Real Time Traffic Monitoring System Using Crowd Sourced GPS Data

by

MD. Al Amin

MD. Rofi Uddin

Supervised by

Mrs. Sadia Hamid Kazi

Abstract

There has always been the necessity of accurate and real time traffic information among the commuters and drivers of big cities. In present times, with the increased use and availability of GPS enabled smartphones, a traffic monitoring system based on cell phone GPS data is highly practical. Vehicles equipped with a smartphone application driving through the traffic of different roads can generate useful information, for example vehicle speed, geo location and information regarding the road. SO we have developed an application using these data, so that it can be sent back to a web service and stored in a database. Later based on these information, a color coded map can be generated that reflects the near real time traffic condition of a city at any given time to any user. Our traffic monitoring system based on this principle requires less physical maintenance, has faster deployment capability and potentially can monitor a large area.

Contents

Part 1: Introduction

1.1 Background

Motorized transportation system is undoubtedly one of the biggest revolutions of twentieth century, and therefore, in modern times. It made commuting from one place to other an easy, convenient yet a controlled mechanism with its ever so complex evolution all over the world. However, as the number of vehicles increased and traffic systems became more and more complex, the necessity of monitoring it clearly became important and crucial. To address this need, various researches have been done throughout the past few decades and numerous systems were developed and deployed. Many countries have started implementation of such systems from quite early on and have made them an integral part of their overall urban planning[1]. However, as effective and operational as they are, some of them are heavily physical infrastructure dependent, thus quite costly[1]. As a result, many of the traditional means are difficult to deploy in developing countries in a large scale and versatile manner. In recent times however, attempts are being made to utilize more multi-purpose and versatile platform, such as smart phones and devices, to gather information in large scale and portray a more complete image of the traffic conditions on the road from there[2].

1.2 Objective

GPS or Global Positioning System is a relatively new technology. Even though around the time of its invention it was primarily used for military purposes, later this technology saw itself in use with numerous civilian applications[3]. In the recent years every smartphones comes equipped with GPS and location services primarily that of Google, to complement and manage applications like Google Maps. Thus human activity can be represented well than ever with GPS data. As a result we saw a tremendous amount of utilization location based apps and services in smartphones in the last few years. Given the fact that a large number of motorists or even commuters are smartphone users, the traffic scenario on the road can also be represented using GPS data. This coupled with the fact that even in developing countries, now a days an ever so increasing number of people are using GPS equipped smartphones[4], a traffic monitoring system based around the concept of collecting and sorting highway specific location data transmitted from the mobile devices and then using those data to represent the amount of traffic present on a specific highway can result in a very practical and versatile traffic monitoring system. Therefore, the objective of this thesis is to develop a real time traffic monitoring framework consisting of a mobile application and a backend web service suitable for rapid and efficient deployment in a country like Bangladesh. For this thesis we will also study different traffic monitoring systems in use that uses similar concepts. We will also study and develop necessary algorithm and workflows for transmission of data back and forth between user-end and server. Moreover, we will investigate the potential drawbacks and scopes for improvements in such a system.

Part 2: Literature Review & Case Study

2.1 Comparative Studies

As stated in the introduction, in the past few decades, there have been several iterations and implementations of both traditional traffic monitoring systems, whereas in recent times, we saw a few attempts to implement revolving around the concept of crowd-sourcing and location data extraction[5]. So, in order to approach the problem and analyze what would be a feasible way to solve it, we need to study the existing systems and how they work.

2.2 Beat the Traffic

“Beat the Traffic” is a provider of vehicle traffic reporting solutions for broadcast media and consumers. With a focus on daily commuters, Beat the Traffic was founded in 2001 by former CEO Andre Gueziec, and now operated by Pelmorex, the parent company of The Weather Network.Headquarters are located in Oakville, Ontario Canada[6].

Fig 1: Beat the Traffic logo[6]

The end service of this company is primary website and mobile applications for both Android and iOS platforms that provides the users with real-time traffic information at any given time. The features of their application/website include:

- Color coded traffic map showing areas of vehicle congestion in the road.
- Live traffic cameras
- Incident reporting system

Besides providing with real time information about traffic situation to the mobile application users, several TV channels and cable networks in US and Canada uses information feed from "Beat the Traffic" to broadcast traffic updates[3].

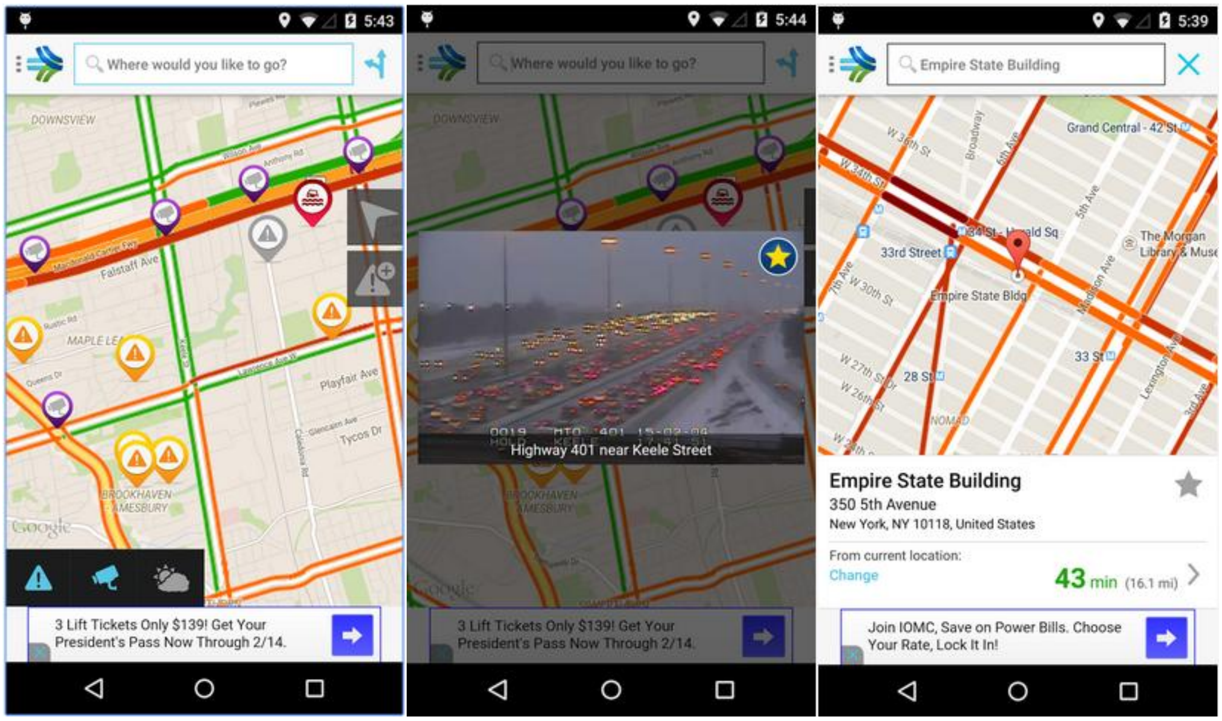

Where would you like to go?
Highway 401 near Keele Street
Empire State Building
Empire State Building
350 5th Avenue
New York, NY 10118, United States
From current location:
Change
43 min (16.1 mi)
3 Lift Tickets Only $139! Get Your President's Pass Now Through 2/14.
Join IOMC, Save on Power Bills. Choose Your Rate, Lock It In!

Fig 2: Screenshots of Beat the Traffic Android app[7]

There are several data sources that the company uses to generate live traffic update. Traditional sources include data providers of US and Canada, traffic cameras, public and private agencies and their own reporters[8]. However, a growing share of their information continues to come from Beat the Traffic users, through a method named "crowdsourcing". This method adds the benefit of data anonymously gathered from phones and navigation systems in cars currently on the roads and highways both in active and passive way. Users can also participate in the process of data collection while sending manual reports. In this fashion, users have been able to report accidents on their routes, hazards, constructions zones, police controls and weather conditions. Compiling information from public and private sources, crowd-sourced content coupled with its own reporting teams, Beat the Traffic provides real-time traffic information and maps for most major cities in the US and Canada.

With more than 600,000 active users, Beat the Traffic app has been named as one of the top five apps for commuter in North America by Lifehacker and CraveOnline. However, their service in only limited to Canada and United States[8].

2.3 Waze

Waze is a GPS-based geographical navigation application program for smartphones with GPS support and display screens which provides turn-by-turn information and user-submitted travel times and route details, downloading location-dependent information over mobile networks. Waze was developed in Israel, funded by early-stage American venture capital firm Bluerun Ventures, and was acquired by Google in 2013[9].

Fig 3: WAZE logo[9]

The way waze is different from other of similar type of apps is, that it is community-driven, gathering complementary map data and traffic information from its users. Like other GPS software it learns from users' driving times to provide routing and real-time traffic updates. On top of that it also support manual reports. People can report accidents, traffic jams, speed and police traps, and from the online map editor, can update roads, landmarks, house numbers, etc. As of January 2012, the app had been downloaded 12 million times worldwide. According to Yahoo! there were nearly 50 million Waze users as of June 2013[10].

Waze can be used anywhere in the world but it requires a critical mass of users to have real utility. Currently only 13 countries have a full base map, the others are incompletely mapped, requiring users to record roads and edit maps. As of 2013 Waze has a complete base map for the United States, Canada, United Kingdom, France, Germany, Italy, Netherlands, Belgium, Israel, South Africa, Colombia, Ecuador, Chile and Panama[10]. This indicates that a customized map is necessary to make such an app useful anywhere in the world, this making it not as effective

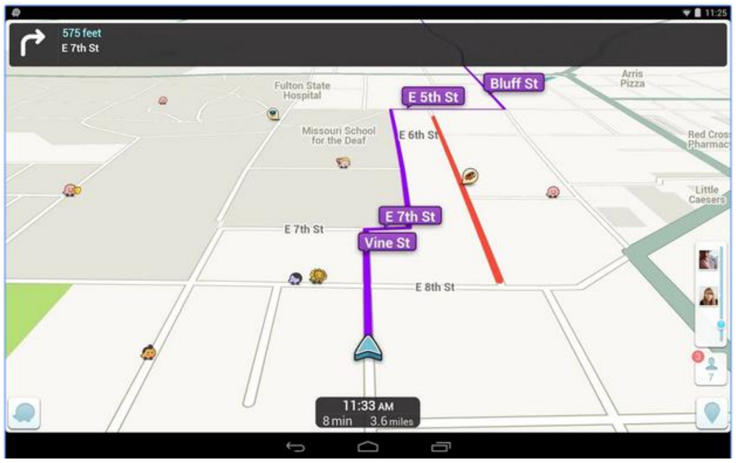


Fig 4: Screenshot of the WAZE Android app[11]

Where there is not much information to start with. Nonetheless, the company has plans to complete maps for other countries in Europe and elsewhere.

2.4 Google Traffic

Google Traffic is a feature on Google Maps which displays traffic conditions in real-time on major roads and highways. Google Traffic can be viewed at the Google Maps website, or by using the Google Maps application on a handheld device[12].

Early versions of Google Maps provided information to users about how long it would take to travel a particular road based on the historic data. This information was not real time and far from accurate. In 2004 Google acquired ZipDash, a company specializing on realtime traffic analysis.In 2007, Google integrated ZipDash's technology into Google Maps, offering traffic data, based on information gathered anonymously from cellular phone users[13].

Google maps also works on the basis of the concept of crowdsourcing, which refers to the process of soliciting electronic information from a large group of people.Google stated: "When we combine your speed with the speed of other phones on the road, across thousands of phones moving around a city at any given time, we can get a pretty good picture of live traffic conditions"[13].

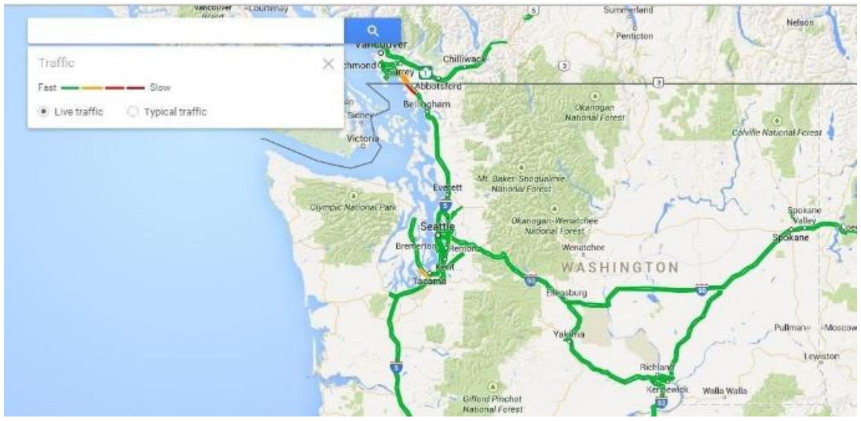


Fig 5: Screenshot of Google maps desktop version[12]

Google Traffic works by analyzing the GPS-determined locations transmitted to Google by a large number of mobile phone users. By calculating the speed of users along a length of road, Google is able to generate a live traffic map. Google processes the incoming raw data about mobile phone device locations, and then excludes anomalies such as a postal vehicle which makes frequent stops. When a threshold of users in a particular area is noted, the overlay along roads and highways on the Google map changes color.

Currently, Google traffic is available in around 50 countries[13]. Unfortunately, it is not available in Bangladesh.

2.5 GO Traffic

Go Traffic is a startup in Bangladesh that aims to provide real time traffic monitoring for Dhaka. The work procedure of their system is similar to the above mentioned systems in a few ways. They have an android application which allows the users to both monitor and report traffic on a particular highway. The main principle of their app is also based on the concept of crowdsourcing. They maintain a database of roads in Dhaka and the application allows the user to submit a manual report at any given time. Then all the reports for any particular road are taken into account at any given time and adds up to the condition of that particular road. Also their team updates these information for a portion of the day manually[11].

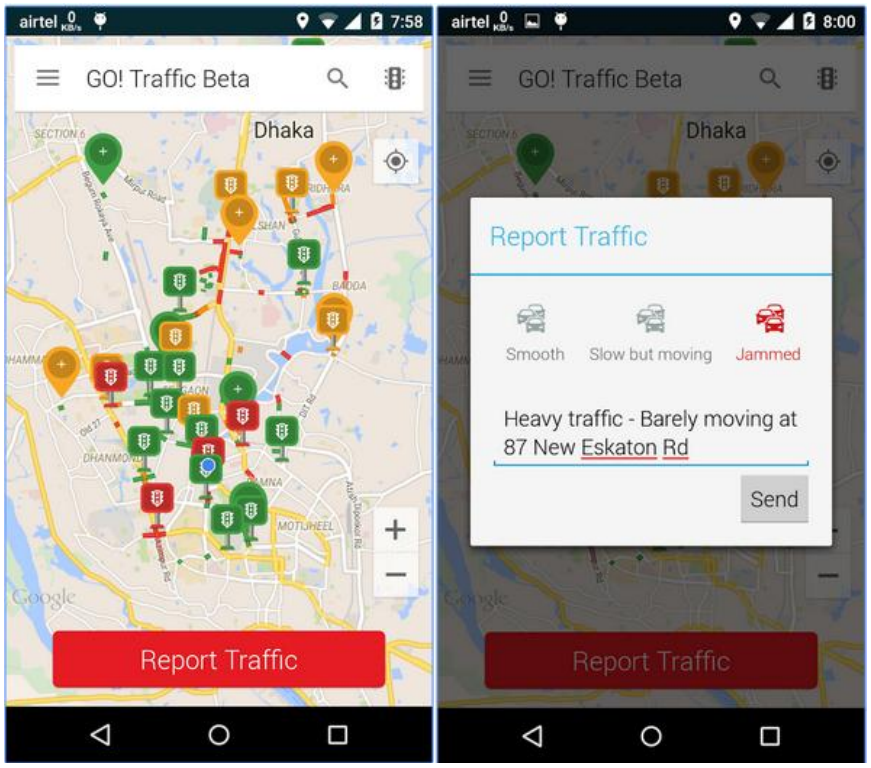


Fig 6: Screencapture of GO Traffic android application[14]

While GO traffic is a very good example of what can be a potentially very good traffic monitoring system, there are some drawbacks to their current iteration of the application. The app lets its user report a location from anywhere and it’s up to the user to inform the system how the traffic condition is like at any given time. Then the system tries to associate that information with the most nearby road and update the condition of that road accordingly. This opens up a potential opportunity of spamming the system.

2.6 Crowd Sourcing: An optimal way

A comparative study of the existing smartphone based traffic monitoring systems shows that cellphone location can and is being used as an effective mean of traffic monitoring in big metropolitans all around the world. It is also evident that any such system is required to be "crowd sourced"- where large number of users will contribute to the system with information and get the results concurrently[15]. Thus, using location data of cellphones form a considerable number of motorists coupled with the features already available in Google maps API and location services, it is possible to construct a versatile traffic monitoring system. However, the amount of features and information available with Google maps and location service varies from country to country. That is the reason why the Google Traffic feature is not available in Bangladesh. So, we will attempt to figure out those challenges and improvise an effective work methodology to make such a framework possible to implement.

Part 3: Algorithm & Workflow Approach

3.1 Data Crowd Sourcing

In business terminology, crowd sourcing referees to the process of obtaining needed services, ideas, or content by soliciting contributions from a large group of people, and especially from an online community, rather than from traditional employees or suppliers. The term is a combination of the words “Crowd” and “Outsourcing”. This mode of sourcing is often used to divide tedious work between participants, and has a history of success prior to the digital age. By definition, crowdsourcing combines the efforts of many self-identified volunteers, where each contributor, acting on their own initiative, adds a small contribution that combines with those of others to achieve a greater result. Hence, it is distinguished from outsourcing in that the work comes from an undefined public, rather than being commissioned from a specific, named group or individual[16].

The concept of crowd sourcing has been used in Information Technology field very often. The main reason behind this is, it enables a greater reach to the consumers while piggybacking a portion of the work on the user community themselves. Mobile crowdsourcing involves crowdsourcing activities that take place on smartphones or mobile platforms, frequently characterized by GPS technology. This allows for real-time data gathering and gives projects greater reach and accessibility. Some examples of mobile crowdsourcing include TaskRabbit, EasyShift, Gigwalk, and Uber[16].

For a GPS based traffic monitoring system data crowd-sourcing is the best solution for a couple of reasons:

1. It removes the necessity for any physical infrastructures on the road.

2. It uses smartphones which is an already existing technology.
3. By collecting data from the users and using that to give service to them, it is possible to remove the complicacy of creating and maintaining a whole different set of assets to provide the service.

Thus to implement and maintain the proposed traffic monitoring system we have consider the whole system as a crowd sourced entity and design the system accordingly.

3.2 City Highway Graph

A map of a city has many roads and road intersections. The intersection of a road is usually a convergence point for two or more roads. If these intersection points are considered "Nodes" and the roads are considered "Edges", then a city map becomes basically a graph[17].

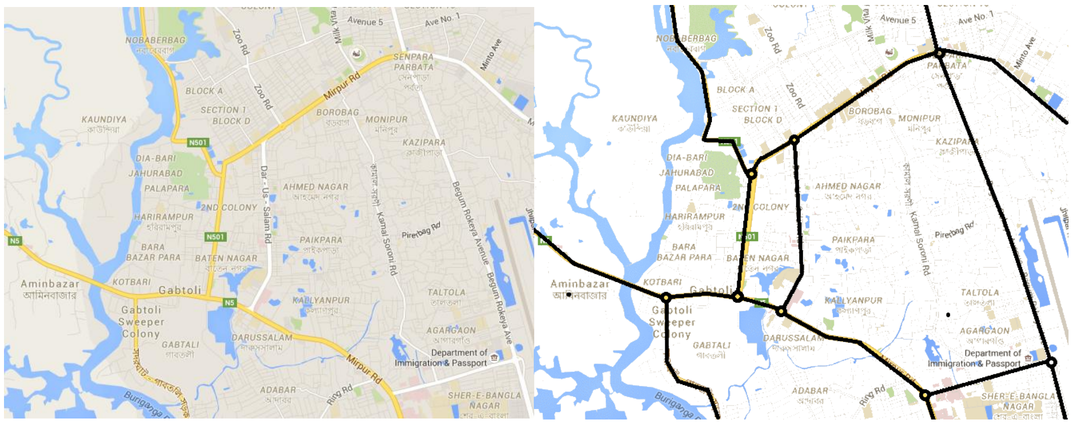


Fig 7: Illustration of a portion of Dhaka city represented as a graph[12]

Now the traffic signals are usually on the both ends of a span of a road. Thus, at any given time, the density of vehicles in between any two nodes in such a city graph represents the amount of traffic on that span of a road. Based on this principle, we will have to maintain a data-structure that can hold both the information about the specific road-spans which will allow the identification of each individual road and also the corresponding information through which traffic density will be determined.

3.3 Metric of traffic measurement

In some previous case studies, we saw that some existing GPS based traffic monitoring apps uses the GPS to only locate the location of the traffic, whereas, they use direct user input to determine the density of traffic. This method is equivalent to a crowd sourced questioner system where the system is only as accurate as the person participating in the poll or submitting the report. This leaves the process vulnerable to spamming the system database with incorrect location data. Also, users are required to manually and actively participate in such information collection process; it may discourage some users to participate at all[18].

A more dynamic way of collecting the necessary data would be an automated process that does the work of collecting, processing and submitting without the active participation of the driver/app user. For this, a metric of measurement has to be used that can be automatically collected from the smart device and easily sent to the backbone system for sorting and calculation. The speed of a vehicle on top of a road can directly be translated as the traffic condition of a road in any given time[19]. Because, speed and traffic density has an inversed relationship with the traffic density of a road. Only one vehicle from one highway can speak for quite a few other vehicles on its surrounding radius. Thus In this approach, even if only one vehicle is reporting its current speed from a particular geo location co-ordinate of a road, it can give us a considerably accurate traffic condition for that span of road. In better case, if there is more than one vehicle, the average speed can give us a better idea of what the traffic condition is like in that span of road.

To illustrate the condition of a road accurately, we also need to set some threshold for a set of speed limit, which will reflect accordingly on the database. For example:

1. 0-15 km/h => Heavy Traffic
2. 16-30 km/h => Medium Traffic
3. Above 30 km/h => Light Traffic

3.4 System Framework Overview

The overall system is mainly divided into two portions, the client side and the server side. The client side is in the form of a mobile application, which has two main functionality modes. Firstly, the **submitter mode**, which is for the purpose of data collection from the "Probe Vehicles". The probe vehicles are the vehicles which are equipped with the app which already in the highways and thus experiencing the road condition they are currently in. In other words, they are probing through the traffic. The submitter aspect of the app continues to communicate with the GPS satellite with the help of Google maps and location services throughout the whole travel time. In a regular interval the submitter collects vital information like geo location, current speed, and current highway thus submits them to the web service. Then the server accepts the automated report and saves them in the relational database for further disposal. The other mode of the client side application is the **monitoring mode**. The monitoring mode can be used by either an user who is not on a highway but just wants to check the condition on the highway, or and user on a highway but want to check the condition of a different highway. Upon the search request of a highway, the application first sends a condition monitoring request to the server. The server side then checks if the highway queried for is a valid highway on which it has information about. Upon finding a highway, information regarding which exists on the server side database, the server fetches the real time information collected from the probing vehiclesup to that point and sends them as a reply to the monitoring app. The monitoring app then decodes that information and illustrates them accordingly to provide the user the information he/she wanted.

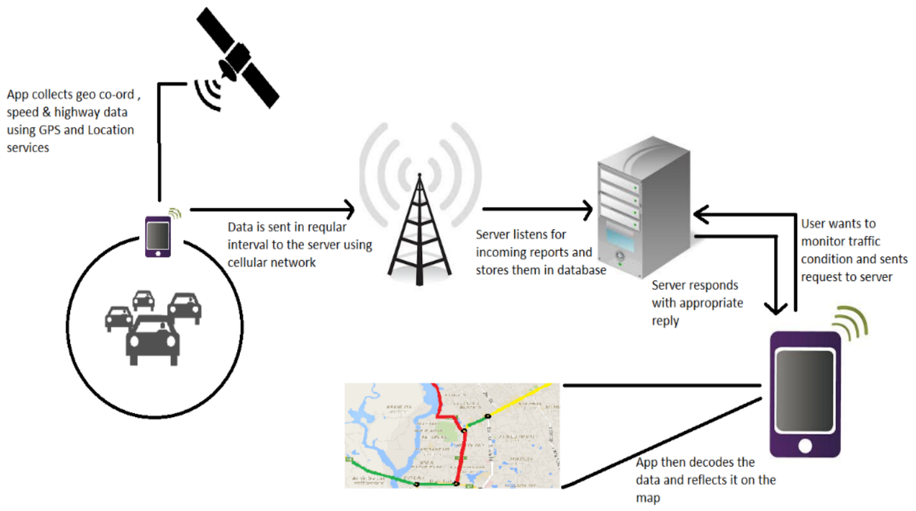

App collects geo co-ord , speed & highway data using GPS and Location services
Data is sent in regular interval to the server using cellular network
Server listens for incoming reports and stores them in database
User wants to monitor traffic condition and sents request to server
Server responds with appropriate reply
App then decodes the data and reflects it on the map

Fig 8: An illustration of the interaction of server and client side application

3.5 Server side algorithm &workflow

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

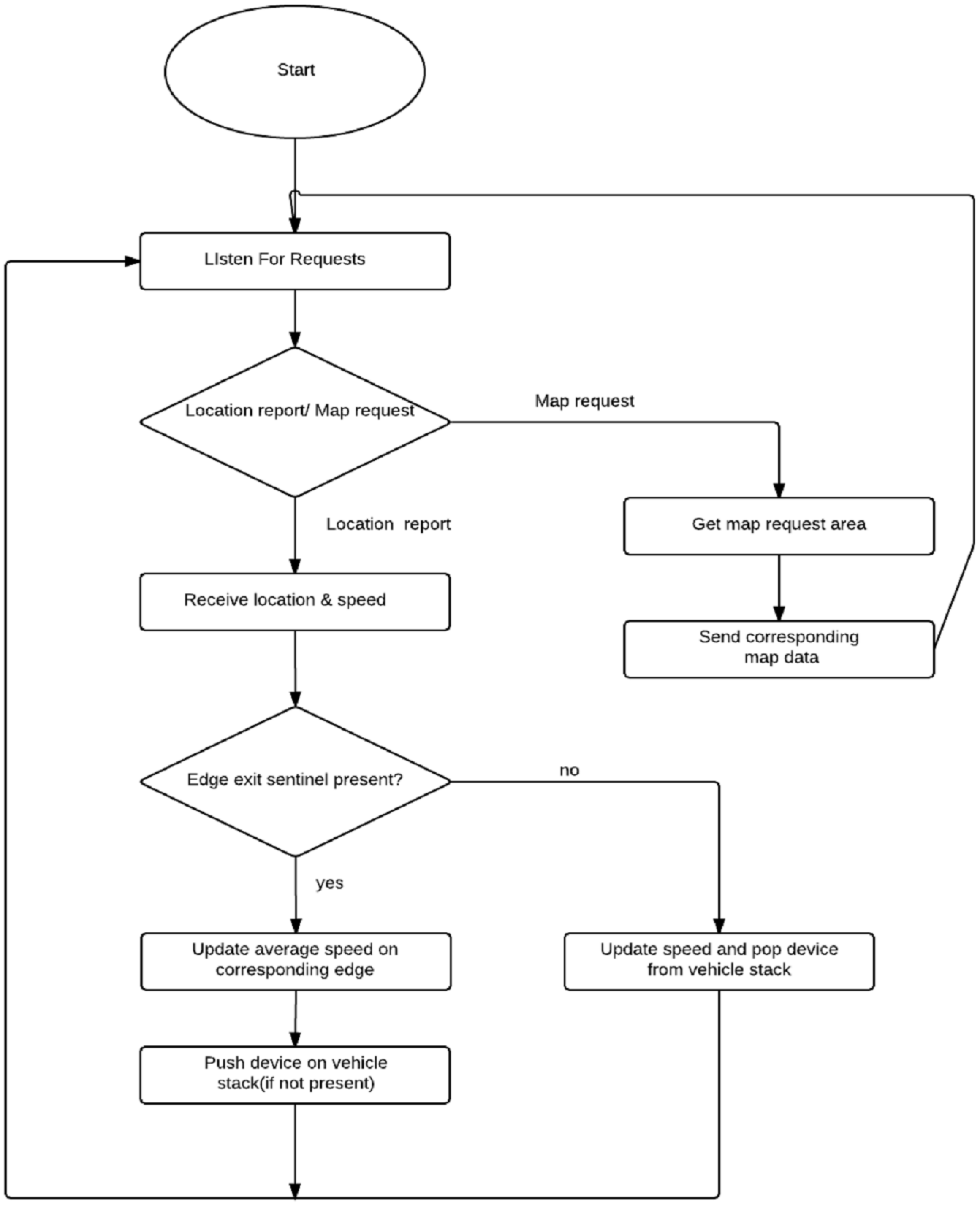

Start
LIsten For Requests
Location report/ Map request
Map request
Get map request area
Send corresponding map data
Location report
Receive location & speed
Edge exit sentinel present?
no
yes
Update average speed on corresponding edge
Update speed and pop device from vehicle stack
Push device on vehicle stack(if not present)

Fig 8: Server side work flow

3.6 Client side algorithm & workflow

On client side, we try to implement a reiterative service which constantly runs in the background and feed data to the server. Data transmission starts once we can decide that the device is on top of a major marked highway and it is indeed a traveling vehicle. Then upon entering an edge/road, we start the interval timer after which we send the location information to the server and reset the timer. The timer interval can be overwritten in one case, when the vehicle entered the intersection point of roads and about to leave the road it was previously. In that case we calculate accordingly and upload the data so that we can start counting anew for the next road.

A Flow-chart below represent the logic demonstrated above:

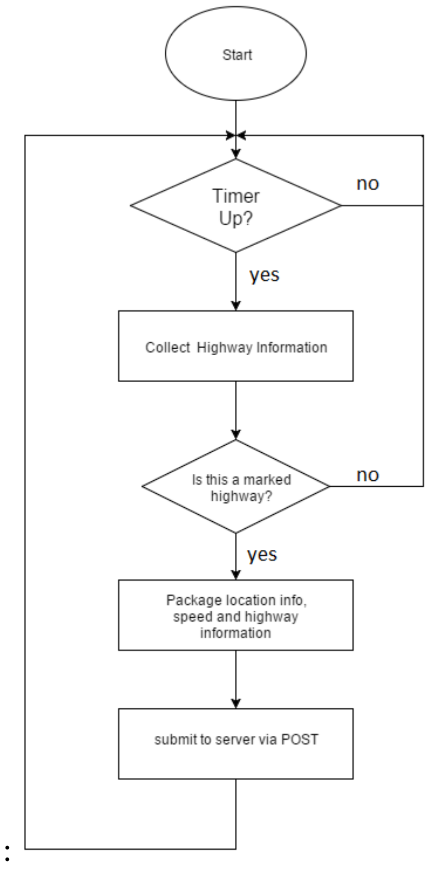

Start
Timer Up?
no
yes
Collect Highway Information
Is this a marked highway?
no
yes
Package location info, speed and highway information
submit to server via POST

Fig 9: Client side submitter work-flow

3.7 Form of Communication between Server side and Client side

The client and server side are going to communicate with formatted data. When a device will report its location it will include co-ordinates, speed and any sentinels in a JSON object and send them to the server. The server will have a graph data structure representing prepared roads and store necessary information about a road in the Edge object reference that represents it. These values will include a device stack, average speed, color marker data etc. When the server will send map data to a requesting device, it will likewise format them in JSON that can be interpreted in the map view directly by the application.

Part 4: Development Environment & Pre requisites

4.1 Android & Google Maps API

Google maps is the desktop/mobile web mapping service that has been developed by Google. It is the primary maps application for Android operation system and also available for usage in all other major mobile operating systems. It offers satellite imagery, street maps, 360° panoramic views of streets (Street View), Google Traffic (country specific) and information regarding public transportation for a few selective countries. While other mapping services do exist, Google maps is one of the most popular in both mobile and in the web. Google Maps for mobile, in August 2013, was the world's most popular app for smartphones, with over 54% of global smartphone owners using it at least once[20].

For developers, Google launched the Google Maps API in June 2005. This allows developers to integrate Google Maps into their websites and mobile apps. Maps API facilitates location and location related operations like accessing cellphone GPS, getting location specific information, determining routes etc[21]. Thus numerous applications exists both for android and other smartphone operating systems that utilizes Google maps API to make their application location aware.

As for the proposed traffic monitoring application, the necessary data can all be accessed using Google Maps API, thus this is the optimal choice for the application. As for the mobile operating system, we choose android as Android is also Google’s product and has full compatibility with Maps API. Also, android has a far better market penetration than any other mobile operating system worldwide, and evidently in Bangladesh too[22].

4.2 Development Environment, Compiler & IDE

For our particular project, since we chose Android as our primary OS to build our software for, we had to use JAVA as our programming language. Henceforth, we set up JAVA development kit and configured our java run-time environment. For building android application, we chose Android Studio as our IDE even though we could use Eclipse with ADT but Google as announced before will no longer support Eclipse due to proprietary issues. Android Studio comes with a compiler named Gradle. Gradle is the next evolutionary step in JVM-based build tools. It draws on lessons learned from established tools such as Ant and Maven and takes their best ideas to the next level[23]. Because Gradle is a JVM native, it allows us to write custom logic in the language we're most comfortable with. Dependency management is employed to automatically download these artifacts from a repository and make them available to our application.Gradle's ability to manage dependencies isn't limited to external libraries. As our project grows in size and complexity, we'll want to organize the code into modules with clearly defined responsibilities. Gradle provides powerful support for defining and organizing multiproject builds, as well as modeling dependencies between projects[24].

4.3 API Keys for Google services

API(Application programming interface) keys are procedure of how authentication, authorization, and accounting are accomplished. For all API calls, our application needs to be authenticated[21]. When an API accesses a user's private data, our application must also be authorized by the user to access the data. For instance, accessing a public Google+ post would not require user authorization, but accessing a user's private calendar would. Also, for all sorts of purposes, all API calls involve accounting. We will summarize the protocols used by Google APIs for our particular project.

4.4 Access Types

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

1. **Simple API access (API keys)**

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

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

2. **Authorized API access (OAuth 2.0)**

   These API calls access private user data. Before we can call them, the user that has access to the private data must grant our application access. Therefore, our application has to be authenticated, the user must grant access for your application, and the user must be authenticated in order to grant that access. All of this is accomplished with OAuth 2.0 and libraries written for it.

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

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

4.5 Class Diagram

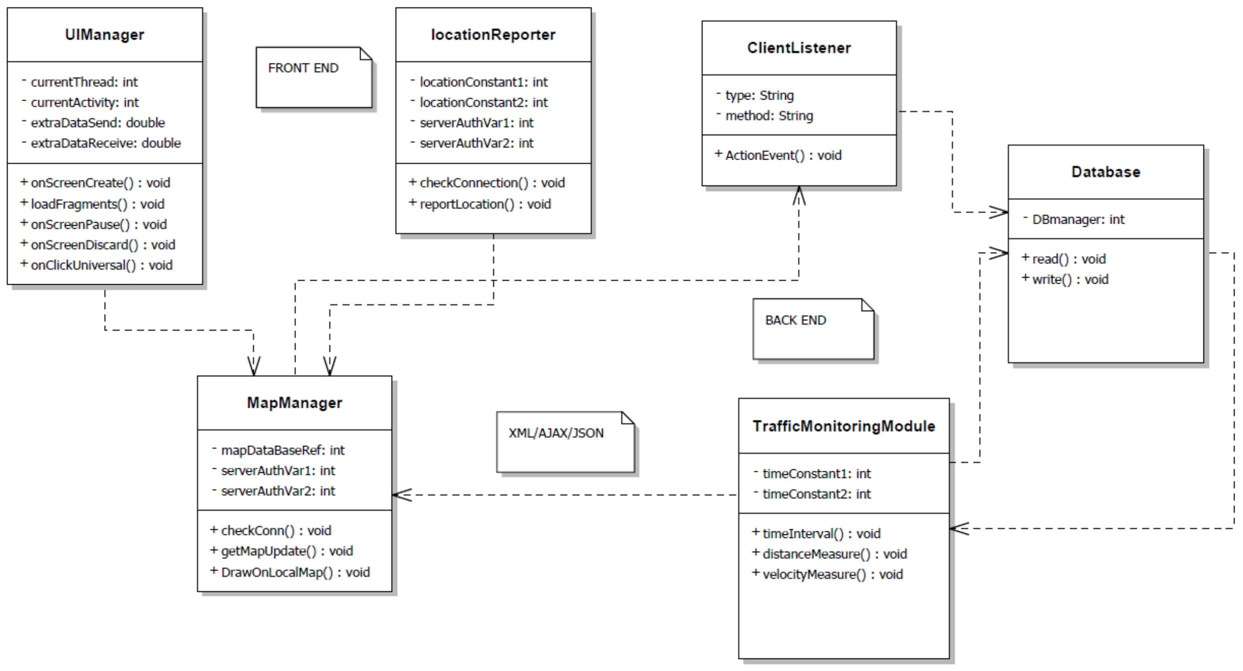


Fig 10: Class diagram of our system

Part 5: Client side implementation of Location and Speed reporting service

As our project is largely based on acquiring finely processed speed data, we had to iterate through multiple options and perform SWOT analysis before we went on to decide which option we will opt in. The purpose was to reduce redundancy of data and increase accuracy as much as possible.

5.1 Comparative Analysis between options in Android Ecosystem to measure speed

Android operating system allows us to measure speed through Google Location API which follows the basic equation:

Speed=(currentGPSPoint - lastGPSPoint) /(time between GPSpoints)

Location API depends on providers such as GPS,WI-FI, Cell to get clues of the GPS points and time differences to record speed of the device. But issues arise which one to go for when each option have matter of trade-offs in accuracy, speed, and battery efficiency.

- **NETWORK SERVICE PROVIDER**: Network Location Provider give clues to locations updates from cell tower and Wi-Fi based location. Due to weak signal, it cannot get proper location fixthereby compromising accuracy. But it is indeed battery efficient as cellphone always remainconnected to cell tower and in most cases Wi-Fi.

- **ACCELEROMETER AND GYROSCOPE**: Apart from Location API to get location and time fix, thereare accelerometer or gyroscope sensors more or less in every device which can be handy tomeasure speed. But for this to work, we need initial speed to be determinate based on thesame referential as the acceleration. This would require some measurements and a lot oftrigonometry, as you would get both values from different sensors at different rates.Furthermore, if the device is in the user hand, the device movements in relation to the car willincrease even more the calculations. So the device has to be fixed to the car to get properspeed measures.

- **GPS PROVIDER**: Perhaps the most reliable provider with respect to aforementioned two. Itgives clues to location updates based on GPS sensor which can be accessed through locationmanager class of android location api. It gets a constant location fix even inside a movingvehicle. The only trade-off can be battery-efficiency when we try to listen for updatescontinuously through location listener without putting thread to sleep. But this can be workedaround with a flow model which contains the procedure for optimum battery-efficiency.

5.2 Flow for obtaining user location

Here's the typical flow of procedures we followed for best location update:

1. Start application.
2. Sometime later, start listening for updates from desired location providers.
3. Maintain a "current best estimate" of location by filtering out new, but less accurate fixes.
4. Stop listening for location updates.
5. Take advantage of the last best location estimate. On each update of location, we save the time of the current update. So, we will have the previous and current location, and time of

update. Then we calculate the distance in meters between those two locations (the old and new one).

A code snippet from our application to calculate distance as follows:

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

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

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

```
/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

As we keep collecting speed data, and on the expiry of each timer, the application sends a timely location report to the server. The data transaction process is discussed in details in part 8 of this paper.

5.3 Obtaining data for map database

As we have discussed earlier, the highways of a city can be represented as a set of geo co-ordinates denoting their both end points. For accurate calculation and visual illustration of each road and their spans, we have to maintain a database of highways in our system.

The traffic jam usually occurs between two signals and signals are usually on the end points of a road. Thus, calculating the density of traffic in between any two traffic signal should prove most effective in any case. However, For Google maps in Bangladesh, no such information regarding traffic signal location is available.

This opts us to approximate the locations of signals and lengths in between the signals manually. The location of signals is usually on the intersecting point of two or more roads. The Google Maps API allows us to compute the intersections between any two roads R1 and R2 by issuing a Directions Request API whose argument is “R1 & R2”. Passing these parameters can automatically calculate the intersection point of two roads. Otherwise, we can also drop pointers on google maps manually to find out the intersection points or probable location of traffic signals[17].

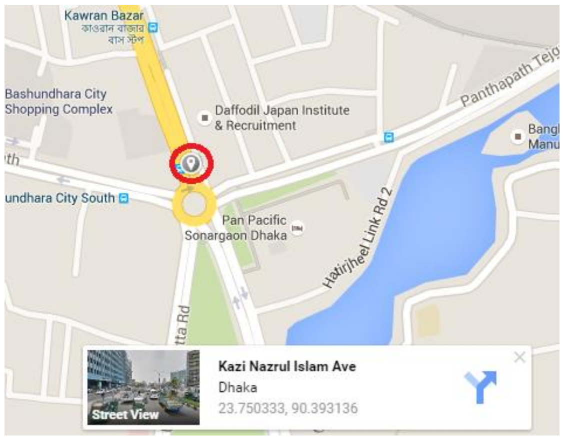


Fig 11: Finding geo ordinates for database in Maps

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

5.4 Translating geo-information to identifying roads

while fetching a mere collection of GPS co-ordinates is fairly easy, translating them to identify a particular road a user is traveling proves to be a daunting task. Google Geocoding API does exactly what we are looking for. It identifies the roads as vehicle was traveling along and provides additional metadata about those roads, such as speed limits. For our purpose, we will be storing speed metrics as metadata for maps API to create visualization on the road.

The term *geocoding* generally refers to translating a human-readable address into a location on a map. The process of doing the opposite, translating a location on the map into a human-readable address, is known as *reverse geocoding*. This service is generally designed for geocoding static (known in advance) addresses for placement of application content on a map.

The following query contains the latitude/longitude value for a location in Farmgate:

https://maps.googleapis.com/maps/api/geocode/json?latlng=23.7550215, 90.3911918&key=*API_KEY*

```
 {
"formatted_address" : "KaziNazrul Islam Ave,Farmgate,Dhaka",
 "geometry" :
{
 "location" : {
 "lat" : 40.714232,
 "lng" : -73.9612889
 }
 }
```

Now that we can translate coordinates to street name we can store information for each road in the server and post process the data in server side.

Part 6: Client side implementation of Google Maps API

With the Google Maps Android API, we can add maps based on Google Maps data to our application. The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. We can also use API calls to add markers, polygons, and overlays to a basic map, and to change the user's view of a particular map area[21]. These objects provide additional information for map locations, and allow user interaction with the map. The API allows you to add these graphics to a map:

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

6.1 MapsView

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

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

```
<meta-data
android:name="com.google.android.maps.v2.API_KEY"
android:value="AIzaSyBuGHO9IEStgSUCKPxVfWrGTZh31FCHP9Y" />
```

6.2 MapFragment

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

6.3 Markers

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

6.4 Changing the GoogleView

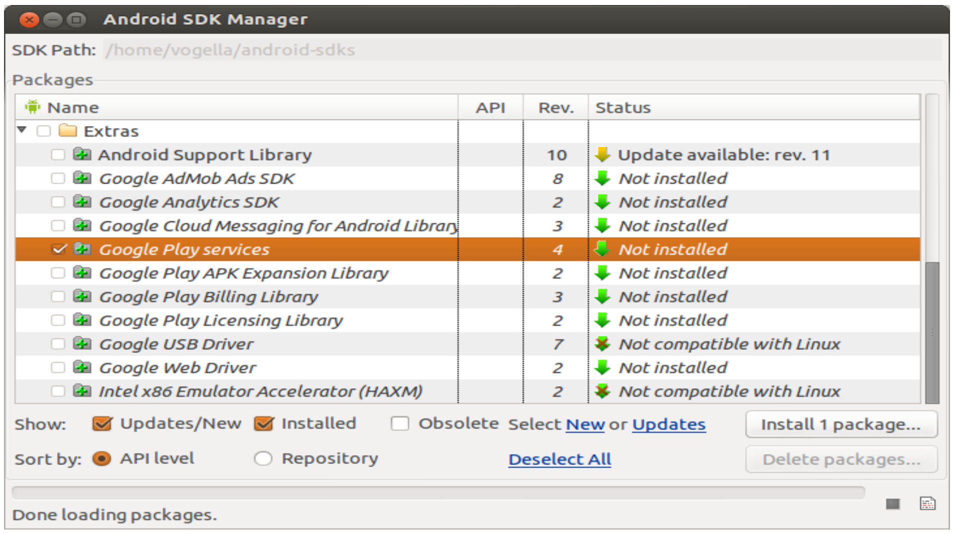
The GoogleMap can be highly customized. The following example code demonstrates the purpose.

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

static final LatLngFarmgate = new LatLng(53.551, 9.993);

privateGoogleMap map;

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

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

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

// Zoom in, animating the camera.

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

6.5 Install Google Play services

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

Google play Services framework to access Google maps API. For this we need to configure this particular framework.


Android SDK Manager

SDK Path: /home/vogella/android-sdks

Packages

Name	API	Rev.	Status
Extras			
☐ Android Support Library		10	Update available: rev. 11
☐ Google AdMob Ads SDK		8	Not installed
☐ Google Analytics SDK		2	Not installed
☐ Google Cloud Messaging for Android Library		3	Not installed
✓ Google Play services		4	Not installed
☐ Google Play APK Expansion Library		2	Not installed
☐ Google Play Billing Library		3	Not installed
☐ Google Play Licensing Library		2	Not installed
☐ Google USB Driver		7	Not compatible with Linux
☐ Google Web Driver		2	Not installed
☐ Intel x86 Emulator Accelerator (HAXM)		2	Not compatible with Linux

Show: ✓ Updates/New ✓ Installed ☐ Obsolete Select New or Updates

Sort by: ● API level ○ Repository Deselect All

Done loading packages.

Fig 12: Installing Google play services for development

Part 7: Acquiring Google API key

7.1 Google console

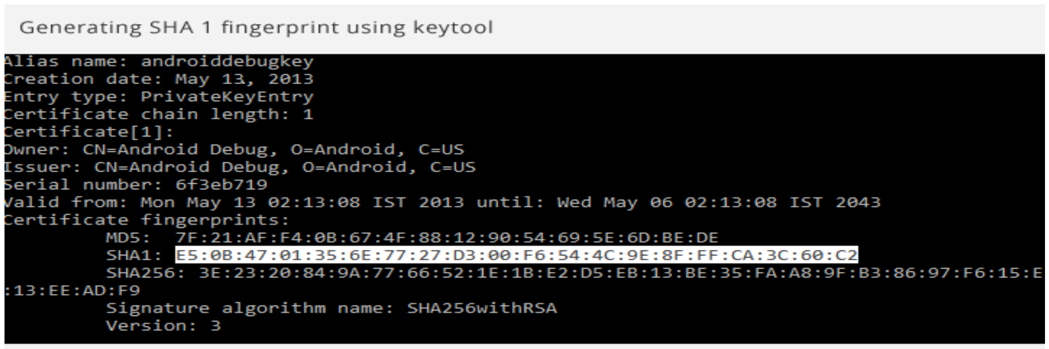
To use Google Maps weneed to create a valid Google Maps API key. The key is free, you can use it with any of your applications that call the Maps API, but it supports only a limited number of users in the free version[25].

We get this key via the Google APIs Console. We had to provide our application signature key and the application package name. This is based on the key with which we sign our Android application during deployment.

7.2 Creating the SHA-1 for your signature key

We need to generate SHA-1 fingerprint using java *keytool*. Open your terminal and execute the following command to generate SHA-1 fingerprint.

keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android



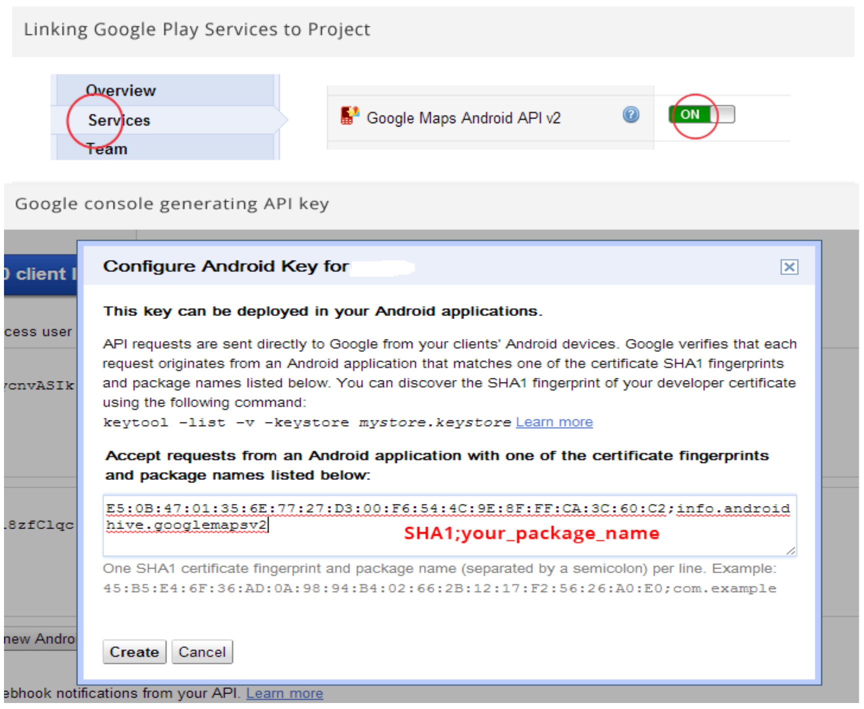
Generating SHA 1 fingerprint using keytool

```
Alias name: androiddebugkey
Creation date: May 13, 2013
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 6f3eb719
Valid from: Mon May 13 02:13:08 IST 2013 until: Wed May 06 02:13:08 IST 2043
Certificate fingerprints:
         MD5:  7F:21:AF:F4:0B:67:4F:88:12:90:54:69:5E:6D:BE:DE
         SHA1: E5:0B:47:01:35:6E:77:27:D3:00:F6:54:4C:9E:8F:FF:CA:3C:60:C2
         SHA256: 3E:23:20:84:9A:77:66:52:1E:1B:E2:D5:EB:13:BE:35:FA:A8:9F:B3:86:97:F6:15:E
:13:EE:AD:F9
         Signature algorithm name: SHA256withRSA
         Version: 3
```

Fig 13: Obtaining fingerprint using Key Tool

7.3 Configuring Google API Console

Now we go to Google console for developer's page and turn on services for Google maps API V2 under services tab. We will paste the sha1 key here for creating unique Android API key for our application.


Linking Google Play Services to Project
Overview
Services
Team
Google Maps Android API v2
ON
Google console generating API key
client I
cess user
ycnvASIk
.8zfClqc
new Andro
Create
Cancel

Configure Android Key for

This key can be deployed in your Android applications.

API requests are sent directly to Google from your clients' Android devices. Google verifies that each request originates from an Android application that matches one of the certificate SHA1 fingerprints and package names listed below. You can discover the SHA1 fingerprint of your developer certificate using the following command:

```
keytool -list -v -keystore mystore.keystore
```
Learn more

Accept requests from an Android application with one of the certificate fingerprints and package names listed below:

```
E5:0B:47:01:35:6E:77:27:D3:00:F6:54:4C:9E:8F:FF:CA:3C:60:C2;info.android
hive.googlemapsv2
```
SHA1;your_package_name

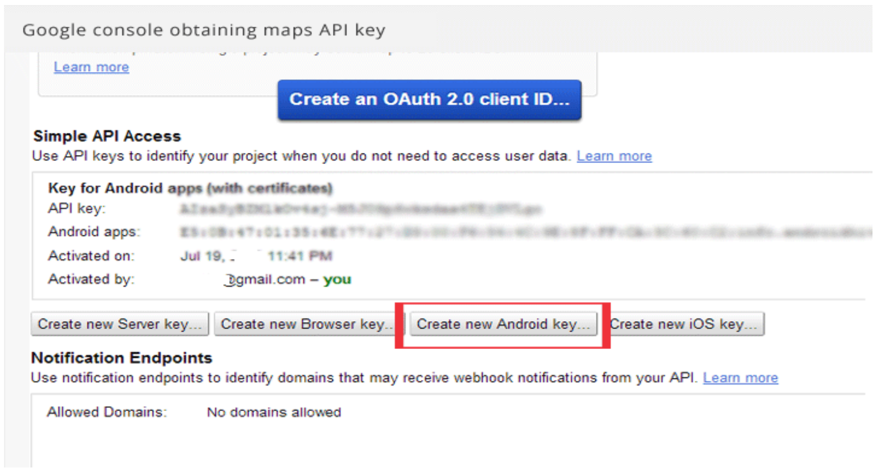
One SHA1 certificate fingerprint and package name (separated by a semicolon) per line. Example:

```
45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example
```

ebhook notifications from your API. Learn more

Fig 14: obtaining SHA1 finger print from the web interface

Now we select **API Access** on left side and on the right side click on **Create new Android key** **i**t will pop up a window asking the SHA1 and package name. We enter our *SHA 1* and our *android project package name* separated by semicolon; and click on create.



Google console obtaining maps API key

Learn more

Create an OAuth 2.0 client ID...

Simple API Access

Use API keys to identify your project when you do not need to access user data. Learn more

Key for Android apps (with certificates)

API key:

Android apps:

Activated on: Jul 19, 11:41 PM

Activated by: @gmail.com – you

Create new Server key... Create new Browser key... Create new Android key... Create new iOS key...

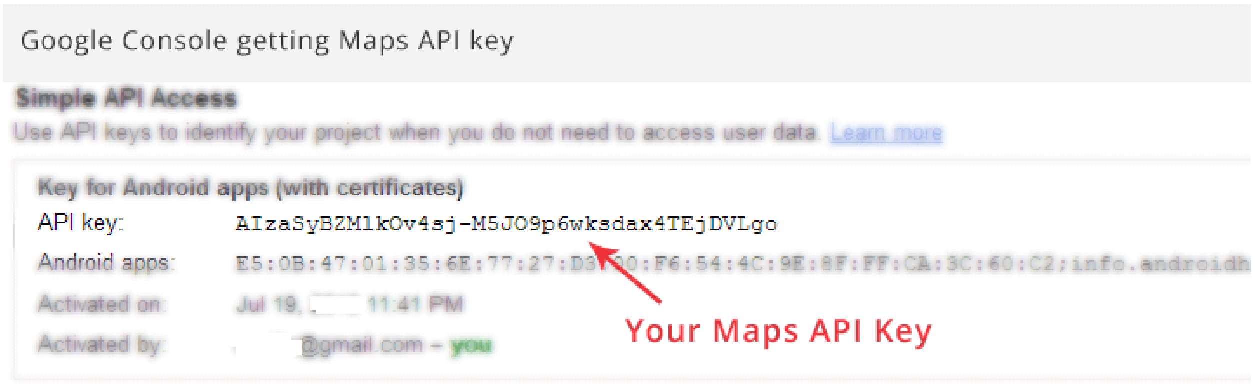
Notification Endpoints

Use notification endpoints to identify domains that may receive webhook notifications from your API. Learn more

Allowed Domains: No domains allowed

Fig 15: Creating OAuth 2.0 Client ID

And finally it will generate API key which is required in our project later.



Google Console getting Maps API key

Simple API Access

Use API keys to identify your project when you do not need to access user data. Learn more

Key for Android apps (with certificates)

API key: AIzaSyBZM1kOv4sj-M5JO9p6wksdax4TEjDVLgo

Android apps: E5:0B:47:01:35:6E:77:27:D3:90:F6:54:4C:9E:8F:FF:CA:3C:60:C2;info.androidh

Activated on: Jul 19, 11:41 PM

Activated by: @gmail.com – you

Fig 16: Obtaining the final API key

7.4 Drawing shapes on map based on retrieved information

Once we decode the necessary information we received from the server we will create MapFragment to supply the refreshed view of the map. This view will contain color coded polylines to visualize the traffic monitoring scenario[21].

Google Maps API for Android offers some simple ways for you to add shapes to your maps in order to customize them for your application. A Polyline is a series of connected line segments that can form any shape you want and can be used to mark paths and routes on the map. To create a Polyline, we first create a PolylineOptions object and add points to it. Points represent a point on the earth's surface, and are expressed as a LatLng object. Line segments are drawn between points according to the order in which you add them to the PolylineOptions object. To add points to a PolylineOptions object, call PolylineOptions.add() .

The following code snippet illustrates how to add a rectangle to a map:

```
// Instantiates a new Polyline object and adds points to define a rectangle

PolylineOptionsrectOptions = new PolylineOptions()

.add(new LatLng(37.35, -122.0))

.add(new LatLng(37.45, -122.0)) // North of the previous point, but at the same longitude

.add(new LatLng(37.45, -122.2)) // Same latitude, and 30km to the west

.add(new LatLng(37.35, -122.2)) // Same longitude, and 16km to the south

.add(new LatLng(37.35, -122.0)); // Closes the polyline.

// Get back the mutable Polyline

Polyline polyline = myMap.addPolyline(rectOptions);
```

To customize appearance and add color, we use various option at our disposal. Like in the follows.

```
Polyline line = map.addPolyline(new PolylineOptions()
.add(new LatLng(-37.81319, 144.96298), new LatLng(-31.95285, 115.85734))
 .width(25)
.color(Color.GREEN)
 .geodesic(true));
```

Part 8: Client side functionality for Server Communication

8.1 Connecting to RESTful Web service

When creating a data driven application as ours, it is necessary to be able to connect from within the application to the web service to send data back and forth. As a result, besides deploying a web service, this has to be taken care of from within the application as well. This is done by utilizing a library that is equipped to handle HTTP connections. There are a handful options to choose from, as they are discussed below.

8.2 Libraries within and outside android SDK

There are two libraries included by default with the Android SDK:

- ***Apache HTTP Client*:**This is a library that is used in many other JAVA applications as well. As a result, those coming from any JAVA related background find it similar. However the version implemented in the android SDK is very much android specific as it use android terminologies and framework calls like all other android libraries. It comes with two HTTP clients: DefaultHttpClient&AndroidHttpClient. They are both extensible java classes this they instantiate just like any other libraries. Apache HTTP client is powerful and flexible, but big and complex in the same time. However, the biggest drawback of using the Apache HTTP client is that, the development of this particular library has been frozen since 2011 by the android framework developer team. Thus whatever features and bugs were there few years ago, they are probably going to be there for the foreseeable feature.

- ***HttpURLConnection*:** This library is an implementation of classes from java.net package. It also uses the URL class for making connections and requests. To process the requests and responses, it uses other classes of the package java.io. It is a fairly lightweight and suited for creating web services. It is being constantly developed and supported by android framework developers and seen as the prime choice for all new android apps.
- **Open source solutions:** There exists some open source solutions as well. For example: OkHttp from square, volley from google and retrofit which is also from square.

While many open source solution provides some unique features, for our case, we choose the HttpURLConnection as it is quite stable and a part of the larger android framework echosystem.

8.3 Using AsyncTask during network requests& submission

In android, each app has a single foreground called the main thread. All the widgets and application elements are maintained in this thread. Thus, when we want to create a network connection from within the application to talk to the remote web service, we cannot do that in the main thread. That causes the main thread to freeze and show the ANR (Application Not Responding) error.

In order to make requests in the background and work with the output in the foreground, we have to use a special class called the AsynTask which is unique to android environment. We use the AsyncTask as an inner class in the main activity. The AsyncTask has many methods most of which are not needed to overwrite as they are necessary to handle the calls back and forth in the application framework. There are three methods that we are going to overwrite to make background network calls and update the display:

- onPreExecute(): On this block the UI elements such as the progressbar is changed from invisible to visible as they start working.
- doInBackGround(): On this block the actual information are gathered and specific methods are called to upload the data to the web service. Inside the doInBackGround another specific method named publishProgress() is called to publish the progress on the screen to let the user know when which information is updated.
- onPostExecute(): This block is hit when the process of updating information is over. Other UI elements, such as re-hiding the progressbar and enabling the "Start uplaoding" switch, these are taken care of in this block.

The data submitter app always keeps on updating the information by sending the data to the server every 30 seconds. Thus an infinite loop is used to keep the process going over and over again with a rest of 30 seconds after each submission. Thus the loop can only be broken when user decides to stop the data submission process by hitting the stop reporting button.

8.4 Sending GET/POST request from within app

Sending GET/POST request from within the android application to the web service requires specific encoding of the URI and then the processing of the response from the web service from within the application. For the encoding of the URI according to the type of request and necessary parameters, we maintain a URL packager class instead of hardcoding the string. The involved classes in this process are as below:

- MainActivity.java
- RequestPackage.java
- HttpManager.java

The RequestPackage.java is a generic class that helps with the encoding of URLs using necessary variables, their getters-setters and JAVA's String builder. The class maintains three variables: URI(the actual link of the web service), method (which holds the type of method this particular request will use) and Hashmap that will be used to hold as many variables that are needed to be submitted according to the specific need.

The HttpManager will do the actual work of Connecting to the web service and getting the reply from the web service using JAVA's HttpURLConnection library.

In the MainActivity during a regular report or a JSON request, first all the necessary data are collected. Then if it is a POST request for data submission, a post request with the post web service's URI is prepared with the help of RequestPackage class. Then in the AsyncTask'sDoInBackground() method, the URI is called with the help of HttpManager. If it is a POST request and it is successfully submitted, then a response will come stating that it's a success. If it is a request for json data, json will be thrown out by the server and then the appropriate class to handle the JSON data will be called to decode the JSON and give particular output.

Part 9: Server Side Implementation

9.1 Purpose of a backend web service

While dealing with a data driven android application such as ours, it is a must to have a backend "Web service" to keep track of all the information. While the data collector side of the application generates several types of data locally, making the same information available to the data monitoring side of the application is important to make it a real time application. Back end Web services are typically made of server side scripts (PHP in our case) and a relational database. The scripts are written to always listen to requests from client side to give the appropriate response in the proper form that can be interpreted by the client. The scripts also listen for data submission (POST requests) to store regular location reports. The database on the other hand actually stores the raw data which is available for manipulation by the scripts.

9.2 MySQL database structure

For the efficient storing and extraction of the necessary data, the construction of the relational database is quit important. In our case there are two entities that we need to keep track about: Highways and Nodes.

The first entity, therefore table, in the database is *HIGHWAYS*. Whenever a location report is trying to be made from a device end, based on the geo location extracted from the GPS data, the user's current location is matched against the database of highways the backend system is maintaining. Similarly, whenever a record is about to be entered in the database, the Highway name of each report has to match some entry on the Highway table. Thus, the highway table in our database has the following fields:

- **id**: This is the primary field of this table and set to auto-increment. The field by itself does not reflect any information, but it maintains the sequence of the entries of the table.
- **name**: This is a varchar type field denoting the name of this particular highway.
- **pointa_lat**: The latitude value of one end point's geo_location of this highway. Like the previous table, the latitude value is of float type and has a length of 10,6.
- **pointa_lng**: The longitude value of one end point's geo_location. Properties are same as the previous field.
- **pointb_lat**: The latitude value of the other end point's geo_location. Properties are same as the previous field.
- **pointb_lng**: The longitude value of other end point's geo_location. Properties are same as the previous field.
- **datetime**: A datetime attribute to denote the time it was entered in the database.

An important aspect of this table is, the contents of this table cannot be updated or modified by the client application. The table is updated by system admins periodically whenever more road specific information are there to be taken in to account in the system.

The second entity in the database is *NODES.* In this case, by a node we refer to a particular vehicle that is reporting about traffic condition from somewhere. The structure of the table is as described below:

- **id**: This is the primary field of this table and set to auto-increment. The field by itself does not reflect any information, but it maintains the sequence of the entries of the table. It is of integer type.
- **device_id**: This field represents the identity of the device that made this particular report. It is an eight-digit string uniquely assigned on every device by the client side app itself.
- **lat**: This is the latitude part of the geo-location from where the report was made. It is of type float and bears a size allocation of 10,6 on the MySQL table structure notation. This means the field store 6 digits after the decimal, plus up to 4 digits before the decimal, e.g. -123.456789 degrees.[26]
- **lng**: This is the lng part of the geo-location from where the report was made. The size and type of this field is same as the lat field.
- **highway_id**: This is a foreign-key attribute that refers to the id field of *HIGHWAYS* table. This denotes the marked highway that the particular report was submitted from.
- **datetime**: A datetime data-type in the MYSQL table structure notation, that denotes the time the report was submitted.

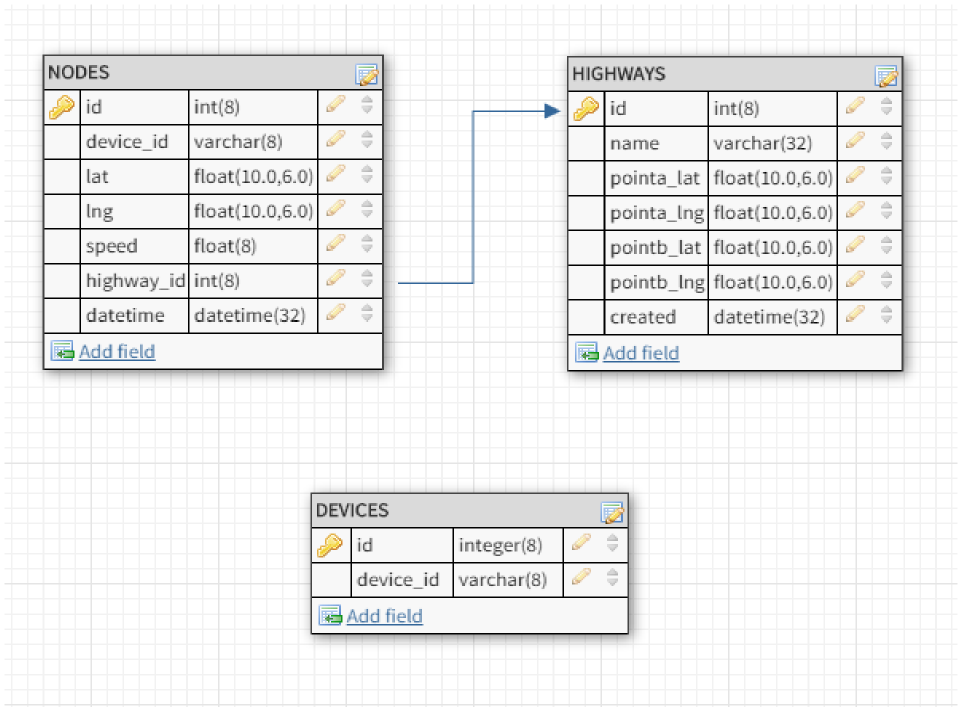
As there can be many NODE vehicles in a highway at any given time, so there can be occurrence of multiple entries in this table with similar highway_id. But one vehicle can be in only highway and before it enters a new highway, it is popped from the previous highway. Thus there cannot be more than one entry of a single device_id in this table at any given time.

Beyond these two relational tables, there is a third table named *DEVICES*. This table has only two fields:

- **id**: This is a general integer field for maintaining the sequence.
- **device_id**: This table holds the eight character unique string ids that are being provided by each individual apps to the corresponding devices.

This table's only purpose to help client end applications to avoid conflicts while assigning devices_id using which they are distinguished in the *NODES* table. As a device_id will not appear twice even in the *NODES* table, there is no need to establish a relation between *DEVICES* table and other tables in the database.

The relation between *HIGHWAYS* and *NODES* table is "one to many", because one *HIGHWAY* can have many *NODES* in it. The schema of the above database is as shown below:



NODES	
id	int(8)
device_id	varchar(8)
lat	float(10.0,6.0)
lng	float(10.0,6.0)
speed	float(8)
highway_id	int(8)
datetime	datetime(32)

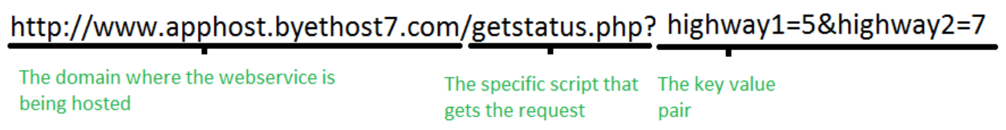
HIGHWAYS	
id	int(8)
name	varchar(32)
pointa_lat	float(10.0,6.0)
pointa_lng	float(10.0,6.0)
pointb_lat	float(10.0,6.0)
pointb_lng	float(10.0,6.0)
created	datetime(32)

DEVICES	
id	integer(8)
device_id	varchar(8)

Fig 17: Database Schema

9.3 PHP scripts & web services

The PHP scripts acts as web services and their URL with case specific parameters that are used by the app side to send particular requests and thus are stored into the application package from the beginning. The structure of these URLs can be broken down to a maximum of 3 segments:


http://www.apphost.byethost7.com/getstatus.php? highway1=5&highway2=7
The domain where the webservice is being hosted
The specific script that gets the request
The key value pair

The web service has to be hosted on a server, thus the domain name comes in the first segment. Then separated with a slash comes the specific script the request is going towards. After that, if it's a GET request there will be a "?" mark followed by one or more "Key/Value" pair. If it is a post request, there will not be a key value pair "&" separated information on the URL.

The Web service contains mainly three scripts that handles all the data:

- **reportsubmit**: This is the script that contains the functionalities to register a new report made by a Node on the road. The script always listens for incoming POST requests made by a device at any point in time. Then it tries to fetch relevant information from the NODES table and goes through three conditional statements.

  a. If a record for the device_id of this specific report does not exist on the table, *INSERT* the corresponding data and return success sentinel.

b. If a record for the device_id of this specific report exists on the table, and if the corresponding highway_id for that particular record matches the highway_id of this report, consider it a speed/location update and just *UPDATE* the specific fields.

c. If a record for the device_id of this specific report exists on the table, and if the corresponding highway_id for that record doesn't match the same field of this report, consider that the Node is no longer on the previous highway and moved to a new highway. Thus, *DELETE* the old entry and *INSERT* the new entry.

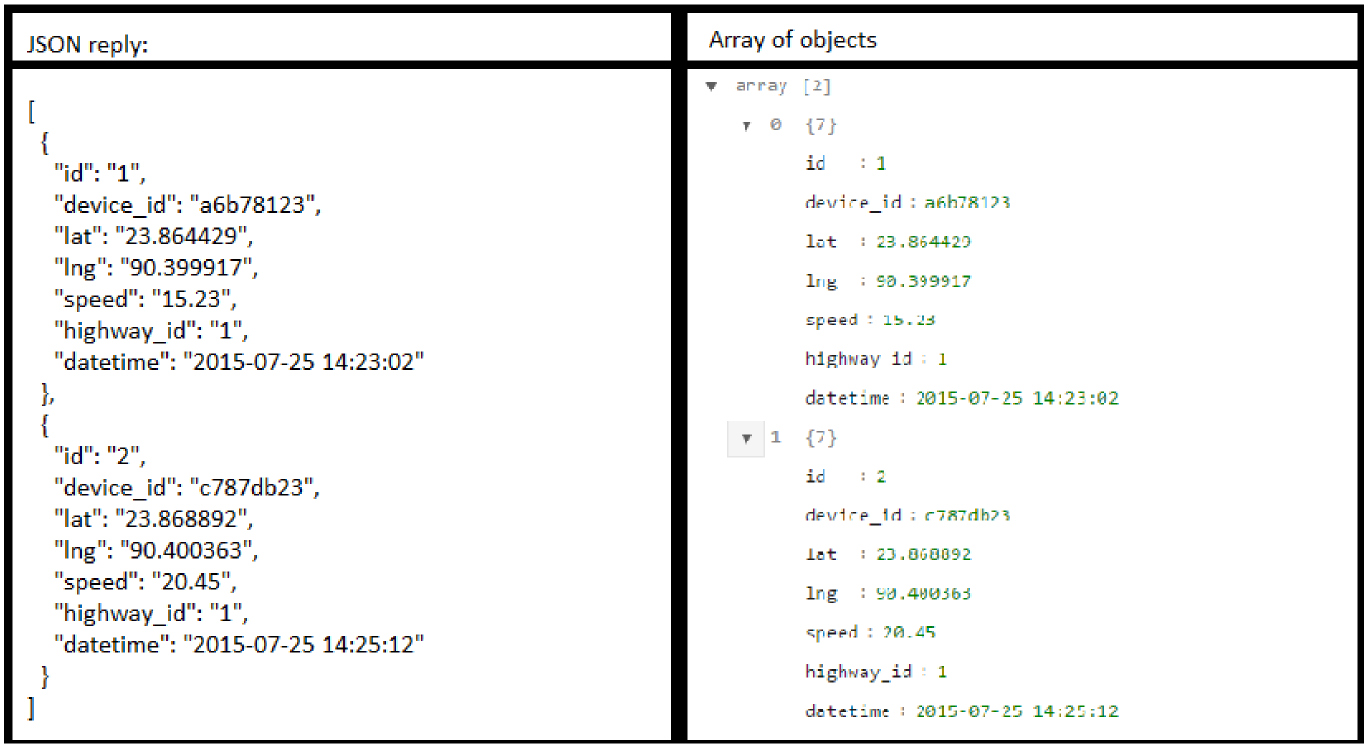
With the above logic, we are essentially creating and maintaining a "Device Stack" for all the highway. Thus whenever later on a highway specific information will be necessary, they can be fetched accordingly with the help of **getstatus.php**script.

- **getstatus**: The purpose of this script is to give response to a particular device that requested all the information regarding a particular fragment of a map. When the user on a device end is loading a particular fragment of the map, the device will send a GET request with the names or ids of the highway that are indexed in that area as the parameters. Then the script will fetch all the entries on the NODES table which has that particular highway's id in its highway_id field and return the result as JSON response.

For example, here is a request URL used by a device:

http://apphost.byethost7.com/getstatus?highway=5

The web service would then reply to such a request in JSON which can directly be interpreted as an array of objects.



JSON reply:

```
[
 {
   "id": "1",
   "device_id": "a6b78123",
   "lat": "23.864429",
   "lng": "90.399917",
   "speed": "15.23",
   "highway_id": "1",
   "datetime": "2015-07-25 14:23:02"
 },
 {
   "id": "2",
   "device_id": "c787db23",
   "lat": "23.868892",
   "lng": "90.400363",
   "speed": "20.45",
   "highway_id": "1",
   "datetime": "2015-07-25 14:25:12"
 }
]
```

Array of objects

```
▼ array [2]
   ▼ 0  {7}
        id   : 1
        device_id : a6b78123
        lat  : 23.864429
        lng  : 90.399917
        speed : 15.23
        highway_id : 1
        datetime : 2015-07-25 14:23:02
   ▼ 1  {7}
        id   : 2
        device_id : c787db23
        lat  : 23.868892
        lng  : 90.400363
        speed : 20.45
        highway_id : 1
        datetime : 2015-07-25 14:25:12
```

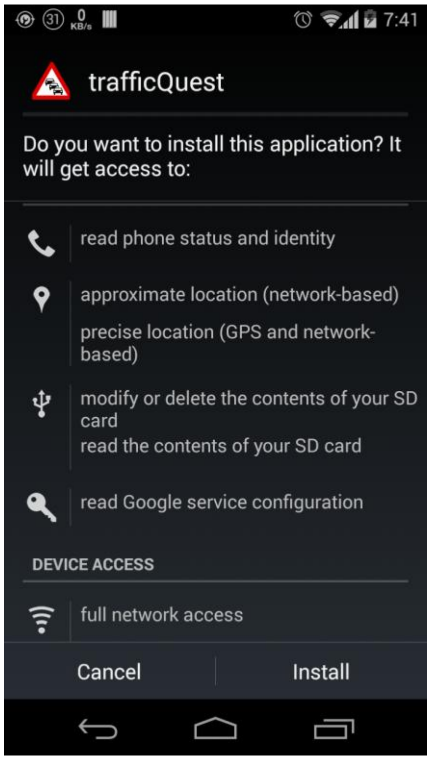
Fig 18: JSON to Array conversion

- **idcheck**: Whenever an app is trying to assign itself an unique device_id, it will try to check if the generated id already exists or not. This is when the **idcheck.php** script comes into play. It listens for a GET request from device end and checks if the id that has been sent off with the GET request is already in the database or not. If yes, it replies with a simple ‘0’ which acts as a sentinel for the device_id already exists. If it doesn’t exist, then it’s okay to use, so the script *INSERT*s the device_id in the *DEVICES* table and replies with a ‘1’ which acts as a sentinel for the id is valid and can be used.

Part 10: App Segment

10.1 Installation and Startup

During the first time installing the package, the app installer manager presents us the required permissions screen. From here we can see what permission the device will take.


trafficQuest

Do you want to install this application? It will get access to:

read phone status and identity

approximate location (network-based)

precise location (GPS and network-based)

modify or delete the contents of your SD card

read the contents of your SD card

read Google service configuration

DEVICE ACCESS

full network access

Fig 19: pre-installation screen

The permissions and their necessity are as followed:

- Phone status and identity: required for communication
- Approximate/precise location: For either GPS based location data and if not present then network based location data

- Modify contents of SD card: required for writing and reading from files in the phone.
- Google service configuration: required to read the API keys
- Full network access: Access data connection

Upon the startup, the app takes some time to load as it connects with the server and loads necessary information. Then it presents with the Maps camera on a certain zoom in a specific default position. The options menu on the top right hand corner allows the app to switch between the location reporter mode and monitoring mode.

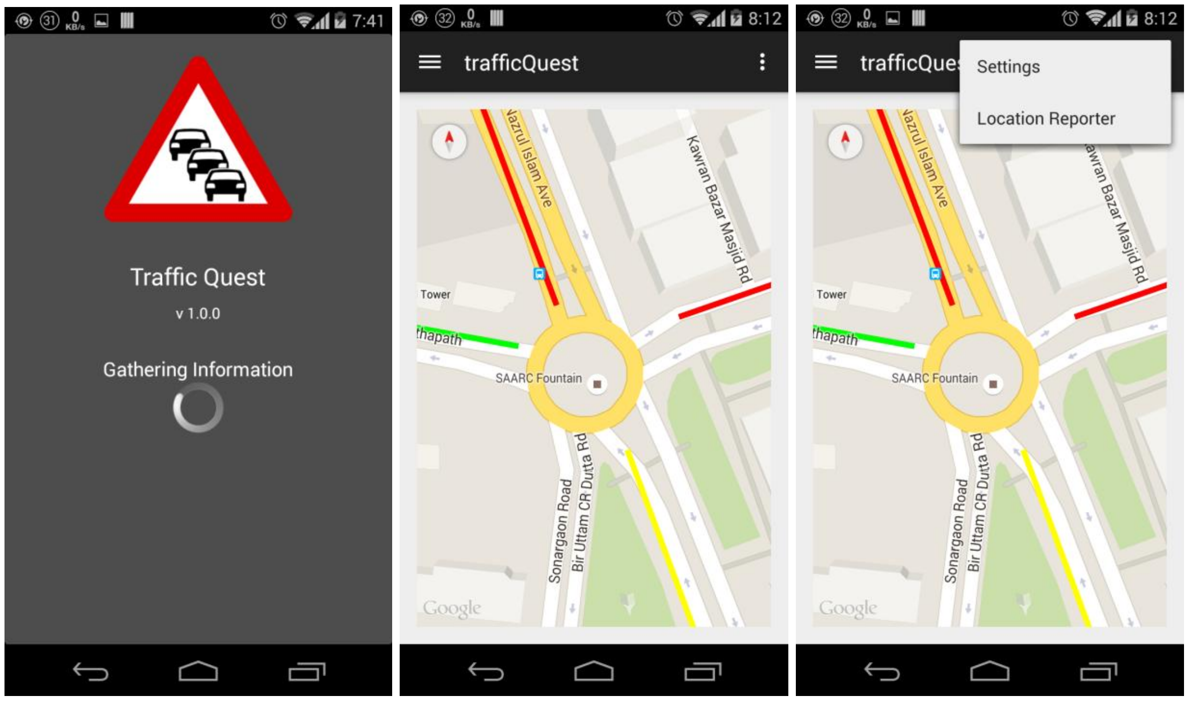


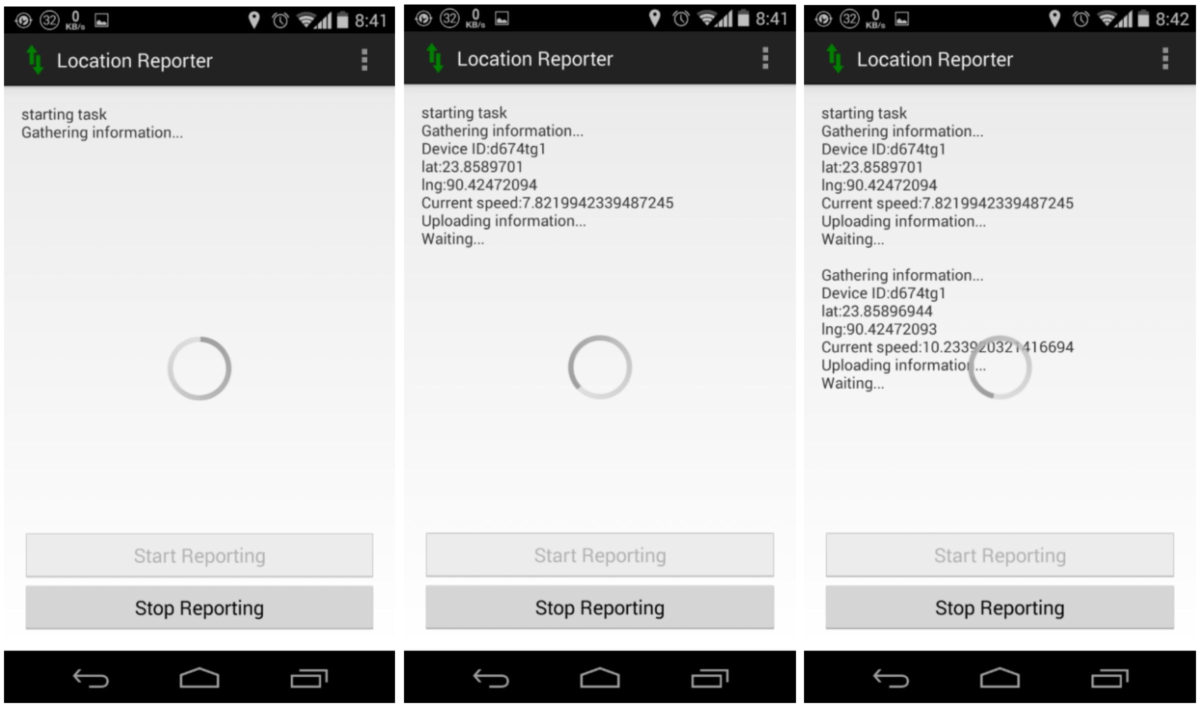
Fig 20: startup screen and default map camera position

10.2 Location reporter function

In this section, we will walk-through the visual elements of our trafficQuest location/speed reporting service.

Our location/speed reporting UI is for demonstration purpose that it produces necessary information and wraps the whole information to send it off to a web service. As we see below upon clicking 'start reporting' button the consecutive view gives us idea that it triggered location and speed reporting module and has started gathering information.

Subsequently, it generates information at an interval of definite period of time and reports thedata to the server.



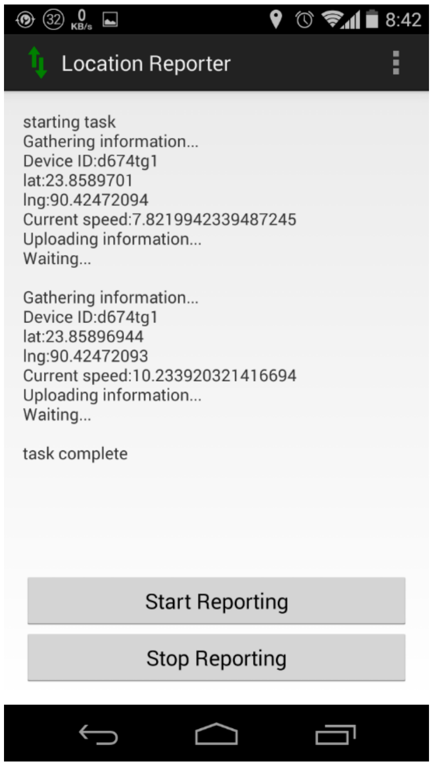
```
starting task
Gathering information...
```

```
starting task
Gathering information...
Device ID:d674tg1
lat:23.8589701
lng:90.42472094
Current speed:7.8219942339487245
Uploading information...
Waiting...
```

```
starting task
Gathering information...
Device ID:d674tg1
lat:23.8589701
lng:90.42472094
Current speed:7.8219942339487245
Uploading information...
Waiting...

Gathering information...
Device ID:d674tg1
lat:23.85896944
lng:90.42472093
Current speed:10.233920321416694
Uploading information...
Waiting...
```

Fig 21: location reporting feature with regular time interval

By pressing the Stop Reporting button the user can stop the reporting procedure at any time. This outputs task complete in the screen and makes the start reporting button press able again for resuming the reporting service.



```
starting task
Gathering information...
Device ID:d674tg1
lat:23.8589701
lng:90.42472094
Current speed:7.8219942339487245
Uploading information...
Waiting...

Gathering information...
Device ID:d674tg1
lat:23.85896944
lng:90.42472093
Current speed:10.233920321416694
Uploading information...
Waiting...

task complete
```

Fig 22: Stopping location reporting functionality

10.3 Traffic monitoring function

As we switch our view from speed reporting service to trafficQuest, our main application for viewing imagery data will connect to the server to request corresponding data of the requested map fragment. As we see on the view, upon acquiring location lock and successful server communication, map manager module draws received information for that particular map fragment.

For easy-to-use navigation, we added navigation drawer to our application which will be populated from server for set of pre-defined most publicly accessed location.

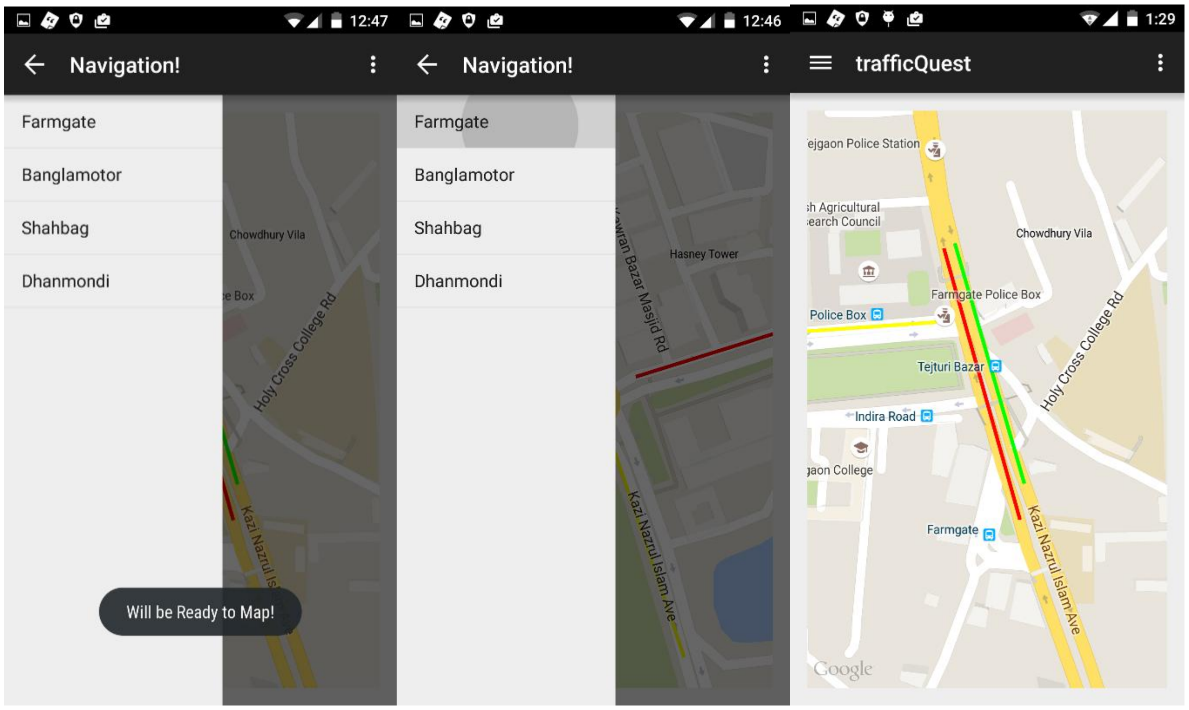


Fig 23: demonstration of Map fragment loading the traffic maps of an area

We for instance populated the application drawer list with locations as shown in the screenshot. And upon clicking farmgate on the right view, we request server a new map fragment to be drawn subsequently in the next page.

As we have further implementation to do, we will add more information for server to feed the application for wider access.

Part 11: Concerns, Challenges& Future Work

11.1 Privacy concern

For a smartphone application that runs on the concept of crowdsourcing and transmitting location data, there is always a big concern about the privacy and security. There has been questions about the method of Google traffic's data collection and how they ensure the anonymity of user data, which Google addressed later on[27]. Another popular traffic monitoring application Waze has been said to be exchanging location data with municipal authorities[28]

For an app like this, the first thing to be noted is what kind of data the application is extracting from the user's mobile phone. In android ecosystem, the operating system maintains this by the "application manifest". It is an XML file contained within the application package that denotes what permissions, therefore data, the app will require from the users of the device. Then upon installation, the list of permissions will be displayed to the user and notify him/her. Anything that was not on the manifest cannot be accessed via the application. So it is important for the user to check if the app is requesting any data that it should not be. This is a major concern in all apps in android echo system and should be verified on user's behalf[29].

Another important aspect is the user credentials. If a user provides any login/signup information for the system to identify a device uniquely, that poses another vulnerability for the app the associate a person to the location data. In our case, following a norm followed by many community driven application these days, we do not require specific username, password or email address during the first time a user starts using the app. Rather we assign a six-digit unique tag in the first startup of the application and register that tag in the server so that later on we can associate the user using just that tag anytime needed. These privacy protection features does not only ensures the users' anonymity but should be checked for whenever using a similar application.

11.2 Data consumption

Location reporting requires data connection to constantly keep sending data to the server. As it will be done on the road, cellular (GSM/ HSPA/ LTE) data connection is a must for such application to work. This is a major concern as this costs the end users a lot of data. Thus in the future, we have plans to design a method of communication with server that would eliminate the necessity of constant communication between the server and client end device application. The reduction can be possible by keeping the device from submitting a new report when the change between a previous report and a new report is negligible. However that would require the server to occasionally ping such a stagnant device and ensure it is not an idle device.

11.3 GPS accuracy

As we have discussed earlier, the geo location and device speed can be figured out from a range of means including GPS, internet connectivity or cell tower triangulation. However, the latter two tend to be not so much accurate and thus not particularly useful in a mechanism as ours where constant connectivity is a must. That's why "to the point" location reporting and status reports are very much dependent on accurate GPS data. GPS accuracy also differs based on factors like the particular GPS modem that is used in a phone and communication with GPS satellite. In our tests we have used the application in phones equipped with two different SoC modems: Qualcomm IZat Gen8A and Gen8B. Both are GLONASS supported and thus results in acceptable amount of accuracy.[30] So, during a large scale implementation (equipping motorists with the app) it is advised to ensure that the device can deliver the accurate data that is needed. In future we intend to work on a better approach to identify the level of accuracy and better adjustment & correction of the data.

Part 12: Conclusion

In this paper, we aimed to outline the possible way of implementing an effective real time traffic monitoring system, address all the aspects of such system and develop a proof of concept application. On the basis of all prior discussions and works in this paper, we believe that implementing such asystem is quite feasible. While the system can work even if there is only one user at any road at any given time, the more users on the road the better the system will work. For that, it is required to either encourage a lot of people to participate the process or deploy it selectively enforcing usage by equipping specific motorists under the supervision of an organization of government. This falls under the business implementation of the overall system and thus falls outside the scope of our study. Their does exist issues about GPS accuracy and data consumption on our current iteration of the mobile application. We do plan to overcome these issues in the future and follow up on the work we have to so far.

References

[1] U.S. Department of Transportation, "Guide to Federal ITS Research".

[2] P. Olson, "Why Google's Waze Is Trading User Data With Local Governments," Forbes, [Online]. Available: http://www.forbes.com/sites/parmyolson/2014/07/07/why-google-waze-helps-local-governments-track-its-users/.

[3] "Global Positioning System - Wikipedia, the free encyclopedia," [Online]. Available: https://en.wikipedia.org/wiki/Global_Positioning_System.

[4] "Bangladesh: mobile data subscriptions 2010-2015," Statistia, [Online]. Available: http://www.statista.com/statistics/218613/mobile-data-subscriptions-in-bangladesh-since-2010/.

[5] "Appcrawler, the app discovery engine," [Online]. Available: http://appcrawlr.com/app/uberGrid/266717.

[6] "Beat the Traffic," [Online]. Available: http://www.beatthetraffic.com/aboutus.htm.

[7] "Beat The Traffic Android app," [Online]. Available: https://play.google.com/store/apps/details?id=triangleSoftware.traffic.android.

[8] "TV Broadcast solution," [Online]. Available: https://en.wikipedia.org/wiki/Beat_the_Traffic#TV_Broadcast_Traffic_Solution.

[9] "Waze," [Online]. Available: https://www.waze.com/about.

[10] "WAZE wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Waze.

[11] "Play Store," [Online]. Available: https://play.google.com/store/apps/details?id=co.gobd.loki.loki&hl=en.

[12] "See traffic on the map," Google, [Online]. Available: https://support.google.com/maps/answer/3093389?hl=en.

[13] "Beat the Traffic used as TV Broadcast Solution," [Online]. Available: https://en.wikipedia.org/wiki/Beat_the_Traffic#TV_Broadcast_Traffic_Solution.

[14] "play store," [Online]. Available: https://play.google.com/store/apps/details?id=co.gobd.loki.loki.

[15] "Merriam Webster," [Online]. Available: http://www.merriam-webster.com/dictionary/crowdsourcing.

[16] "Crowdsourcing Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Crowdsourcing.

[17] A. Costanzo, "Using GPS data to monitor road traffic flows in a metropolitan area: methodology," Department of Electrical, Electronics and Computer Engineering,.

[18] Y.-J. Byon, A. Shalaby and B. Abdulhai, "A Study on Travel Time Collection and Traffic Monitoring Via GPS Technologies," IEEE, ITSC conference , 2006.

[19] J. I. M. C. A. Obuhuma, "Use of GPS With Road Mapping For Traffic Analysis," Internation Journal of Scientific & Technology Research, 2012.

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

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

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

[23] "Gradle release notes," [Online]. Available: https://docs.gradle.org/current/release-notes.

[24] "Gradle features," [Online]. Available: https://docs.gradle.org/current/release-notes#new-and-noteworthy.

[25] "Google APIs documentation," [Online]. Available: https://developers.google.com/console/help/new/.

[26] "Google Maps Tutorial Article," [Online]. Available: https://developers.google.com/maps/articles/phpsqlsearch_v3?csw=1#createtable.

[27] "Google Traffic," [Online]. Available: https://en.wikipedia.org/wiki/Google_Traffic.

[28] "News Mashable," [Online]. Available: http://mashable.com/2011/11/16/traffic-tech/.

[29] A. Henry, "Android permission concerns," [Online]. Available: http://lifehacker.com/5991099/why-does-this-android-app-need-so-many-permissions.

[30] "Qualcomm product specsheet," [Online]. Available: https://www.qualcomm.com/documents/snapdragon-800-processor-product-brief.

[31] "About Beat The Traffic," [Online]. Available: http://www.beatthetraffic.com/aboutus.htm.

[32] "WAZE Wiki," [Online]. Available: https://wiki.waze.com/wiki/Problems,_bugs_and_limitations.

[33] L. Drell, "Cities Using Tech to Alleviate Traffic," Mashable, 16 11 2011. [Online]. Available: http://mashable.com/2011/11/16/traffic-tech/.

[34] "WAZE android app," [Online]. Available: https://play.google.com/store/apps/details?id=com.waze.