Real-Time Public Vehicle Mobile Tracking System Using Global Positioning System Technology

J.C. Ogbonna[#1], C.E. Nwokorie[#2], J.N. Odii[#3], C.C. Ukaegbu[*4]

[#]*Department of Computer Science, Federal University of Technology Owerri, Imo State, Nigeria*

[*]*Department of Computer Science, Akanu Ibiam Federal Polytechnic Unwana, Ebonyi State, Nigeria*

Abstract *- Intelligent Transportation Systems (ITS) are gaining recognition in developing countries today. This paper focuses on the implementation of a Real-Time Public Vehicle Mobile Tracking System using GPS Technology. The system includes global position determining mobile phones located in the vehicles for determining the location of the vehicles along their routes. The system allows the public transit vehicle management and the passengers to view the tracking history of a target vehicle on a GIS Map through the internet. The mobile application installed on a GPS enabled mobile phone (Bus Simulator), periodically reads the current position of the vehicle using GPS technology, the data is sent via a mobile wireless communication network from the mobile phone to a Webserver where predictions such as the Estimated Time of Arrival (ETA) and Link's Travel Time (LTT) is computed. The vehicle's position data, ETA and LTT is then stored in the database for live and past tracking. This system enhances the LTT computation and ETA prediction in order to reflect the current traffic situations. Results show that by storing the GPS location updates in the mobile device local database when there is no network connectivity between the Webserver and the Bus Simulator, and by transmitting the stored GPS location updates to the Webserver whenever the network connectivity is re-established, the LTT computation as well as the ETA prediction is improved. This system is useful in any public vehicle transportation company, and will benefit both the management and the commuters.*

Keywords *- Estimated Time of Arrival (ETA), Geographical Information System (GIS), Global Positioning System (GPS), Intelligent Transportation Systems (ITS), Link's Travel Time (LTT).*

I. INTRODUCTION

Intelligent Transportation Systems (ITS) are advanced applications which aim to provide innovative services relating to different modes of transportation and traffic management and enable various users to be better informed and make safer, more coordinated, and smarter use of transport networks.

One application area of ITS is in the field of vehicle tracking system, more especially the Public Transport Tracking System and is in operation in many parts of the developed countries today. The system reduces the private car usage, which in turn reduces traffic congestion, fuel consumption and pollution on our highways.

When travelling with public vehicles, the travellers normally want to know the accurate timing for the vehicle arrival. Excessive long waiting time at bus stops may drive away the worried travellers and make them unwilling to take public vehicles. Currently, most public vehicle operating companies have been providing their schedules on the web/bus stops, freely available for the travellers. The vehicle schedules only provide limited information (e.g. working hours and time intervals, etc.), which are usually not timely updated. Although these services offer useful information, they are far from being satisfactory to the passengers. For example, many unpredictable factors may delay the schedule of a bus (e.g., traffic conditions, harsh weather situation, vehicle breakdown etc.). The accurate arrival time of next bus will allow travellers to take another transport as an alternative instead, and therefore mitigate their anxiety and reduces/eliminates their long waiting experience.

One of the problems faced by public transport systems today is low patronage. One of the reasons for this problem is that passengers normally do not know the public vehicle scheduling information and its arrival timing information at a particular bus stop. Therefore, it is possible to motivate a passenger to opt for public vehicles having predefined routes by providing the vehicle location and time of arrival at respective bus stops through notification boards at each bus stop or more conveniently, through a mobile application. Large cities with fleet of vehicles require a system to determine location of movement of passenger vehicles at a given time. Hence, there is need for a system that provides a systematic transportation management, provide passengers/commuters with the information as regards to an accurate estimated time of arrival

(ETA) for the next available vehicle, provides a broad set of interface options for passengers to enable them make decisions about which bus stop to go to and which bus to catch up with.

A number of studies have been carried out in the past on the problem of bus arrival time prediction in the area of intelligent transportation systems. Past studies shows that so many methods have been used and so many algorithms have been developed to solve the problem of predicting the estimated time of arrival of Public Vehicles (from their source bus stops to their destination bus stops) to the commuters, putting into consideration the wide range of factors that could affect their arrival times at their downstream stops. These methods (historical data methods, statistical data methods, Kalman filters, and Artificial Neural Networks) have been used to GPS data collected from transit vehicles in order to predict the public transit arrival time. In each of these methods, GPS data collected from the transit vehicles are transmitted through a mobile wireless communication network to a central workstation (webserver) where predictions such as LTT and ETA are made, but none of these developed algorithms puts into consideration the effect of what happens when there is low/no network connectivity between the bus station and the webserver, more especially places where there is no network coverage. Data could be lost as a result of low/no network coverage during data transmission between the bus simulator and the webserver which leads to inaccurate LTT computation for the link(s) where there is low/no network coverage, which eventually leads to inaccurate ETA prediction.

This research paper suggested and implemented an efficient way to enhance the ETA prediction by ensuring that the link's travel time for all the links in a given route reflects the present/current real time traffic situations at any given point in time.

II. LITERATURE REVIEW

The tracking of vehicles has become a very popular research topic in recent years and there are several techniques which have been applied such as Vehicle-Card technique, Cellular Infrastructure technique, Dead-Reckoning technique, and Global Position System (GPS) technique/technology. GPS is a system composed of a network of 24 satellites of the United States Department of Defence, which were originally used in military services and later allowed for commercial use. It is a free and a key technology for giving a device its position.

Recent studies show that so many methods such as historical data, statistical methods, Kalman Filters and Artificial Neural Networks (ANN) have been used to GPS data collected from transit vehicles, in order to predict public transit arrival time.

In 1999, D'Angelo, Al-Deek and Wang (1999) used a non-linear time series model to predict a flight travel time on a highway. They compared two cases: the first model used only speed data as a variable, while the second model used speed, occupancy, and volume data to predict travel time. The model proved that the single variable model using speed was better than the multivariable prediction model.

In another research carried out by Weigang et al. (2002) a model to estimate bus arrival times at bus stops using GPS information was developed. The model consists of a main algorithm calculating the estimated arrival time and two sub algorithms to determine the position and the speed of the bus on the road. First, the routes were partitioned into a number of short straight lines called links, and they modelled these lines as first-degree equation in a plane. Second, when using the speed information from the GPS, the arrival time at the stop point will be infinite if the vehicle is stationary. In order to solve this problem, they made use of historical bus travel speed along the route segment and current speed of the bus derived from the GPS data. With the improved method and empirical calibration, the results from the developed model were satisfactory in the implementation and experiment, but the problem with their model is the size of the historical data needed to predict the arrival time.

An expected time of arrival statistical model using average travel time developed by Chung and Shalaby (2007) solved the problems associated with the historical travel speed. The model predicted arrival time from the input of two categories: the last several days' historical data and the current day's operational conditions. The study incorporated schedule information and weather conditions but did not consider the dwelling time at bus stops. Performance evaluation of their proposed model showed lower levels of prediction error than moving average and regression models. Furthermore, Sun et al. (2007) developed a model with slight modification to the model proposed by Weigang et al. (2002). Their proposed prediction algorithm is based on the average speeds of route segments rather than the historical average velocity. It also leads to a superior result, because this model only estimates the speed to the end of the route segment instead of the speed to downstream bus stations, when the bus is far from the station. They carried out a case study on a real bus route. The comparison of the two algorithms showed an improvement in their own algorithm.

Ganesh et al. (2012) and Chandurkar et al. (2013) developed an implementation of a Real-time Passenger Information System for a Public Transport System using average speed model respectively. In each case, the system comprises of vehicle-mounted units, bus station units and a server located at the transport company premises. The vehicle unit reports the current position of the vehicle to a central server periodically via General Packet Radio Service (GPRS). An Estimated Time of Arrival (ETA)

algorithm running on the server predicts the arrival times of buses at their bus stops based on real-time observations of the buses' current Global Positioning System (GPS) coordinates. This information is displayed and announced to passengers at bus stops using station units, which periodically fetch the required ETA from the server via GPRS. The features of their system include: (a) a route-creator utility which automatically creates new routes from scratch when a bus is driven along the new route, and (b) voice tagging of the stops and points of interest along any route.

One of the main limitations of the system developed by Ganesh et al. (2012) and Chandurkar et al. (2013) is that their ETA prediction algorithm respectively, does not consider the effect of what happens when there is no network connectivity between the bus station and the server, more especially places where there is no network coverage. Data could be lost as a result of low/no network coverage during data transmission between a Bus Simulator and the Webserver which could lead to incorrect historical average velocity computation, inaccurate travel time computation for those links, as well as inaccurate information required during automatic route generation.

In this research paper, we propose an efficient model that improved the accuracy of LTT computation and ETA prediction in a public vehicle tracking system, considering a wide range of factors that could affect the vehicle arrival time/arrival time prediction.

III. Research Methodology

With the need for a generic approach to modelling, design and management of public vehicle tracking system, an Object Oriented Analysis and Design Methodology that combines ideas from object-based and system-engineering development was adopted. A step-by-step procedure was used to transform an operation concept through a system-level design. Public vehicle tracking system application areas can be organized into conceptual class hierarchies suitable for reuse. The Unified Modelling Language (UML), a collection of several diagrams capable of modelling public vehicle tracking system behaviour and structure was employed. Key aspects of front-end development include a use-case model for a visual representation of high-level management functionality and a domain model for building an understanding of the domain in which the management system is being developed.

In order to enhance the ETA prediction by ensuring that the link's travel time for all the links in a given route reflects the present/current real time traffic situations at any given point in time, the mobile application called Bus Simulator stores the GPS location updates in the mobile device local database when there is no network connectivity between the Bus Simulator and the Webserver, and finally transmits the stored GPS location updates to the Webserver whenever the network connectivity is re-established. These GPS location updates on getting to the webserver are used to refine the links travel times for the passed links, and this gives a better approximation to ETA prediction.

The system was implemented using C#.Net, ASP.Net, Windows Phone SDK, SQL Server database, SQLite mobile phone database, IIS Webserver, JavaScript, JQuery, AJAX, JSON, and GIS Map APIs such as Google Map, Bing Map, and Open-Street Map for location displaying.

IV. The System Architecture

The new system consists of a data collection unit in each vehicle called Bus Simulator (a mobile application that runs on a GPS enabled mobile device), a central location server (Webserver), and user interfaces (Application Simulators) for the commuters. Figure 1 shows the architecture of the system with the various units.

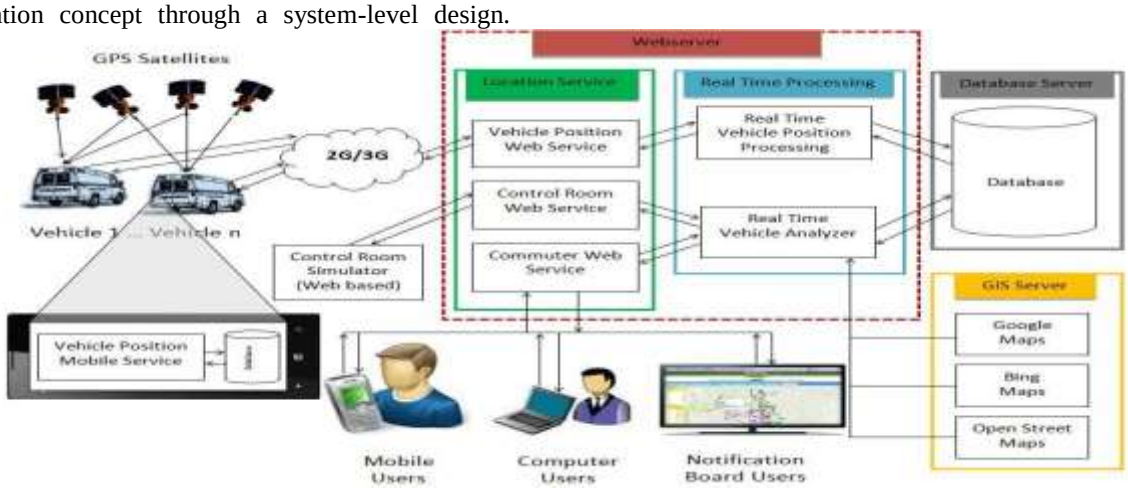

Figure 1: Architecture of the System

A. The Vehicle Unit

The vehicle unit is made up of a Bus Simulator App installed on a GPS enabled mobile device that periodically computes the vehicle's current/present location (latitude, longitude, speed, time etc.) and transmits this location update to the Webserver using a mobile wireless communication network. The Vehicle Position Mobile Service does this automatically, and if there is no connectivity between the Bus Simulator and the Webserver, then the Bus Simulator stores the location update in its local database, which can be transmitted to the Webserver whenever the network connectivity is re-established.

B. The Vehicle Position Web Service Unit

The Vehicle Position Web Service receives location updates from a number of vehicles and passes the request to the Real Time Vehicle Position Processing unit, which handles the link updater, as well as ETA computation; updates the database and finally sends a success/failure signal through the vehicle position web service to the Bus Simulator.

C. Real Time Vehicle Position Processing Unit

Real Time Vehicle Position Processing Unit receives signal from the Vehicle Position Web Service and determines the vehicle's current link, calculates the passed links travel time as well as predicting the arrival time of the vehicle to each of its subsequent bus stops, updates the database and finally transmit a success/failure signal to the Vehicle Position Web Service which communicates back to the Bus Simulator through an HTTP get request using a mobile wireless communication network.

D. Control Room Simulator Unit

Control Room Simulator Unit is a web base application that enables the administrators to interact with the services of the Control Room Web Service.

E. Control Room Web Service Unit

Control Room Web Service Unit authenticates the System Administrator as well as the Scheduling Officers. It allows the administrators to update the database and analyze the information in the database through the Real Time Vehicle Analyzer Unit.

F. Commuter Web Service Unit

Commuter Web Service is the web service that enables the Mobile Users, the PC Users, and Notification Board Users to access the tracking history of a given vehicle along a given route at any given point in time. The Commuter Web Service fetches the vehicle tracking history from the database through the services of Real Time Vehicle Analyzer, and uses the services of the GIS Server APIs such as Google Map, Open Street Map or Bing Map to analyze the vehicle tracking history.

G. GIS Server Unit

A Geographic Information System (GIS) is a system designed to capture, store, manipulate, analyze, manage, and present all types of spatial or geographical data. With the help of the GIS APIs such as Google Map, Open Street Map and Bing Map, each vehicle's tracking history can be generated and published to the Commuter Web Service available for the end users.

H. Computer Users

The tracking history of each vehicle is provided to the computer users through the Commuter Web Service unit. Computer users can view the present position of all the buses on the route map through a mobile app, PC or a Notification Board.

I. Bus Stop Notification Board

The real time arrival information of buses at bus stops is provided in the form of rolling displays. It helps the passengers to make efficient use of time. When this information is disseminated to passengers, they can spend their time efficiently and reach the bus stop just before the bus arrives, or take alternate means of transport if the bus is delayed. This unit periodically fetches the required ETA from the server via a mobile wireless communication network.

J. Mobile Users

In today's world, technology is highly needed on our fingertips. This is a mobile application which helps the mobile users to view the tracking history of a particular bus. The passenger supplies the route, vehicle id, and the schedule detail to the app, and the entire map of the city, the tracking history and the ETA will be displayed on the screen. It fetches the ETA of the requested route and provides the real time information to the passenger. This will make the public transport system competitive and passenger-friendly.

V. IMPLEMENTATION

This system is implemented using the following algorithms:

A. Bus Simulator Tracking Algorithm

Algorithm A shows the algorithm for the Bus Simulator 'Start Tracking' event. The Bus Administrator (Driver/Conductor) will select the route and the schedule information from the Bus Simulator App and hits on the 'Start Tacking' button. Then, at every 30 seconds interval, the algorithm will compute the GPS coordinates of the bus, check whether internet connectivity exist so as to transmit the current GPS coordinates as well as the stored GPS coordinates to the webserver ; if the internet connectivity does not exist then the computed GPS Coordinates will be stored in the mobile device local database.

Algorithm A: Bus Simulator Tracking Algorithm

```
Selects the route;
Selects the Schedule Information;
For every 30 seconds {
    Compute the current GPS location information (Latitude, Longitude, Speed, Time and Direction) of the vehicle;
    If Internet Connection exist and ConnectionCost < DataBundleAvailable {
        Transmit the current computed GPS location information to the Webserver;
        If GPS location information exist on the local database {
            For each GPS location information on the local database {
                Retrieve the GPS location information on the local database;
                If Internet Connection exist and ConnectionCost < DataBundleAvailable {
                    Transmit the retrieved GPS location information to the Webserver;
                    If (successfully transmitted) {
                        Delete the GPS location information from the local database;
                    }// End If successfully transmitted
                }// End If Internet Connection exist
            }// End For each GPS location info
        }// End If GPS location information exist
        Else {
            Store the current computed GPS location information on the local database;
        }// End Else
}// End For every 30 seconds
```

B. Route Generator Algorithm

A novel method has been developed to automate the process of creating new routes and populating the database, with little human intervention. Figure 2 shows a bidirectional graph used in ETA prediction for calculating the estimated time of arrival. The bus stops are represented as nodes and the route is in the form of sequence of links

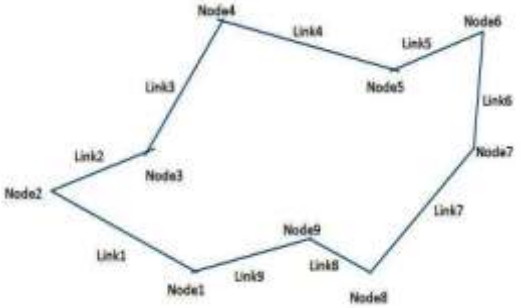

Figure 2: Route Creation Identified by Links

A particular route is identified by its unique id. To create a new route, the Bus Simulator unit is taken along the required route. The unit keeps logging the position coordinates along the route in the local memory at a fast rate. Whenever a bus-stop needs to be indicated, the driver/ conductor enters the name of the bus-stop and click on the submit button. At the end of the journey, the Bus Simulator transmits these position coordinates to the Webserver which receives the position updates and creates nodes and links for the new created route. The algorithm considers three nodes (n_i, n_{i+1}, n_{i+2}) at a time and checks how collinear they are. In Figure 3, if the distance **x** (the Euclidean distance between node n_i and node n_{i+1}) is closer to the threshold (100m in the implementation) as compared to the distance **y** (the Euclidean distance between node n_{i+1} and n_{i+2}) then the second node is retained, this link's length becomes **y** and the travel time becomes $t_{i+1} - t_i$; otherwise the second node is skipped and a new node tuple (n_i, n_{i+2}, n_{i+3}) is considered and this link's new length becomes **z** (the Euclidean distance between node n_i and n_{i+2}) and the travel time becomes $t_{i+2} - t_i$. The algorithm repeats starting with the second node in the tuple. However, all bus-stops are included irrespective of the linearity constraint.

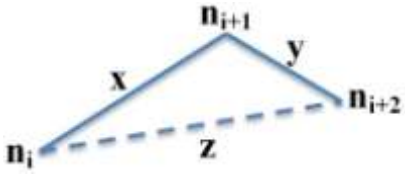

Figure 3: Route Creation Algorithm Node Tuple

The Euclidean distance between two points **P_1(lon1, lat1) and point P_2(lon2, lat2)** can be calculated using the following equation:

dlon = lon2 - lon1
dlat = lat2 - lat1
x = (sin(dlat/2))^2
*y = cos(lat1) * cos(lat2) * (sin(dlon/2))^2*
a = x + y
*c = 2 * atan2(sqrt(a), sqrt(1-a))*
*d = R * c (where R is the radius of the Earth)*

Algorithm B shows the algorithm for the Route Generator.

Algorithm B: Route Generator Algorithm

```
NewLinksList = Null;
LocationList = Null;
Link = Null;
SerialNumber = 1;
Select the vehicleID and the Scheduling Type;
LocationList = Retrieve the list of all the location updates for the selected route ordered by gpsTime;
LocationListCount = The number of items in the
```

```
LocationList;
If(LocationListCount > 1) {
  //At least we have up to 2 points/nodes
   Node1 = LocationList[0];
   Node2 = LocationList[1];
   NewLength = Euclidean distance between Node1 and Node2;
   Link.RouteID = The selected RouteID;
   Link.FirstNode = Node1;
   Link.LastNode = Node2;
   Link.Length = NewLength;
   Link.SerialNumber = SerialNumber++;
   Link.TravelTime = Link.LastNode.gpsTime – Link.FirstNode.gpsTime;
   NewLinksList.Add(Link);
   Route.RouteID = The selected RouteID;
   Route.SourceNode = Node1;
   Route.Length = Link.Length;
   For (int i =2; i < LocationListCount; i++) {
      //Start the optimization process
      Node1 = Link.FirstNode;
      Node2 = LocationList[i];
      NewLength = Euclidean distance between Node1 and Node2;
      OldLength = Link.Length;
      //Use a ternary operator to determine //between the two pints, the one closer to 100
      OptimizedLength = (|(NewLength - 100)| < |(OldLength-100)|) ? NewLength : OldLength;
      If ((OptimizedLength == NewLength) && (Link.LastNode.IsBusStop == False)) {
         Link.RouteID = The selected RouteID;
         Link.FirstNode = Node1;
         Link.LastNode = Node2;
         Link.Length = OptimizedLength;
         Link.TravelTime=Link.LastNode.gpsTime – Link.FirstNode.gpsTime;
         Route.Length += (Link.Length – OldLength);
      }// End If
      Else {
         Link.RouteID = The selected RouteID;
         Link.FirstNode = Link.LastNode;
         Link.LastNode = Node2;
         NewLength=Euclidean distance between Link.FirstNode and Link.LastNode;
         Link.Length = NewLength;
         Link.TravelTime=Link.LastNode.gpsTime – Link.FirstNode.gpsTime;
         Link.SerialNumber= SerialNumber++;
         NewLinksList.Add(Link);
         Route.Length += Link.Length;
      }// End Else
   }// End For
   Route.DestinationNode = Link.LastNode;
   Route.Update(); // Database update
   Delete previous nodes and links for this route from the database;
   For each Link in NewLinksList{
      Insert Link.Node1 into Node Table;
      Insert Link.Node2 into Node Table;
      Insert Link into Link Table;
   }// End For each
   Display a success message;
}// End If
```

C. Real Time Vehicle Position Unit Algorithm

Real Time Vehicle Position Unit algorithm consists of three major segments: Current Location Detector, Link Updater, and ETA Calculator.

i) Current Location Detector: Current Location Detector algorithm determines the exact location of the bus on the linearized route based on the current GPS location update.

ii) Link Updater: Link updater calculates the LTTs required by the ETA calculator. Whenever a bus position update is received from the vehicle unit, the link updater calculates the travel times for all links traversed by the bus from the previous known position. The link updater requires distance of each link. The weighted average of the previous value and the actual travel time obtained for the current bus is stored as the link travel time in the Link table. To compute the average velocity, the weights are 90% for the previous average velocity and 10% for the current velocity. To compute average link travel time, the weights are 70% for the previous average and 30% for the current value. For an update rate of two per minute used in trial runs, these weights give a good approximation of the average values.

Link updater locates the bus position along the current route of the bus. The link updater then calculates the time required to reach the end of the current link and updates the estimated end time information in the Bus Position table. If the bus enters a new link, the entry time for the new link is stored in the Bus Position table against the bus and the travel time for all the crossed links is calculated. This time is also the exit time for the previous link. The time difference between the exit time and the previously recorded link entry time gives the link travel time for the crossed links. The travel times for links are a function of their lengths. Thus, when more than one link is traversed between updates, the individual link travel times are computed as fractions of the total travel time, with the fraction for link i being the ratio of the length of the i^{th} link to the sum of lengths of traversed links. This ensures that among the traversed links, shorter links have smaller travel times and longer links have larger travel times. The computed link travel times are

averaged with their previous values and the Link table is updated.

For instance, suppose that the last position update received from a bus corresponds to position P_1 and the position update received just now corresponds to position P_c as shown in Figure 4; then let's discuss the actions of the link updater algorithm for this example.

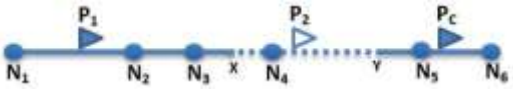

Figure 4: Link Travel Time calculation diagram

The route section consists of the links N_1-N_2, N_2-N_3, N_3-N_4, N_4-N_5, N_5-N_6, where N_i are the nodes comprising the route. In the first pass of the link updater algorithm, the latest bus position is mapped onto the route by checking every link in the route, starting from the current link. In this case, the bus is found to be in the link N_5-N_6. Since it is a new link, the entry time into that link (for this bus) is set to the current time. In the second pass of the algorithm, travel time for the crossed link(s) N_1-N_2, N_2-N_3, N_3-N_4 and N_4-N_5 are updated by partitioning the total travel time among individual link travel times, based on their lengths. Hence, link N_1-N_2 gets a larger travel time than link N_2-N_3. Assuming point X to point Y (i.e. X-Y) represents the area where there is no network coverage, and P_2 represents the position update stored in the mobile device local database as a result of no network connectivity, then at point Y (when the network connection has been re-established) the Bus Simulator will submit the position update P_2 to the webserver which will be used to calculate/refine the travel time for the crossed links N_1-N_2, N_2-N_3, and N_3-N_4. On receiving the current position update P_c, the last position update received from a bus corresponds to position P_2, and the travel time for the crossed link(s) in this case N_4-N_5 is calculated. For the final link, the time to reach the end of the link (node N_6) is found by dividing the Euclidean distance between the bus position and N_6, by the average velocity of the bus. Database updates for bus position, current link, link travel times, estimated end time and average velocity happens at every iteration of the algorithm.

iii) ETA Calculator: This program takes the current bus position, link travel times and estimated time to link-end to predict the ETA for all bus stops. ETA at a bus stop is the time taken for the bus to reach a bus stop. It is calculated as the sum of travel times of the links, starting from the current bus position, up to the given bus stop.

Algorithm C shows the algorithm for the Real Time Vehicle Position Unit when it receives a GPS position update from a particular vehicle.

Algorithm C: Real Time Vehicle Position Unit Algorithm

If (GPS PositionUpdate is valid) {
Store the GPS location updates in the database;
//Current Location Detector
Links = Retrieve a list of all links along this route ordered by their SerialNumber;
D_0 = 10000; //Start D_i with a large number
i =1;
For each (Link in Links) {
L_{cu} = Euclidean distance between Current Vehicle Location and Link.LastNode;
L_{cd} = Euclidean distance between Current Vehicle Location and Link.FirstNode;
L_{ud} = Link.Length;
$D_i = L_{cu} + L_{cd} - L_{ud}$; //The link with the smallest D_i is considered the current link.
If ($D_i < D_{i-1}$) {
j = Link.SerialNumber;
}// End If
i++;
}// End For each
// j holds the serial number of the link with the smallest D_i
//LTT Updater
CurrentLink = Select link from the database where serial number = j
v_a = PreviousPositionUpdate.AVGSpeed;
v_c = CurrentPositionUpdate.Speed;
d = The Euclidean distance between CurrentPositionUpdate and CurrentLink.LastNode;
$v_j = (90.0 * v_a + 10.0 * v_c) / 100.0$;
CurrentPositionUpdate.AVGSpeed = v;
T = a/v_j; //The estimated time to the end of link with serial number j
CurrentPositionUpdate.LinkEstimatedEndTime = T;
L_c = CurrentPositionUpdate.SerialNumber;
L_p = PreviousPositionUpdate.SerialNumber;
PositionUpdateTimeInterval=CurrentPositionUpdate.
EntryTime - PreviousPositionUpdate.EntryTime;
If ($L_c <> L_p$) {
NumberOfPassLinks = $|L_c - L_p|$;
LengthOfPassedLinks = 0;
For (int k=0; k<NumberOfPassLinks; k++) {
LengthOfPassedLinks=Links[L_p + k].Length;
}// End For
For (int k=0; k<NumberOfPassLinks; k++) {
Links[L_p + k].TravelTime = (70.0 * Links[L_p + k].TravelTime + 30.0 * (PositionUpdateTimeInterval * (Links[L_p + k].Length / LengthOfPassedLinks)) / 100.0;
Links[L_p + k].Update();
}// End For
//ETA Calculator

```
    Update bus-stops set ETA = 0 where the serial number < j;
    If (Links[Lc].LastNode.IsBusStop == True){
      Store the value of T as the ETA of this Bus-Stop;
    }// End If
    For each (Link in Links where the serial number > j) {
      T += Link.TravelTime;
      If (Link.LastNode.IsBusStop == True) {
        Store the value of T as the ETA of this Bus-Stop;
      }// End If
    }// End For each
  }// End If (Lc <> Lp)
}// End If (GPS PositionUpdate is valid)
```

VI. RESULTS/SYSTEM OUTPUT

This Real-Time Public Vehicle Mobile Tracking System using GPS Technology was tested in Nigeria. The routes used for the test are shown in Figure 5 and Figure 6 respectively. In Figure 5, the source bus stop is Amajieke Douglas Road Imo State and the destination bus stop is Okwuzi Bus Stop ONELGA Rivers State. In Figure 6, the source bus stop is Okwuzi Bus Stop ONELGA Rivers State and the destination bus stop is Old GRA Port-Harcourt Rivers State.

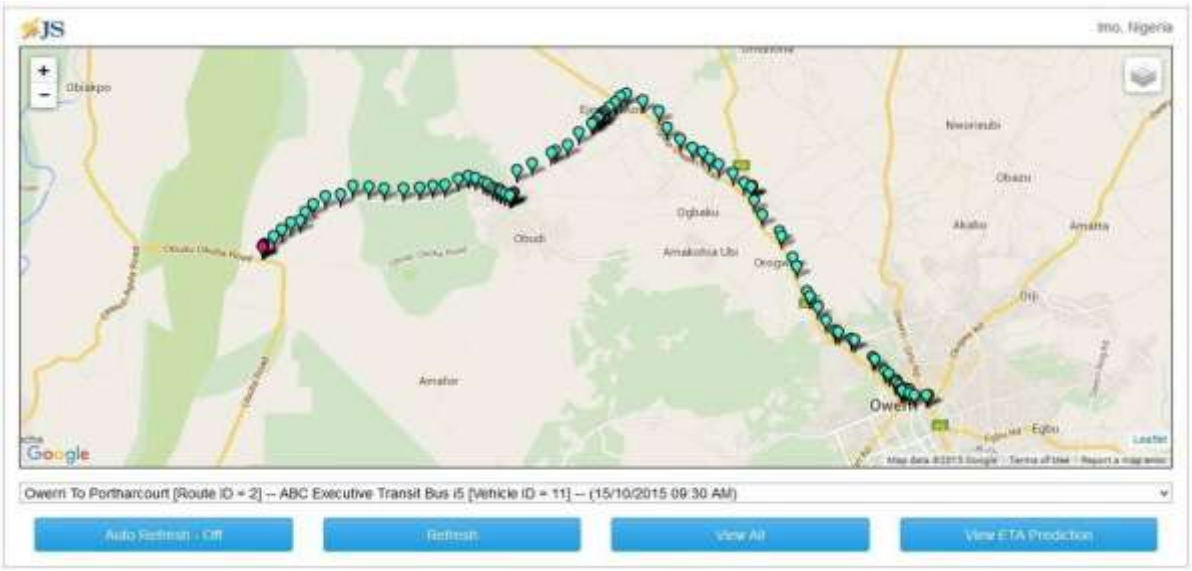

Figure 5: Route Map Tracking History (Amajieke Douglas Road Imo State to Okwuzi Bus Stop Rivers State)

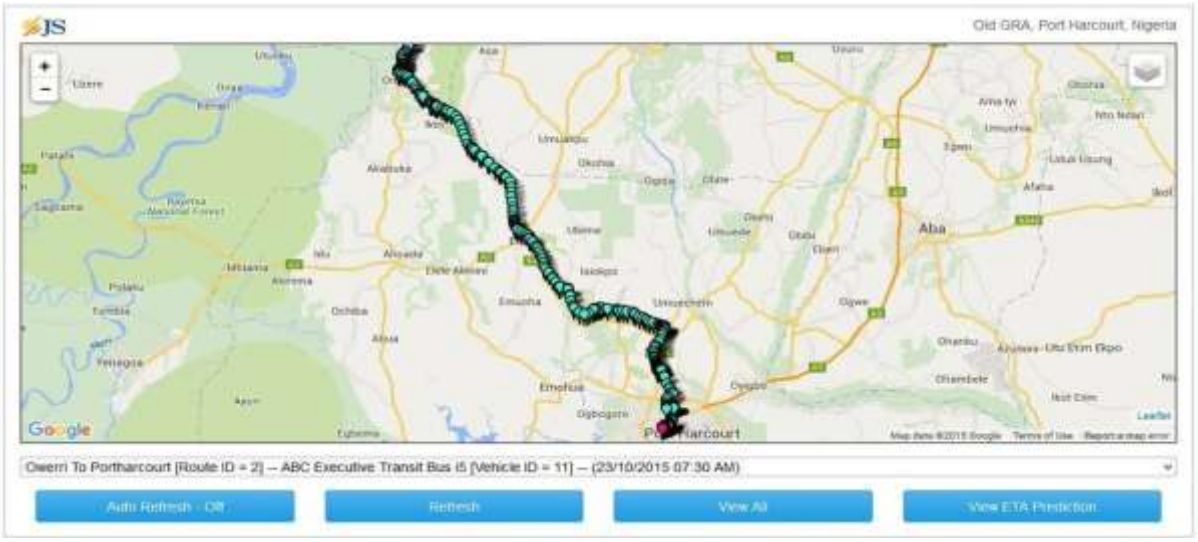

Figure 6: Route Map Tracking History (Okwuzi ONELGA Rivers State to Old GRA Port-Harcourt Rivers State)

Initially, when the vehicle unit (Bus Simulator Mobile App) is activated, it retrieves the vehicle information from the local database, and as well allows the Bus Operator to retrieve the routes and schedules information either from the local database or from the Webserver. When the journey begins, the bus operator hits the "Start Tracking" button as shown in Figure 7; and at every 30 seconds interval, current position update for the vehicle in question is generated and transmitted to the webserver (when there is network connectivity) or stored in its local database pending when the network connection will be re-established.

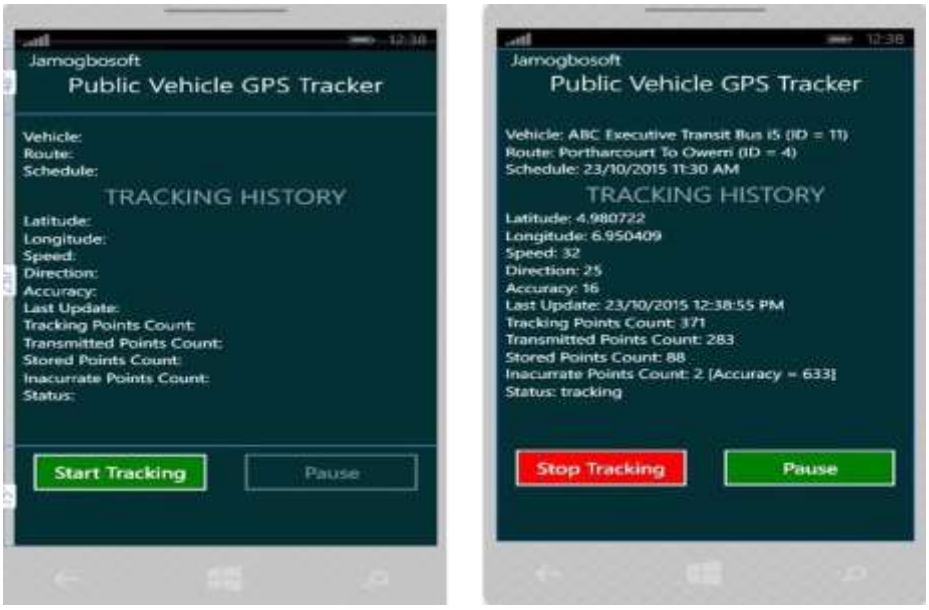

Figure 7: Bus Simulator Track Vehicle Page

Figure 8: ETA Prediction Page

Figure 8 shows the Estimated Time of Arrival (ETA) of a vehicle with Vehicle Id = 11 that moved along the route with Route Id = 2 within the hours of 7:30AM and 11:16AM on the 23rd day of October 2015.

Table 1 and Figure 9 shows the table and the chart of the tracking history as a result of low/no network coverage respectively.

Table 1: Low Network Coverage Tracking History

	Amajieke to Okwuzi	Okwuzi to GRA Port-Harcourt	Okwuzi to Amajieke	GRA Port-Harcourt to Okwuzi
Total Points	104	387	132	507
Transmitted Points	68	283	93	388
Stored Points	36	104	39	119

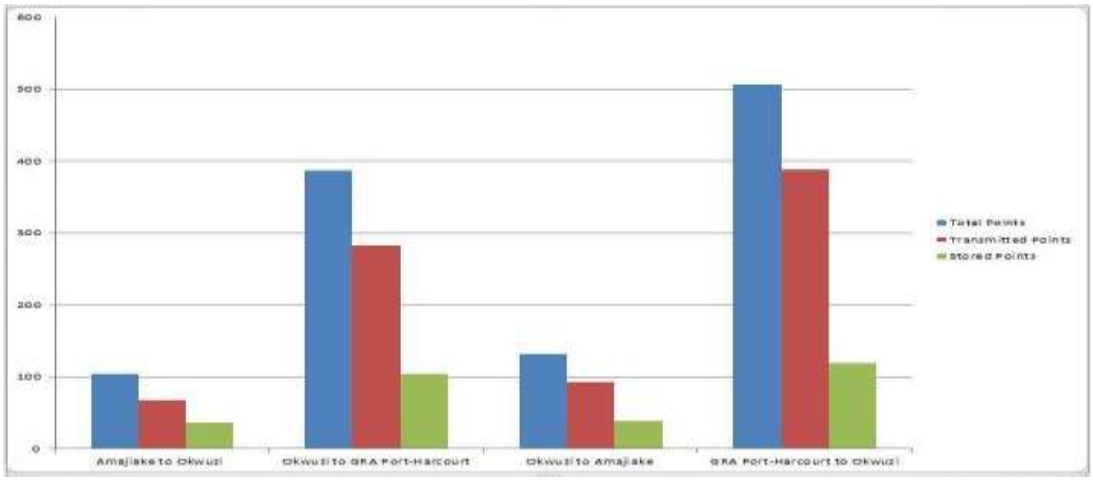

Figure 9: Low/No Network Coverage Tracking History Chart

VII. CONCLUSION

This research work proposed an efficient intelligent transportation system that: provides commuters with exact location of buses on a GIS map (such as Google map, Bing Map and Open-Street Map) through notification board at bus stops or more conveniently through a website and a mobile application; predicts and provides the arrival time of a bus at all bus stops along its route to the passengers/commuters.

In conclusion, this work is useful to public vehicle transportation companies, as successful implementation will not only reduce the long waiting experienced by passengers, but will increase productivity, improve dispatching, reduce the number of private vehicle usage, reduce the quantity of fuel consumption, as well as the rate of pollution on our highways.

REFERENCES

[1] Chandurkar, M., Mugade, S., Sinha, S., Misal, M. & Borekar, P, (2013), Implementation of Real Time Bus Monitoring and Passenger Information System. *International Journal of Scientific and Research Publications*, 3(5), 1-5.

[2] Chung, E., & Shalaby, A., (2007), Expected time of arrival model for school bus transit using real-time global positioning system-based automatic vehicle location data, *Journal of Intelligent Transportation Systems*, 11(4), 157-167.

[3] D'Angelo, M., Al-Deek, H. & Wang, M., (1999), Travel-time prediction for freeway corridors, *Transportation Research Record, Journal of the Transportation Research Board*, (1676), 184-191.

[4] Ganesh, K., Thrivikraman, M., Kuri, J., Dagale, H., Sudhakar, G., & Sanyal, S., (2012), Implementation of a real time passenger information system, *arXiv preprint arXiv:*1206.0447.

[5] Sun, D., Luo, H., Fu, L., Liu, W., Liao, X., & Zhao, M., (2007), Predicting bus arrival time on the basis of global positioning system data. *Transportation Research Record: Journal of the Transportation Research Board*, (2034), 62-72.

[6] Weigang, L., Koendjbiharie, M., de, M., Ricardo, C., Yamashita, Y. & Maciver, A., (2002), Algorithms for estimating bus arrival times using GPS data, In *Intelligent Transportation Systems, 2002, Proceedings of the IEEE 5th International Conference on* (pp, 868-873), IEEE.