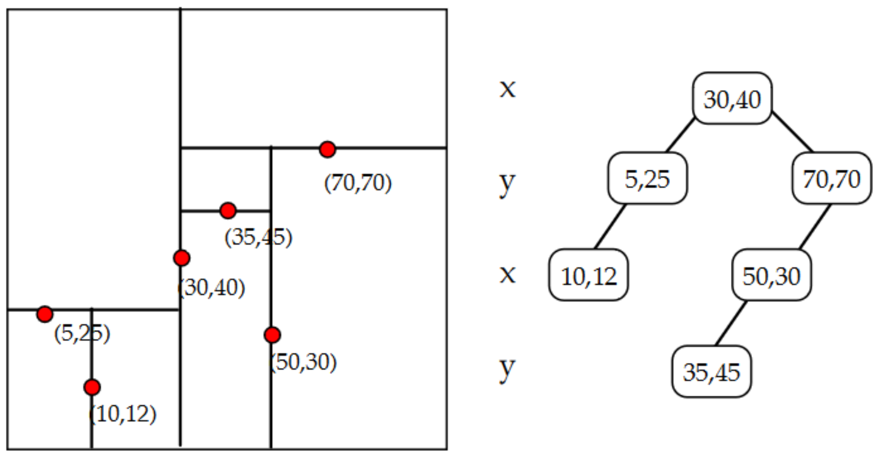


Figure 2.6 KD Tree with Parameter Values

In the research, the system choices K-d tree structure for data storing and extracting because of this tree is most suitable for spatial database. For data balancing the system requires K-d tree because It is sorted in any multiple dimensions.

The following algorithm is the procedure for constructing the KD-tree.

Algorithm BuildingKDTree(A,weight)

```
begin
        if A includes only one point
            then storing this point and return a leaf
                  else if weight value is equal to even
            then Split A with the median x-coordinate on vertical line l
                  else Split A with the median y-coordinate on horizontal line l
         Vleft ← BuildingKDTree (A1, weight+1), A1++
         Vright ← BuildingKDTree (A2, weight+1)A2++
         Create a node (V) saved l,
        Act Vleft with the left child of (V), and act Vright with the right child of (V).
return (V)
end
```

2.4 Traffic Considerations in Real-time Transportation

As the fastest or shortest routing strategy in local traffic optimization, there are many traffic routing standards. However, the lack of a global perspective may lead to even worse choices and even traffic congestion. Since the traffic is dynamic, and any static policy is inherently inadequate. It follows that need a dynamic policy with real-time traffic considerations. This dynamic policy has a global perspective of the real-time traffic status and it will always select a currently global optimal path from origin place to the destination.[24]

We can consider the route selection problem as a search problem, and our goal is to search a path from the source to the destination which costs us less time and/or money. The research will introduce the A* search into this project that use the traffic status as a heuristic. It has mentioned that a route selection problem is a graph search problem. And it can model the real-time transportation route selection with traffic considerations problem as an informed search problem, specifically an A* search problem.[24] Since these are not only trying to find a shortest route among two or three vertices in the graph, but a path that will cost us less time and money. Intuitively, the traffic situation is the heuristic in our search. In order to model this problem, the system needs to setup data structures for each road and for traffic congestions.

2.4.1 Demonstration and Traffic Analysis

The other component is the simulator that interprets these two files and search for an optimal path. And the simulator is consisted by two main parts that one is the A* searching engine and the other is the Map-traffic interpreter. The interpreter will translate the map and traffic information into inner data structures and serve for the A* searching engine. The structure of the project is designed as following Figure 2.7:

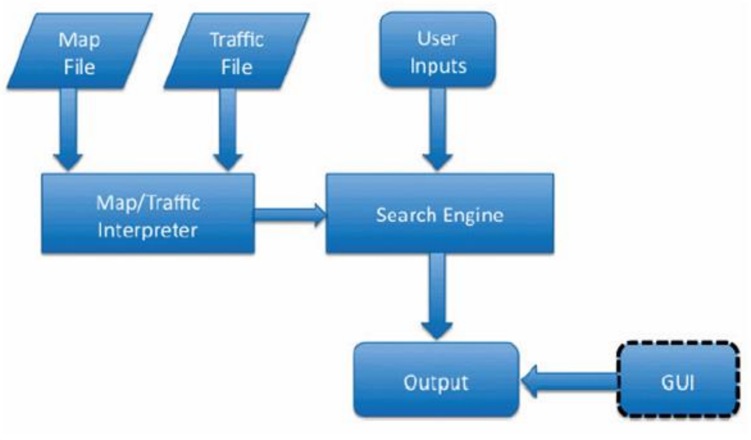


Figure 2.7 Traffic Consideration Flow Diagram

2.4.2 The Choose of Control Route

For a route search problem, there are two major options to evaluate the route, one is the total distance and the other is the total time cost. The system chose total distance to be an evaluation function in this project as like the Figure 2.8. There are two reasons that do not choose the total time cost to be the evaluation function. First, the time cost depends on the length values of the path and the speed that both of them could be dynamic. And second, if there is a traffic congestion, the duration of the congestion cannot be evaluated. Then it is implausible to use the time cost as the evaluation variable [24].

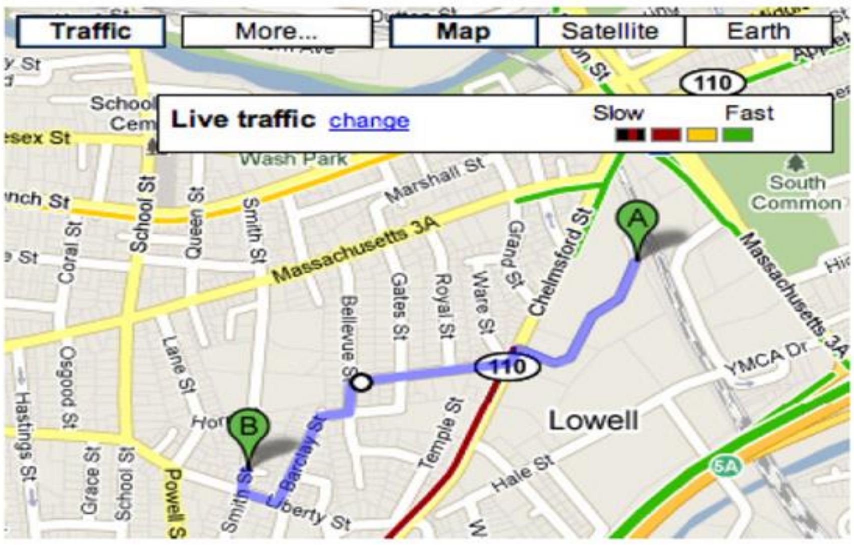


Figure 2.8 Optimal Route without Traffic Congestion

2.5 Recover Traffic Jam Through Intelligent Routing System

In related work, they further proposed an intelligent traffic application based on vehicle network called CHIMERA (through traffic classification mechanism and rerouting algorithm to avoid congestion). This system is able to detect when congestion is formed and intelligently balance traffic to distribute the density of the vehicle to avoid greater congestion in the near future. Therefore, CHIMERA uses the information stored by the vehicle to understand the traffic conditions of AoI. Using this data, this regularly categorizes each road by congestion level. By applying a rerouting algorithm that considers the degree of jams state, then can maintain regular traffic in AoI [37].

To prove the validity of the recommendations, they compared CHIMERA with some of the major formulas found in the related works through simulation. After compared to three well-known solutions: Through "routing", they show that the proposal is more effective in predicting congestion and properly redirecting vehicles to perform correct vehicle traffic load balancing. [37]

2.5.1 Steps of The CHIMERA System

In this process, CHIMERA and ITS consisting of three main processes will be described, in detail below :(A) A model for representing a city road and a working method of a vehicle data collecting program are proposed. (B) describes a technique for classifying the degree of congestion of each street. Finally, (C) proposes a rerouting algorithm that balances traffic density based on the degree of congestion on each road. [37]

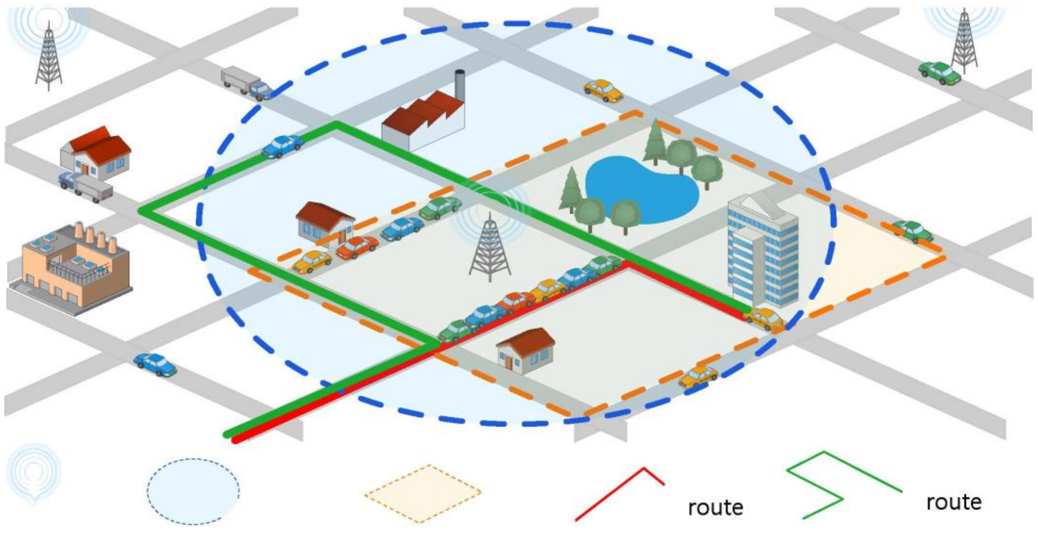


Figure 2.9 CHIMERA Entities: Congestion, Vehicle and Routes

The fuel consumption result is directly related to driving time, parking time and driving distance. In the part of comparing to OVMT, this increase is due to the longer average travel of these solutions. As shown in Figure 2.9, however, due to shorter travel distances and shorter traffic jam times, CHIMERA reduced fuel consumption by approximately 6% [37].

2.6 Summary

The chapter has five review collections: There are Comparative Studies, Traffic Analyzing and Optimal Routing Systems, Methods for Traffic Estimation, Traffic Considerations in Real-time People Transportation and Recover Traffic Jam Through an Intelligent Routing System in this chapter. In comparative studies it defied the difference of traffic concepts, API and works. This explained Android & Google Maps API, Google Map Traffic, and Checking Traffic Flows from GPS Signals. Methods are important in the thesis. The chapter introduced the many methods for traffic estimation: Fuzzy logic, Personal navigation assistants and map-matching, Topological network-based algorithms and K-D Tree Algorithm and so on.

These includes location-based services (LBS), moving object structures, synthetic moving objects, and traffic statement system reviews. It introduces the challenges of location-based services (LBS) and discusses location providers with a location upgrade strategy. It also describes a review of the relevant literature and previous work on updating the policy. Then, the moving object index structure is discussed in conjunction with related work.

In addition, the chapter describes Traffic Considerations in Real-time Prevent Traffic Congestions Through an Intelligent Transportation reviews. A complete intelligent transportation system needs to perform many tasks, such as maintaining and detecting the user's current location and conducting targeted searches in the correct area (Yangon City). Also, using GPS for the current location is not enough to determine the location, especially indoors. In traffic consideration review, it described Demonstration and Traffic Analysis for proving the using method to get the efficient result. These issues have been solved by using Google API and Steps of the CHIMERA in this system. The variety of references illustrated such as GPS, Location service, moving object index structures and location update policies and detect traffic statement tie together background theory, technique, and algorithm. And next chapter will discuss theorical methods of background of the thesis.

CHAPTER 3

BACKGROUND THEORY

This chapter presents the background theory of the traffic analysis and Avoiding Traffic Congestion and Optimal Routing (ATCOR) model uses for estimating the traffic congestion condition of the system. This chapter describes decision rule topological map matching algorithm and classification of the shortest path (SP) problem used in this system. It uses the modified A* algorithm, therefore, this section also provides the general shortest path process, the properties of A*, and explains how is going to be used to model traffic detecting and routing are described.

3.1 Map Matching

The Intelligent Transportation System (ITS) applications such as road traffic congestion and traffic detecting management used GPS (Global Positioning System) technology to collect location data for events, accidents or vehicles located in two or three dimensions. This information is combined with a Geographic Information System (GIS) to determine the path of events and events to identify points such as road signs or moving vehicles. The vehicle trajectory showed on the digital routing map is not at the top of the road described the real world.

3.1.1 Dynamic Traffic Routing

In these years, public transport and routing systems became more complex and crowded. Traffic complex is an important problem that affects upon travelers both economically and mentally. Finding the best route in a big city is also very difficult using maps. These issues include traffic accidents, traffic jams, air pollution, and environmental impact [42]. When it is working towards this achievement, traffic conditions can change over time and dynamic traffic routing is required.

Real-time, real information and updated data on traffic conditions can be collected through loop detectors, detection vehicles. But, the use of this information and data to provide efficient services, is still fall off behind. The aims of this research is to solve the problem of dynamic routing, which guides vehicles such as cars, motorcycle, bicycle through the urban routing pattern using the fastest route, getting into number of traffic conditions on the roads [42].

3.1.2 The Standard of GIS and Location Based Service

The Geographic Information System (GIS) represents a new ideal for information system organization and design. Its basic aspect is to use location service as the basic for building information systems. Transportation or routing is geographical in nature, so the application of GIS is related to transportation because of the spatially dispersed nature of routing-related data and the need for different types of networking level analysis, statistical analysis and implement to do [42]. Space GIS has technologies that can dramatically improve the efficiency and productivity of many simple transportation and path finding applications.

The influence of GIS technology on the development of road information systems is technical in nature. The basic concept is to completely lift the decision-making process in transport engineering. As a good example, you can better develop route directions and avoid congestion in the GIS environment. In this application, GIS is used as a powerful method for identifying and detecting city traffic jams and planning optimal routes based on the routes with the shortest time / shortest distance / lower cost, and graphical functions allow you to display several routes and the order in which they are built. This allows the user to understand the system flow back of the routing design [42].

Over the past decades, it has witnessed the rapid emergence of mobile information terminals with Internet access (smartphones, LCD, in-vehicle computers, etc.), mobile / integrated computing, spatial information technology using GIS and GPS. The result is a new generation for mobile services called as location-based services (LBS). It provides geographic information and geographic processing capabilities to mobile users over the Internet and wireless networks [42]. Location classification work related to location-based services was initiated by global industry initiatives such as the OpenGIS Consortium and the Location Interoperability Forum (LIF) formed by Motorola, Ericsson, and Nokia.

3.1.3 The Architecture of Routing and Navigation Service

Navigating guides can distinguish within the distribution and centralization route guides. First, the mobile client uses its on-board computer to obtain its own position on real-time to get traffic information provided via a CD-ROM static roadmap or wireless network. However, mobile networks are expensive, so these have limited bandwidth network, and have poor connection performance, delivering detailed traffic information to all mobile users is expensive. In addition, geoprocessing takes time, and mobile devices usually have limited memory and computing power. As a result, it may take some time to perform the calculations locally, and in some cases it may not be possible. Navigation services, on the other hand, are often used in critical and crime situations (eg, 199 Police emergency services) that require near up-to-minute query responses and lengthy route guidance information to facilitate decision making.

A centralization routing guide relies on the Traffic Management Center (TMC) to answer to route queries sent from mobile users. In this case, use a client-server architecture to reduce query response time. Instead of providing a complete data set using a centralized GIS server, it performs geoprocessing tasks and returns query results. This service can provide users with step-by-step navigation instructions on the best route to the desired destination through text or map screens. This can also warn drivers about future problems such as traffic congestions and accidents. To discover query results to mobile users within activated delay, you need an efficient algorithm to get the desired navigation information quickly. Therefore, it can accommodate many mobile clients.

3.1.4 Data Structure by KD Tree

In k-dimensional structure, Cartesian coordinates can be showed by k-dimensional vectors, and the vector components represent the positions of points along the i axis. It shows to be vector space, so all operations in vector space can be used with these Cartesian coordinates. This includes scalar addition and multiplication, dot product and norm [27].

$$\mathrm{U}.\mathrm{V} = \sum_{i=1}^{n} U_i V_i \qquad \text{Equation (3.1)}$$

The KD tree is very similar to a one-dimensional binary search tree and shares a portion of the same algorithm. At each level, the binary search tree divides a series of points around the points stored on the nodes. Similarly, the KD tree divides a set of vectors along a k-dimensional hyperplane orthogonal to the axis and passes through the vectors stored on the nodes. The dimension proved by the split axis dimension [38].

The optimal combination of split strategy and search algorithm seems to be a strategy that splits the most deviating dimensions with detailed research. When grouping multiple datasets, A * search can overcome the initial depth search, which is a pure guess [38]. Splitting the space along the larger dimension creates a good tree for the search, but seems not very effective for query optimization. The median along the dimension may not be the best place to split the points. Given some clustering patterns, it may be better to divide the space along the boundary of two dense clusters.

Henry Stern [38] explained that KD trees can be adapted to handle sparse high-dimensional arrays that are common in information retrieval systems. The A* search combines the term mapping of the document and produces good results from the lowest position of the structure. The KD tree can be used for clustering algorithms. The cluster centroid can be randomly selected from a series of points, and a simple "shift" around the KD tree can find all points within a certain distance of the centroid.

3.2 Types of the Shortest Path Case

Some researchers tend to group the types of problems for the shortest paths, in slightly different ways, but can usually distinguish between shorter paths calculated in one-on-one, one-on-one, or all-at-all. Given a graph, you may need to find the shortest path from a single initial node v to all other nodes in the graph. This is called the shortest path problem from a single source. Therefore, all the shortest paths from v to all other nodes form a short root tree that covers each node in the graph. Another problem is finding all the shortest paths between all the pairs of nodes in the graph. This is called the shortest route problem for all couples. One way to solve the shortest path problem for all couples is to solve the shortest single path source for all possible source nodes in the graph.

The Dijkstra algorithm is an efficient way to solve the optimal path problem of a single source with a positively weighted direct graph with real-valued link costs. Many of today's shortest path algorithms are based on Dijkstra's approach. There is also a relatively simple problem with a pair of shorter paths, and the shortest path between the first node and the destination node must be determined. In the worst case, such problems are difficult to solve as a single source [42].

3.2.1 Haversine Formula

The Haversine formula determines the distance of the great circle based on the length and latitude of the two points of the ball. There is important in navigation, it is a special case of a more general formula in a spherical trigonometric function, the sinusoidal law, which relates to the sides and angles of a spherical triangle [48].

Calculating the geographic distance on Earth, the Haversine can easily calculate the distance of the great circle. The system needs a point closer to the user's current location. After calculating the distance using Haversine, it releases the best route to the user's desired destination as the following Figure 3.1.

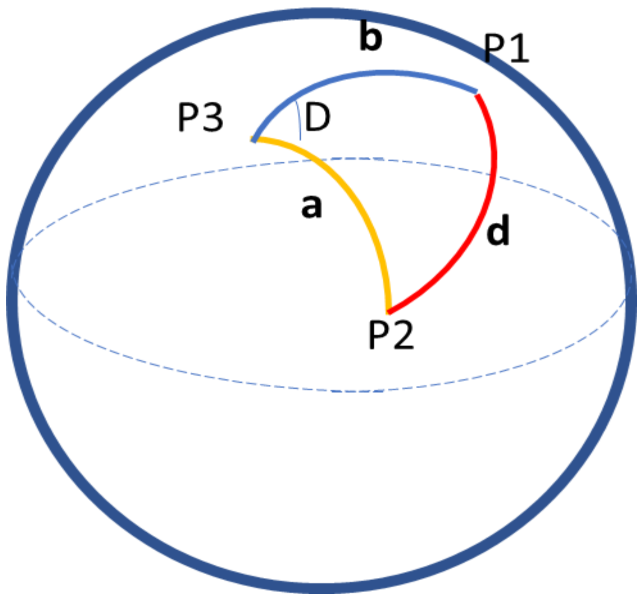


Figure 3.1 Distance by Haversine Formula

3.2.1.1 Law of Cosines

In fact, JavaScript uses 64-bit floating point numbers, which provide precision for 15 significant digits. According to my estimation, with this precision, the simple spherical law of cosine formula ($\cos c = \cos a \cos b + \sin a \sin b \cos C$) gives good results until a few meters of the surface of the Earth. For many purposes of geodesy (if not for astronomy), this makes the simple law of cosine a reasonable one-line alternative to the haversine formula. It can be selected according to the programming language, the processor, the encoding context, the available trigger functions, and so on. Also, for very small distances, an equal rectangle approximation can be appropriate [51].

Law of cosines:

$$d = \operatorname{acos}(\sin \varphi 1 \cdot \sin \varphi 2 + \cos \varphi 1 \cdot \cos \varphi 2 \cdot \cos \Delta\lambda) \cdot R \qquad \text{Equation (3.2)}$$

The Haversine formula uses the latitude and longitude measured along the surface to calculate the shortest distance between two points on the sphere. This is very important for navigation. Haversine can be expressed as a trigonometric function:

$$haversine(\theta) = sin^2(\frac{\theta}{2})$$

The haversine of the center angle is calculated by the following formula:

$$\left(\frac{d}{r}\right) = haversine\ (\emptyset_2 - \emptyset_1) + \cos(\emptyset_1)\cos(\emptyset_2) haversine\ (\lambda_2 - \lambda_1)$$

Where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km);

Solving d by applying the inverse haversine or by using the inverse sine function, it gets:

$$d = 2r\ sin^{-1}\left(\sqrt{sin^2\left(\frac{\emptyset_2 - \emptyset_1}{2}\right) + \cos(\emptyset_1)\cos(\emptyset_2)\, sin^2(\frac{\lambda_2 - \lambda_1}{2})}\right)$$

3.3 Traditional Shortest Path Procedures for Networks

Route finding can be applied to many types of networks, including roads, public services, water, Power, communications and computer networks, so the total number of algorithms developed over the years is very large. Only network types are involved. The marking algorithm is the most common and effective algorithm for solving SP problems. These algorithms use a label for each node that corresponds to the shortest temporary path length pk for that node. Algorithms are implemented to update these tags to find the shortest path [42]. The marking algorithm can be divided into two groups: a marker construction algorithm (LS) and a marker modification algorithm (LC).

At each iteration, the LS algorithm permanently establishes the node tag as the shortest valid path to the initial node, thereby adding the shortest path vector for the component at each step. The LC algorithm does not create tags permanently. At the same time, all components of the shortest path vector are obtained. Tags are set to an estimate of the shortest path from a particular node to each iteration. At the end of the algorithm, a start tag [42] is saved for each node. This represents the node before the shortest route to the current node. As a result, the last step of the algorithm determines only the set of routes P_k= {P_1, ..., P_k}. Backspace is used to build the shortest route to each node. Typical tag construction algorithms include Dijkstra and A* algorithms. The algorithm is an example of a tag correction algorithm.

3.3.1 Dijkstra's Algorithm

Dijkstra's algorithm was named after the inventor and influenced the calculation of route calculations. This can be done by starting from the primary node of the object, accessing the nodes on the network, and repeatedly investigating the nearest node that has not yet been investigated. Divide the chart into two groups to add subsequent nodes to the set of nodes to be examined. S is the node with the shortest path to the starting node, and S' is the node with the shortest path to the starting node. Unknown Initially, S contains all nodes. After the test, when the node moves from S' to S, the node set S will "grow". At each algorithm step, the next node to be added to S is determined by the priority queue. The queue contains the S nodes prioritized by distance tag, which is the current node up to the start node. The cost of the shortest route of node u at the top of the priority queue is examined, added to S, and its external links are relaxed. The estimated distance from node v is determined by the distance tag of u and the transmit link cost (u , V) is smaller than the distance tag of v. The algorithm returns and processes the next node at the top of the priority queue. The algorithm ends when the destination is reached or the priority queue is empty. Dijkstra's algorithm can solve the SP single-source problem by computing the shortest path tree from one source node to all other nodes. The pseudo-code of Dijkstra's algorithm is described below.

```
Dijkstra function (G, start)
(1) d [start] = 0
(2) S = ∅
(3) S '= V∈G
(4) S ≠≠ while
```

```
(5) do u = minimum (S ')
(6) S = S U {u}
(7) About each link (u, v) sent from u
(8) do if d [v]> d [u] + w (u, v) // relax (u, v)
(9) d [v] = d [u] + w (u, v)
(10) Previous [v] = u
```

3.3.2 A* Algorithm

Since this algorithm does not apply heuristics, the Dijkstra algorithm cannot be used to calculate the shortest route from a single starting node to a single destination. Scale equally in all directions and search for search areas that are too large and unnecessary before finding the target. The Dijkstra algorithm is a version of BFS that is guaranteed to find the best route, but is not widely applied due to the relatively high computational cost. This led to the development of heuristic research. In heuristic research, the A* algorithm is widely considered the most efficient method.

The A* algorithm is a heuristic variant of the Dijkstra algorithm that applies the principles of artificial intelligence. Similar to the Dijkstra algorithm, the search space is divided into two sets. S is a node whose shortest known path to the starting node is known and S 'is a node whose shortest unknown path to the starting node is unknown. Unlike Dijkstra's algorithm, it considers not only the distance between the destination node and the starting node, but also the distance between the destination node and the destination node [42].

In algorithm A*, g (n) is called the starting distance and represents the cost of the starting node route to any node n, and h (n) is estimated as the target distance representing the estimated distance. From node n to the destination. Since the path is not yet complete, this value is unknown and h (n) must be "guessed". This is where heuristics are applied. In general, a search algorithm is said to be acceptable if it is guaranteed that the shortest path from the starting node to the destination node is always found. If the heuristic used in the A* algorithm does not overestimate the cost or distance from the destination, this indicates that the A* algorithm is acceptable.

The heuristic is called an acceptable heuristic because it makes an A * search acceptable.

If heuristic estimation is specified as zero, this algorithm works in the same way as the Dijkstra algorithm. In many cases it is not practical to calculate, but the best heuristic is the smallest actual distance to the target. An example of a practical heuristic that can be tolerated is the linear distance from the investigated node to the target, which estimates how close the target is. The A * algorithm estimates two distances g (n) and h (n) in the search and classifies each node with an equation.

$$f(n) = g(n) + h(n) \qquad \text{Equation (3.3)}$$

This expand node n that always has the smallest f(n). Therefore, A* avoids considering addresses that are not ideal, and the search address can effectively reach the target. This can reduce the calculation time. Therefore, the A* algorithm is faster than the Dijkstra algorithm and finds the shortest path between a single pair of nodes. This algorithm is an example of the first best search.

3.3.3 Comparing Algorithms with Time Performance

Search algorithm efficiency is an important issue in route planning because it is related to the practicality and effectiveness of the search algorithm. For some time, the search algorithm it consumes cannot be applied to real applications. The system needs to perform a performance analysis of different types of algorithms. Performance and Complexity analysis involves two aspects: time and space. Because space and space savings are usually the result of increased processing time and vice versa, time and space algorithm requirements are often contradictory. However, advances in computer hardware have made it possible to provide enough memory in most computing environments, and the current main focus is on the time complexity of the algorithm.

There are two basic operations for calculating the shortest route. One is an addition calculation that provides the starting distance of the current node based on the weight of the previous node and the link between them. The other is a comparison operation that provides a short path to the starting node. And assume that the time costs for these two operations are equal. The complexity of time is measured by the frequency of operation most commonly used by previous algorithms. Looking at the pseudocode of the Dijkstra algorithm, the main loop from step 5 to step 10 requires more computation time. In step 5, the algorithm finds the node with the shortest starting distance. Need | C | first time comparison, |C | second -1 hour, etc. Therefore, the time complexity of the node search is |C | + (|C | −1) +...+1 = O (|C |2). In steps 8 through 10, the algorithm examines all links connected to the current node for addition and comparison operations. Examine all links on the network from the overall search view | E | Time. Therefore, the final time complexity of Dijkstra's algorithm is O (|C |2 + | E |) = O (|C |2).

The A* algorithm calculates only the shortest path between a single pair of nodes, so its temporal complexity is calculated differently. If the average grade of the network is indicated by d and the search depth (that is, the level crossed by the tree search until the target is found) is indicated by h, the temporal complexity of the A* algorithm is O (dh). For these two algorithms, comparison of time complexity between is shown in Table 3.1.

Table 3.1 Time Performance Comparison between Two Algorithms

	Dijkstra Algorithm	A* Algorithm
Time Complexity	O (|C |2)	$O(d^h)$

In previous section, it is recommended that the shortest route from your current location to a known destination is a typical navigation service query. Based on a comparison of previous time complexity, A * is an efficient algorithm that solves the SP problem. |C |. Therefore, the time complexity of Dijkstra's algorithm is much larger than A *. This is because it involves redundant calculations to solve the single-pair SP problem. Since they can be applied by other shorter path problems, they can be used in other scenarios discussed later in the system. A * can answer the first type of query proposed in the section, but it is a static approach and is not the best solution. In a dynamic environment, A * needs to recalculate the shortest path from zero (0) each time there is a value of change in traffic statements. In addition, this from the view, improvements are needed to adapt to the dynamic environment.

3.4 Virtual Machine Database and Cloud Computing

This system uses cloud computing as a back-end server to reduce processing time and improve performance. Cloud computing is as a new way of technology where dynamically scalable, reliable and often virtualized resources are served as services on the Internet. With cloud computing process, users can access programs, collect data as a storage and apply application as a development platform on the Internet through a variety of devices, including PCs, mobiles, laptops, smartphones and PADs, through the services provided by cloud computing providers [16]. Cloud computing technology offers the advantages of cost effective, high availability performance and ease of expansion and so on. It has five characteristics:

i. **On-demand self-service**: Users can configure cloud computing resources primarily through a web-based self-service portal (management console) without human intervention.

ii. **Extensive network access**: Online access to cloud computing resources to support heterogeneous client platforms such as mobile devices and workstations.

iii. **Resource Pool**: Serves multiple customers from the same physical resource and securely separates resources at a logical level.

iv. **Fast resiliency**: Resources are provided and released on request, and/or automatically based on triggers or parameters. This will ensure that your application has the required capacity at any time.

v. **Measurement services**: Check, measure and report (turnaround) resource usage in a transparent manner based on usage. In short, paid for.

Cloud computing also provides a range of services. These services are

i. **Software as a Service (SaaS)**: A software that runs on a computer owned and managed by a SaaS provider compared to software installed and managed on a user's computer. The software is accessible via the public Internet and is usually subscribed monthly or yearly.

ii. **Infrastructure as a Service (IaaS)**: refers to IT devices and resources as a service. This includes virtual machines (VM) with guaranteed processing power and bandwidth reserved for storage like database and take Internet access.

iii. **Platform as a Service (PaaS)**: Similar to IaaS, but also includes the operating systems (OS) and make services required for a particular application.

iv. **Data Storage as a Service (DaaS):** Provides storage for customers, developer including bandwidth requirements for archiving. And this accept data collecting as a cloud database.

In three types of cloud computing: public, private, and hybrid, it uses Google's public cloud (Google App Engine (GAE)).

i. **Public cloud**: In a public cloud (external cloud), an external provider of a third-party site dynamically provides processing resources over the Internet through a web application or web service. Public cloud providers typically provide access control mechanisms to their users. Public clouds are managed by third parties' applications, and methods from different customers may be mixed on servers, storage systems, and cloud networks [16].

ii. **Private cloud**: A private cloud (internal cloud) refers to cloud computing on a private network. The difference between a private cloud and a public cloud is that in a private cloud-based service, data managed within the organization, limit and privacy make changes in these, without network bandwidth limitations, security risks, and legal requirements to use public cloud services. A private cloud can be created and managed by your IT organization or cloud service provider [16].

iii. **Hybrid Cloud**: A hybrid cloud is an environment cloud that combines public and private cloud models. Hybrid Cloud introduces the comparison of determining how to deploy public and private cloud applications. These clouds are typically created by companies and management responsibilities are shared between the company and the public cloud provider. Hybrid clouds leverage services in public and private spaces. Hybrid clouds can provide the necessary services when companies need services that use public and private clouds.

3.4.1 Mobile Cloud Computing

Use the GPS sensor built into your phone to collect traffic data and display the results to it. As a result, the benefits of cloud computing enable users to use infrastructure, platforms and software with low cost and on-demand flexibility. Mobile cloud computing provides mobile users such as clients with data storage, collect data and information and give the processing services in the cloud without having to have powerful device configurations (such as CPU speed) because all resource-intensive applications and projects computing can be performed in the cloud.

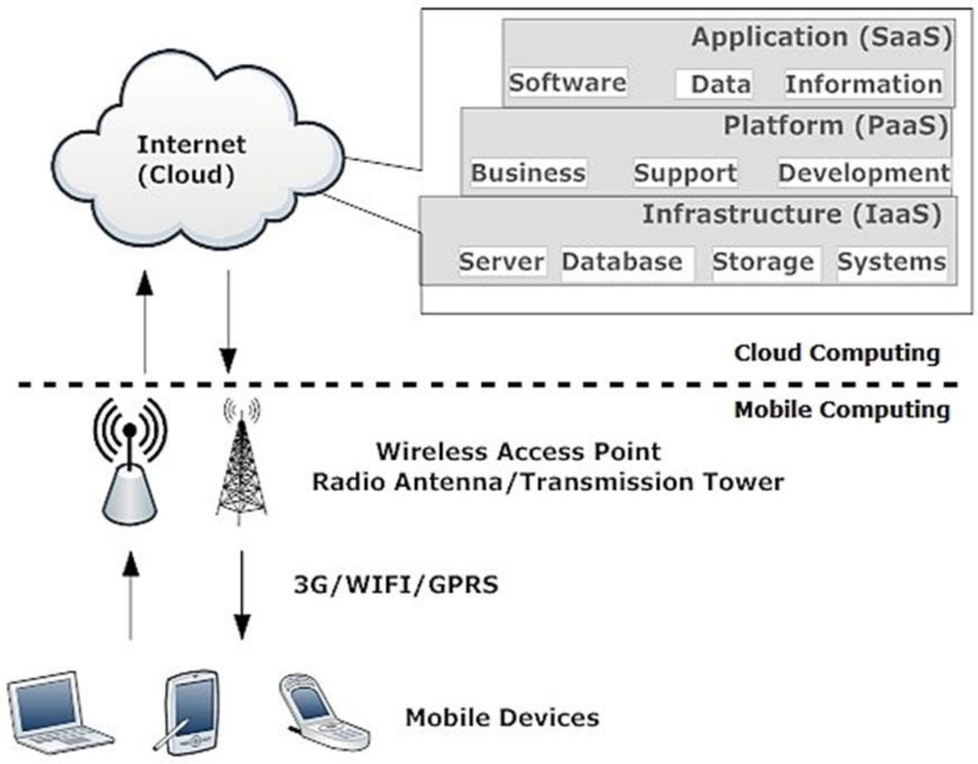


Figure 3.2 Architecture of Mobile Cloud Service

In a mobile cloud computing architecture, mobile devices are connected over mobile networks through base stations that establish on the cloud that is Internet and it controls connections and functional interfaces between the network and mobile devices. Requests and results information from mobile users are transmitted to a central processor connected to a server providing mobile network services.

3.5 Summary

Firstly, this chapter presents Map Matching included Dynamic Traffic Routing, The Role of GIS and Location Based Service and The Architecture of Navigation Service and the data structure by KD tree. The purpose of this study is to solve the problem of dynamic routing. It guides the car through the city's road network using the fastest route considering road traffic conditions. This explains that to show query results to mobile clients within an available time, efficient algorithms are needed to quickly retrieve the needed routing information. Thus, it can be accommodating large numbers of mobile users (client). The chapter explains the sphere of databases with moving objects. The procedures of the KD shaft and the spindle shaft, which are used to compare the proposed index shaft for performance evaluation, are explained and discussed. The index approaches, especially for the nearest neighbor techniques and two types of interval sizes, are also described.

Secondly, the Classification of the Shortest Path (SP) Problem was presented by this chapter. It explains algorithms of shortest path finding such as Classical Shortest Path Algorithms for Static Networks, Dijkstra’s Algorithm and A* algorithm. And then this included the relationship between cloud technologies and mobile devices with location update policies and then discuss the cloud database services. And then the chapter presents the detailed process of the structure-based methods in indexing technologies. Finally, this chapter also discusses Virtual Machine Database and Cloud Computing and Mobile Cloud Computing. Next chapter also presented the step by step procedure for the design and implementation of the proposed traffic avoiding system.

CHAPTER 4

DESIGN AND IMPLEMENTATION OF TRAFFIC AVOIDING SYSTEM

This chapter describes the step by step procedure for the design and implementation of the proposed traffic avoiding system. This chapter presents the process collects GPS traffic data and sends the detecting and routing results of desired source and destination to user by coloring the road segment on Google Map.

The demonstration of the program includes indexing the cellular location data set, and the proposed structure of the KD index tree is first presented in this chapter. The detailed process is discussed briefly, starting with the Haversine formula about finding the closest point to the current location in a range. The demonstration program provides a clear explanation of the virtual machine cloud services for the database structure and detail.

This implementation includes two main parts: the client-side program and the server-side implementation. As for the customer, he first defines two types of client applications: obtaining the location and updating the location through the GPS tracking algorithm. It then sends the current location to the application server with the Google Map Google Map API and the server that supports the dataset cloud database. On the server side, there are collections of registered mobile locations with identification number, latitudes and lengths, and point names that index user locations in the service range through the proposed indexed KD tree, which are in the desired searching area with the routing algorithm. All mobile users in the distance range will be requested to announce their location and traffic. All stages of system services are implemented by the client-server architecture, which is accepted as an abstraction layer with flexibility and free connectivity. As for requirements throughout this work, the following eight objectives are pursued:

i. To get the user's current location from the mobile provider called Google API
ii. To sign up for Google and Cloud Tech, which generates a unique token code on your mobile

iii. To communicate between a server and a client, not only by sending the token ID, but also the latitude and longitude of the current location and traffic instructions.

iv. Troubleshoot complex data from client and server

v. Support for the cloud database with a relevant index tree structure

vi. To get points from the locations closest to your mobile range

vii. To perform the comparison with another index structure.

viii. To complete works with an effective interpretation and an optimal route of evaluation results.

4.1 Process of Traffic Avoiding and Routing System

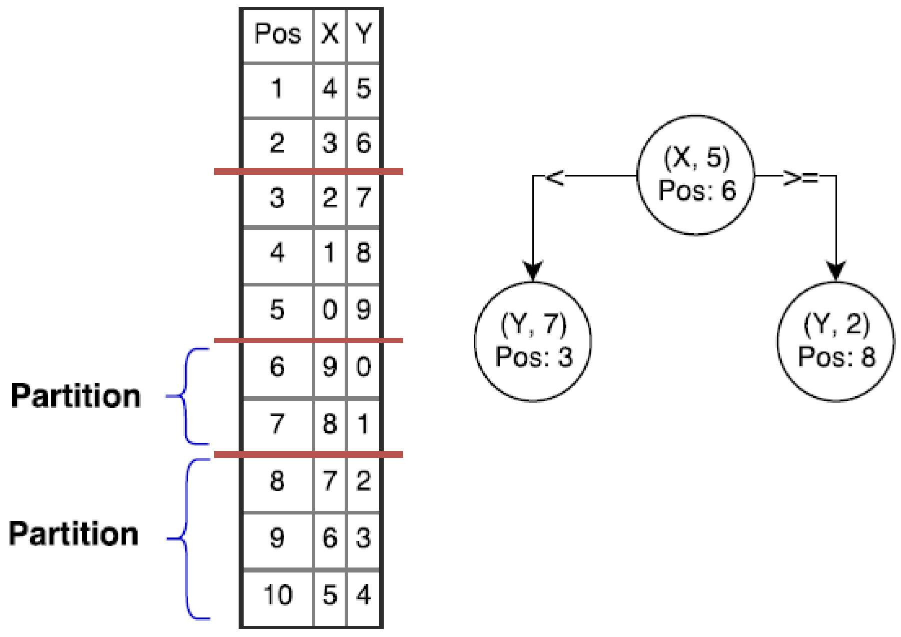
The system displays the results of detected traffic congestion conditions for the source and destination desired by the user. The system has two aspects: Client and server. Customers First, create road segment data using latitude, longitude and road conditions. Then map these segments to the GPS data collected using the map matching algorithm. After understanding the GPS data for each segment and current location, calculate historical traffic data based on this data and store it in the cloud [48]. When users search for the required traffic conditions for the desired source and destination, it collects data from such GPS sources and destinations from users in real time. Then, combine the GPS data with the road segment in real time and use the map matching algorithm to find the location of the vehicle. Therefore, this historical segment data and real-time traffic data are used for these segments to get the traffic congestion of the segments that users need to use the Google Map Traffic Layer. After obtaining the traffic possibilities of the source and destination that the user expects, it shows the traffic congestion by red coloring on the road, heavy colors in blue and green in normal conditions on Google Maps.

4.1.1 KD Tree Indexing Algorithm

The K-d tree is a useful data structure for different applications, such as searches involving multi-dimensional search keywords (eg, interval search and close-range search). The K-d tree is a special case of partitioning binary space trees. The following is the process of creating a KD tree algorithm.

```
function KDtree (list of points PList, integer depth)
{ // Mark axis based on depth because axis cycles through all values
    var integer axis: = depth mod k;
     Mark median by axis from PList;
    // Creating node and subtrees
    var tree_node node;
    node.location :=  to median;
    node.leftChild := to KDtree(points in PList that before median, depth+1);
    node.rightChild := to KDtree(points in PList that after median, depth+1);
    return node;
}
```

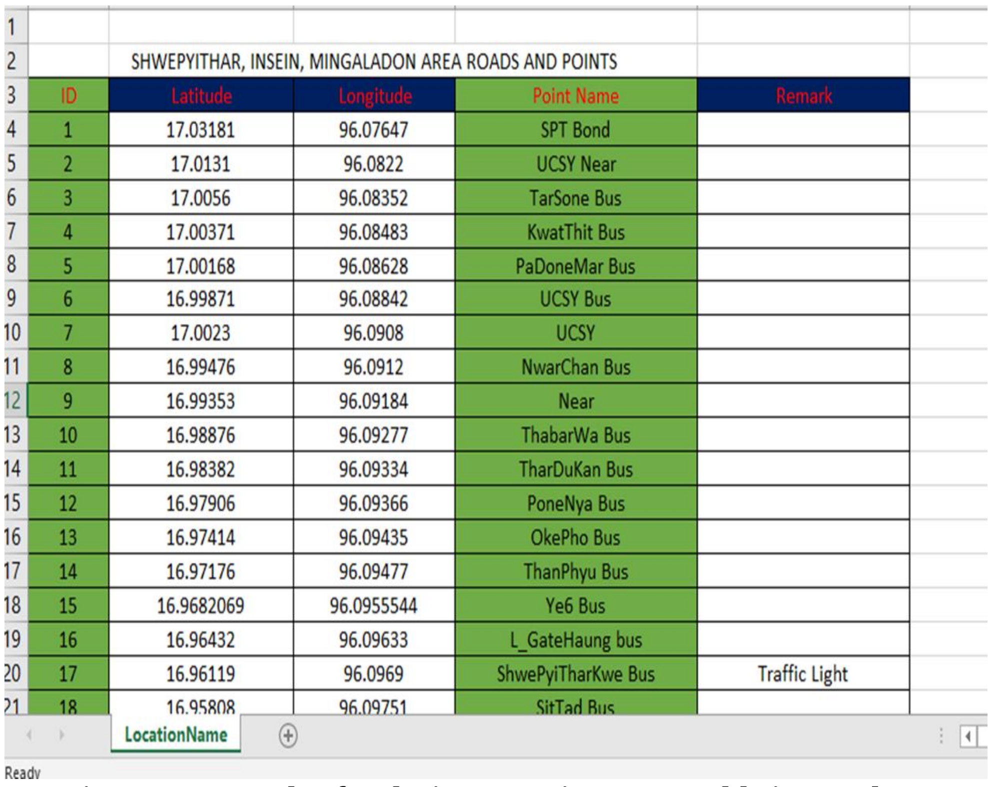
KD-Tree is a summary of binary search trees to multidimensional data. Each node of the KD tree contains a key, a distinct column, and up to two indicators for its child nodes. Traditional methods use the median of each column to divide the data level. The three elements of each level are concentrated on a specific dimension around the robin selection. Figure 4.1 depicts a KD tree indicating the X and Y size. The root indexes the X dimension in the value of 5. The next level determines the next Y dimension in the median of the new segment defined by X. X when $X < 5$, otherwise 2. The aquarium with intervals $X > 5$ and $2 < Y < 10$ is required to pass through the tree until the node (Y, 2) is reached, and the column partition [23] is scanned from position 8 to the end as shown in Figure 4.1.



Pos	X	Y
1	4	5
2	3	6
3	2	7
4	1	8
5	0	9
6	9	0
7	8	1
8	7	2
9	6	3
10	5	4

Figure 4.1 Building KD Tree Indexing

The following Figure 4.2 describes the data table in database that storing and indexing location data. These are location ID, Latitude, Longitude and point names.

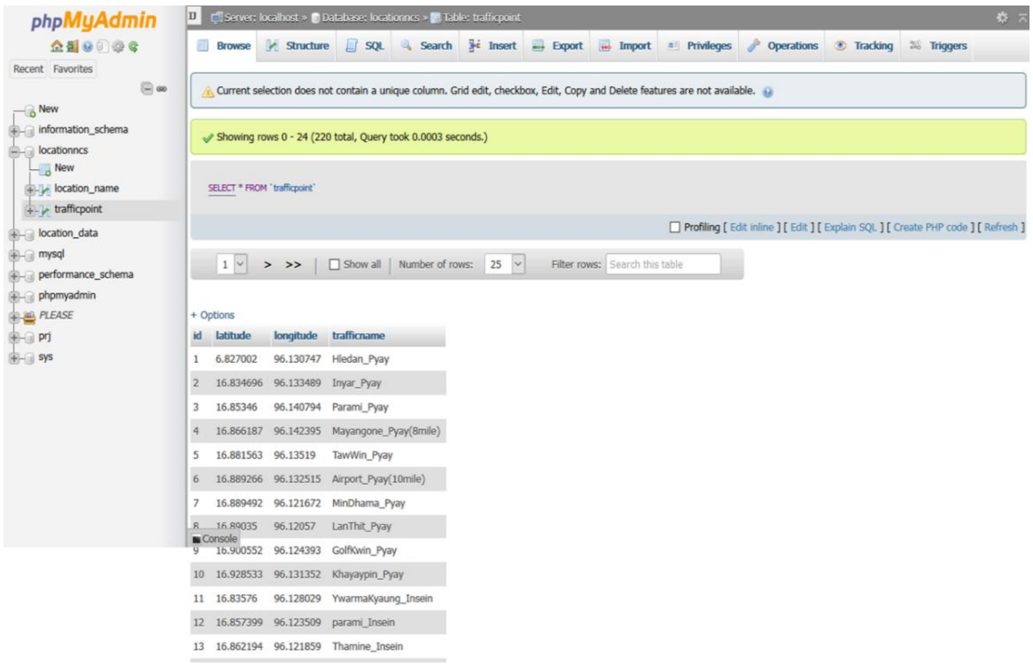


SHWEPYITHAR, INSEIN, MINGALADON AREA ROADS AND POINTS

ID	Latitude	Longitude	Point Name	Remark
1	17.03181	96.07647	SPT Bond	
2	17.0131	96.0822	UCSY Near	
3	17.0056	96.08352	TarSone Bus	
4	17.00371	96.08483	KwatThit Bus	
5	17.00168	96.08628	PaDoneMar Bus	
6	16.99871	96.08842	UCSY Bus	
7	17.0023	96.0908	UCSY	
8	16.99476	96.0912	NwarChan Bus	
9	16.99353	96.09184	Near	
10	16.98876	96.09277	ThabarWa Bus	
11	16.98382	96.09334	TharDuKan Bus	
12	16.97906	96.09366	PoneNya Bus	
13	16.97414	96.09435	OkePho Bus	
14	16.97176	96.09477	ThanPhyu Bus	
15	16.9682069	96.0955544	Ye6 Bus	
16	16.96432	96.09633	L_GateHaung bus	
17	16.96119	96.0969	ShwePyiTharKwe Bus	Traffic Light
18	16.95808	96.09751	SitTad Bus	

Figure 4.2 Sample of Indexing Location Data Table in Database

The system used Phpmyadmin that using for data managing and storing. These are location ID and point names and Traffic_Light data.



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 24 (220 total, Query took 0.0003 seconds.)

```
SELECT * FROM `trafficpoint`
```

id	latitude	longitude	trafficname
1	6.827002	96.130747	Hledan_Pyay
2	16.834696	96.133489	Inyar_Pyay
3	16.85346	96.140794	Parami_Pyay
4	16.866187	96.142395	Mayangone_Pyay(8mile)
5	16.881563	96.13519	TawWin_Pyay
6	16.889266	96.132515	Airport_Pyay(10mile)
7	16.889492	96.121672	MinDhama_Pyay
8	16.89035	96.12057	LanThit_Pyay
9	16.900552	96.124393	GolfKwin_Pyay
10	16.928533	96.131352	Khayaypin_Pyay
11	16.83576	96.128029	YwarmaKyaung_Insein
12	16.857399	96.123509	parami_Insein
13	16.862194	96.121859	Thamine_Insein

Figure 4.3 Sample of Indexing Traffic-Point Data Table in Database

4.1.2 Implementation Haversine Algorithm

This is calculating the geographic distance on Earth. This can easily calculate the distance of the great circle (the shortest distance between two points on the sphere).

The system needs a point closer to the user's current location. After calculating the distance using Haversine, it releases the best route to the user's desired destination. You can calculate the center angle Haversine between two points, where r is the radius of the earth, d is the distance between two points, λ_1, λ_2 is the latitude of two points, $\emptyset_1$, $\emptyset_2$ is the longitude of two points, for example:

$$haversin\left(\frac{d}{r}\right) = haversine\ (\emptyset_2 - \emptyset_1) + \cos(\emptyset_1)\cos(\emptyset_2) haversine\ (\lambda_2 - \lambda_1)$$

Equation (4.1)

Where φ is latitude, λ is longitude, and R is the radius (average radius = 6.371 km);

By applying reverse or using a reverse sine function to resolve d, it gets:

$$d = 2r\ sin^{-1}\left(\sqrt{sin^2\left(\frac{\emptyset_2 - \emptyset_1}{2}\right) + \cos(\emptyset_1)\cos(\emptyset_2)\ sin^2(\frac{\lambda_2 - \lambda_1}{2})}\right)$$ Equation (4.2)

Proving by Substitution with Latitudes and Longitudes (degree values) in Haversine Equation

d=2* 6371 sin $^{-1\,*}$

=

(

$\sqrt{sin^2(16.99476-17.0023)/2)+\cos(17.0023)\cos(16.99476)\ sin^2\,(96.0912-96.0908)/2)}$

)

=

(

$\sqrt{(4.329496919*10-9)+(0.9562930186)\ *(0.9563314909)\ *(1.218469674*10-11)}$

)

= $(\sqrt{(4.329496919*10-9)+(1.114330887*10-11)}\ )$

= $(\sqrt{4.340640288*10-9}\ )$ = 6.588353533*10^{-5}

= 2* 6371 sin^{-1} (6.588353533*10^{-5}) = 2*6371* 3.774848517*10^{-3}

= 0.839 km //(Distance Value)

4.1.3 Algorithm for Haversine Formula

The Haversine formula is used for calculating Distance between two points and implementing the system in through Java code:

```
public class DistanceCalculator {
     private double Radius;
     // R = earth's radius (mean radius = 6,371km)
   // Constructor
   DistanceCalculator(double R) {
      Radius = R;
   }
     public double CalculationByDistance(GeoPoint StartP, GeoPoint EndP) {
      double lat1 = StartP.getLatitudeE6()/1E6;
      double lat2 = EndP.getLatitudeE6()/1E6;
      double lon1 = StartP.getLongitudeE6()/1E6;
      double lon2 = EndP.getLongitudeE6()/1E6;
      double dLat = Math.toRadians(lat2-lat1);
```

```
        double dLon = Math.toRadians(lon2-lon1);
        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));
        return Radius * c;
    }
}
```

4.2 Processes of Client Application and Server

Part (A): Client side

Process (i). A client application that acquires GPS data at a user's location via a GPS service.

Process (ii). A client application developed for the Android platform.

Part (B): Server side

Process (i). A server that proposed cloud server that application and storage server which data acquired by a client application.

Process (ii). A module includes in the server side that is used to track this traffic information on the map.

4.2.1 Prerequisites for Using of Client Application and Server

The following documents are required for client application and server.

i. Android Studio 3.5.1 or later (to create better android application)

ii. The device that has Google Play Store app (to get the application)

iii. Android version 5.0 or higher (to compatible for application)

iv. Google APIs configurations (to take the data of Google Map traffic)

v. The Android SDK Manager with Google Repository (to create the relevant application)

vii. Tomcat v9.0 Server at localhost (to test in local before public cloud)

viii. XAMP Control Panel version 3.3.3 or later (to test in local)

4.2.2 Client-Side Application

The user interface design application is very simple and user friendly. The main navigation content of the client application is a list, showing the current location, detecting the best traffic and routing, showing the traffic statement and user guide. The location button is a collection of GPS location data sent by the application server.

Both the update position of the related historical data for the update time and the position status of the current position, which are confirmed by the Haversine formula, are stored in the position log. The location is provided by the Google API and helps you quickly and accurately locate your phone by switching between active location providers (such as GPS, network providers, cellular networks, etc.).

When the application server sends the required data to the registered mobile, location data updates and traffic status are sent to the mobile and saved to the application. In this application, the data are available and supported to see the traffic statement and optimal route with distance(km) by English language. Moreover, user can get the nearest points from their current location and avoiding the Traffic-lights points by seeing the result.

4.2.3 Server-Side Implementation

It is used to get important concepts of auto-configuration, initial dependencies and command line interpreter. In this system, most processes are dynamic, such as obtaining the refresh location of mobile users, searching a continuous range of queries, a dynamic index tree of mobile users in range, and sending the results to the target client application. Therefore, the Spring scheduler is the main controller of the system.

On the server side, deploy a multithreaded listener. The server continues to listen for location reports created by devices or devices that request the current map view for a particular region. If the device on a particular road reports its location, get the coordinates and speed. Then make sure the timer has been replaced. This indicates that you are leaving a particular road and that you can no longer report the condition of that road. In that case, the system implements the update data in cloud database by KD tree algorithm and Haversine formula with historical data to get nearest point from current location on the application of server. Otherwise, just take the report as a normal entry and update the route speed cost accordingly. Then listen again for additional requests. When a device requests a real-time view of traffic on a particular road, it receives the request and looks for information related to that edge. The system implements a routing pattern algorithm and sends data in a format that can be formatted. Interpreted and implemented by the client application system and draws traffic segments on the map [1].

The collection of information for mobile users is stored in the database. Once the client application is installed and its location is available, the token ID and its current location are received from the application server using this location provider and special system called Google API. The current location is updated as soon as the server receives user locations in the mobile application. Locations for all registered mobile locations are available and connected to Google Map in the mobile user application.

The process of sending application server data, identification number, traffic status, central latitude and longitude location, the service radius with kilometers is provided in the application. In addition, the system implemented the workaround using a Distance Matrix Web API service or a Directions API service. If you specify a departure time in the request, the response contains the duration and duration_in_traffic fields. Therefore, it can find out if duration_in_traffic is much longer than the duration and decide if there is a jam somewhere along this path through Google Map in the server application.

4.3 Steps of The System flow of ATCOR System (Avoiding Traffic Congestions and Optimal Routing)

This general description is the flow chart of Figure 4.4. The client application uses a GPS receiver to capture the user's location. Designed for the Android platform. The server processes the captured data and analyzes the traffic routing in the cloud database. This method performs some steps.

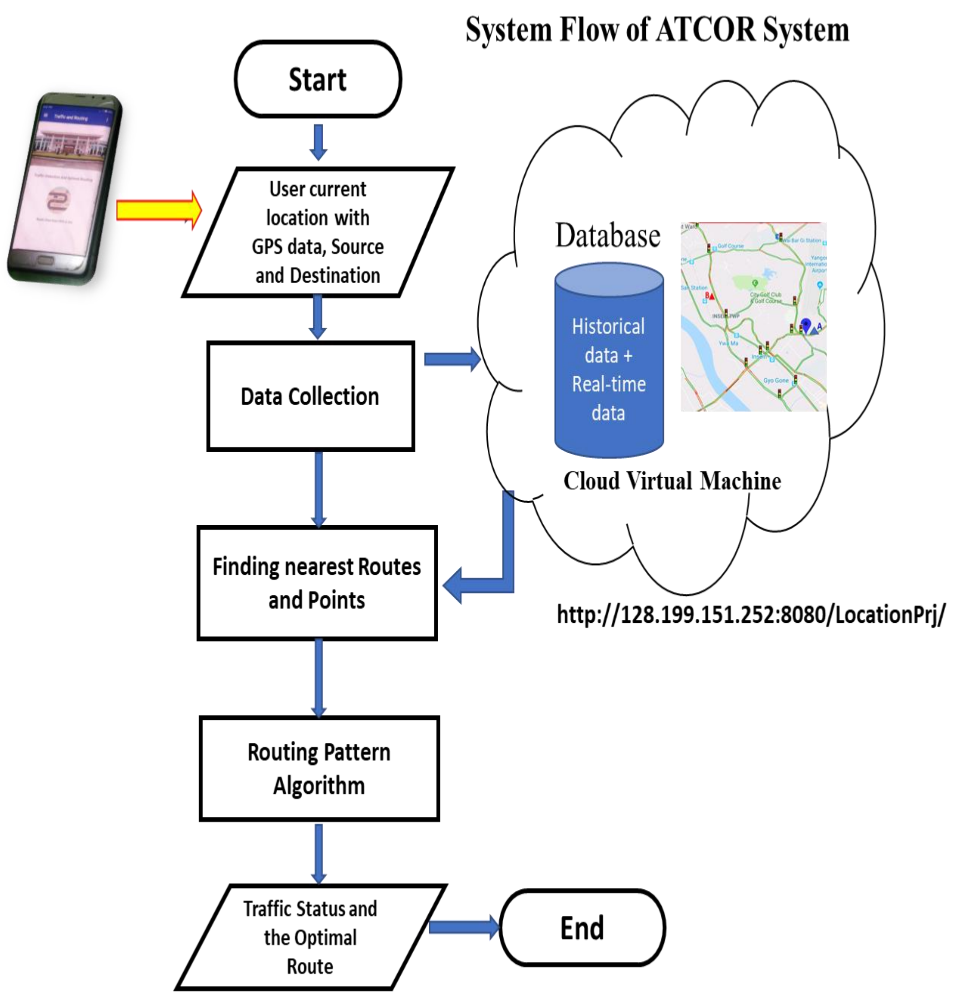
Step (1): GPS is collected from the transportation network by the customer's GPS receiver (mobile GPS tracking unit) to capture the user's current location using longitude and latitude.

Step (2): Update the location data in the cloud database. And save GPS location data and historical GPS and route traffic data as an index file by KD tree of cloud database (VM).

Step (3): Extract latitude and longitude, speed, distance, time, direction and get the closest point in Haversine Formula.

Step (4): Detect the maximum amount of traffic jams through the Google traffic layer using the routing pattern algorithm

Step (5): Manipulate traffic analysis and optimal route using routing pattern algorithm (modified algorithm-A *). The end user then gets the avoiding traffic route results via mobile as a client.



System Flow of ATCOR System
Start
User current location with GPS data, Source and Destination
Data Collection
Finding nearest Routes and Points
Routing Pattern Algorithm
Traffic Status and the Optimal Route
End
Database
Historical data + Real-time data
Cloud Virtual Machine
http://128.199.151.252:8080/LocationPrj/

Figure 4.4 Flow Diagram of ATCOR System

4.4 Pre-Processing

Preprocessing is the process of establishing road segments data and calculating historical data from pre collected GPS trajectories data from mobile phone equipped vehicles to use in checking up to date traffic congestion statements for users' needed source and destination.

4.4.1 Research Area and Data Collection

The research area is Yangon City. Now, there are seven townships where currently research areas in this approach: There are Collected locations data and Traffic-lights in Yangon city (33 townships). As Figure 4.5, these GPS data and historical traffic data are stored and extracted in the Cloud database (VM).

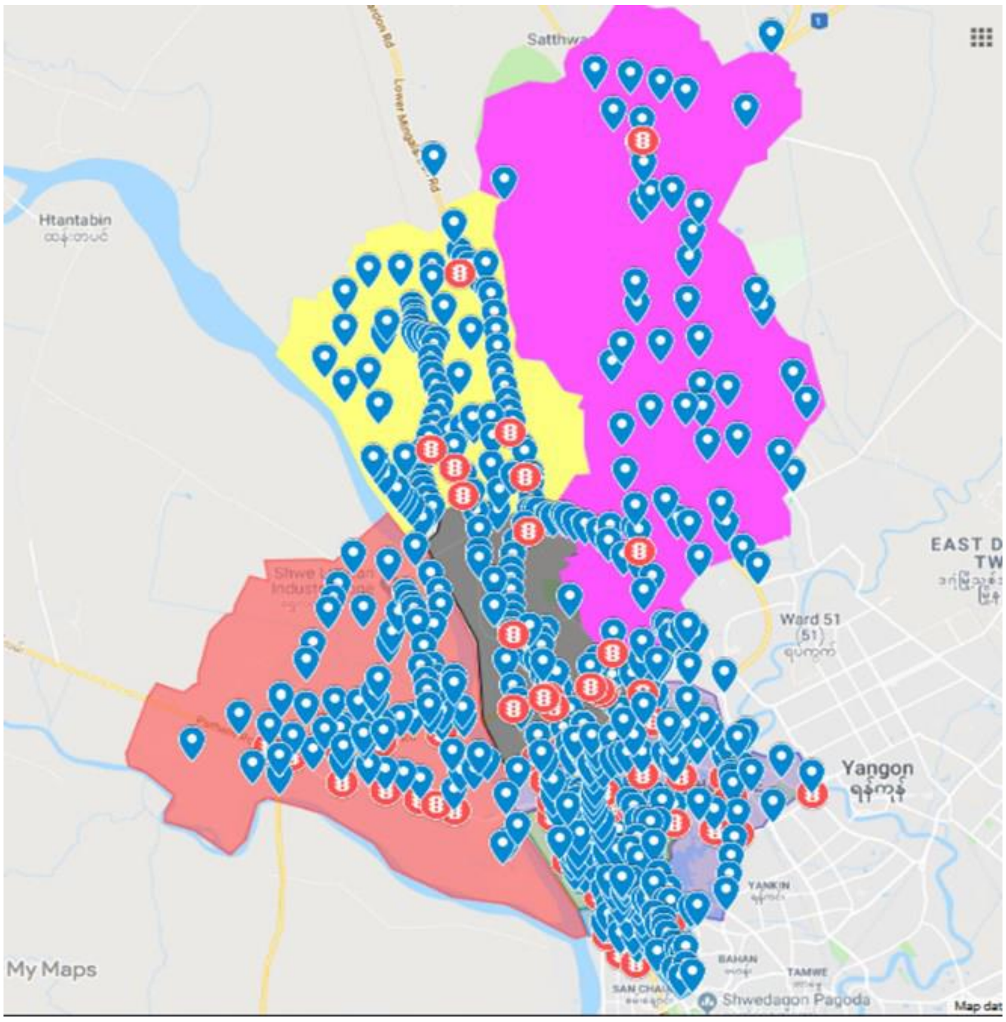
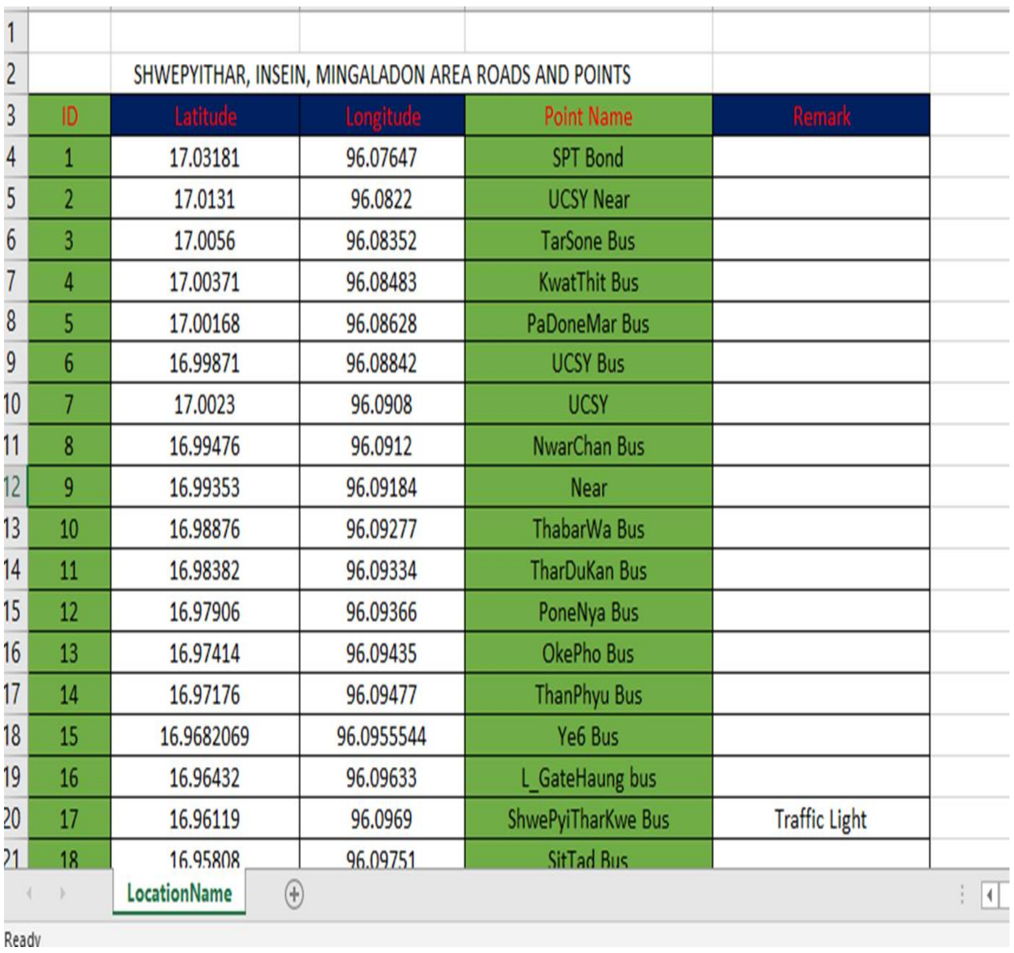


Figure 4.5 GPS Locations and Traffic-Lights in Yangon

By the above Figure, the system collected and marked the location data on the customized map. These points are bus stops, traffic points, intersections points and stations. In each point includes their data attributes such as ID, Latitude, Longitude, Point name and Remark. These points collected data file is the facts of these points and their names as the Figure 4.6. It includes Source ID, Latitude, Longitude and point names.



SHWEPYITHAR, INSEIN, MINGALADON AREA ROADS AND POINTS

ID	Latitude	Longitude	Point Name	Remark
1	17.03181	96.07647	SPT Bond	
2	17.0131	96.0822	UCSY Near	
3	17.0056	96.08352	TarSone Bus	
4	17.00371	96.08483	KwatThit Bus	
5	17.00168	96.08628	PaDoneMar Bus	
6	16.99871	96.08842	UCSY Bus	
7	17.0023	96.0908	UCSY	
8	16.99476	96.0912	NwarChan Bus	
9	16.99353	96.09184	Near	
10	16.98876	96.09277	ThabarWa Bus	
11	16.98382	96.09334	TharDuKan Bus	
12	16.97906	96.09366	PoneNya Bus	
13	16.97414	96.09435	OkePho Bus	
14	16.97176	96.09477	ThanPhyu Bus	
15	16.9682069	96.0955544	Ye6 Bus	
16	16.96432	96.09633	L_GateHaung bus	
17	16.96119	96.0969	ShwePyiTharKwe Bus	Traffic Light
18	16.95808	96.09751	SitTad Bus	

Figure 4.6 GPS Data of Location Points and Point Name

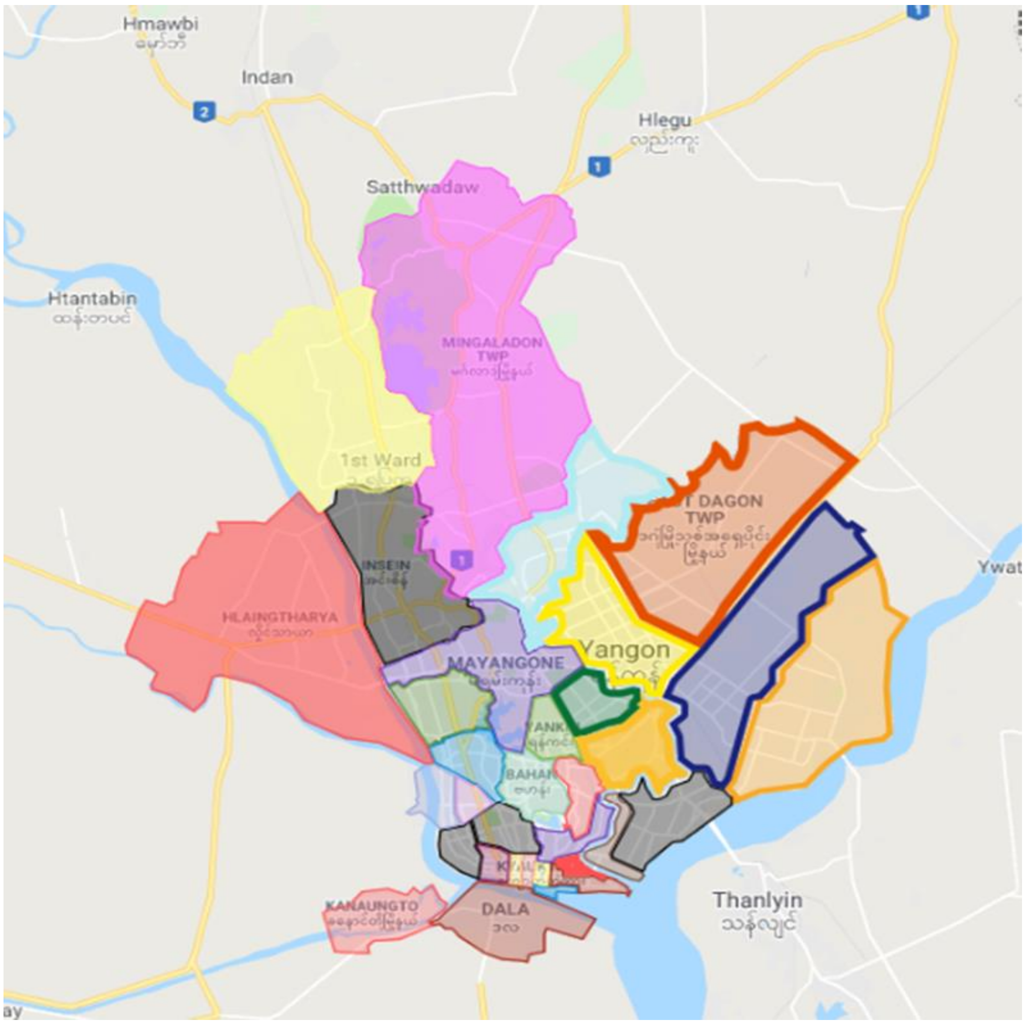
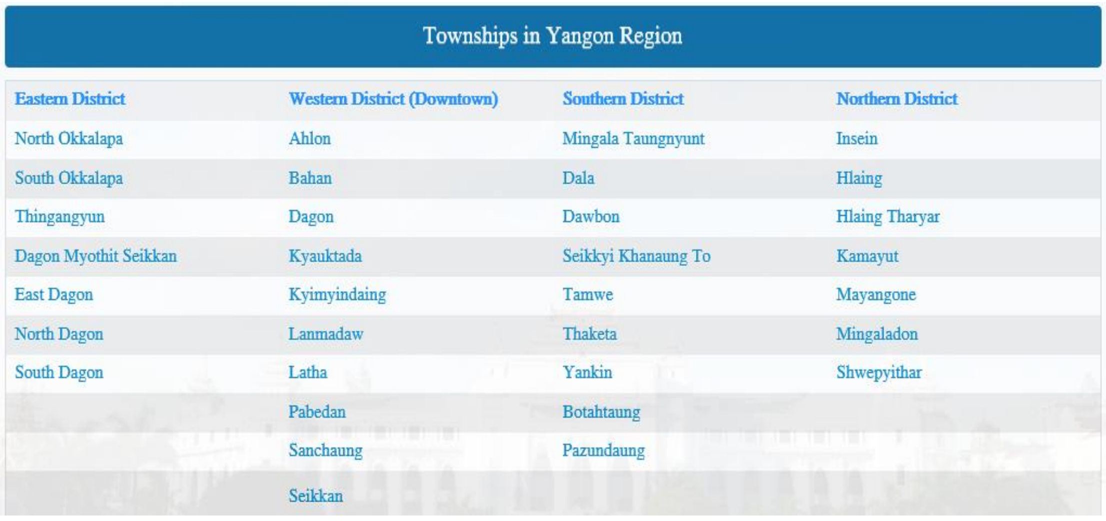


Figure 4.7 Research Area in Yangon Townships Map

Figure 4.7 shows the areas of townships in Yangon that the survey areas for implementing of the system. And Figure 4.8 is the list of townships in Yangon.



Townships in Yangon Region

Eastern District	Western District (Downtown)	Southern District	Northern District
North Okkalapa	Ahlon	Mingala Taungnyunt	Insein
South Okkalapa	Bahan	Dala	Hlaing
Thingangyun	Dagon	Dawbon	Hlaing Tharyar
Dagon Myothit Seikkan	Kyauktada	Seikkyi Khanaung To	Kamayut
East Dagon	Kyimyindaing	Tamwe	Mayangone
North Dagon	Lanmadaw	Thaketa	Mingaladon
South Dagon	Latha	Yankin	Shwepyithar
	Pabedan	Botahtaung	
	Sanchaung	Pazundaung	
	Seikkan		

Figure 4.8 List of Townships in Yangon

Townships in Yangon Region are Eastern District(North Okkalapa, South Okkalapa, Thingangyun, Dagon Myothit Seikkan, East Dagon, North Dagon, South Dagon), Western District (Downtown) (Ahlon, Bahan, Dagon, Kkyauktada, Kyimyindaing, Lanmadaw, Latha, Pabedan, Sanchaung, Seikkan), Southern District (Mingala Taungnyunt, Dala, Dawbon, Seikkyi Khanaung To, Tamwe, Thaketa, Yankin, Botahtaung, Pazundaung), and Northern District (Insein, Hlaing, Hlaing Tharyar, Kamayut, Mayangone, Mingaladon, Shewpyithar). See as Figure 4.7 and Figure 4.8.

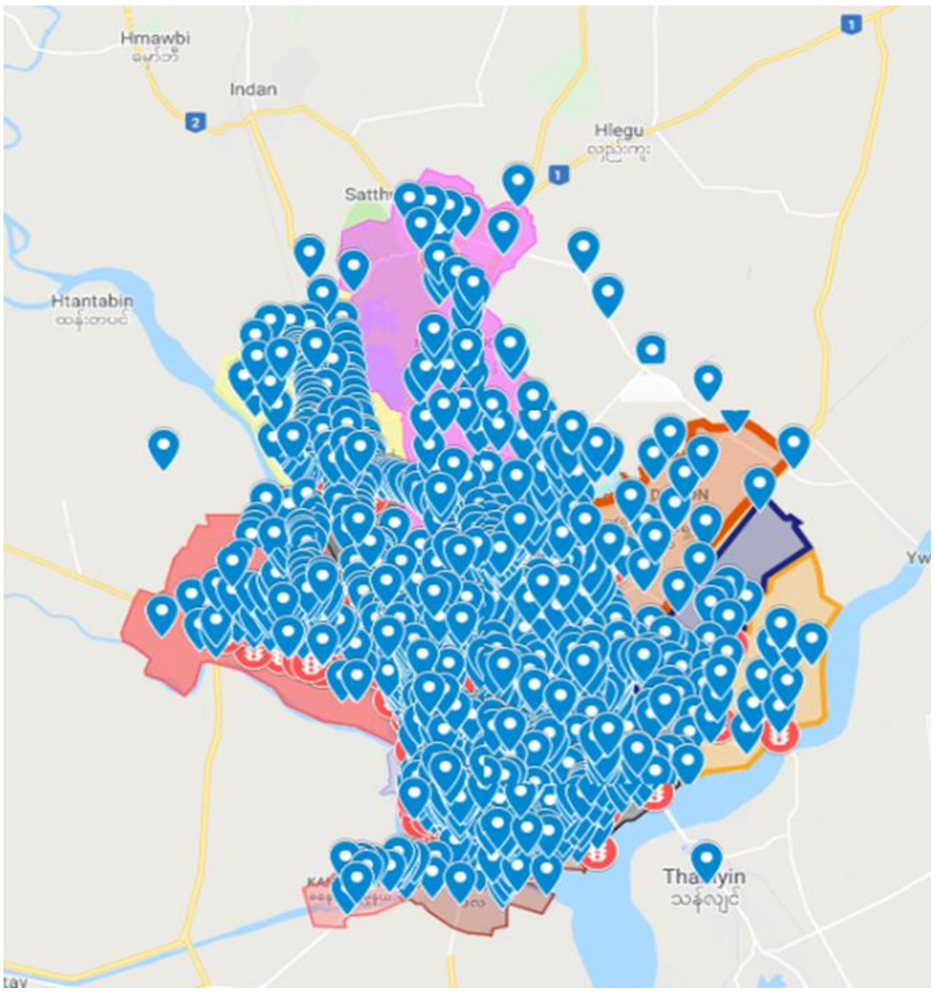


Figure 4.9 Historical GPS Location Points in Yangon Area

There are Collected locations data and Traffic-lights in Yangon Area. These are over 2000 locations points and 220 Traffic-Lights and more node points as the above Figure 4.9.

4.4.2 Historical Traffic Data

This is The List of Total Traffic-Lights in Yangon City. The number of traffic-light in the whole of Yangon is 220 points. There are many points in the whole of country. But this research area is Yangon city and Traffic-Lights points are collected and analyzed in just the Yangon City. There are 220 Traffic-lights as shown in the list YCDC.

Table 4.1 Locations Points of Traffic-Lights in Yangon

No.	District	Quantity			Remark
		Light-Point	Stand	LED	
1	Eastern	46	173	92	
2	Western	56	207	143	
3	Southern	57	199	144	
4	Northern	61	223	122	
TOTAL		220	802	501	

And then these Traffic-light data are captured on the Map of the system. There are 220 points of road Traffic-lights collected in Yangon as the above Table 4.1. In these 220 of Points and 802 of Stands and 501 of LEDs.

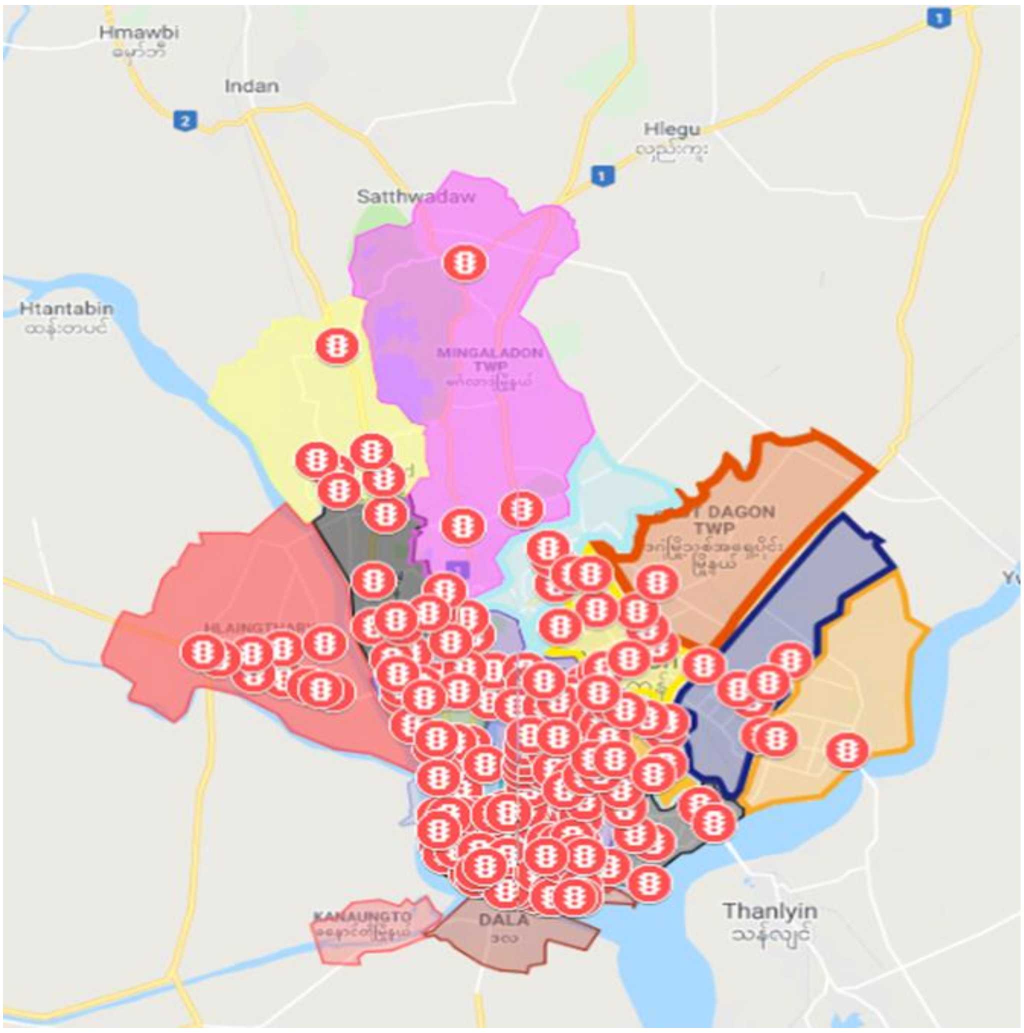
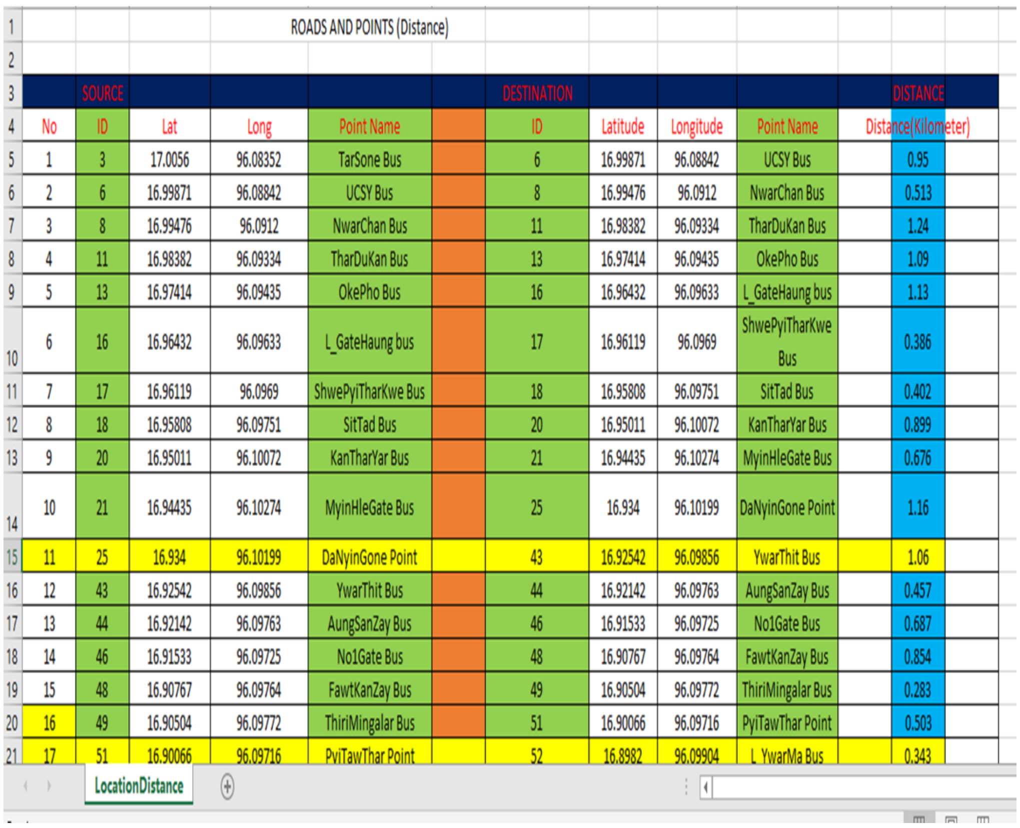


Figure 4.10 Locations of Traffic-Lights in Yangon City

By the above Figure 4.10, the location data is collected and marked on the customized map. These points are bus stops, traffic points, intersections points and stations. In each point includes their data attributes such as ID, Latitude, Longitude, Point name and Remark. These points collected data file is the facts of these points and their distances as the Figure 4.11. It includes Source ID, Destination ID and Distance. It defined the distance unit between each point by km (Kilometer).



ROADS AND POINTS (Distance)

SOURCE						DESTINATION				DISTANCE
No	ID	Lat	Long	Point Name		ID	Latitude	Longitude	Point Name	Distance(Kilometer)
1	3	17.0056	96.08352	TarSone Bus		6	16.99871	96.08842	UCSY Bus	0.95
2	6	16.99871	96.08842	UCSY Bus		8	16.99476	96.0912	NwarChan Bus	0.513
3	8	16.99476	96.0912	NwarChan Bus		11	16.98382	96.09334	TharDuKan Bus	1.24
4	11	16.98382	96.09334	TharDuKan Bus		13	16.97414	96.09435	OkePho Bus	1.09
5	13	16.97414	96.09435	OkePho Bus		16	16.96432	96.09633	L_GateHaung bus	1.13
6	16	16.96432	96.09633	L_GateHaung bus		17	16.96119	96.0969	ShwePyiTharKwe Bus	0.386
7	17	16.96119	96.0969	ShwePyiTharKwe Bus		18	16.95808	96.09751	SitTad Bus	0.402
8	18	16.95808	96.09751	SitTad Bus		20	16.95011	96.10072	KanTharYar Bus	0.899
9	20	16.95011	96.10072	KanTharYar Bus		21	16.94435	96.10274	MyinHleGate Bus	0.676
10	21	16.94435	96.10274	MyinHleGate Bus		25	16.934	96.10199	DaNyinGone Point	1.16
11	25	16.934	96.10199	DaNyinGone Point		43	16.92542	96.09856	YwarThit Bus	1.06
12	43	16.92542	96.09856	YwarThit Bus		44	16.92142	96.09763	AungSanZay Bus	0.457
13	44	16.92142	96.09763	AungSanZay Bus		46	16.91533	96.09725	No1Gate Bus	0.687
14	46	16.91533	96.09725	No1Gate Bus		48	16.90767	96.09764	FawtKanZay Bus	0.854
15	48	16.90767	96.09764	FawtKanZay Bus		49	16.90504	96.09772	ThiriMingalar Bus	0.283
16	49	16.90504	96.09772	ThiriMingalar Bus		51	16.90066	96.09716	PyiTawThar Point	0.503
17	51	16.90066	96.09716	PyiTawThar Point		52	16.8982	96.09904	L_YwarMa Bus	0.343

Figure 4.11 Distance Data for Locations of Points

The following sequence indexing numbers are Distance values analyzed by Haversine and historical distance data. And then the sample of Demonstration of Calculation of Distance and Historical Traffic Data by distance format type shows as the following:

Historical Location_Distance File:

1:2,2.17
2:3,0.851:1,2.17
3:4,0.252:6,0.95:2,0.85 1:174,2.79:185,0.905
4:5,0.274:3,0.252
5:6,0.399:4,0.274
6:7,0.511:8,0.513:5,0.399:3,0.95:184,1.78
7:6,0.511
8:9,0.145:11,1.24:6,0.513:237,1.46
9:10,0.54:8,0.145
10:11,0.581:9,0.54
11:12,0.508:13,1.09:10,0.581:8,1.24:197,1.75:198,0.731
12:13,0.562:11,0.508
13:14,0.254:16,1.13:12,0.562:11,1.09:203,1.15
14:15,0.404:13,0.254
15:16,0.443:14,0.404
16:17,0.386:15,0.443:13,1.13:91,3.57
17:18,0.402:16,0.386:238,0.554

```
18:19,0.744:17,0.402:239,0.694
19:20,0.211:18,0.744
20:21,0.676:19,0.211:240,1.16
21:22,0.165:25,1.16:20,0.676:242,0.581
22:23,0.186:21,0.165
23:24,0.468:22,0.186:241,1.09
24:25,0.339:23,0.468
25:26,0.148:105,0.894:24,0.339:21,1.16:28,0.586:42,0.466
26:27,0.125:25,0.148
27:28,0.297:26,0.125
28:29,0.179:25,0.586:27,0.297:32,0.838
… … …
```

4.4.3 Illustration for Structure of Location Distance file to calculate Distance

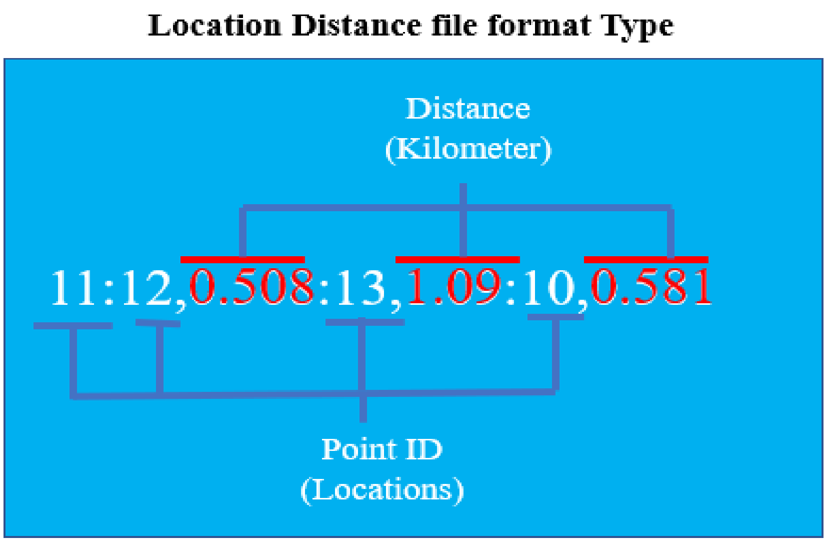


Figure 4.12 Calculation Type of Location Distance

As above the demonstration Figure 4.12, these numbers are the location points IDs and distance (kilometer) between these each point in location_distance index file. This file is for use calculation of distance to get the shortest distance and optimal path in program. In these the distance between point ID 11 and 12 is 0.508 km, the distance between point ID 11 and 13 is 1.09 km, and the distance between point ID 11 and 10 is 0.581 km each other.

4.5 Processes of the ATCOR System

There are organized the following procedures for approaches of the system:

Process (1)- Client-Server Web base system, Server and Database on Cloud

Process (2)- Current Location

Mobile GPS Tracking Unit

Process (3)- Find Nearest Routes and Points

User determine the destination

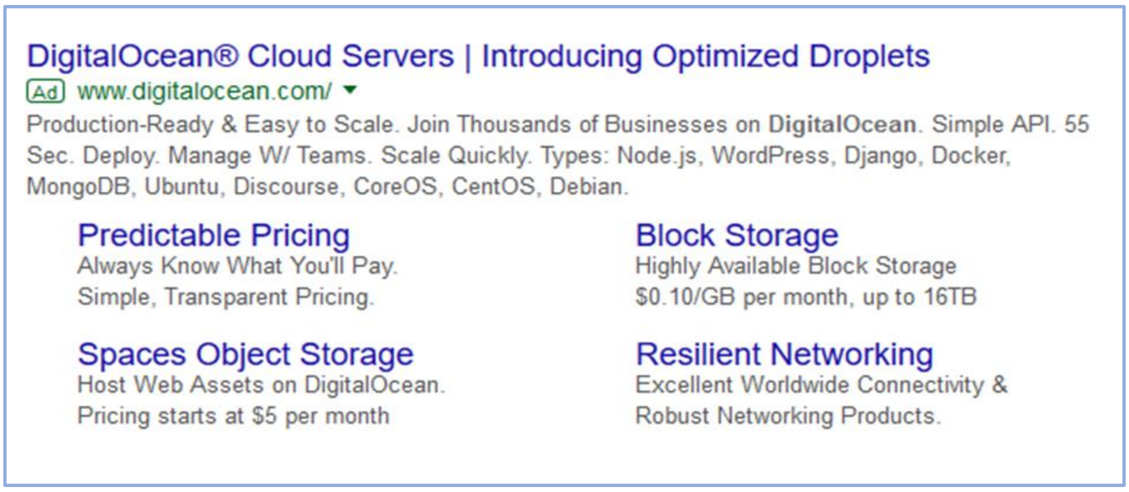
Get nearest Points and traffic points by Haversine Formula

Process (4)- Detecting and Avoiding the traffic congestion on the routes

Routing Pattern Algorithm

4.5.1 Client-Server Web Base System, Server and Database on Cloud

In these process (1), the system chooses the Cloud Virtual Machine to use as Server and database. This is Digital Ocean product for storage and uses some budget 5$ per month. The following Figure 4.13 is the official website of Digital Ocean service.



DigitalOcean® Cloud Servers | Introducing Optimized Droplets

Ad www.digitalocean.com/

Production-Ready & Easy to Scale. Join Thousands of Businesses on DigitalOcean. Simple API. 55 Sec. Deploy. Manage W/ Teams. Scale Quickly. Types: Node.js, WordPress, Django, Docker, MongoDB, Ubuntu, Discourse, CoreOS, CentOS, Debian.

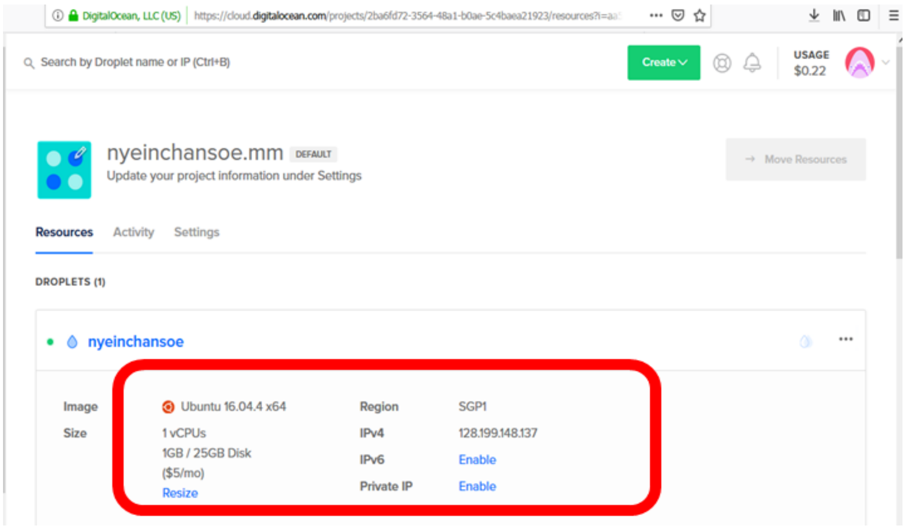
Predictable Pricing
Always Know What You'll Pay.
Simple, Transparent Pricing.

Block Storage
Highly Available Block Storage
$0.10/GB per month, up to 16TB

Spaces Object Storage
Host Web Assets on DigitalOcean.
Pricing starts at $5 per month

Resilient Networking
Excellent Worldwide Connectivity &
Robust Networking Products.

Figure 4.13 Web Site for Using Cloud Server and Database



DigitalOcean, LLC (US) https://cloud.digitalocean.com/projects/2ba6fd72-3564-48a1-b0ae-5c4baea21923/resources?i=aa

Search by Droplet name or IP (Ctrl+B)

Create

USAGE $0.22

nyeinchansoe.mm DEFAULT

Update your project information under Settings

Move Resources

Resources Activity Settings

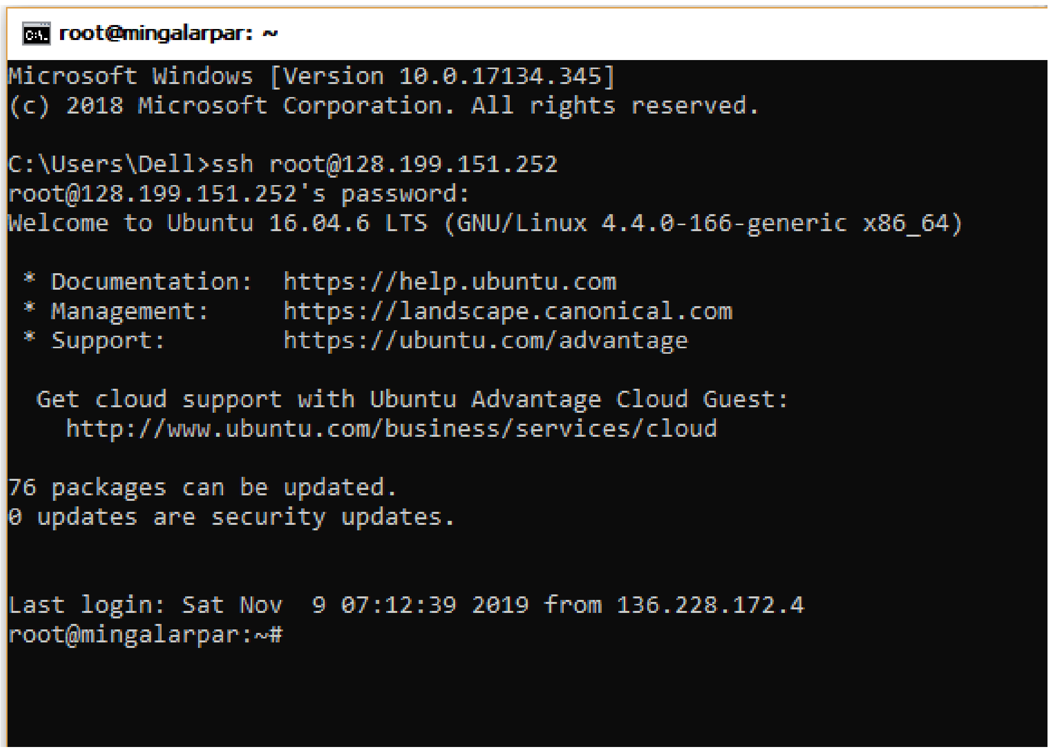
DROPLETS (1)

nyeinchansoe

Image	Ubuntu 16.04.4 x64	Region	SGP1
Size	1 vCPUs 1GB / 25GB Disk ($5/mo) Resize	IPv4	128.199.148.137
		IPv6	Enable
		Private IP	Enable

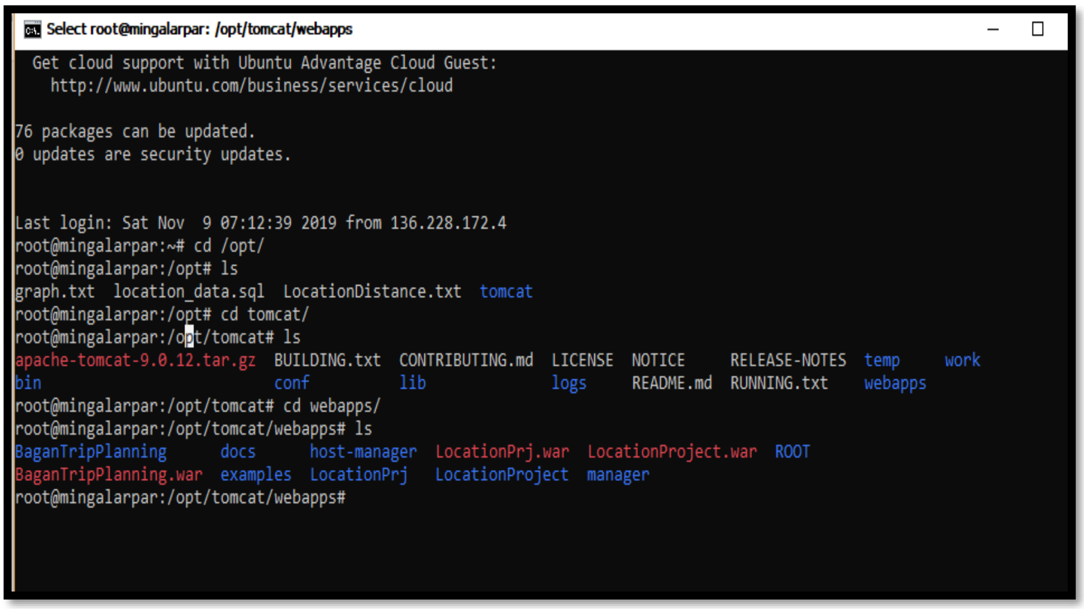
Figure 4.14 Web Service of Cloud Database

The VM is High performance this provide developer cloud service that help to deploy and scale application that run simultaneously on multiple computers. This specification is provided public IP V4 and V6 and then storage size is 25Gigg Disk as shown in Figure 4.14. It uses Linux command to apply these server and database on cloud VM as like the following Figure 4.15, 4.16, 4.17,and 4.18. This gets one VM account and password.



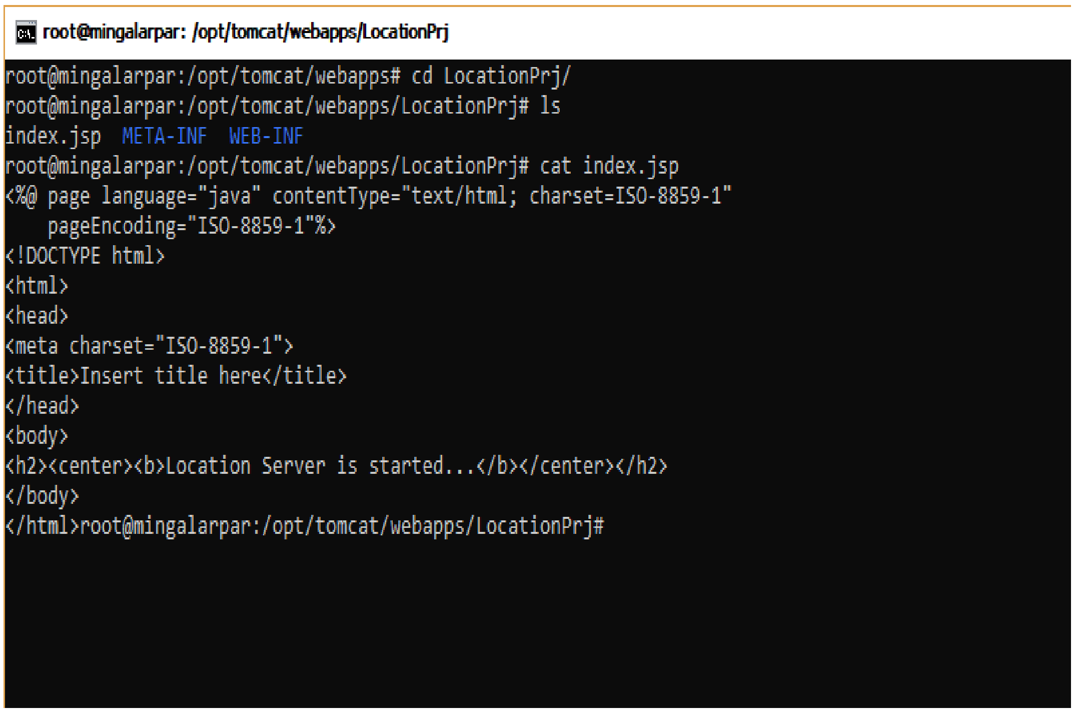
```
root@mingalarpar: ~
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dell>ssh root@128.199.151.252
root@128.199.151.252's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-166-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

76 packages can be updated.
0 updates are security updates.


Last login: Sat Nov  9 07:12:39 2019 from 136.228.172.4
root@mingalarpar:~#
```

Figure 4.15 Linux Commands to Enter Server on Cloud



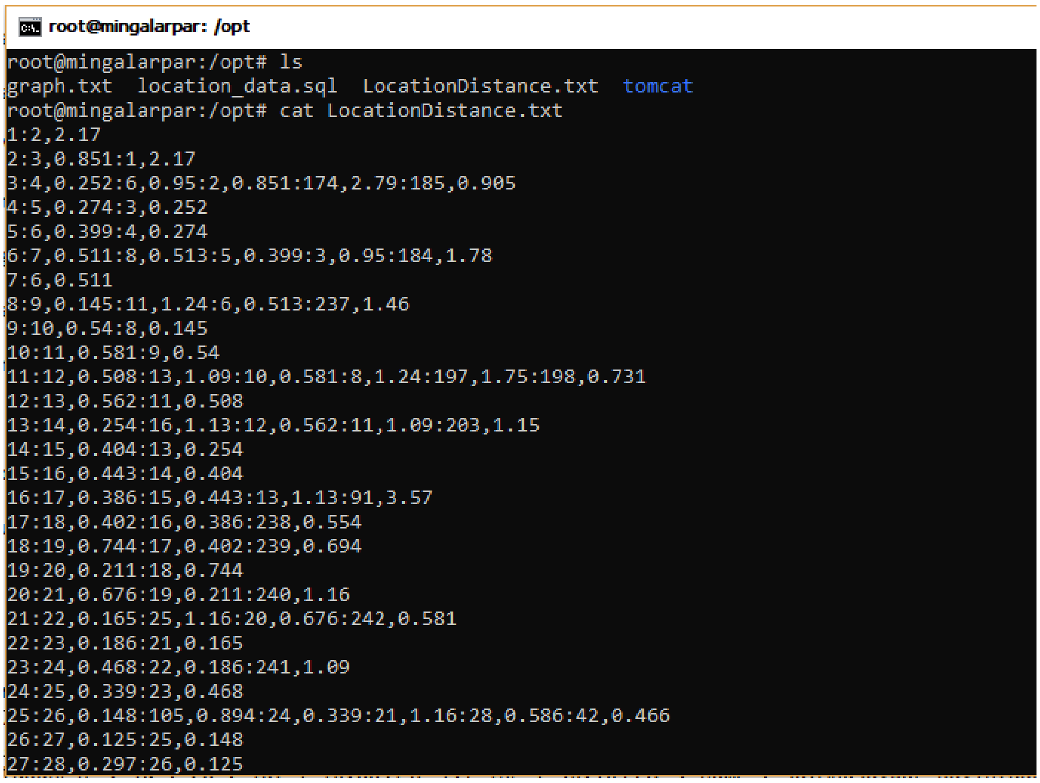
```
Select root@mingalarpar: /opt/tomcat/webapps
  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

76 packages can be updated.
0 updates are security updates.


Last login: Sat Nov  9 07:12:39 2019 from 136.228.172.4
root@mingalarpar:~# cd /opt/
root@mingalarpar:/opt# ls
graph.txt  location_data.sql  LocationDistance.txt  tomcat
root@mingalarpar:/opt# cd tomcat/
root@mingalarpar:/opt/tomcat# ls
apache-tomcat-9.0.12.tar.gz  BUILDING.txt  CONTRIBUTING.md  LICENSE  NOTICE     RELEASE-NOTES  temp     work
bin                          conf          lib              logs     README.md  RUNNING.txt    webapps
root@mingalarpar:/opt/tomcat# cd webapps/
root@mingalarpar:/opt/tomcat/webapps# ls
BaganTripPlanning      docs      host-manager  LocationPrj.war  LocationProject.war  ROOT
BaganTripPlanning.war  examples  LocationPrj   LocationProject  manager
root@mingalarpar:/opt/tomcat/webapps#
```

Figure 4.16 Linux Commands to Get Project in Tomcat Server



```
root@mingalarpar: /opt/tomcat/webapps/LocationPrj
root@mingalarpar:/opt/tomcat/webapps# cd LocationPrj/
root@mingalarpar:/opt/tomcat/webapps/LocationPrj# ls
index.jsp  META-INF  WEB-INF
root@mingalarpar:/opt/tomcat/webapps/LocationPrj# cat index.jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2><center><b>Location Server is started...</b></center></h2>
</body>
</html>root@mingalarpar:/opt/tomcat/webapps/LocationPrj#
```

Figure 4.17 Linux Commands to Start Server on Cloud



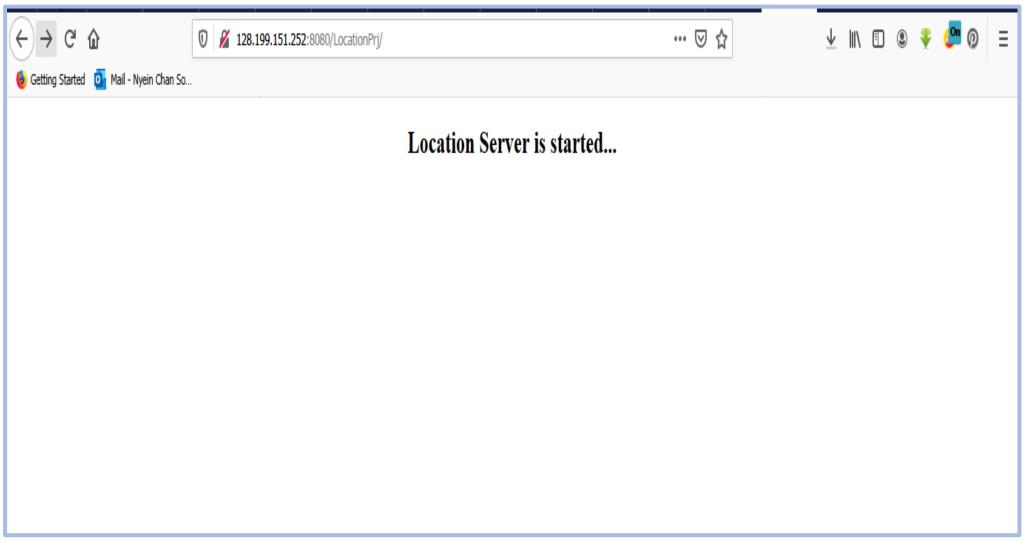
```
root@mingalarpar: /opt
root@mingalarpar:/opt# ls
graph.txt  location_data.sql  LocationDistance.txt  tomcat
root@mingalarpar:/opt# cat LocationDistance.txt
1:2,2.17
2:3,0.851:1,2.17
3:4,0.252:6,0.95:2,0.851:174,2.79:185,0.905
4:5,0.274:3,0.252
5:6,0.399:4,0.274
6:7,0.511:8,0.513:5,0.399:3,0.95:184,1.78
7:6,0.511
8:9,0.145:11,1.24:6,0.513:237,1.46
9:10,0.54:8,0.145
10:11,0.581:9,0.54
11:12,0.508:13,1.09:10,0.581:8,1.24:197,1.75:198,0.731
12:13,0.562:11,0.508
13:14,0.254:16,1.13:12,0.562:11,1.09:203,1.15
14:15,0.404:13,0.254
15:16,0.443:14,0.404
16:17,0.386:15,0.443:13,1.13:91,3.57
17:18,0.402:16,0.386:238,0.554
18:19,0.744:17,0.402:239,0.694
19:20,0.211:18,0.744
20:21,0.676:19,0.211:240,1.16
21:22,0.165:25,1.16:20,0.676:242,0.581
22:23,0.186:21,0.165
23:24,0.468:22,0.186:241,1.09
24:25,0.339:23,0.468
25:26,0.148:105,0.894:24,0.339:21,1.16:28,0.586:42,0.466
26:27,0.125:25,0.148
27:28,0.297:26,0.125
```

Figure 4.18 Linux Commands to Extract Distance Index on Cloud

When the following command “cat” in root@server cmd for show the result of database and two index files (tables) such as location_name and traffic point as follows:

```
root @ mingalarpar: / opt # cat location_data.sql
- phpMyAdmin SQL Dump
- versione 4.7.0
- https://www.phpmyadmin.net/
-
- Host: 127.0.0.1
- Tempo di generazione: 3 dicembre 2018 alle 14:00
- Versione server: 10.1.25-MariaDB
- Versione PHP: 7.1.7

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
INIZIA L'OPERAZIONE;
SET time_zone = "+00: 00";

/ *! 40101 SET @OLD_CHARACTER_SET_CLIENT = @@
CHARACTER_SET_CLIENT * /;
/ *! 40101 SET @OLD_CHARACTER_SET_RESULTS = @@
CHARACTER_SET_RESULTS * /;
/ *! 40101 SET @OLD_COLLATION_CONNECTION = @@
COLLATION_CONNECTION * /;
/ *! 40101 SET NAMES utf8mb4 * /;
```

And then the url http://128.199.151.252:8080/LocationPrj/ in web browser is called to start the server for proving the server status as the following Figure 4.19.



Location Server is started...

Figure 4.19 Server Home Page

4.5.2 Current Location and Mobile GPS Tracking Unit

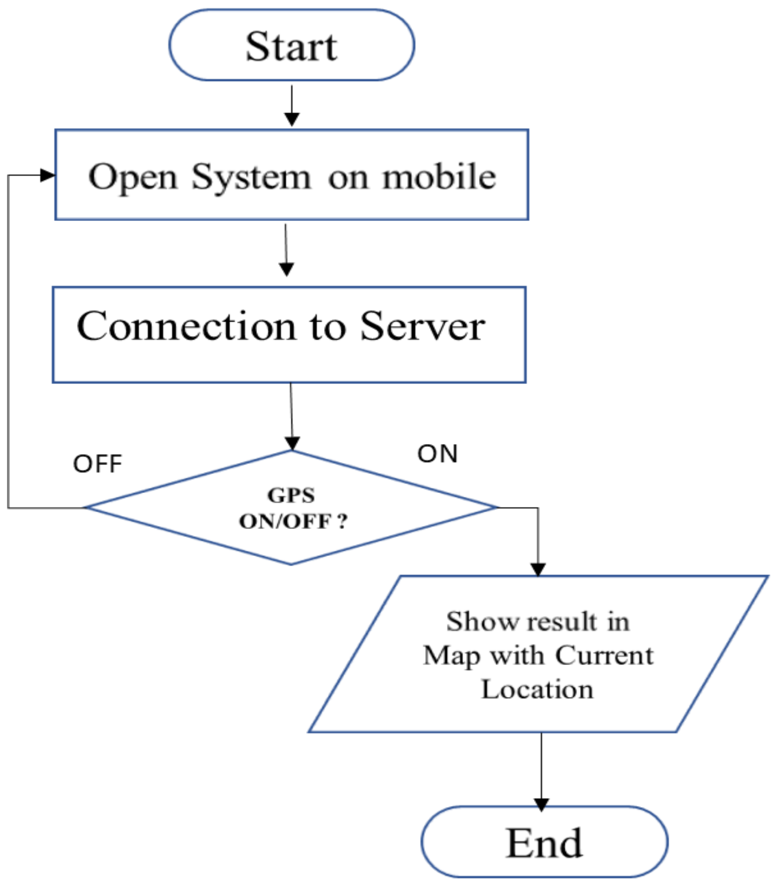


Figure 4.20 Flow Diagram of Client Side

In the process (2), on the client side, it tries to implement an iterative service that always runs in the back and provides data to the database and server. Data requesting and transmission begins when the device is at the top of the marked main road and can actually be determined to be a moving vehicle. Next, when you enter the edge / road, start the connection server after logging into the system, send the data of location information to the server and restart it by the timer. If the user GPS is not available, the system will take the notification “GPS Service On”. Finally, client-side application shows the current location with the detail of latitude and longitude on the map as the above Figure 4.20. The system then immediately implements sending a request to the server-side application. The timer interval may be overridden when a vehicle enters a road intersection and has previously attempted to leave the road. In that case, calculate and load the data accordingly so that the next pass can start counting again.

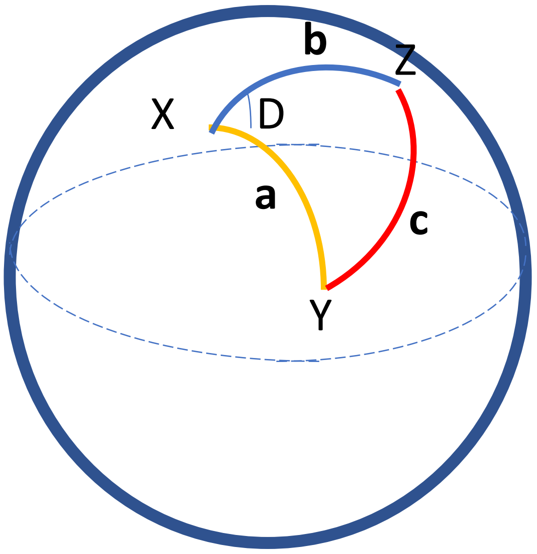
The client side and server side communicate with the formatted data. When the device reports its location, the JSON object contains coordinates, speed, and sentinel and is sent to the server. The server has a graphical data structure that represents a prepared road and stores the necessary information about the road in an Edge object reference that represents it. These values include device stack, average speed, color marker data, and so on. When the server sends map data to the requesting device, the application formats it in JSON file that can be interpreted and transmitted directly in the map view [1].

4.5.3 Find Nearest Routes and Points

In the process (3), it is calculating the geographic distance on Earth. It can easily calculate the distance of the great circle. The system needs a point closer to the user's current location. After calculating the distance using Haversine, it releases the best route to the user's desired destination.

If it has two different latitudes - the longitude values of two different points on the earth, then with the Haversine formula, this can easily calculate the distance of the great circle. It verifies the haversines law to get the equation, as shown Figure 4.21.

$$haversin\left(\frac{d}{r}\right) = haversine\ (\emptyset_2 - \emptyset_1) + \cos(\emptyset_1)\cos(\emptyset_2) haversine\ (\lambda_2 - \lambda_1)$$



$$\boldsymbol{haversin}\ (\mathrm{c}) = \boldsymbol{haversin}\ (\boldsymbol{a} - \boldsymbol{b}) + \boldsymbol{sin}(\boldsymbol{a})\boldsymbol{sin}(\boldsymbol{b})\ \boldsymbol{haversin}(\boldsymbol{D})$$

Figure 4.21 Haversine Formula

Table 4.2 Assignments and Parameters for Haversine Equation

Haversine Formula:	
Assignment	Parameter Values
a =	$\sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$
c =	$2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$
d=	R*c
φ *is latitude,* λ *is longitude,* R *is earth's radius (mean radius = 6,371km);*	

4.5.3.1 Get Nearest Routes by Haversine Algorithm

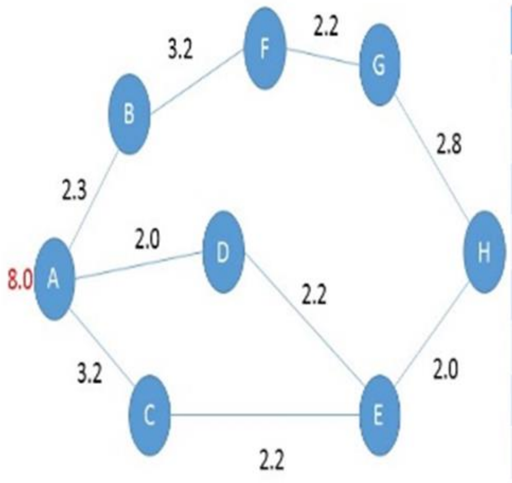
This system uses the following algorithm, modified from the Haversine formula and the A * algorithm: To complete the search for the best route while saving process time and data storing, the algorithm repeats the following steps:

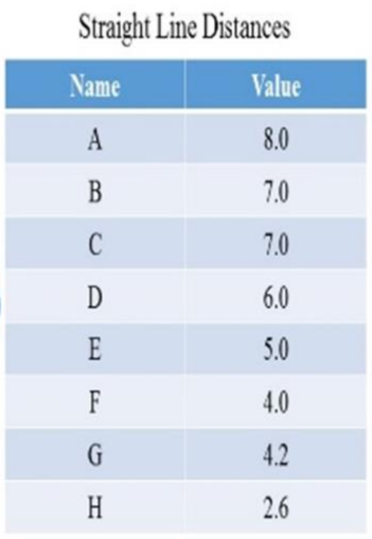
Algorithm (Haversine):

```
1. Start the program and first open the list in file.
2. Enter the starting point node into the empty set of the first-list.
3. Take the minimum cost of the first-list and move to the last-list.
4. Check all current drop-down nodes to determine adjacency.
5. (Haversine d = R * c) Meanwhile, get the closest point.
6. If there are no adjacent nodes in the first-list, calculate the cost of the node
   and record it as the formula f = g + h.
7. Enter the cost into the first-list.
8. If the adjacent node is already in the first-list, record the cost of the current
   node as the previous cost. If not, change the previous node and recalculate the
   cost.
9. Add the start and end node costs to the last-list.
10. End the search and close the list.
```

4.5.3.2 Example of The Optimal Path Processing Example:

This is finding the best route as an edge connecting roads in a considered municipality. First, define source = A, destination = H, and function cost. For the node cost f, a differential cross-section equation is given: $f = g + h$, where g is the distance from the origin to the current node, and h is the estimated distance from the current node to the end. As following Figure 4.22, the best first search approach in the system.




Straight Line Distances

Name	Value
A	8.0
B	7.0
C	7.0
D	6.0
E	5.0
F	4.0
G	4.2
H	2.6

Figure 4.22 Example of the Optimal Path Calculation

As for the equation, f(B)=9.3, f(C)=10.2, f(D)=8.0 and the minimum is D. With Calculating, three possible routes is:

A, B, F, G, H=28.3

A,C,E,H=22

A,D,E,H=19.8

Finally, the system gets the shortest path of minimum value for the routing is:

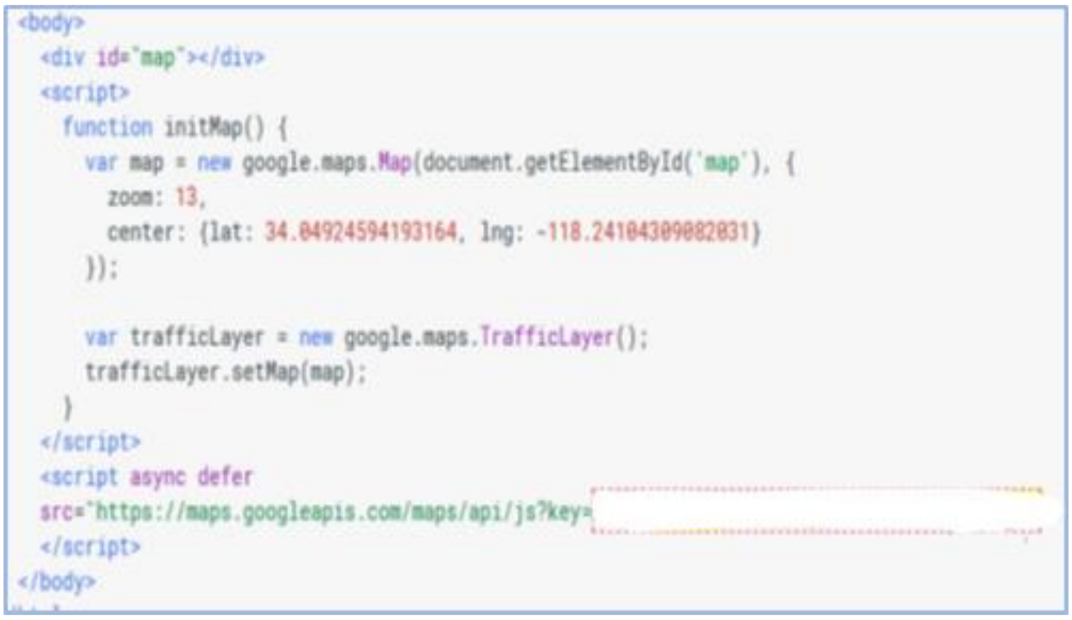
$$A \rightarrow D \rightarrow E \rightarrow H$$

4.5.4 Detecting and Avoiding the Traffic Congestion on The Routes

In the process (4), Google Maps data layer provides a container for storing data in any geospatial space. You can use the data layer to store custom data and display GeoJSON data in Google Maps. Using the Google Maps JavaScript API, you can mark maps with multiple overlays, including markers, polylines, and polygons.

Each of these annotations combines style information with location data. The Google Map data class is a container for storing arbitrary geospatial data. Instead of adding these overlays, you can use the data layer to add any geographic data to the map. If the data contains geometry such as points, lines, and polygons, by default, the API displays them as markers, polylines, and polygons. You can create these features as you would normally overlap and apply style rules based on other dataset properties.

The system step is to get information about traffic congestion. According to step (4), the Google Maps JavaScript API allows you to add real-time traffic information to the map using the Traffic Layer object (if supported). Traffic information is updated frequently, but will not be updated immediately. The Google platform supports traffic data by applying the following feature code as shown in Figure 4.23.



```
<body>
  <div id="map"></div>
  <script>
    function initMap() {
      var map = new google.maps.Map(document.getElementById('map'), {
        zoom: 13,
        center: {lat: 34.04924594193164, lng: -118.24104309082031}
      });

      var trafficLayer = new google.maps.TrafficLayer();
      trafficLayer.setMap(map);
    }
  </script>
  <script async defer
  src="https://maps.googleapis.com/maps/api/js?key=
  </script>
</body>
```

Figure 4.23 Sample Code for Traffic Layer

4.5.4.1 Routing Pattern Algorithm

The proposed system implements the work around using a Distance Matrix API web service or Directions API web service. If specify a departure time in the request, the response will contain fields duration and duration_in_traffic. Therefore, the system can Figure out if duration_in_traffic is much bigger than duration and decide if there is a traffic jam somewhere on this route.

In order to display traffic data, the user must consider the following issues, and it ensures that the user's current location is detected on our map. Make sure the user has available traffic data between your current location and your destination. The routing model algorithm can also test the traffic analysis program using the improved A* algorithm. Initialize the map correctly and then set the traffic data after detecting your current location. In the study, the proposed and implemented the modified equation is:

$$f = g+h \ (d=R.C) \qquad \text{Equation (4.3)}$$

Where $c = 2 \cdot \text{atan2} *(\sqrt{a}, \sqrt{(1-a)})$, $a = \sin^2 *(\Delta\varphi/2) + \cos \varphi 1 * \cos \varphi 2 * \sin^2 * (\Delta\lambda/2)$

The following program is Routing Pattern Algorithm (modified A*) by pseudo code.

Routing Pattern Algorithm:

```
1 Enter node_start in the OPEN list with f(node_start)= h(node_start)
2, and the OPEN list is not empty {
3 Get the lowest node node_current from the open list
4 f(node_current)= g(node_current)+ h(node_current)
5 though (Haversine d = R * c) got the nearest point; rest
6 if (h = d) in the open list {f1 = g1 + h1(d)
7}
8 If node_current is node_goal, found a solution; rest
9 generate each state node_successor after node_current
10 The node_successor of each node_current {
11Set successor_current_cost = g(node_current)+ w(node_current, node_successor)
12 if node_successor is in the OPEN list {
13 If g(node_successor)successor_current_cost continues (on line 23)
14} Otherwise, if node_successor is in the CLOSED list {
15 if g(node_successor)successor_current_cost continues (on line 23)
16 Move node_successor from the CLOSED list to the OPEN list
17}Other {
18Add node_successor to the OPEN list
19 sets h(node_successor) to the heuristic distance of node_goal
20}
21 set g (node_successor) = successor_current_cost
22 set the parent of node_successor to node_current
23}
24 Add node_current to the CLOSED list
25}
26 If (node_current!= Node_goal) error exits (OPEN list is empty)
```

4.5.4.2 Loading Data from The Same Domain

The Google Maps data layer provides likes a container for arbitrary and accuracy geospatial data (including GeoJSON). Then it can download it using the map.data.loadGeoJson () method when our data is in a file hosted on the same domain as the Maps JavaScript API. The file must include to the same domain, but it can host it in a different subdomain

map.data.loadGeoJson ('data.json');

4.5.4.3 Loading Data Across Domain

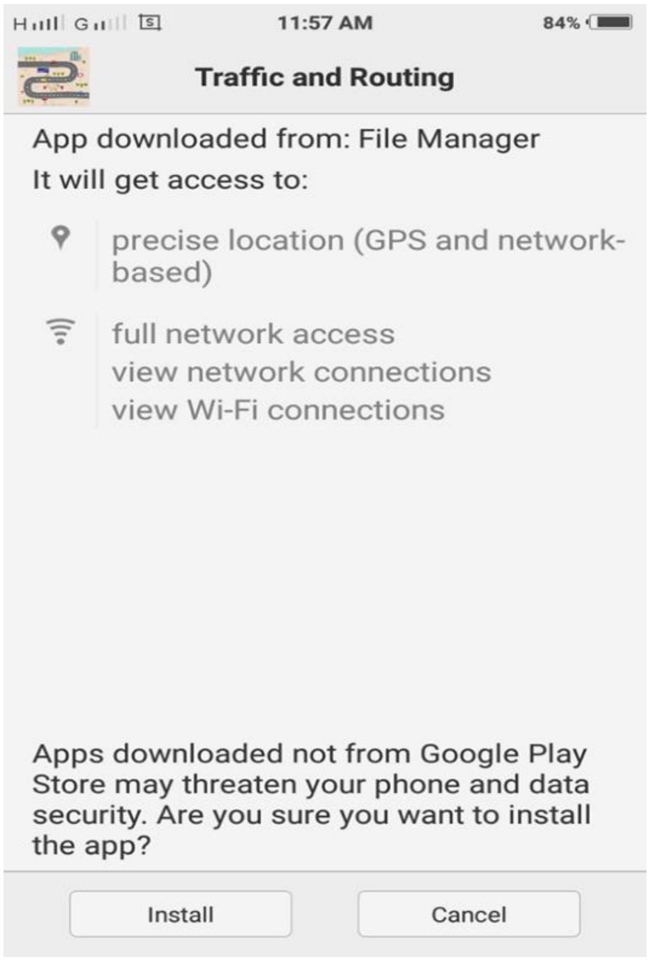
The system may also request a domain from our data, if the domain configuration enables such a request. A standard for these changes is called Cross source resource sharing (CORS). After that it can cross domain requests, if there is a domain, which include such a declaration response header:

Access-Control-Allow-Origin:

Next, to keep them in the object to see in Stratus in the advertisement, it is to paste bus congestion to the pull stop, Google, and other circumstances of the journey. This comic can save the layer (country stick) and find a way to improve the specified output. But now it is recommended to implement routes using the modified A * algorithm.

4.5.4.4 Installing Application on The Mobile Phone

Firstly, when the proposed system has to use our proposed application to take the route with traffic less navigation, it installs this android application (Traffic_and_Routing.apk) in our mobile phone. The system requirements in mobile for this application are precise location that get GPS and network base technology and full internet network access. The system has to recommend android software version is 6.0 and above. Therefore, this application is work for more facilities and relevance for routing system on mobile phone. The following Figure 4.24 is the installing of the application on android version 8.0 phone. After that user will get already, use this application on their mobile phone. Figure 4.25 shows the working of this android application to get the result of traffic avoiding and optimal routing with sample source and destination.



App downloaded from: File Manager
It will get access to:

precise location (GPS and network-based)

full network access
view network connections
view Wi-Fi connections

Apps downloaded not from Google Play Store may threaten your phone and data security. Are you sure you want to install the app?

Figure 4.24 Installing page of Traffic and Routing

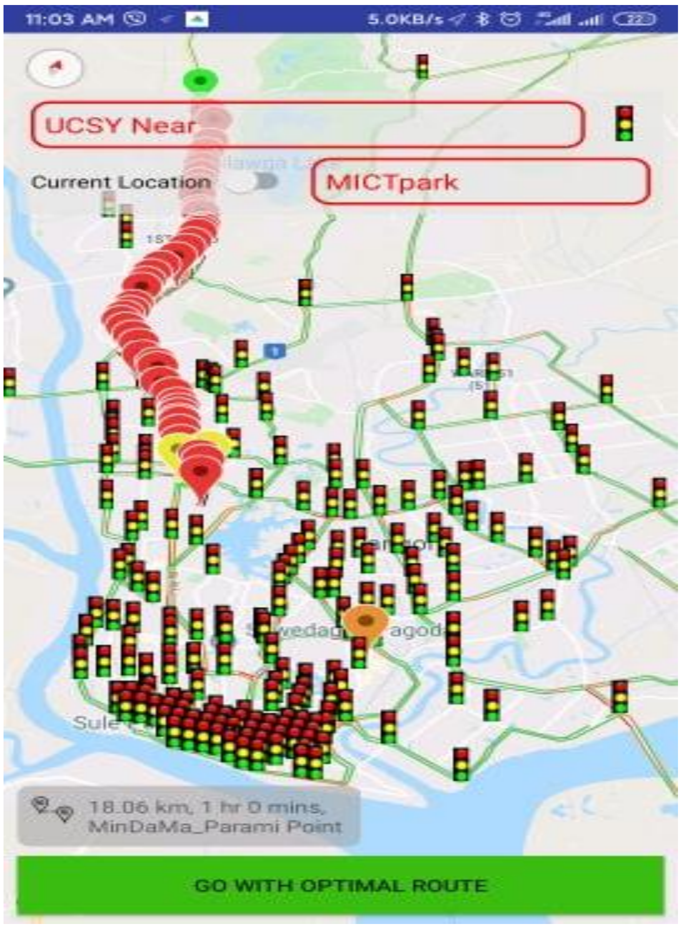


Figure 4.25 Optimal Route Information on Mobile Application

4.5.4.5 Testing for Optimal Route without Traffic Congestion

In this test, it assumes Considering the route from Source A, to Destination B in Insein Township, Yangon. There are three possible routes in map: (1) Pyay-Thirimingar, (2) Insein- Lower Mingalardon, (3) Insein- Upper Mingalardon. And this will pass four traffic congestions as the following Figure 4.26.

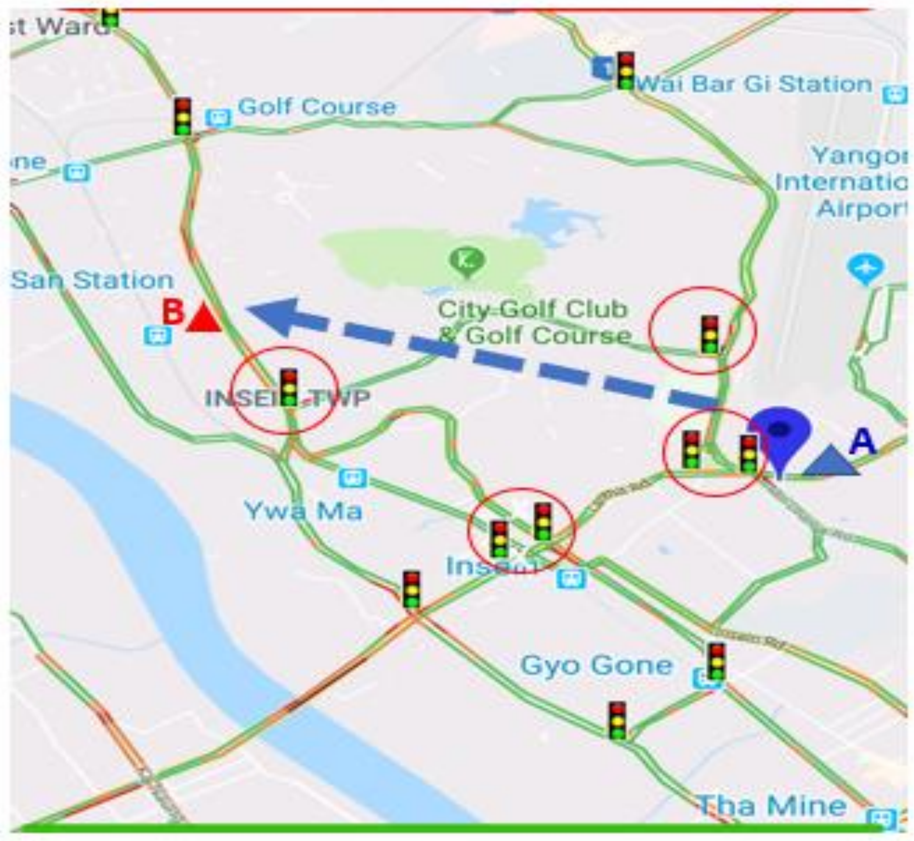


Figure 4.26 Sample Direction and Traffic Status Map

Firstly, the proposed system tests in Google Map for normal route from source A to Destination B. The result gets following path:

Source A, Sawbwargyigone Bus_Stop, "Sawbwargyigone Traffic_Point," PhaYarShay Bus_Stop, ShanKone Bus_Stop, Insein Park Bus_Stop, "Insein Park Traffic_Point," Insein Hospital(lower lane) Bus_Stop, YwarMa Bus_Stop, Pyidawthar Bus_Stop, "Pyidawthar Traffic_Point," ThiriMingalar Bus_Stop, PhawKan Bus_Stop, and Destination B as shown in Figure 4.27.

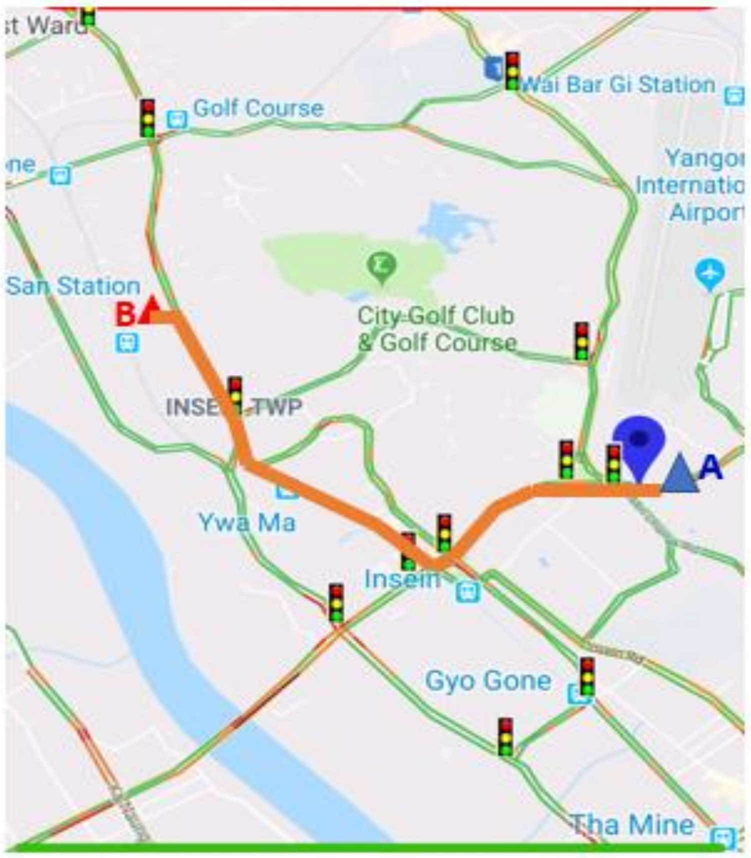


Figure 4.27 Google Normal Direction

And then, the proposed system tests using ATCOR application for optimal route from source A to Destination B. The result gets following path:

Source A, KyaukTawGyi Bus_Stop, PhaYarShay Bus_Stop, ShanKone Bus_Stop, Insein Park Bus_Stop, "Insein Park Traffic_Point," Insein Hospital(Upper mingalardon Rd) Bus_Stop, MaNyaKa Bus_Stop, HydePark Bus_Stop, BoKone Bus_Stop, ThiriMingalar Bus_Stop, PhawKan Bus_Stop, and Destination B as shown in Figure 4.28.

In this testing, the excellent result is the optimal route without traffics that this can avoid two traffic areas in the possible route than normal route. As another good advantages are this system can choice source pointes where user want to be and if user haven't need other source as for current location, that is also complete action for the optimal route. And then the system shows detail of the optimal route such as estimate time and distance between source and destination.

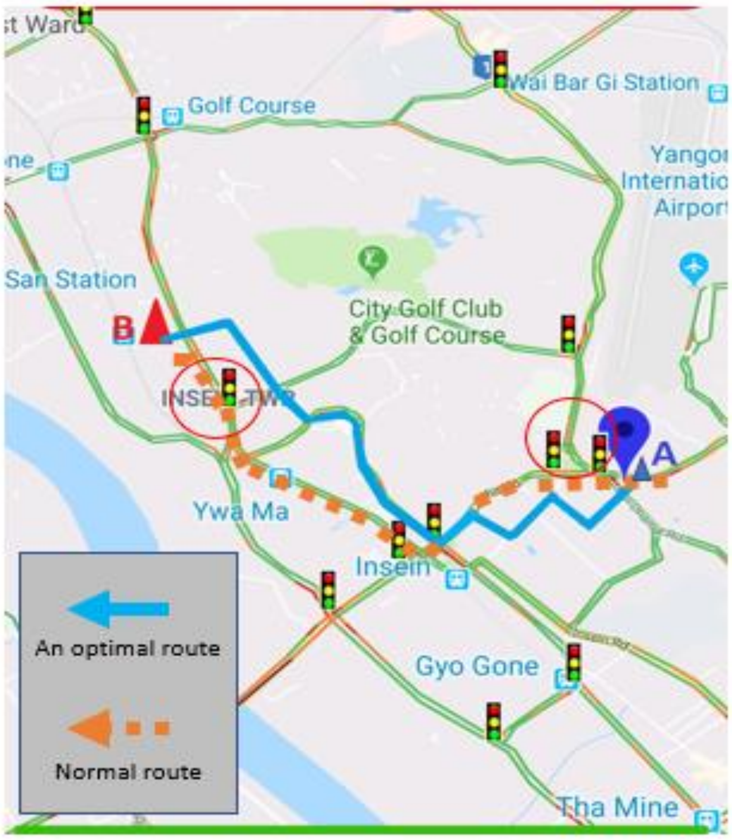


Figure 4.28 Optimal Route without Traffic Congestion

4.6 Guidance Application for User

In the system, other assistant for the clients to use the application is Guide for User that is called "How to Use". In Menu Bar, the last bottom "How to Use" help the users how to check, implement, work and route with optimal and that is avoiding traffic to the destination on the digital map. There are four instructions in this service as the follow numbering:

(1) User Registration

- If user want, he can register on the application

(2) Show My Location

- Click the button to see user current location. User must get the result with Latitude and Longitude Location on Map. And then back button on the Phone.

(3) Traffic Detection and Optimal Route

- Click the button to get the optimal route without traffic congestions. In these user choices and enter location points into source and destination drop box above on the screen. After that click Find and Go Button on bottom. Now user can get the results. Another way is user can choice current location that it is like as source point and then enter location point in destination box.

(4) Show Traffic-Status

-This service is "Show Traffic-Lights Status" function and user click this button from Menu and then user get the result of Traffic-Light (Traffic Points) around the user current location and statements of Traffic-lights in the Yangon City.

4.7 Summary

This chapter presents Process of Traffic Avoiding and Routing System with the proposed methods. The proposed system is implemented to collect data and defied research area so it works for pre-processing. The system gets the mobile user's current location from Google API and updates them from GPS location algorithm. The system proposes to build an index tree that can take advantage of dynamic range queries. The required performance of this proposed index structure is evaluated using haversine formulas and algorithms. In the historical database there are two index data files: location_name file and traffic_name file. This is Digital Ocean Cloud Virtual Machine for using database with indexing tree datasets. Then the implementation of the proposed system is discussed in chapter 5 with structure of program, system demonstration, and detail process of the system. These are organized the following procedures for approaches of the system: Process (1)- Client-Server Web base system, Server and Database on Cloud, Process (2)- Getting Current Location, Process (3)- Find Nearest Routes and Points, Process (4)- Detecting and Avoiding the traffic congestion on the routes. Chapter 5 will discuss experimental results for the research after implementation and processes as explained these chapter.

CHAPTER 5
EXPERIMENTAL RESULTS

This chapter presents the evaluation and experimental result of the optimal routing and traffic congestion system.

5.1 Evaluation on Road Traffic Avoiding System

This system detects the road traffic congestion with three states: jam state, heavy state, and normal state. The traffic states described with four traffic parameters: jam zone, rush hour, speed, Traffic-light for weeKDays and weekends.

The result of traffic avoiding and optimal routing in UCSY_SawbwarGyigone road segment is as shown in Figure 5.1. The system take source point (SawbawarGyigone) as current location and destination point (UCSY). It shows the optimal route with avoiding traffic on Insein road from source to destination. The figure shows the result that the optimal route and the location points with red color.

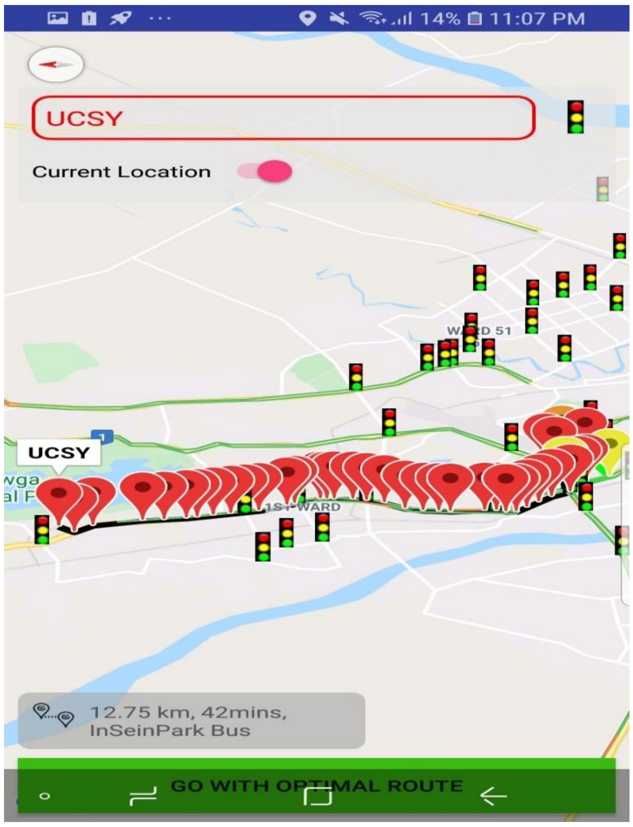


Figure 5.1 Optimal Route Map with Location

The users can apply the system very easily and quickly for the traffic route, they need to know. The proposed system provides the users with the services as shown in figure 5.2.

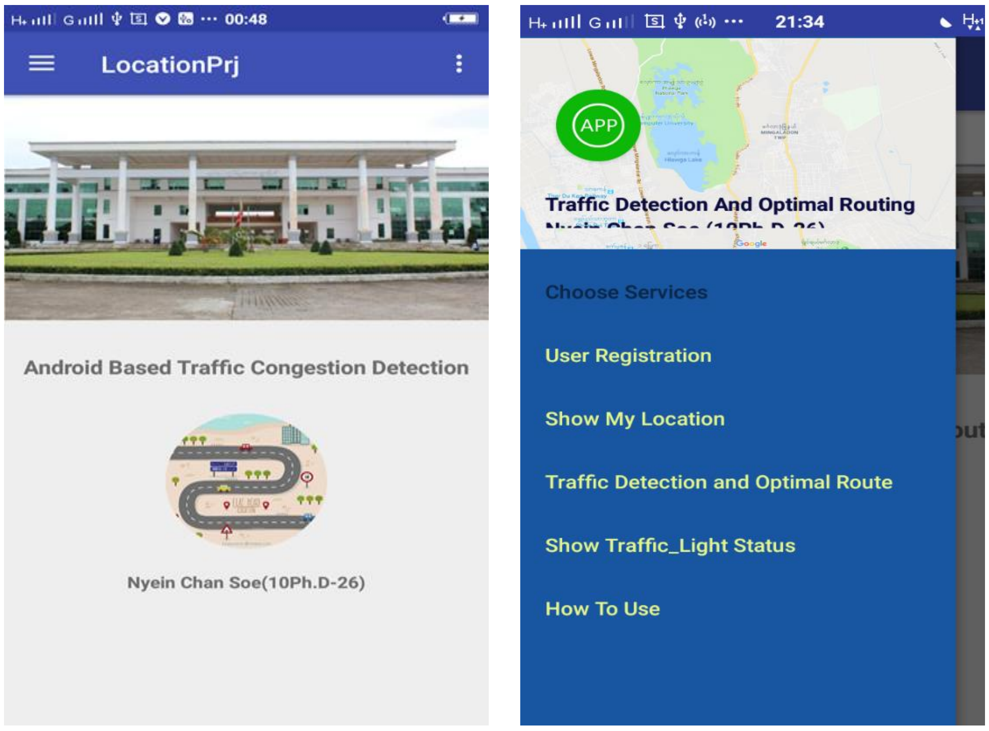
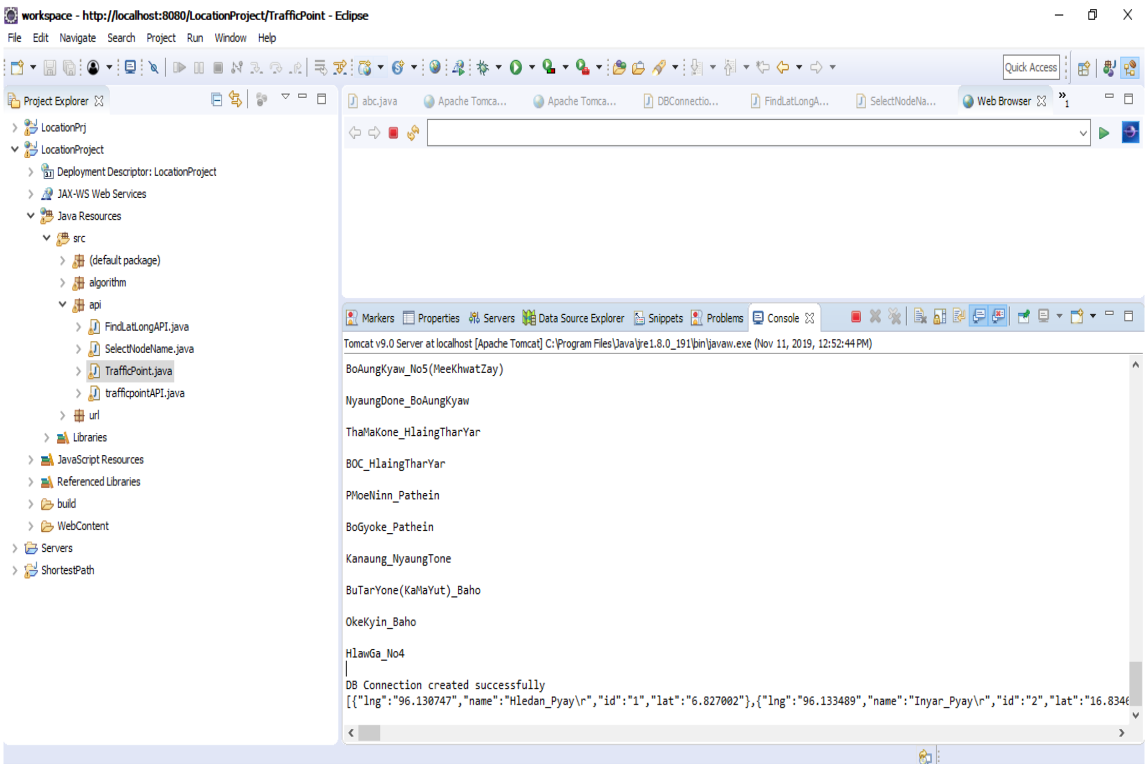

00:48
LocationPrj
Android Based Traffic Congestion Detection
Nyein Chan Soe(10Ph.D-26)
21:34
APP
Traffic Detection And Optimal Routing
Choose Services
User Registration
Show My Location
Traffic Detection and Optimal Route
Show Traffic_Light Status
How To Use

Figure 5.2 System Services Page

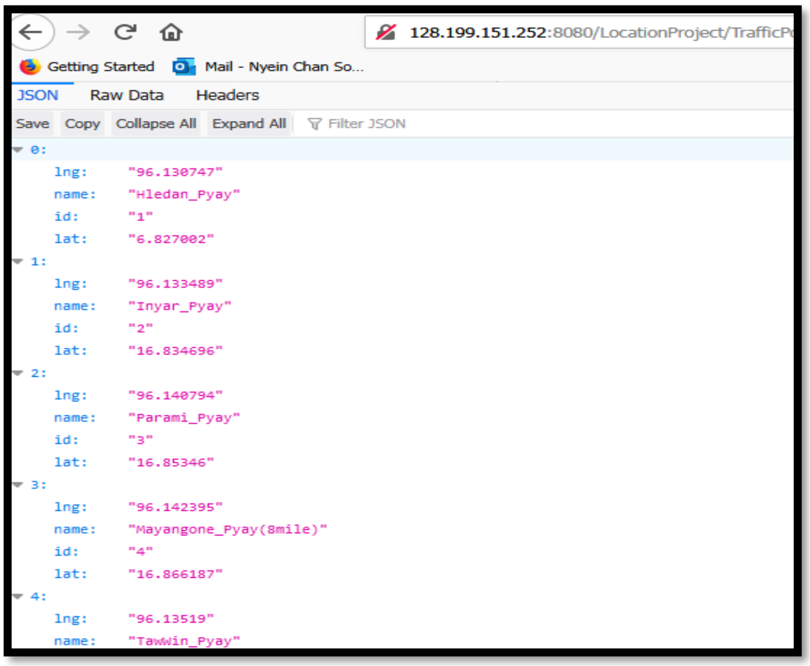
In server side, the traffic points result implemented by eclipse application. This testing is using in local by Eclipse IDE for Web Developers and android users, Version: 4.7.1 Oxygen Release and Apache Tomcat v9.0 Server at localhost to get the real result. After testing the data in local, the system implements in the public server with these real resulted data in the (128.199.151.252:8080) cloud database.



```
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (Nov 11, 2019, 12:52:44 PM)
BoAungKyaw_No5(MeeKhwatZay)

NyaungDone_BoAungKyaw

ThaMaKone_HlaingTharYar

BOC_HlaingTharYar

PMoeNinn_Pathein

BoGyoke_Pathein

Kanaung_NyaungTone

BuTarYone(KaMaYut)_Baho

OkeKyin_Baho

HlawGa_No4

DB Connection created successfully
[{"lng":"96.130747","name":"Hledan_Pyay\r","id":"1","lat":"6.827002"},{"lng":"96.133489","name":"Inyar_Pyay\r","id":"2","lat":"16.834
```

Figure 5.3 Traffic Point Result (Local Testing)

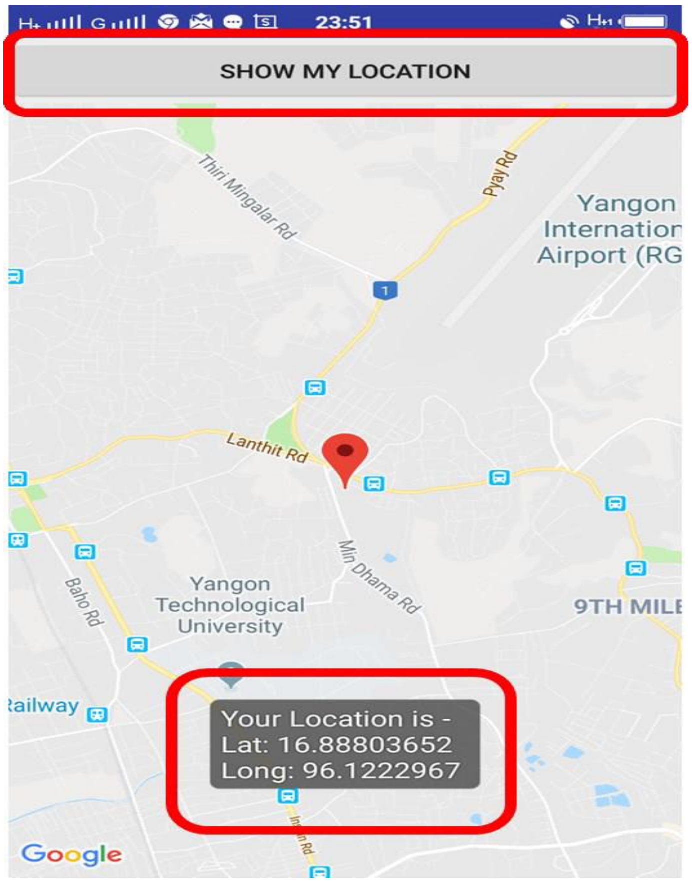
And then the system call the project URL that is http://128.199.151.252:8080/ LocationPrject/TrafficPoint in web browser to start the server for proving the server status as following Figure 5.4.



```
▾ 0:
    lng:     "96.130747"
    name:    "Hledan_Pyay"
    id:      "1"
    lat:     "6.827002"
▾ 1:
    lng:     "96.133489"
    name:    "Inyar_Pyay"
    id:      "2"
    lat:     "16.834696"
▾ 2:
    lng:     "96.140794"
    name:    "Parami_Pyay"
    id:      "3"
    lat:     "16.85346"
▾ 3:
    lng:     "96.142395"
    name:    "Mayangone_Pyay(8mile)"
    id:      "4"
    lat:     "16.866187"
▾ 4:
    lng:     "96.13519"
    name:    "Tawwin_Pyay"
```

Figure 5.4 Traffic Point Result on Web Browser

In the experiments, the system collects the user's current location, then detects most of the traffic jam area such as Sawbwargyigone, Khahayaypin Point, Danyingone Point, and Computer Studies University intersect intersect in Insein, Mingaladon, and the municipalities of Yangon, Shwepita. Then the system shows the optimal route with routes information (as shown in Figure 5.7) using Modified-A * algorithm. Figure 5.5 shows the user current location with latitude and longitude by using GPS in android. The user chooses the best route (using the orange arrow) but avoids other lanes. Also, refrain from most traffic jams through traffic data on Google Maps.



Your Location is -
Lat: 16.88803652
Long: 96.1222967

Figure 5.5 User Current Location in GPS

The proposed system provides one of services that “Show Traffic-Lights Status”, to assist the people who are citizens and travelling tourists. This service provides the Traffic-Light (Traffic Points) information of target downtown area in the Yangon City to the users as shown in Figure 5.6 (A) and Figure 5.6 (B).

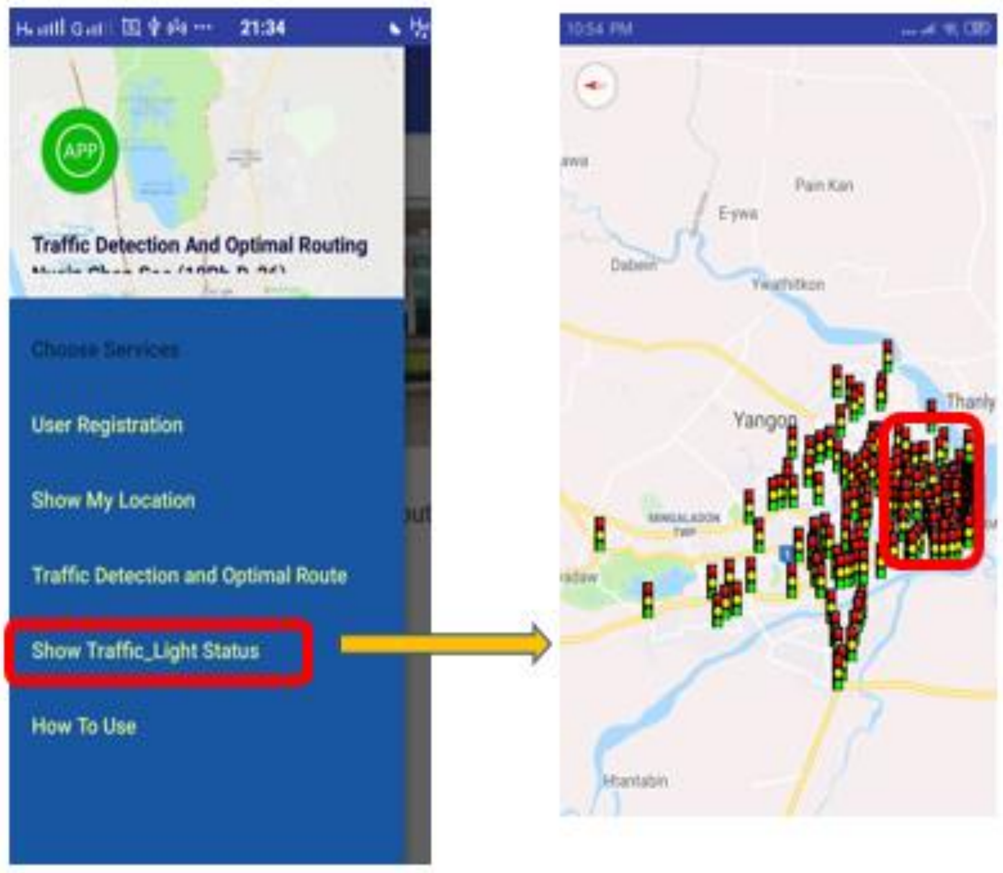


Figure 5.6 (A) Traffic-Light Status Map

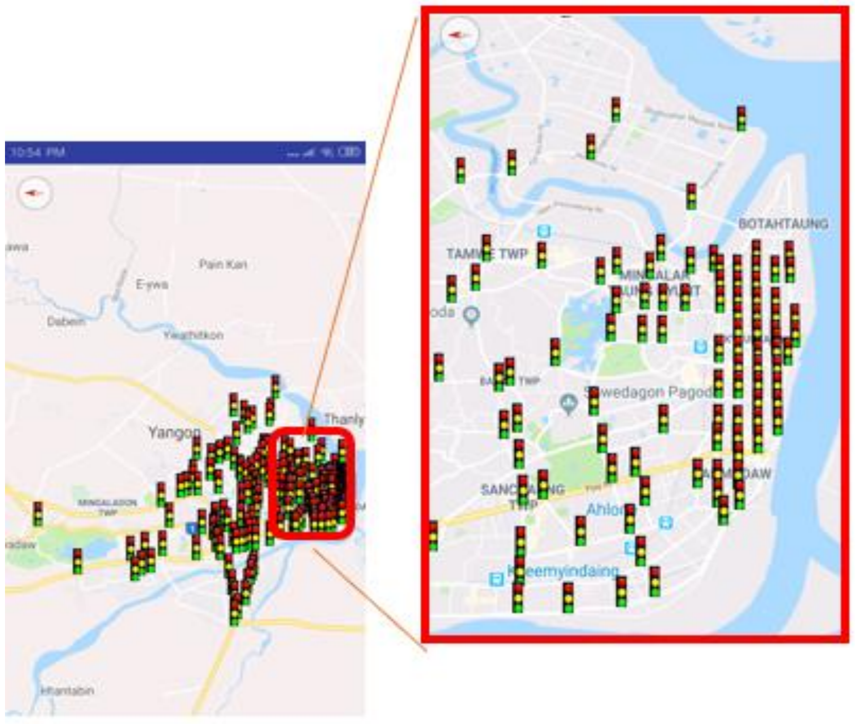


Figure 5.6(B) Traffic-Light Status of Downtown Area

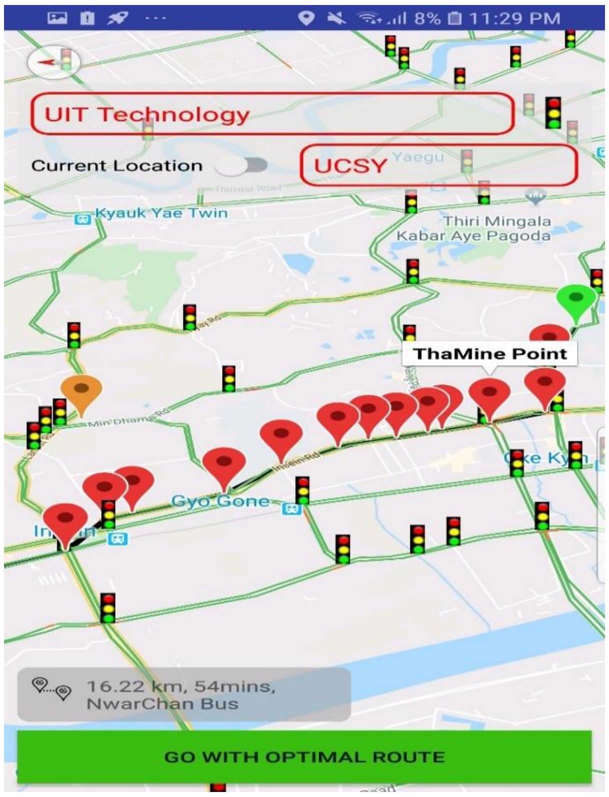
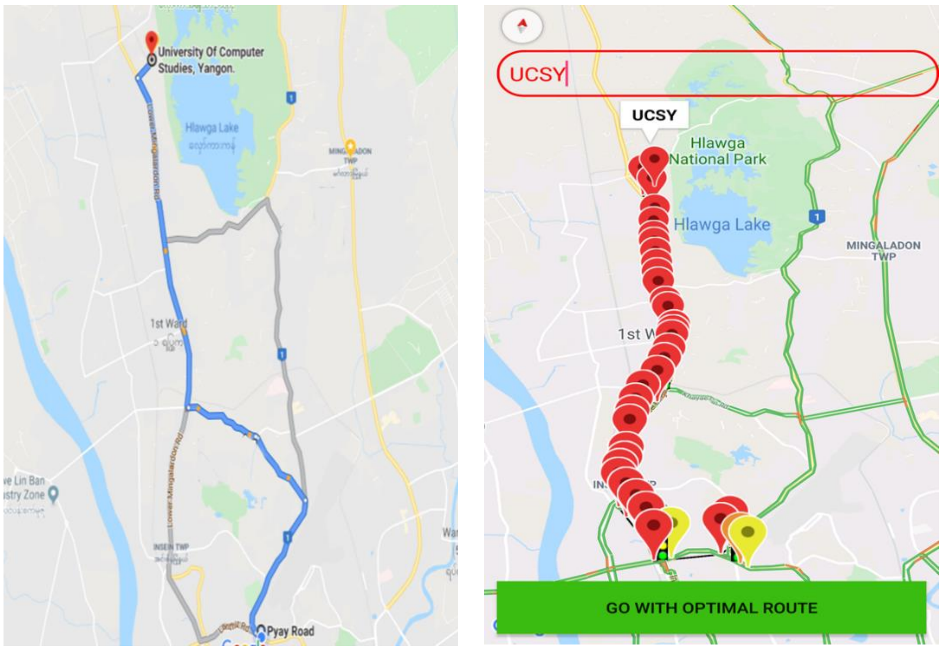


Figure 5.7 Optimal Route Map

As the above Figure 5.7, for taking the accuracy result of proposed algorithm, the processing time of this algorithm is depending on the expanded nodes. By comparing with A* algorithm, it estimates two distances g (n) and h (n) in the search and classifies each node with an equation, f(n) = g(n) + h(n). In heuristic research, the A* algorithm is widely considered the most efficient method. But the proposed system modified A* algorithm equation with Haversine formula like as an equation, f = g+h(d=R.C), where R is earth's radius (mean radius = 6,371km). This modified algorithm is also called Routing Pattern Algorithm (RPA).

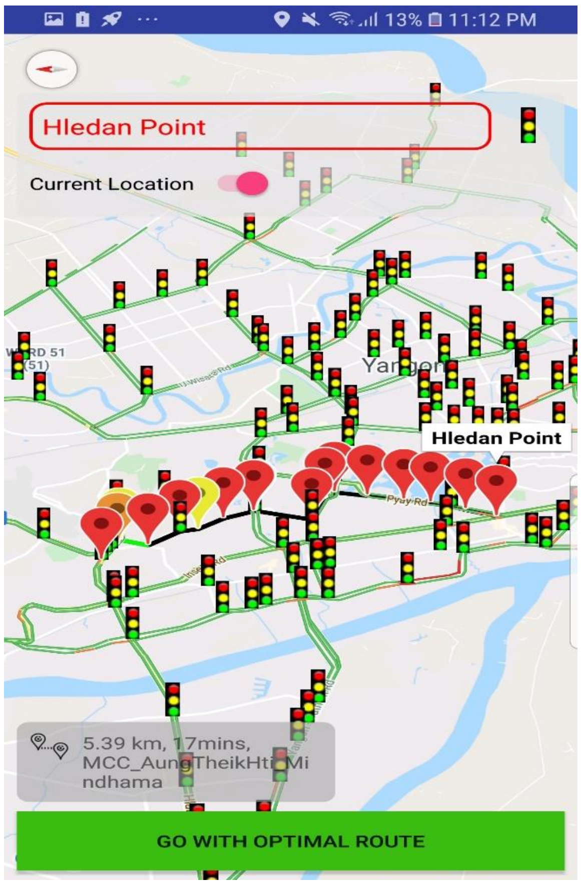
The modified algorithm reduces more processing time because of the shortest path search starts from source and expands node that towards destination and our algorithm uses the same process as A* algorithm but not allowed accumulated cost of edge plus the time complexity. Therefore, it proves the proposed Modified-A* algorithm or RPA reduce the finding process than another shortest path algorithm.

The chapter discusses the different path between Google Map and ATCOR Map in Figure 5.8(a) and (b). Figure 5.8(a) shows the result of normal traffic route on Google Map and Figure 5.8(b) shows the Traffic-lights, bus stops and the optimal route that avoiding the traffic jam areas by proposed system (ATCOR) Map.



(a) **(b)**

Figure 5.8 Comparison Map for Traffic Status



5.39 km, 17mins, MCC_AungTheikHtiMi ndhama

Figure 5.9 Time and Distance Values Map

The proposed system provides users with time and distance information between their source to destination and also provides users with optimal routing information between their source and destination as shown in Figure 5.9. The system take the user current location from GPS on mobile and then detect the traffic status and provides the optimal route information related to destination place(see Figure 5.10). Figure 5.11 also shows the result of optimal route information with the user desired destination place. If the user request the desired place from their source, the system provides the optimal route informatin with the minimum distance and duration related place to the current location and their desired destination as shown in Figure 5.12. These results are efficient and optimal route to drive the goal with accurately routing information in the system.

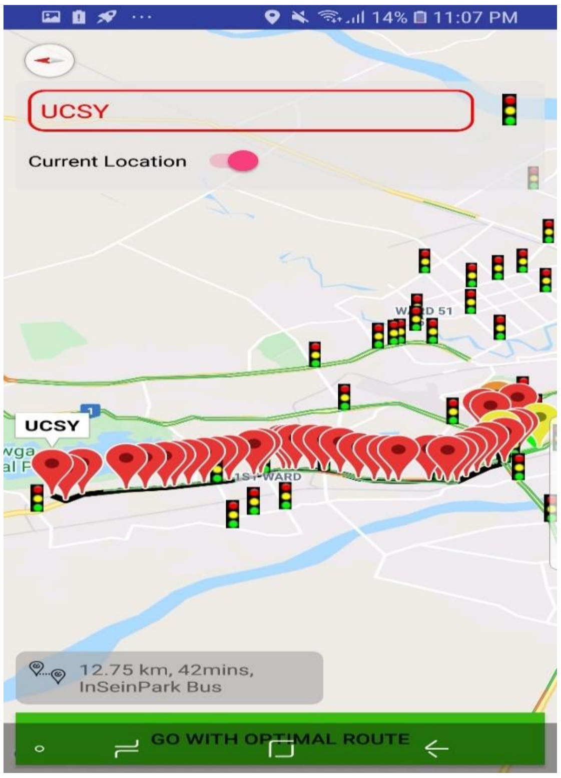


Figure 5.10 Map of Optimal Route Information using GPS

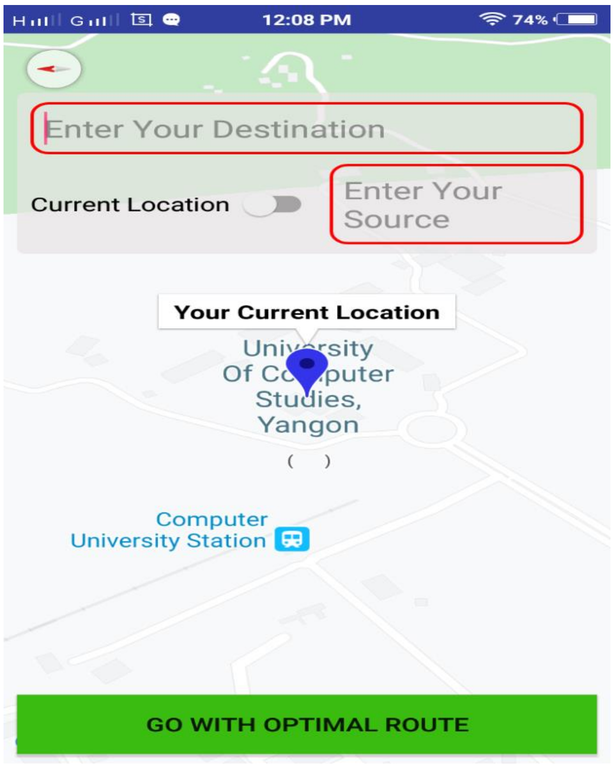


Figure 5.11 Map of User Desired Destination Point

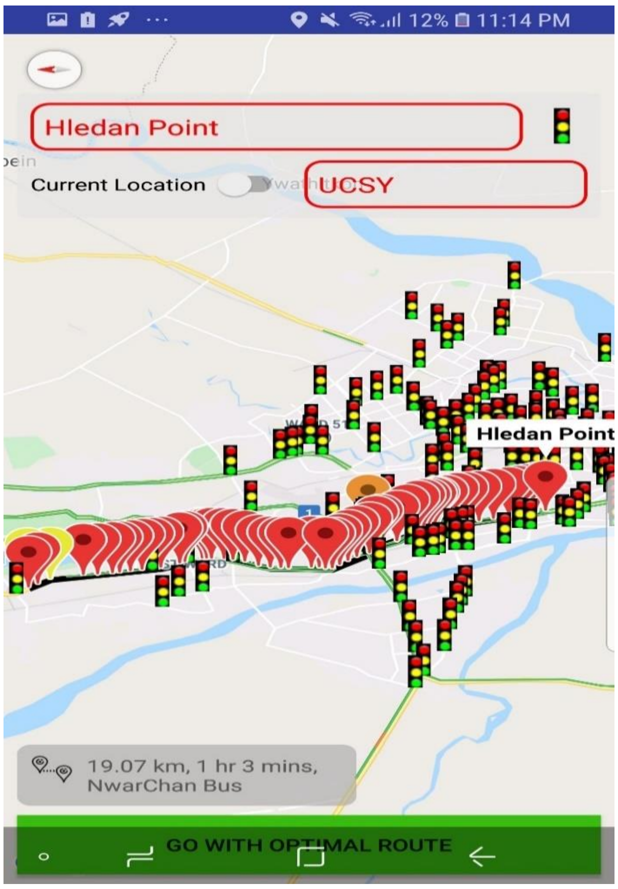


Figure 5.12 Map of Optimal Route Information by User Desired Place

5.2 Advantages and Features of The Application

This application is GPS Navigation with live location and user can drive with Map and Traffic less. This navigation application helps user to track all visited locations on map with time. And it tracks user mobile device location on map where ever user go its keep on updating the locations. It also maintains history of visited locations. GPS route locator helps user to find driving directions from user current location on map to destinations.

There are many features of the application as following:

(i) Easily track all places user visited in Yangon

(ii) Track location on map and take history of locations

(iii) View clear and simple Map

(iv) Drive directions accurately

(v) Take the best accurate location tracker on map

(vi) Find address and locations on map with saving time

(vii) Support different map view like optimal route and Traffic-light status

5.3 Summary

This chapter focuses on the experimental results of index queries, modified A* algorithm, and the system provides the user for traffic avoiding and optimal routing with minimum distance and time. The nearest routes information can be retrieved by Haversine method. This is a simple calculation like a distance-based formula. However, large mobile location data can be time-consuming, because each object in the dataset must be searched sequentially. Sorting data before the search query is appropriate for unbalanced index structures and highly maintained the GPS locations. Therefore, the proposed system pre-assorted neighbor index KD tree structure supports powerful range lookup queries and nearest neighbor queries. In particular, it is important to easily use indexing and traffic status for optimal route to user. In addition, Haversine method supports the nearest location, so it can easily send results to mobile users who are closest to the desired service area. According to the experimental results, the proposed system also provides the optimal routing system with reducing of runtime and processing time. The KD tree, that relevant for finding the range of GPS positions. In addition, the proposed RPA algorithm offers a nearer search radius with modified Haversine and A* algorithm that is more impressive and closer than the distance-based method. Moreover, the system selects the appropriate index structure and procedures that routing pattern algorithm and traffic level have optimized for availability and efficiency of optimal routing. Then chapter 6 will conclude with the summary of the thesis and future research.

CHAPTER 6

CONCLUSION AND FURTHER EXTENSION

In this thesis, Avoiding Traffic Congestion and Optimal Routing (ATCOR) system presented to reduce the time consuming by avoiding congested roads. The road traffic congestion problems faced in our everyday life mostly in urban area. If the current traffic congestion of the desired road segment at the starting point and destination known, the system can avoid congestion. Therefore, the congestion state of road traffic is avoided in the system.

Traffic Control and Management was a major concern to the Government in Yangon, road users and the Traffic Department of the Yangon Police. In the current times, Yangon has been experiencing extremely heavy traffic especially towards Central Business District (CBD). This discussed this is as a result of the developed in the number of mobile in vehicles especially private cars and public vehicles that are heavy load than the roads can accommodate. Another, the manual systems of traffic control and analysis of intelligent traffic lighting systems on major roundabouts are very weak for Yangon transportation. It was also contributing to congestions and traffic jams all over the Yangon region. These are the motivations to do the research for less traffic congestions on the Yangon roads. Moreover, other motivations were weakness and disadvantages of some navigation system for our unity. The system introduced the about of that in Chapter 1 by Introduction.

The chapter 2 had five review collections: There were Comparative Studies, Traffic Detecting and Optimal Routing Systems, Methods for Traffic Estimation, Traffic Considerations in Real-time People Transportation and Recover Traffic Jam through an Intelligent Routing System in this chapter. This is the related works and literature review for our research. It defied the difference of traffic concepts, API and works in comparative studies. This explained technique of Android & Google Maps API, Google Map Traffic, and Detecting Traffic Flows from GPS Signals. This chapter discussed the many methods for traffic estimation: Fuzzy logic, Personal navigation assistants and map-matching, Topological network-based algorithms and, K-D Tree Algorithm and so on. This chapter also includes revisions to location-based services (LBS), moving object structures, synthetic moving objects, and traffic declaration systems. Presents the challenges of location-based services (LBS) and discusses location providers and location update strategies. He also reviewed literature review and work on using policies and updating information. In this chapter explained KD tree index structures with related work.

The thesis was performing the each of chapters about the introduction, objectives, background theories, using methods and technologies, formula to solve the traffic problem, implementation the project and finally analysis result. In these the chapter 3 is reached, this chapter presented map matching included dynamic traffic routing, the role of GIS and LBS and the architecture of navigation service and the data structure by KD tree. The invention of this study was to solve the problem of dynamic routing. Drive the car through the city's road network, using the fastest route, given the traffic conditions. This explained that, in order to show the results of querying mobile clients in an available time, efficient algorithms are needed to quickly retrieve the necessary routing information. Thus, it can accommodate a large number of mobile (customer) users. The chapter presented the scope of databases with moving objects. The KD tree procedures and the pre-classification interval tree are explained and discussed as the procedures used to compare the proposed index tree for performance evaluation. Index approaches are described, especially for the nearest neighbor techniques and two types of range sizes. Secondly, the shortest classification of the problem (SP) presented in this chapter. Explains the classic short path algorithm for short path search algorithms such as static networks, Dijkstra algorithms and A* algorithms. Then, it covers the relationship between cloud technology and mobile devices, followed by cloud database services. And then the system described the detailed process of structure-based methods in indexing technologies. Finally, this chapter discusses virtual machine databases and cloud computing systems, as well as mobile cloud computing.

Chapter 4 presents a detailed process for estimating road traffic congestion systems. Although there are other traffic avoiding systems that can estimate travel time and vehicle density, there are enough sensors on the road network to get complete information and accurate. Because it doesn’t have enough sensors, and use mobile phone GPS data on the vehicle to avoid traffic congestion. It also addresses the uncertainty of GPS data that may not match when GPS data is matched to a road segment to determine the location of the vehicle. The system uses historical traffic data and real-time traffic data as well as hidden Markov models to estimate traffic congestion on each street segment. Cloud computing is used as a back-end server to perform these processes with better performance and reduce processing time. Another work in this chapter presented the wizard for customers to use the application is the Guide for User, called "How to Use".

Chapter 5 presented the results of the assessment of traffic status. According to the evaluation results, the estimation error is higher every minute, and the accuracy of the avoiding congestion segment depends on the acquired data. If real-time data is not accessible when estimating traffic status, that will reduce accuracy. This chapter focuses on the experimental results of index queries and the closest time to execute a query on a mobile object. Because the system requires a sequential search by each object in the dataset, large amounts of location data can take too long. Therefore, these queries can be easily recovered using Haversine expressions, which are simple calculations, such as distance-based expressions. Preprocessing before the query is suitable for an unbalanced index structure with many mobile objects.

Thus, the KD index tree structure receives strong search requests and closest requests. In particular, it is more relevant to use cellular object indexing and easily search for traffic status. In addition, he accepts to get the closest location from Haversine, so the notification results can be easily sent to mobile users who come from the closest location in the desired service area. According to the results of the experiment, the proposed tree structure of the proposed location gives better results: taking implementation time and less processing. The proposed index tree, the KD tree, is relevant for searching across mobile locations. In addition, the proposed index structure provides impressive ranges and closer neighbors than distance-based methods. In addition, the system selects the appropriate structure and indexing procedures that the routing pattern algorithm and Google traffic layer have optimized for optimal route availability and efficiency. Moreover, Chapter 6 concludes with an abstract of the proposed system, conclusions, and future work.

6.1 Summary

The system looks for the traffic congestion area and selects the optimal route based on the current location and time. Algorithm A* is a graph search algorithm that finds the best path from a specific initial first node to a specific target node in the mapped area. Modified A* is suitable when both the origin and destination are known to determine the best route. In addition, it provides users with accurate maps with more efficient routing results and saves more time designing upon other causes of the database structure. Because the accuracy of the system depends on the available data, this road congestion avoiding system is the most useful tool when sufficient accuracy data is provided. And then GPS service is becoming the standard function of mobile phones, and with the spread of mobile phones to the population, there is a high possibility that sufficient data will be active in the coming future, so the proposed method is intended to the traffic avoiding and optimal routing.

In addition, this system not only displays the results of traffic jam conditions at the source and destination of all interested areas in Yangon, but also shows the optimal route for teal-time. Most people can use this application to easily avoid traffic jams and avoid traffic jams. The system gives the detail information of traffic on the routes such as duration, distance and locations data and information of Traffic-Lights in Yangon City. Moreover, User Interface Design is designed in an appropriate way that up to date information. The system has to load and analysis the datasets in the whole of country after processing of current research area (Yangon city). Therefore, this system can extend for road segments with many lanes. This system proposes detecting and avoiding the road traffic congestion system that shows the traffic congestion states of user on Google Map.

This search includes dynamic attributes that represent floating position coordinates, but they can be used on other hybrid systems that have dynamic attributes. The proposed system provides the user with minimum distance and time for the route information of their desired places. This system is excellent of better server-side applications that can automatically detect based on geographic analysis. In addition, this system consists of mobile cloud services, therefore it exists as a permanent network that maintains thousands of connections for each client application between server and user device.

This system shows efficient method for traffic avoiding system of Yangon City in Myanmar Nation. The system shows the traffic congestion areas of user's desired road segment of current time. Mobile phone with GPS-enabled can help their position and velocity accurately. The system used methods that are Speed of traffic and routing pattern algorithms and Haversine methods. Software Requirements are QGIS and eclipse (JAVA), Android Studio, Map API and so on. Hardware requirements are computer/Laptops GPS Devices/ GPS enabled mobile phones and GPS Server.

An excellent system for storing and managing these types of data is the Geographic Information System (GIS). GPS tracking data obtained from a mobile phone, tablet or desktop computer. View cars as they move in real time, or use the historical playback feature, which provides constant access to vehicle travel history. In the proposed system, GPS tracker installed on each selected vehicle in the road network. In the recent years, Yangon has been experiencing extremely heavy traffic especially towards the City downtown and Central Business District. This is because of the upgraded in the number of mobile in vehicles especially private cars and Public Service Vehicles hence heavy load than the roads can accommodate. Furthermore, the manual systems of traffic control and analysis together with lack of intelligent traffic lighting systems on major roundabouts.

The congestion is directly related to the dynamic interaction between demand and traffic volume. If traffic demand is high and road capacity is low, congestion may occur. Variables such as speed, strength or density can be used to measure traffic conditions. In order to successfully real-time traffic congestion, information about past and current network status is required.

In the system, the control server receives GPS data via GPS tracker and matching and calculate optimal route in the system to show result mobile user. Moreover, it also provides the accurate map for more efficient results for traffic state from GPS data and saving more time other. In the recent years every person uses smartphones mostly. Every smartphone come equipped with GPS to manage route applications. Therefore, human activities can be represented well the result with GPS data. Users need to get the valued facts (travel time, route distance and speed of vehicle) for not delaying and congest to reach their goals.

6.2 Further Extension

The system is implemented for location-based services that automatically send data by the application server. It should take into consideration the reconstruction of any platform and building any database for route information of historical GPS data in the upper Myanmar. The proposed system focused in the Yangon city only but this is recommended in developed cities or Mandalay for further extension.

LIST OF ACRONYMS

3D	Three-Dimensional
2D	Two-Dimensional
A*	A Star model
ATCOR	Avoiding Traffic Congestion and Optimal Routing
API	Application Programming Interface
ATM	Automatic Teller Machine
CBD	Central Business District
CPU	Central Processing Unit
CNN	Condensed Nearest Neighbor
DBMS	Database Management System
DO	Digital Ocean Technology
GAE	Google App Engine
GIS	Geographic Information Systems
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
ID	Identification
ITS	Intelligent Transportation System
J2SE	Java 2 Standard Edition
J2EE	Java 2 Enterprise Edition
JDK	Java Development Kit
JSON	Javascript Object Notation
KD	K Dimensional
KNN	K Nearest Neighbor
KDNN	K Dimensional Nearest Neighbor

LA	Location Areas
LBS	Location-Based Service
LC	Label Correcting
LS	Label Setting
MACID	Media Access Control Identification
MBR	Minimum Bounding Rectangle
MCC	Mobile Cloud Computing
MIS	Mobile Information System
MOD	Moving Object Database
NN	Nearest Neighbor
OTDOA	Observed Time Difference of Arrival
RAM	Random Access Memory
RF	Radio Frequency
RFID	Radio Frequency Identification
RNN	Reduced Nearest Neighbor
PSV	Public Service Vehicles
SDK	Software Development Kit
SQL	Structured Query Language
SP	Shortest Path
TIS	Traffic Information System
TMS	Traffic Management Centers
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
VM	Virtual Machine
VSL	Variable Speed Limits

WKNN	Weighted K Nearest Neighbor
WIFI	Wireless Fidelity
XAMP	Stack for web server Apache, MariaDB and PHP
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

AUTHOR'S PUBLICATIONS

[P1] Nyein Chan Soe and Thin Lai Lai Thein, University of Computer Studies, Yangon Myanmar, “**An Intelligence Optimal Routing System for Transportation**”, ICCA 2017, International Conference, University of Computer Studies, Yangon Myanmar.

[P2] Nyein Chan Soe and Thin Lai Lai Thein, University of Computer Studies, Yangon Myanmar, “**Information of Traffic Congestion Area in Urban Transportation (Yangon)**”, CSTD 2017, Defence Services Academy, Pyin Oo Lwin, 1st November 2017.

[P3] Nyein Chan Soe and Thin Lai Lai Thein, University of Computer Studies, Yangon Myanmar, “**Routing and GPS Tracking System in Urban Transportation (Yangon)**”, ICCA 2018, International Conference, University of Computer Studies, Yangon Myanmar.

[P4] Nyein Chan Soe and Thin Lai Lai Thein, University of Computer Studies, Yangon Myanmar, “**An Efficient method of Optimal Routing System for Transportation (Yangon)**”, 18th ISCIT 2018, Bangkok, Thailand, September 2018. #IEEE, Q3 index SCI Conference

[P5] Nyein Chan Soe, Thin Lai Lai Thein and Thida Aung, University of Computer Studies, Yangon Myanmar, “**GPS Tracking and Traffic Monitoring System in Urban Transportation**”, GCCE 2018, Japan. #IEEE, Q2 index SCI Conference

[P6] Thiri Yadanar and Nyein Chan Soe, WYTU(IT) and University of Computer Studies, Yangon Myanmar, “**The Opinion Mining from Social Media by Using Support Vector Machine(SVM) Algorithm**”, International Journal of Science, Engineering and Technology Research, India, Volume7, Issue9, September 2018.

[P7] Nyein Chan Soe and Thin Lai Lai Thein, University of Computer Studies, Yangon Myanmar, “**Intelligence Routing Plan using K-d tree and Modified-A* with Google Map Data**”, ICCA 2019, International Conference, University of Computer Studies, Yangon Myanmar.

[P8] Nyein Chan Soe and Thin Lai Lai Thein, University of Computer Studies, Yangon Myanmar, “**Haversine Formula and RPA Algorithm for Navigation System**”, PAPER ID: 3671089, International Journal of Data Science and Analysis (IJDSA) Science Publishing Group, (ISSN Print: 2575-1883; ISSN Online: 2575-1891).

BIBLIOGRAPHY

[1] MD. Al Amin, MD.Rofi, Supervised by Mrs. Sadia Hamid Kazi, ‘Real Time Traffic Monitoring System Using Crowd Sourced GPS Data’

[2] Carola A. Blazquez Universidad Andres Bello Department of Engineering Science Chile,”A Decision-Rule Topological Map-Matching Algorithm with Multiple Spatial Data”.

[3] Cory A. Brose Saint Mary's University of Minnesota, Graduate Studies in Resource Analysis, 700 “Geographic Information Systems for Spatial Analysis of Traffic CollisionLocations in La Crosse, Wisconsin”, Terrace Heights #10; Winona, Minnesota, 55987, USA.

[4] F. Cannavò, G. Nunnari, D. Giordano, C. Spampinato, “Variational Method for Image Denoising by Distributed Genetic Algorithms on GRID Environmentc”. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE '06. 227-234, IEEE, 2006.

[5] A. Costanzo, A. Faro, D. Giordano, M. Venticinque, “Wi-City: A federated architecture of metropolitan databases to support mobile users in real time,” IEEE Proc. Int. Conf. on Computer & Information Science (ICCIS), Vol.2, 586-591, 2012.

[6] A. Costanzo, A. Faro, D. Giordano, “Wi-City: living, deciding and planning using mobiles in Intelligent Cities,” 3rd International Conference on Pervasive and Embedded Computing and Communication Systems, PECCS, Barcelona, INSTICC, 2013

[7] Alfio Costanzo, B. Herrei, Department of Electrical, Electronics and Computer Engineering, University of Catania, “Using GPS data to detect road traffic flows in a metropolitan area: methodology and case study”, viale Andrea Doria 6, Catania, 95125, Italy

[8] A. Crisafi, D. Giordano, C. Spampinato, “GRIPLAB 1.0: Grid Image Processing Laboratory for Distributed Machine Vision Applications”. Proc. Int. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE '08. 188-191, IEEE, 2008.

[9] M. David, “Developing Websites with jQueryMobile,” Focal, 2011.

[10] ESRI, 380 New York Street Redlands, California 92373-8100 usa, “GIS

Solutions for Highway and Roadway Management", Copyright © 2011 Esri. All rights reserved. Esri.

[11] A. Faro, D. Ethan, "Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection". IEEE Transactions on Intelligent Transportation Systems, Vol.12(4), 1398 - 1411, 2011.

[12] A. Faro, D. Giordano, C. Spampinato, "Evaluation of the Traffic Parameters in a Metropolitan Area by Fusing Visual Perceptions and CNN Processing of Webcam Images," IEEE Transactions on Neural Networks, Vol. 19 (6), 1108-1129, 2008.

[13] A. Faro, C. Spampinato, "Integrating Location Tracking, Traffic Analyzing and Semantics in a Layered ITS Architecture. Intelligent Transport Systems," vol.5(3), 197-206, 2011.

[14] A. Faro, E. Zaha, A. Musarra, "Ontology Based Mobility Information Systems". Proc. of IEEE Systems, Men and Cybernetics SMC03, vol.3, 4288-4293, 2003.

[15] S. Fischer, H. Koorapaty, E. Larsson, and A. Kangas, "System performance evaluation of mobile positioning methods," in IEEE Vehicular Technology Conference, 1999, vol. 3, pp. 1962–1966.

[16] Borko Furht, "Handbook of Cloud Computing – 2010, Cloud Computing Fundamentals", Return to Bookmetrix summary, Handbook of Cloud Computing, About Affiliation, DOI -10.1007/978-1-4419-6524-0_1

[17] D. Giordano, A. Faro, "Wi-City: Efficient Strategy with using mobiles in Intelligent Cities," 3rd International Conference on Pervasive and Embedded Computing and Communication Systems, PECCS, Barcelona, INSTICC, 2013

[18] Cheng Hian Goh*, Hongjun Lu*, Department of Information Systems and Computer Science, National University of Singapore, "Indexing temporal data using existing B +-trees", Data & Knowledge.

[19] M. Wylie-Green and P. Wang, "GSM mobile positioning simulator," in IEEE Emerging Technologies Symposium: Broadband, Wireless Internet Access, 2000, pp. 874–878.

[20] Yangon Region Government, Yangon City Development Committee;

YCDC, Japan International Cooperation Agency; JICA, JICA Study Team, "Yangon 2040 The Peaceful and Beloved Yangon —A City of Green and Gold—", The Strategic Urban Development Plan of the Greater Yangon March, 2013.

[21] M. Hartl, "Ruby on Rails 3," Addison Wesley, 2011.

[22] Ryan Jay Herring, the degree of Doctor of Philosophy in Industrial Engineering and Operations Research in the Graduate Division of the University of California, "Real-Time Traffic Modeling and Estimation with Streaming Probe Data using Machine Learning", Berkeley Committee in charge: Professor Laurent El Ghaoui, Chair Professor Zuo-Jun Shen Research Advisor, Professor Alexandre Bayen Fall 2010

[23] Pedro Holanda1, Matheus Nerone2, Eduardo C. de Almeida2and Stefan Manegold11CWI, Amsterdam - The Netherlands2UFPR, Curitiba , "Cracking KD-Tree: The First Multidimensional Adaptive Indexing(Position Paper)" - Brazilholanda@cwi.nl,{manerone,eduardo}@inf.ufpr.br, manegold@cwi.n

[24] Ke Huang, University of Massachusetts Lowell Computer Science Lowell, "Real-time Transportation Route Selection with Traffic Considerations," MA 01854, khuang@cs.uml.edu

[25] Jan Jannink, Stanford University, Computer Science Dept. Stanford, "Implementing Deletion in B+ Trees," CA 94305.

[26] Kun Mande, Stanford University, Computer Science Dept. Stanford, "B+ Trees for index structure," CF 56605.

[27] Celso Goncalo Dias Kulir, GIS/RS Rresearcher, State Office of Public Safety, DETRAN- State Vehicles Trqaffic Department, "Vehicles Traffic Management and GIS Planning System- A Study if DETRAN Project- Parana State /Brazil", Av. Visitor Ferreia do Amaral, 2940-ZIP 82800 900, Curitiba, PR, Brazil, International Achieves of Photogrammetry and Remote Sensing. Vol. xxxi, Part B4, Vienna 1996.

[28] G. Leduc, "Road Traffic Data: Collection Methods and Applications, Working Papers on Energy, Transport and Climate Change," N.1, JRC European Commission, 47967, 2008.

[29] H. T. Lwin, T. T. Naing, University of Computer Studies, Yangon,

Myanmar, Professor, Software Department, University of Computer Studies, Yangon, Myanmar “Estimation of Road Traffic Congestionusing GPS Data”, International Journal of Advanced Research in Computer and Communication EngineeringVol. 4, Issue 12, December 2015.

[30] Dr. Mohammed Otair, Department of Computer Information Systems, Amman Arab University, Amman, Jordan, “Approximate KNearest Neighbour Based Spatial Clustering Using K-D Tree”, International Journal of Database Management Systems (IJDMS) Vol.5, No.1, February 2013.

[31] Ben Pfaff, "An introduction to binary search tree and balanced tree" label binary search tree library, volume 1: source code, version 2.0.3, free software foundation, inc, (2004).

[32] Shekhar K. Rahane, Prof. U. R. Saharkar , Pg student in Civil Engineering (Construction and Management), Dr. D. Y. Patil Institute of Engineering and Technology Ambi, University of Pune, Maharashtra, India, “Technique Identification For Road Traffic Congestion Solution In Talegaon Dabhade State Highway- 55”, Journal Of Information, Knowledge And Research In Civil Engineering ISSN: 0975 – 6744| NOV 13 TO OCT 14 | Volume 3, Issue1.

[33] V.L. Sauter, “Decision Support Systems for business intelligence,” Wiley, 2011.

[34] V.L. Sauter, M. Rashfold, “Decision Support Systems for IOTs,” Wiley, 9.8.2011.

[35] W. Shi, Q-J. Kong, Y. Liu, “A GPS/GIS Integrated System for Urban Traffic Flow Analysis,” Proc. of the 11th International IEEE Conference on Intelligent Transportation Systems Beijing, China, 2008.

[36] Anitha Selva Sofia, Nithyaa.R2, Prince Arulraj.G3 Assistant Professor, Department of Civil Engineering, SNS College of Technology, Coimbatore, Tamilnadu, India, “Minimizing the Traffic Congestion Using GIS”, IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013, ISSN: 2320 – 8791 www.ijreat.org

[37] Allan M. de Souza1, Roberto S. Yokoyama1,3, Institute of Computing, University of Campinas, Brazil, “Real-Time Path Planning to Prevent Traffic Jam Through an Intelligent Transportation System,” 2016 IEEE

Symposium on Computers and Communication (ISCC)

[38] Henry Stern, Partial Fulfillment of The Requirements For The Degree Of Bachelor Of Computer Science, “Nearest Neighbour Matching Using KD-Trees”, Dalhousie University Halifax, Nova Scotia August 2002

[39] S. Tao, V. Manolopoulos, A. Rusu, “Real Time Urban Traffic State Estimation with A-GPS Mobile Phones as Probes,” Journal of Transportation Technologies, N.2, 22-31, 2012.

[40] S. Tao, S. Rodriguez, “Urban Traffic Estimation Using LBS System, Journal of Transportation Technologies, 2014.

[41] SHA TAO, A. SURU, Licentiate Thesis KTH Royal Institute of Technology Stockholm, “Mobile Phone-based Vehicle Positioning and Tracking and Its Application in Urban Traffic State Estimation”, Sweden 2012

[42] Qiang Wu, UNIVERSITY OF CALGARY Incremental Routing Algorithms For Dynamic Transportation Networks, Department of Geomatics Engineering, “Incremental Routing Algorithms For Dynamic Transportation Networks” (URL:http://www.geomatics.ucalgary.ca/research/publications/GradTheses.html) January 2006

[43] J. Zhai, J. Jiang, Y. Yu, J. Li, “Ontology-based Integrated Information Platform for Digital City,” IEEE Proc. of eWireless Communications, Networking and Mobile Comp., WiCOM '08, 2008.

[44] Ke Zhang, Guangtao Xue, Shanghai Jiao Tong University, Shanghai, China, “A Real-Time Urban Traffic Detection Algorithm Based on Spatio-Temporal OD Matrix in Vehicular Sensor Network*”, doi:10.4236/wsn.2010.29080 Published Online September 2010.

[45] "About Maps API," [Online]. Available: https://en.wikipedia.org/wiki/Google_Maps#Google_Maps_API.

[46] "Maps API documentation," [Online]. Available: https://developers.google.com/maps/documentation/android/.

[47] "Android - Statistics & Facts," Statistia, [Online]. Available: http://www.statista.com/topics/876/android/.

[48] https://en.wikipedia.org/wiki/Haversine_formula

[49] https://en.wikipedia.org/wiki/K-d_tree

[50] https://en.wikipedia.org/wiki/K-d_tree

[51] https://www.movable-type.co.uk/scripts/latlong.html

[52] https://developers.google.com/maps/documentation/javascript/layers