Vol. 7(25), Jul. 2017, PP. 3454-3473

Contents list available at IJMEC

International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)

Journal Homepage: www.aeuso.org

PISSN: 2411-6173 - EISSN: 2305-0543

Performance Analysis of a Real-Time Cloud Based Bus Tracking System with Adaptive Prediction

T.P. Fowdur* and M.N. Khodabacchus

Department of Electrical and Electronic Engineering, University of Mauritius

Réduit, Mauritius

Phone Number: *+2304037888*

****Corresponding Author's E-mail:*** *p.fowdur@uom.ac.mu*

Abstract

Many public transport users rely on real-time estimations of bus arrival times at transit stops to plan their trips. The accuracy of such predictions depends on the prediction algorithms used as well as the system architecture. In this paper a real-time cloud based bus tracking system is developed using a microcontroller with a GPS/GSM shield to record and relay the position of the vehicle being tracked to a cloud server. A control station then accesses the vehicle position data in real-time and applies prediction algorithms to the data. The predictions are then sent to the cloud server which relays the information to the end users on their mobile phones. The prediction system at the control stations can adaptively chooses between five different prediction algorithms namely ARIMA, KNN, regression, polynomial fitting and Moving Average based on the RMSE criterion. A detailed performance comparison between the five algorithms and the adaptive one has also been made on two different bus routes for ten week days. It was observed that the adaptive algorithm provides a lower MSE than all the other predictive algorithms when compared with the actual bus arrival times.

Keywords: *Adaptive prediction; real time, Arduino, GPS, GSM, Cloud server*

1. Introduction

Bus transport is the mainstream public transportation medium in several countries. With advances in the field of ICT and Internet of Things (IoT), it has now become a common practice for several urban transportation operators to use to use location reporting systems such as GPS devices on-board their fleet, with the primary purpose of monitoring and managing their fleet as well as providing arrival time information to bus users [17]. These systems are normally enabled with a GPS based location tracker with some form of connectivity such as GSM to send the positional data collected for predicting the arrival times. Predicting bus arrival times is in fact a well established problem and several techniques such as Kalman Filtering [7], Support Vector Machines [23], and K-Nearest Neighbors [8,15] have been used to create prediction models. There are also several research that have proposed different bus arrival time prediction systems. An overview is given next. In [17] methods for predicting bus arrival times using location reports from GPS tracking devices were investigated. The system developed employed on a complex event processing engine that incorporated an algorithm to predict the estimated arrival time in real time continuously. The developed system in the first phase was evaluated using a vehicle simulator that generated vehicle trajectories along real public transportation routes. In

Article History:
Received Date: Jan. 19, 2017
Accepted Date: Apr. 12, 2017
Available Online: May. 01, 2017

[6] the authors used data from a local transit agency, to create prediction models for bus arrival times with the K-Nearest Neighbors and Kernel Regression techniques employing seven sets of features. It was observed the optimum feature provided a RMSE of 34 seconds which was a major improvement over the baseline model which generated arrival time predictions with a RMSE of 99 seconds. A feasibility analysis on the application of IoT in bus transportation systems in Singapore was performed in [2]. A technical architecture was designed for an app where IoT can be used to predict arrival timings of buses as well as the crowd inside each bus [2]. In [21], a model-based algorithm for real-time automated bus arrival prediction was developed which accounted for the time and space fluctuations involved during travel.

The Double Seasonal Holt-Winter's Exponential Smoothing technique was used to predict the travel time of buses, which allows accounting for time-trend and seasonality in the data series. The results showed that the proposed model could provide travel time estimates within an accuracy of approximately 10%. In [9] a GPS based bus arrival time prediction system was proposed which mainly requires low cost cellular signals. The system was self-calibrating and proved to be functional and easily deployable with minimum equipment set required [9]. A new system for predicting the expected bus arrival times was proposed in [10]. Real-time location data from GPS receivers was combined with average travel speeds of individual route segments by the prediction algorithm. The algorithm also took into account historical travel speed, temporal and spatial traffic variations. A geographic information system–based map-matching algorithm was used to project each received location onto the underlying transit network. The proposed system was could satisfactorily predict bus arrival times and provided perfect performance in predicting travel direction.

Although several papers have evaluated the performance of bus arrival time prediction algorithms, very few have reported the use of an adaptive algorithm. Moreover, despite the well established bus network in Mauritius, passenger information systems are yet to be deployed. This paper proposes the use of an adaptive algorithm which could select between the most appropriate predictor for a given observation by using the Root Mean Square Error (RMSE) criterion. A real time cloud-based bus arrival time prediction scheme has also been set up. The system consists of a GPS/GSM tracking device installed in a bus. The tracking will then relay the bus location to a cloud server. A control station will then access the location data from the server and perform predictive analytics using five different algorithms, namely Autoregressive Integrated Moving Average, K-Nearest Neighbors, regression, polynomial regression and Moving average. The RMSE criterion was then used to adaptively select the best predictor to provide bus arrival time estimates to the end users via the cloud server. The data was collected on two bus routes in Mauritius for a period of 10 week days during the peak hour. It was observed that the adaptive algorithm significantly outperformed the other entire prediction algorithms by providing a MSE of only 0.015 with respect to the actual arrival time.

The organization of the paper is as follows. Section two gives an in-depth background on the most widely used prediction techniques as well as the adaptive algorithms used. Section three describes the complete hardware and software architecture of the proposed bus prediction system. Section four deals with system testing and performance analysis. Section six concludes the paper.

2. Background

There are several prediction algorithms that can be incorporated into a bus arrival time prediction system. The main algorithms investigated in this work are as follows:

1. Moving Average
2. Linear regression
3. Polynomial regression
4. Autoregressive Integrated Moving Average (ARIMA)
5. K-Nearest Neighbors (KNN)
6. Neural Network (NN)
7. Adaptive algorithm

Moving Average (MA) is a time series prediction algorithm which is based on the averages of the previous observation. Each average is computed by dropping the oldest observation and including the next observation [13]. In this work, a simple moving average scheme is adopted and is given by the equation:

$$Y_{i+1} = \frac{1}{n}\sum_{k=i-n}^{i} Y_k \quad (1)$$

Where Y_{i+1} is the predicted value (speed of vehicle), n is the number of previous observations and Y_k is the value of the *k* previous observations.

The linear regression technique fits a straight line equation through the data that predict y-axis values based on x-axis[24].The model is expressed as:

$$T_{i+1} = a + bX_i \quad (2)$$

Where T_{i+1} is the predicted time, a is the intercept on the y-axis, *b* is the slope of regression and X_i is the independent variable. The least square method is used to estimate the intercept and slope of regression. Once the parameters are obtained, the predicted time of T_{i+1} at any given point X_i (distance in this example) is calculated as $a+bX_i$.

In polynomial regression, a dependent variable is regressed on the powers of an independent variable [18]. Polynomial regression is expressed as:

$$T_{i+1} = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \cdots + \beta_k X_i^d \quad (3)$$

Where β_0 is the y-intercept, each β_i , i = 1,2,..,d , is the slope of regression with respect to the variable X_i and *d* is the degree of polynomial.

Another time series algorithm is the ARIMA (Autoregressive Integrated Moving Average). ARIMA models are denoted by ARIMA (p, d,q) where p,d,q are numbers representing the order of autoregression, degree of differencing and order of moving average. ARIMA (0,1,0) is used in this work since the speed of the vehicle does not have the same statistical properties over time[14]. The prediction equation of this model is:

$$Y_{i+1} = \mu + Y_{i-1} \quad (4)$$

Where μ is the average period-to-period change in *Y* and Y_{1-1} is the previous value.

K-Nearest Neighbors is an example of a machine learning algorithm. In this scheme, K-nearest observations are combined to obtain a prediction. The Euclidean Distance metric (Equ.5) is used to retrieve the K instances closest to the unknown observation. From [11], the optimal value of K is equal to the square root of the number of instances in the data set.

$$Euclidean\ Distance\ Metric = \sqrt{(x_j - x_k)^2} \quad (5)$$

Where x_j is the unknown observation and x_k is the observation recorded. Once all the distances between the unknown observation and all the observations are computed, the K observations with smallest distances are selected for prediction. The KNN prediction is the average of the K nearest observations and calculated as follows:

$$Y_{i+1} = \frac{1}{K}\sum_{i=1}^{K} y_k \quad (6)$$

Where K is the neighborhood size and y_k is the closest sample observation

Neural Networks (NN) are statistical models that can solve non-linear complex problems. A Neural Network first learns from samples and captures functional relationships among the data. After the learning process, the system is able to produce predictions from input values which were not in the training samples. The NN is composed of a set of neurons arranged in layers (input, hidden and output)

which process the information. The hidden layer generates weights that define the connection between input and output. The outputs are calculated as shown in Equ.7 [3].

$$a = f(\sum_{i=1}^{n} w_i \, p_i + b) \tag{7}$$

Where a is the predicted result, p_i is the value of ith input, w_i is the value of ith input, b is th bias and f is the activation function. The activation function implemented is a sigmoid function which is used to determine relationship between inputs and outputs of the network. The learning process is performed by the back-propagation algorithm. The steps involved in the process are as follows:

1. Initialize the hidden layer with random weight.
2. Training data are fed to the neural network
3. Normalize the input data in the range (-1,1) [3]using equation 8.
4. If the desired output is not obtained, then a back-propagation signal is used where the weights in the hidden layer are adjusted.

$$X_s = \frac{2(X - X_{min})}{X_{max} - X_{min}} - 1 \tag{8}$$

Where X_s is the normalized value,X is the original value, X_{min} and X_{max} are the minimum and maximum value of X.

The proposed architecture of the neural network is shown in figure 1.

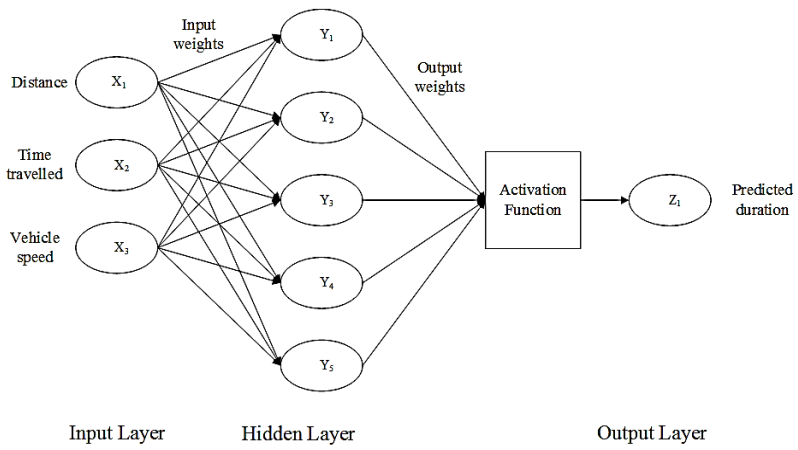


Figure 1: Neural network architecture

In the input layer, the number of neurons is the same as the number of input data. The vehicle information such as the distance travelled, time taken and the speed of the vehicle are used as input to forecast the duration time. With a trial and error approach, the number of neurons in the hidden layer is decided to be seven. The neurons in the neural network are connected to each other by a communication links which are associated with weights. Epoch [1] is the iteration through which the neural network is presented with a new input pattern and the network's weights are updated.

The aim of the adaptive algorithm is to select the best prediction from all the prediction algorithms adaptively for any given time instant. The adaptive algorithm used in this work is based on the RMSE criterion. A cross validation [20] technique is implemented on the data to generate training and test data set. Each prediction algorithm uses a training set (t_1,t_2,…t_s) to estimate a forecast for t_{s+1}. The size

of the training data set is then incremented each turn to t_1-t_{s+1}, t_1-t_{s+2}, etc. and the prediction process is repeated. The squared error formula is used to compute the error deviation.

$$Squared\ Error = (p_i - p_0)^2 \tag{9}$$

Where p_i is the predicted value and p_0 is the expected value. The performance of each prediction algorithm is then evaluated using the root mean square error (RMSE) as follows:

$$Root\ Mean\ Square\ Error(RMSE) = \sqrt{\frac{1}{v}\sum_{i=1}^{v}(p_i - p_0)^2} \tag{10}$$

Where v is the number of data points in test data, p_i is the predicted value and p_0 is the expected value. The algorithm with the lowest RMSE is chosen as predictor for the next travel time, t_8.

3. Proposed bus tracking system

A block diagram of the proposed system is given in Figure 2.

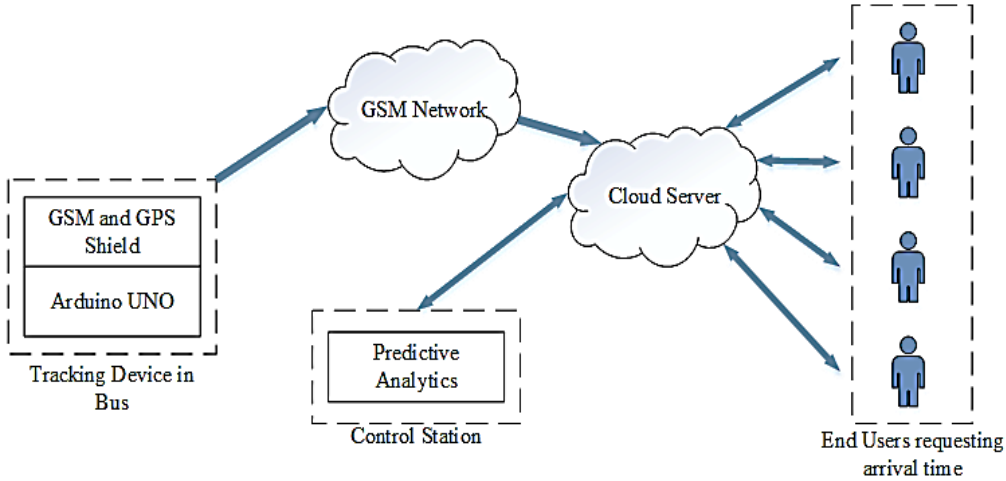


Figure 2: Block diagram of proposed system

The vehicle to be tracked (Bus) is equipped with a tracking device which is an Arduino microcontroller[26] mounted with a Global Positioning System(GPS) and Global System for Mobile communication(GSM) shield. The vehicle location i.e. GPS coordinates are transmitted in real-time to the cloud server via the GSM network. The control station obtains the GPS coordinates from the cloud server and performs predictive analytics to predict the arrival time of the bus at the destination. This prediction is sent back to the cloud to the end users' devices. The end users' devices contain an android application to query for the arrival times. The next subsections describe the hardware and software configuration details for the vehicle tracker, cloud server, control station and end user application.

3.1. Vehicle tracking configuration and programming

The core elements of the vehicle tracking system are:

1. Arduino UNO Microcontroller
2. Adafruit GPS Module
3. SIM900 GSM/GPRS Module

The Arduino UNO [26] is the brain of this system. It controls the GPS and GSM/GPRS modules simultaneously. It has a programmable flash memory and a series of input/output pins to interface external shield.

The Adafruit GPS V3 [25] is incorporated in the system to provide vehicle position and timing information in real-time from the Global Positioning System (GPS).

The SIM900 GSM/GPRS module [5] establishes the connection between the tracking device and remote server. It makes use of a SIM card and cellular antenna. The GPRS service enables information to be transmitted over the GSM network as HTTP packets. The SIM900 shield is controlled by using a set of AT commands [27] through serial communication.

The proposed circuit design is given in Figure 3:

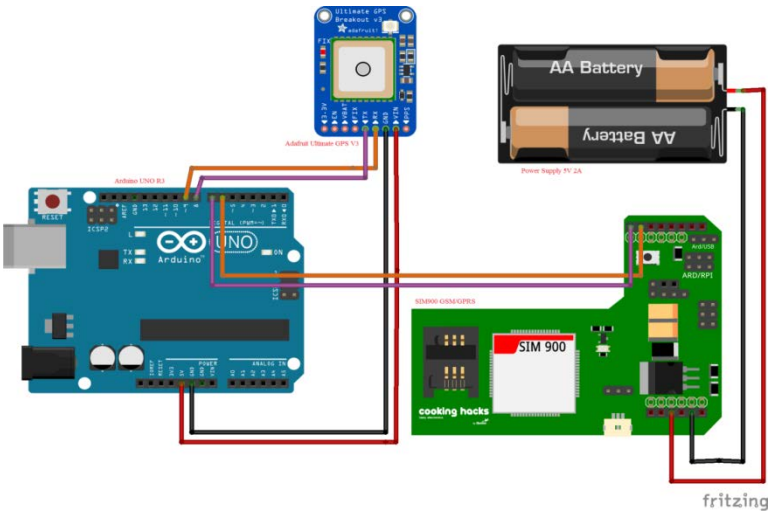


Figure 3: Circuit design

The GPS module and microcontroller are interconnected by four wires. Two wires are used to power on the GPS module from the microcontroller's 5V output. The other two are signal wires which connect the GPS Transmit (TX) and Receive (RX) to the microcontroller's digital input pin 8 and 9.

The GSM shield is powered by an external supply of 5V with a minimum current of 1.5A. To use Pin 7 and 8 as Transmit (TX) and Receive (RX), the jumper setting is set to software serial mode.

An Arduino Integrated Development Environment (IDE) is installed in order to program the Arduino Uno microcontroller. The Arduino IDE compiles the written program in C/C++ language and uploads it to the board via the Universal Serial Bus (USB) port. External libraries are added to the Arduino IDE for accessing data from the GPS and GSM modules. They are described as follows:

1. The SIM900 GSM library which consists of AT commands list to operate the GSM shield.
2. An Adafruit GPS library [29] which is used for communicating and extracting data from GPS National Marine Electronics Association (NMEA) sentences.
3. A Software Serial library that enables serial communication on other digital input/output pins.
4. An AltSoftSerial library [30] that allows Arduino Uno to communicate with another serial device (GPS and GSM).

The overall structure of the Arduino program codes is given as follows:

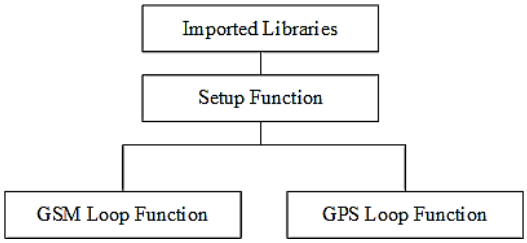


Figure 4: Structure of Arduino program code

The Arduino Setup function initialises the GPS and GSM module and sets the data rate in bauds. In this system, a baud rate of 19200 is used which is compatible with both the GPS and the GSM/GPRS modules.

The Arduino Loop function runs the program consecutively allowing it to change and respond. There are two loops defined in this work, GPS and GSM. The GPS loop constantly reads data from navigation satellite and passes the decoded information to the GSM. The GSM monitors its input port and transfers the location and time acquired from the GPS through General Packet Radio Service (GPRS) to the Uniform Resource Locator (URL) defined using HTTP GET [12] method.

The HTTP GET method requests data from the server which in turn returns a response to the client submitted the HTTP request. In this system, the microcontroller transfers the GPS coordinates to the remote server by inserting the GPS values in the URL of a GET request. The AT commands are entered in the GSM program code to initiate and run the HTTP service. The algorithm is illustrated in Figure 5.

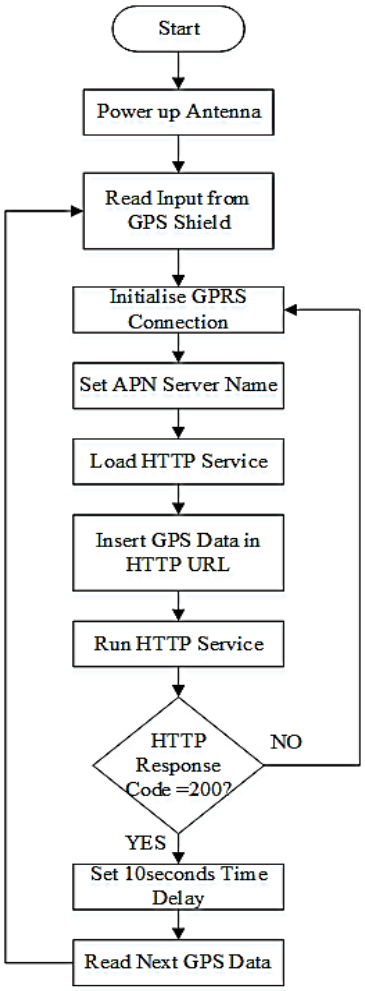


Figure 5: Arduino GSM/GPRS loop

3.2. Cloud server set up and programming

In this work, a cloud server is set up that provides services such as data storage, access and computation without requiring resource usage at the end user. MySQL [16] and PHP [22] are the backbone of the cloud server where the interaction with Google Application Programing Interface (API) [28] service results in a complete system interfacing between end user and control station. Fig.6 shows the structure of the cloud server proposed.

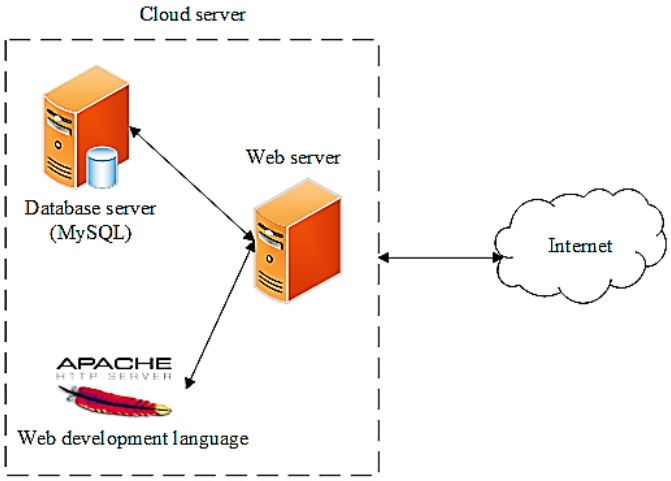


Figure 6: Cloud server structure

The cloud system is run on the web host server which integrates various development languages and software known as Apache [4]. PHP is a web development language which is embedded in Hypertext Transfer Protocol (HTML) page to store received GPS information into a table created by a MySQL database server. With an authentication key from a Google server, the API server enables the cloud server to access a Google map in order to obtain the travelling distance and location address.

The database structure implemented is given in Tables 4(a) and 4(b).

Table 1: Database structure

(a) GPS record table

Field	Type	Size
ID(PK)	Integer	3
Time	Timestamp	15
Latitude	String	15
Longitude	String	15
Distance	Double	5
Location	String	50

(b) User request table

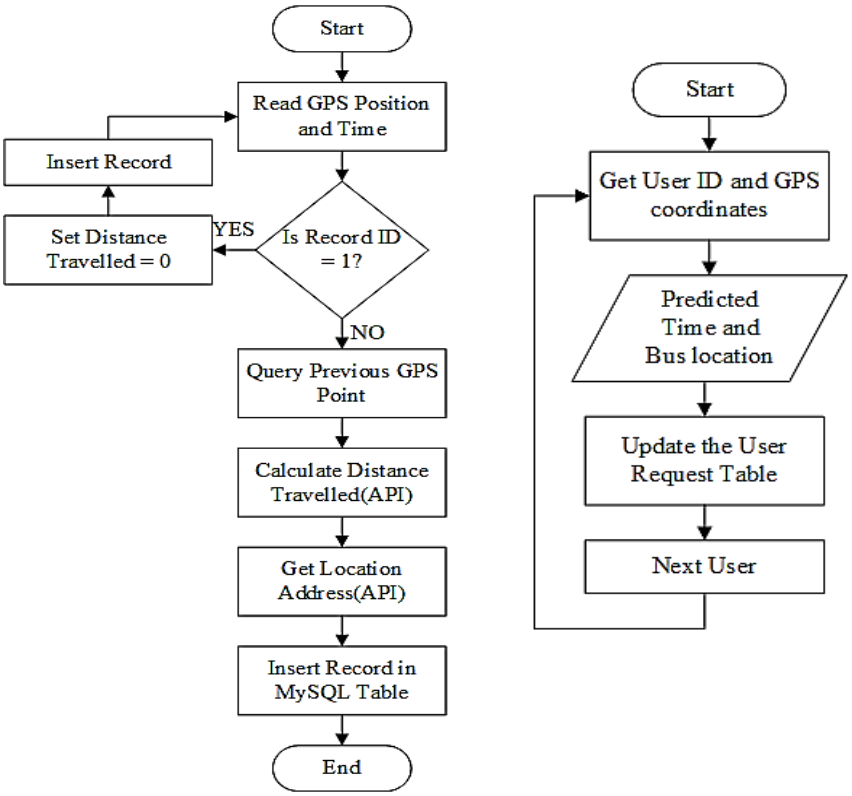
Field	Type	Size	Attribute
UserID(PK)	Integer	3	Auto increment
Latitude	String	15	
Longitude	String	15	
Bus Number	Integer	3	None
Arrival Time	Timestamp	15	
Location	String	50	

The GPS record table stores the received GPS data obtained from the microcontroller. It consists of a Record ID field (primary key) and vehicle's location. The Distance field indicates the distance travelled by the vehicle. The location field stores the address information received from the Google API.

The User request stores the end user GPS position, requested bus number, bus current location and predicted arrival time. It consists of the User ID field as primary key to track multiple user requests. The field has an auto increment attribute which assigns sequence numbers automatically when a user request is inserted.

A PHP file (dbconnect.php) is written that directly connects to the database server in order to manipulate the table. A second PHP file (index.php) is created which is responsible for interfacing the Arduino microcontroller. The PHP command $_GET [] is used to retrieve the transmitted data in the URL. The Google API is executed to obtain the travelled distance and address location of the bus. The vehicle information is then entered into the GPS record table by running the MySQL insert command. The flowchart (Fig.7a) illustrates the algorithm.

The cloud server holds a third PHP file that allows the user mobile application to relay GPS information to a database server. The HTTP GET method URL and PHP command $_GET [] are used in the same way in this file. Fig.7b shows the processes associated with the cloud server and end user.



(a) Cloud server with Arduino (b) Cloud server with end user request

Figure 7: Cloud server flowchart (a) Cloud server-Arduino (b) Cloud server and end user request

The API service to calculate the distance considers latitude and longitude values of two GPS points. The cloud server uses the PHP statements and server parameters, including the API key, to initiate a remote connection to Google map service. The Google map is queried and returns a response (calculated distance) to the cloud server. To acquire the address of a location, the cloud server transmits the GPS coordinates to the Google map service via PHP statements and the received content is extracted and converted to a formatted address.

3.3. Control station configuration and predictive analytics

The control station has two functions which are linked to each other. The first one is to communicate with the cloud server to monitor the vehicle position in real time. Secondly, the arrival time is predicted by applying several predictive analytic techniques. Figure 8 illustrates these two functions:

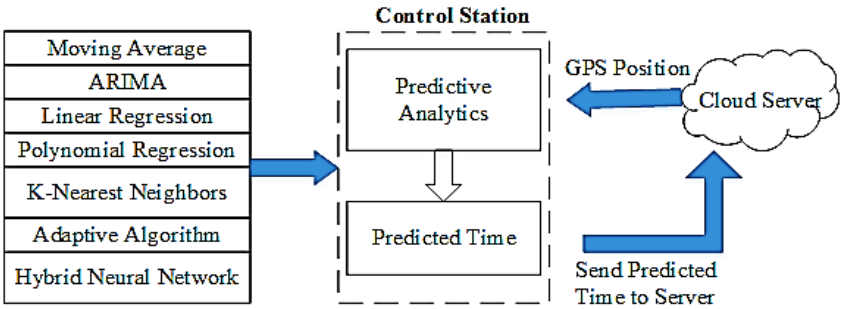


Figure 8: Function of control station

In this work, the control station is developed on the Java platform using the Netbeans IDE. The main user interface is shown in Fig.9.The user interface consists of three frames; bus GPS information, destination point and predicted arrival time. In the bus GPS information frame, the vehicle information such as latitude, longitude and time are updated from the cloud server and displayed in the list box. In addition, the API information (distance and address location) are displayed in corresponding text fields.

In the destination frame, a command button is inserted which retrieves the end user's destination address from the user request table. The address location, latitude and longitude are then displayed in the text fields with their labeling.

The arrival time prediction frame consist of three text fields and one combo box option which displays the list of prediction algorithm available. The calculated distance from destination as well as the estimated arrival time is inserted in two text fields respectively. The estimated arrival time is calculated using equation 11 and the predicted arrival time is inserted in the third text field.

$$\text{Estimated arrival time} = \frac{Distance\ from\ destination}{Vehicle\ current\ speed} \tag{11}$$

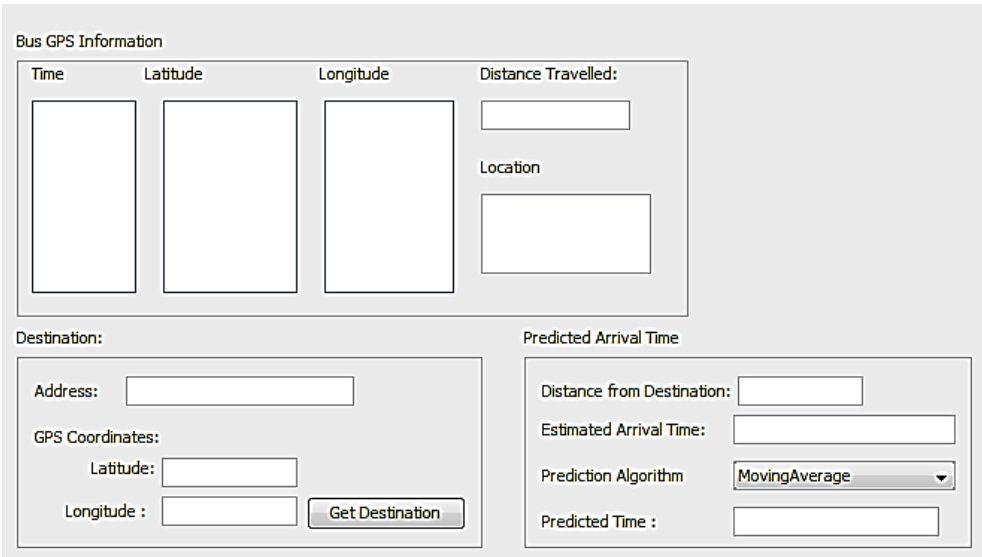


Figure 9: Control station interface

A block diagram illustrating the application method structure and their interconnections is given in Fig10.

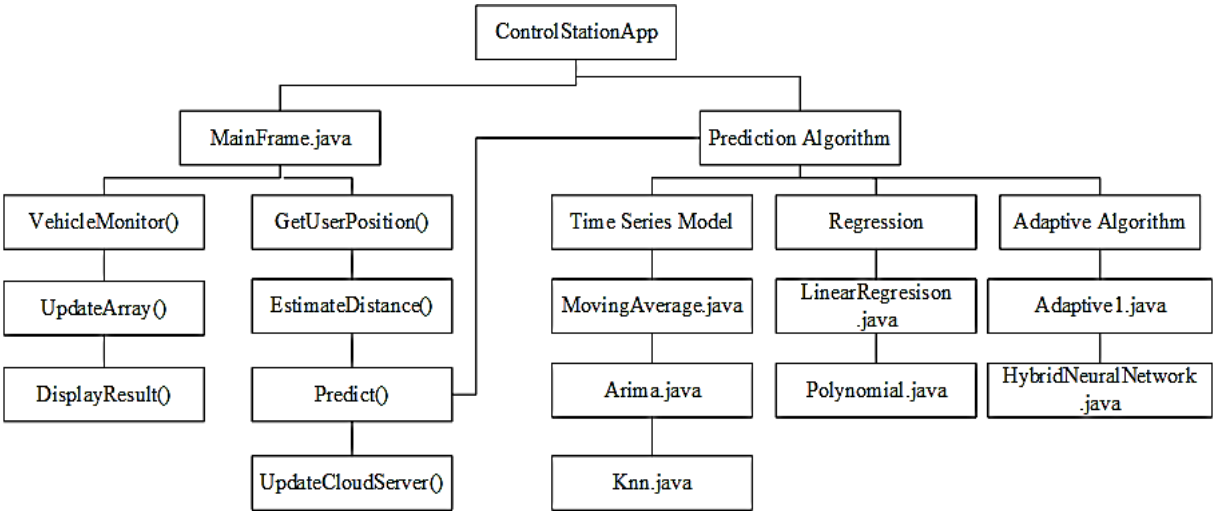


Figure 10: Method structure of control station

The application is made up of eight Java classes which consist of one Main Frame Class and seven predictive analysis classes. The Main Frame (MainFrame.java) runs the application and graphical components to display the form shown in Fig.9. In addition, it connects the application to the cloud server to retrieve both the vehicle and end user GPS data and update the arrival time. Also, the main class uses the Google API service to calculate the vehicle's distance from the destination. The predictive analytics classes use the vehicle information retrieved from the cloud server to predict the bus arrival time to its destination.

The Main frame java class is made up of seven functions. The VehicleMonitor() function retrieves the vehicle information in real-time from the cloud server. The UpdateArray() method is responsible for updating the array data structure while the DisplayResult() method displays the information on the user interface.

The other functions are used for predicting the arrival time and updating the cloud server. The GetUserPosition() executes a query to retrieve the end user's GPS location which is then used by the Google API to estimate the distance from the destination. The Predict() function uses a prediction algorithm to forecast the bus arrival time and updates the arrival time information on the cloud server by using the UpdateCloudServer() function. The flowchart in Fig.11 describes the real-time vehicle monitoring function of the control station.

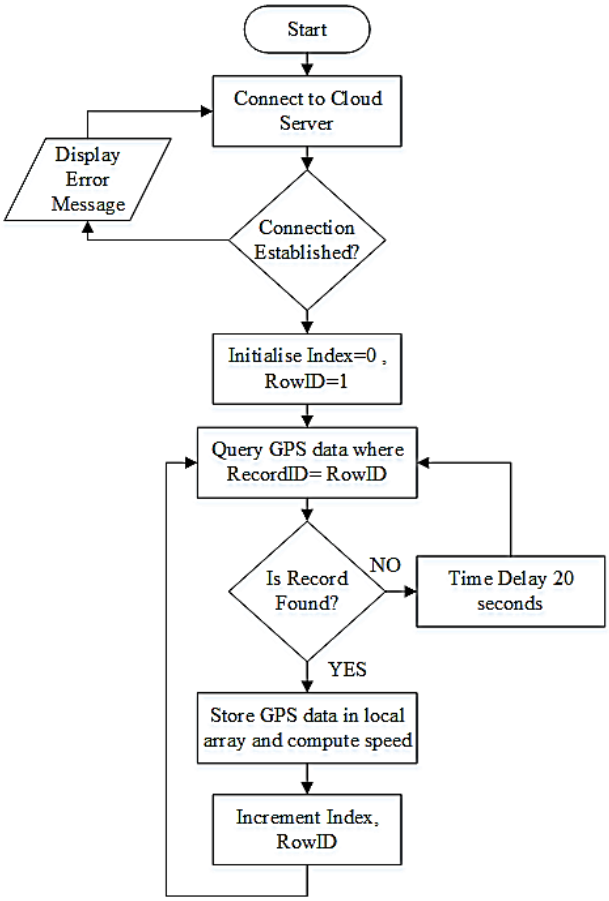


Figure 11: Vehicle monitoring flowchart

The system first accesses the cloud server using the MySQL connection syntax DriverManager.getConnection. If the connection fails, the application will display the reason behind the failed connection. After connection is established, the cloud server storage is continuously queried with a conditional statement and a pointer variable, Row ID. A copy of the GPS record is then stored in the application's local array.

The real-time monitoring system stores vehicle information such as latitude, longitude, address, duration and distance travelled. The speed of the vehicle for each GPS position is computed using the following formula:

$$Speed\ of\ vehicle, (kmh^{-1}) = \frac{Distance\ between\ two\ GPS\ points}{Time\ Taken} \quad (12)$$

Predictive analytics techniques are then applied to the information gathered. The following steps describe the process:

1. The user position from the cloud server is retrieved and Google API is used to calculate the distance from the user's destination
2. The actual vehicle information is stored in an array data structure.
3. Using the set of information, the arrival time for the next 0.3 kilometer is calculated. A range 0.3km is chosen to improve the accuracy of the algorithms.
4. The algorithm's array data set is updated with new vehicle information from the cloud server.
5. Steps 3 and 4 are repeated until the total predicted distance is equal to the calculated destination distance.
6. The arrival time at the destination point is transmitted to the cloud server where the information is relayed to the user mobile application.

The prediction algorithm class consists of ten methods which implement the different prediction algorithms used in this work. The description of the prediction algorithms proposed are described below and the adaptive pseudocode is given next:

1. The Moving Average algorithm implemented by MovingAverage.java class which uses the vehicle speeds to forecast arrival time.
2. Arima.java class implements the ARIMA model and considers the vehicle speeds as input.
3. Linear regression technique implemented by LinearRegression.java class
4. Polynomial regression technique employed by Polynomial.java class with degree equals to two.
5. Polynomial regression with degree equals to three
6. K-Nearest Neighbors algorithm programmed by Knn.java class.
7. Adaptive algorithm using RMSE only implemented by Adaptive1.java class.
8. Hybrid Neural Network; combination of adaptive algorithm with Neural Network system constructed using HybridNeuralNetwork.java class.

The adaptive algorithm applies the RMSE criterion in each loop to choose the algorithm with the best performance. In the hybrid neural network, the selected algorithm is used at the input to predict vehicle travelling speed. The pseudo codes are given in Fig 12 and Fig.13.

```
Initialise array variables, Distance to Predict= 0.3 km
Copy actual vehicle speed, distance and time in array defined
Initialise two pointer variables, start and end of the array
Do
    Calculate RMSE for last 6 records
     Select algorithm with lowest RMSE
     Predict arrival for Distance to Predict
        If  New Vehicle Record Added Then
            Update current array variable
        Update array pointer variable
    Else
        Increment array pointers
    If Total Distance = Destination Distance Then
        Break
    Else If (Total Distance+ 0.3)> Destination Distance Then
            Distance to Predict = Difference
    Else
        Distance to Predict = 0.3 km
While (True);
```

Figure 12: Adaptive pseudocode

From Fig.12 and Fig.13 it can observed that the size of test data during the cross validation process is set to six. Since a window of 30 records is used, an optimal ratio for splitting the data set is 80-20. The reason behind this ratio is the Pareto principle [19].

In the hybrid model, the neural network is first trained with previous records to adjust the network's weights. The output of prediction algorithm is then used as input to the neural network to obtain a predicted arrival time.

```
Initialise array variables, Distance to Predict= 0.3 km
Copy actual vehicle speed, distance and time in array defined
Initialise two pointer variables, start and end of the array
Do
  Calculate RMSE for last 6 records
  Select algorithm with lowest RMSE
  Predict Vehicle Speed using Algorithm for Distance to Predict
      Train Neural Network with array data set
      Pass the calculated Vehicle Speed to the Neural Network
      Predicted Arrival Time = Neural Network Output
        If  New Vehicle Record Added Then
            Update current array variable
            Update array pointer variable
        Else
            Increment array pointers
    If Total Distance = Destination Distance Then
        Break
    Else If (Total Distance+ 0.3)> Destination Distance Then
            Distance to Predict = Difference
    Else
        Distance to Predict = 0.3 km
While (True);
```

Figure 13: Hybrid neural network pseudocode

The performance of the prediction algorithms is assessed in details in Section four.

3.4. The user mobile application

The user application is implemented on the Android platform using android studio. The function of the mobile application is to transmit the user GPS coordinates to the cloud server and display the predicted arrival time updated by the control station. Fig.14 shows the processes performed by the application.

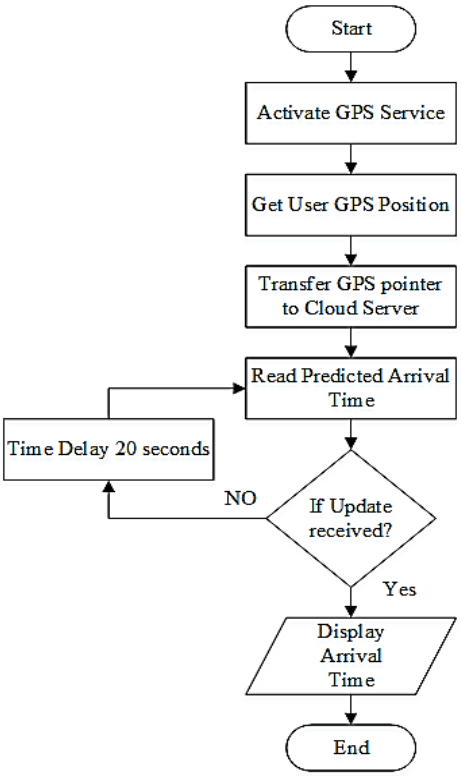
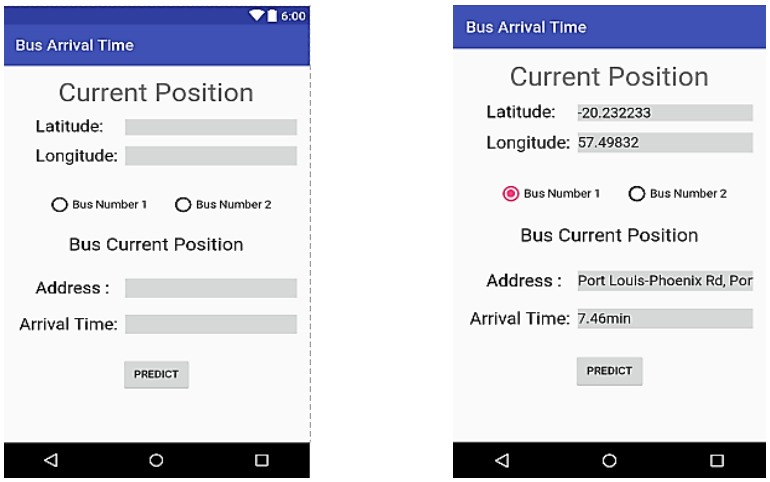


Figure 14: User application processes

When the application is run on a mobile device, the user interface given in Fig.15a is displayed. To demonstrate the operation, a smartphone was configured with the application developed. The coordinates of the user and moving bus are displayed and the predicted arrival time is updated continuously from the cloud server.Fig.15b shows a typical output from the application.

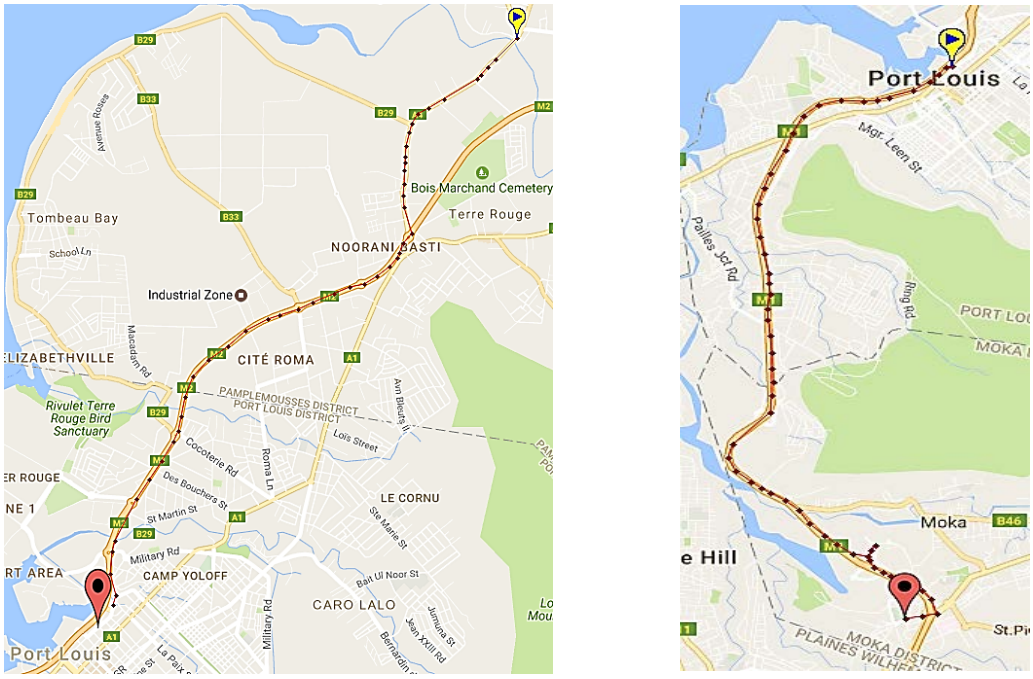


(a) (b)

Figure 15: User interface of mobile application

4. System testing and performance analysis

The performances of the predictive algorithms as well as the adaptive one were tested on two routes in Mauritius as shown in Figure 16. The yellow marker is the source and red marker is the destination.



(a) (b)

Figure 16: Map location (a) Route 1 (b) Route 2

Table 2 gives the details of the routes and data collection.

Table 2: Route details

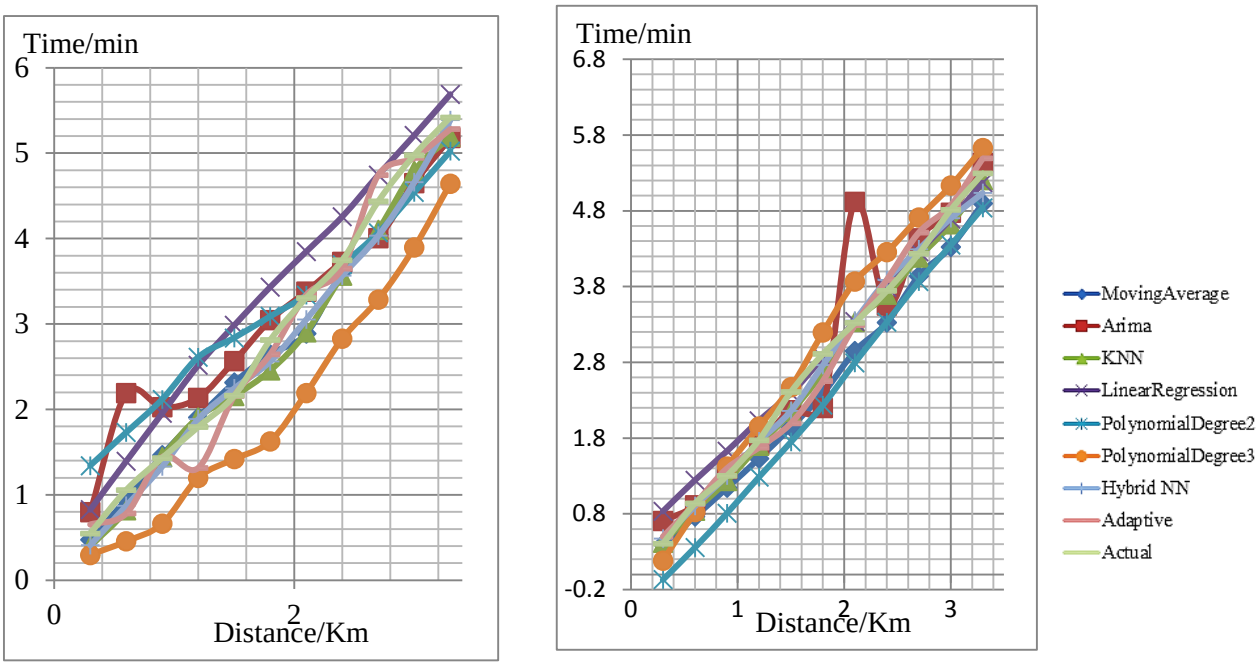
	Route 1	Route 2
Source	Arsenal	Port Louis
Destination	Port Louis	Reduit
Distance	6.8km	12km
Data collection interval		
Morning	7h00 – 7h30	7h30 – 8h15
Afternoon	16h00 – 16h30	15h00 – 15h30

Since KNN and Neural Network are data dependent methods, the algorithm parameters set during testing phase are given in Table 3.

Table 3: Algorithm parameter

K-Nearest Neighbors	**Neural Network**
K = 6(square root of number instances)	Epoch value = 10000

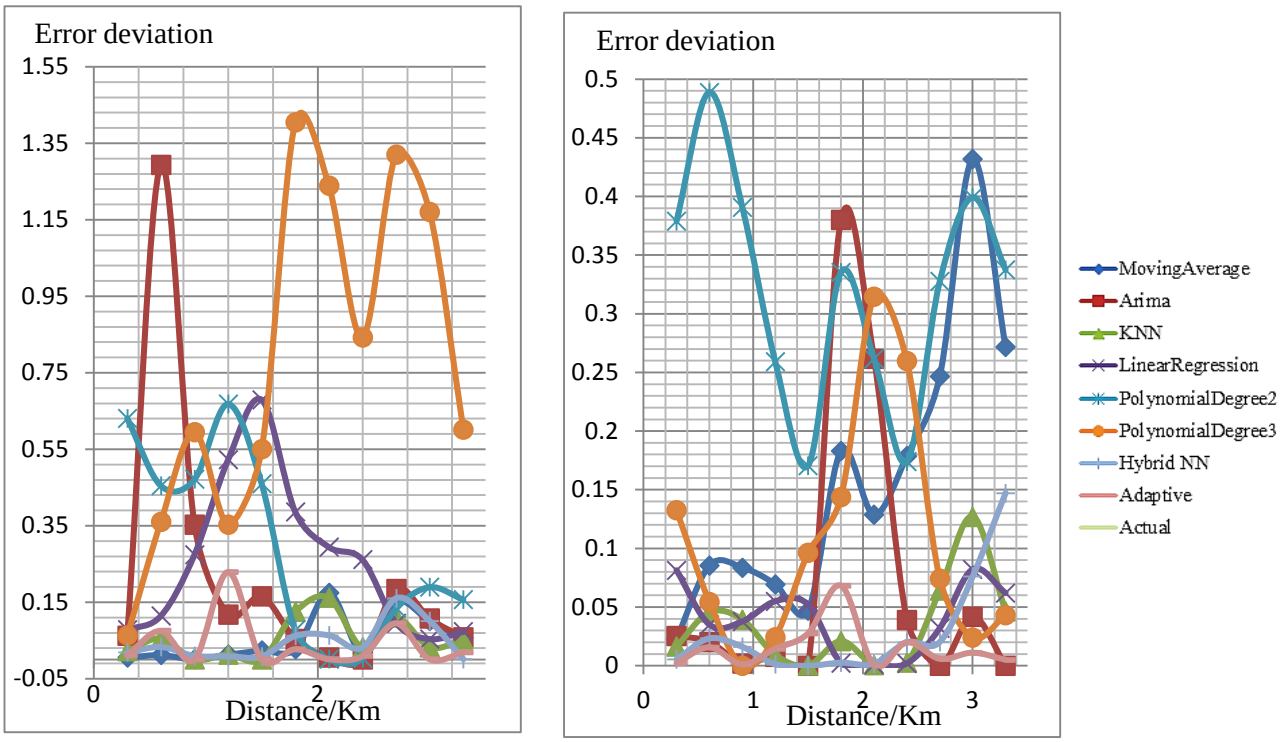
The performances of the algorithms are assessed in terms of the predicted and actual arrival times for a given distance between the source and destination. The Mean Squared Error (MSE) between the predicted and actual arrival times have also been used to compare their performances. For each route, tests were performed on ten week days and the results presented represent the average of the ten recorded data sets. Figure 17 shows the graph of predicted time against distance travelled for the eight algorithms as well as the actual arrival times for route1. It is observed that the best performance is obtained with the adaptive algorithm since it is closest to the actual reading.



(a) Route 1 morning readings (b) Route 1 afternoon readings

Figure 17: Route 1 prediction results

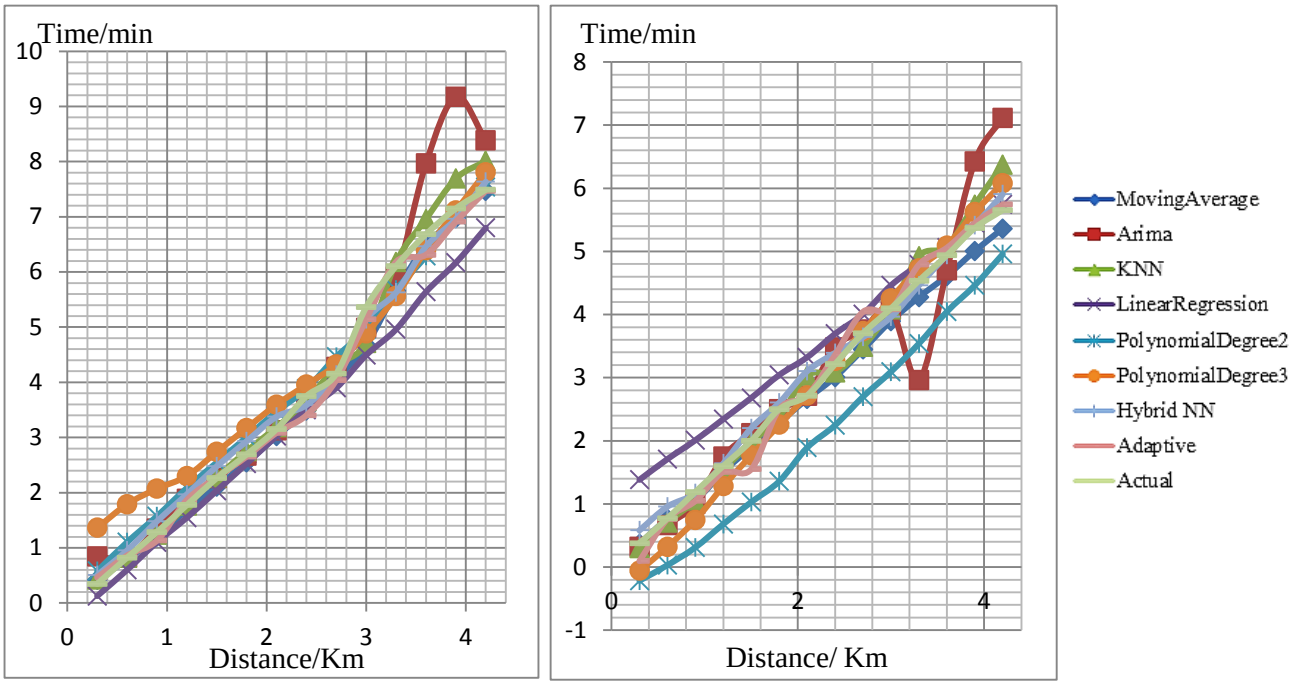
Figure 18 shows the graph of MSE against distance for the eight algorithms for route 1. It is observed that the lowest MSE is achieved by the adaptive algorithm. Table 4 gives the average MSE of the eight algorithms over the whole journey from the source to the destination for route 1. Again it is observed that the adaptive algorithm provides the lowest MSE.



(a) Morning readings (b) Afternoon readings

Figure 18: Route 1 error performance

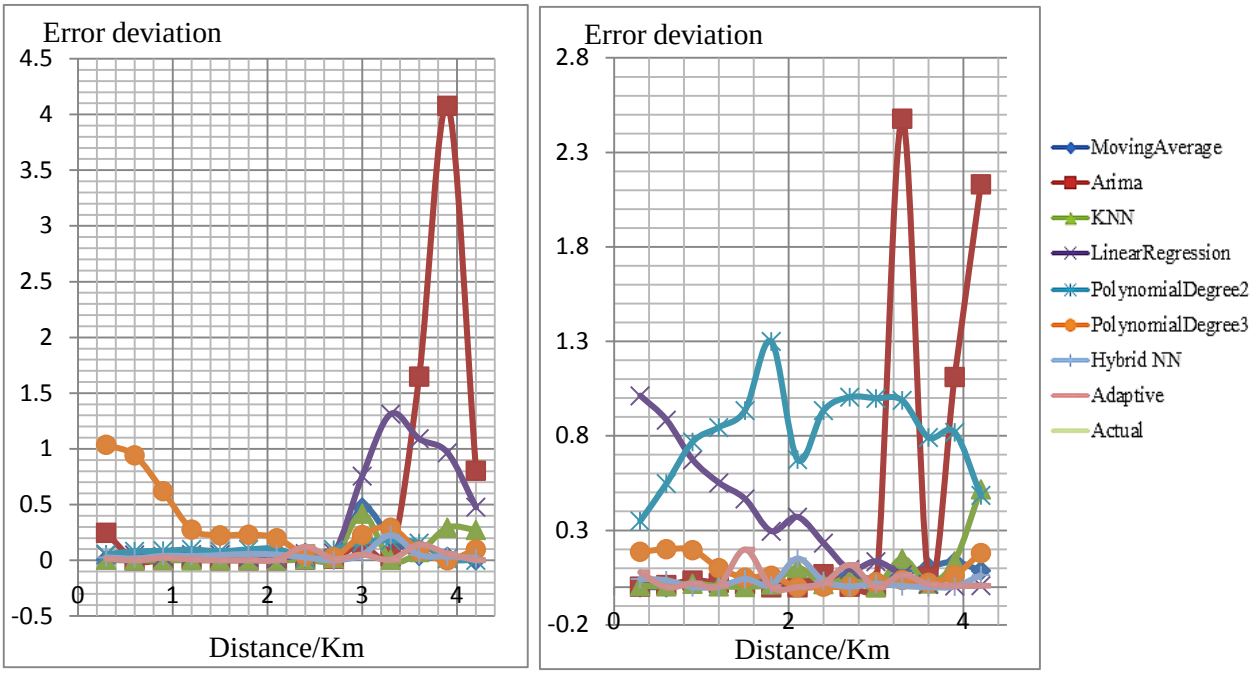
Figure 19 shows the graph of predicted time against distance travelled for route 2.



(a) Route 2 morning readings (b) Route 2 afternoon readings

Figure 19: Route 2 prediction results

From the graph, it is observed that the best performance is obtained with adaptive algorithm employing RMSE. The MSE for the eight algorithms is computed and results are shown in Fig.20.



(a) Route 2 morning readings (b) Route 2 afternoon readings

Figure 20: Route 2 error performance

As observed, the lowest MSE is obtained with adaptive algorithm. Table 4 gives the average MSE for the eight algorithms for route 1 and route 2.

Table 4: Average MSE for route 1 and route 2

Algorithm	**Route 1**		**Route 2**	
	Morning	**Afternoon**	**Morning**	**Afternoon**
Moving Average	0.053	0.158995	0.064714	0.044897
ARIMA	0.218053	0.070806	0.49994	0.424243
K-Nearest Neighbors	0.054758	0.033887	0.079384	0.074169
Linear Regression	0.2571669	0.0402039	0.357656	0.343679
Polynomial Regression Degree 2	0.294926	0.320359	0.088066	0.817438
Polynomial Regression Degree 3	0.772343	0.106123	0.307628	0.080951
Adaptive Algorithm	0.042548	0.015568	0.031379	0.02929
Hybrid Neural Network	0.045074	0.028612	0.049059	0.038602

From table 4, the adaptive algorithm provides the lowest MSE compared with other algorithm.

ACKNOWLEDGEMENT

The authors would like to thank the University of Mauritius for providing the necessary facilities to conduct this research.

CONCLUSION

This paper compared the performances of an adaptive prediction algorithm and hybrid algorithm with five prediction techniques (Moving average, ARIMA, KNN, Polynomial regression, linear regression). A real time cloud based bus arrival time prediction system was proposed which consists of an in-vehicle tracking device, a control station and smartphone application. The tracking device is composed of a microcontroller, GPS and GSM/GPRS which acquires the bus location and transmits the information to the cloud server. On the other end, a control station interface developed in Java has been implemented which accesses the location data of the end user and moving vehicle from the server and it also performs predictive analytics. The RMSE criterion was used to adaptively select the best predictor to estimate the bus arrival time. The arrival information is then relayed to the user's smartphone application through a web interface written in PHP. The performance of the proposed algorithm was evaluated and was found achieve an average MSE of 0.0297.This promising result can be used to implement an Advanced Public Transportation System and improve the reliability of the public transport system. The results indicate that the proposed adaptive algorithm is capable of providing a maximum accuracy in predicting bus arrival time. Further study may incorporate the adaptive algorithm in a traffic congestion prediction system as well as developing a user interactive interface to provide travel time information.

References

[1] A basic introduction to neural networks Available at: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html (Accessed: 03 January 2017).

[2] A. Menon, R. Sinha, D. Ediga, S.Iyer, 'IMPLEMENTATION OF INTERNET OF THINGS IN BUS TRANSPORT SYSTEM OF SINGAPORE', Asian J Eng Res/Vol. I/Issue IV/July-Sept.2013/08-17.

[3] Amita, J., Sukhvir Singh, J. and Pradeep Kumar, G. (2015) 'PREDICTION OF BUS TRAVEL TIME USING ARTIFICIAL NEURAL NETWORK', INTERNATIONAL JOURNAL FOR TRAFFIC AND TRANSPORT ENGINEERING, 5(4), pp. 410–424. doi: 10.7708/ijtte.2015.5(4).06.

[4] Apache HTTP server (2017) in Wikipedia. Available at: https://en.wikipedia.org/wiki/Apache_HTTP_Server (Accessed: 10 January 2017).

[5] Arduino GPRS shield - Geeetech Wiki (2014) Available at: http://www.geeetech.com/wiki/index.php/ Arduino_GPRS_Shield (Accessed: 20 December 2016).

[6] Catherine M. Baker and Alexander C. Nied, 'Predicting Bus Arrivals Using One Bus Away Real-Time Data', 2013

[7] Cathey, F.W., & Dailey, D.J. (2003) A prescription for transit arrival/departure prediction using automatic vehicle location data, Transportation Research Part C 11 (2003), pp. 241-264

[8] Dalia Tiesyte, and Christian S. Jensen (2008) Similarity-based Prediction of Travel Times for Vehicles Traveling on Known Routes. ACM GIS 2008, Irvine, CA, USA.

[9] Deeksha Bhardwaj, Sandip Daphal, and Anand Nerkar, "Predicting Bus Arrival Time with GPS on Android Application", International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064, Volume 5 Issue 1, January 2016.

[10] Dihua Sun, Hong Luo, Liping Fu, Weining Liu, Xiaoyong Liao, and Min Zhao, "Predicting Bus Arrival Time on the Basis of Global Positioning System Data", Transportation Research Record: Journal of the Transportation Research Board, No. 2034, Transportation Research Board of the National Academies, Washington, D.C., 2007, pp. 62–72. DOI: 10.3141/2034-08

[11] Duda, R.O., Stork, D.G. and Hart, P.E. (2000) Pattern classification: Pt.1: Pattern classification. 2nd edn. New York: John Wiley & Sons.

[12] HTTP methods GET vs POST (2016) Available at: http://www.w3schools.com/tags/ref_httpmethods.asp (Accessed: 27 December 2016).

[13] Hyndman, R.J. (2009) Available at: http://robjhyndman.com/papers/movingaverage.pdf (Accessed: 26 December 2016).

[14] Introduction to ARIMA models Available at: https://people.duke.edu/~rnau/411arim.htm (Accessed: 20 December 2016).

[15] Mathieu Sinn, Jin Won Yoon, Francesco Calabrese and Eric Bouillet 'Predicting arrival times of buses using real-time GPS measurements'. International IEEE Conference on Intelligent Transportation Systems 2010, Anchorage, AK, USA.

[16] MySQL (2017) in Wikipedia. Available at: https://en.wikipedia.org/wiki/MySQL (Accessed: 06 January 2017).

[17] Naveen Nandan, 'Online Grid-Based Dynamic Arrival Time Prediction Using GPS Locations', International Journal of Machine Learning and Computing, Vol. 3, No. 6, December 2013.

[18] Ostertagová, E. (2012) 'Modelling using polynomial regression', Procedia Engineering, 48, pp. 500–506. doi: 10.1016/j.proeng.2012.09.545.

[19] Pareto principle (2016) in Wikipedia. Available at: https://en.wikipedia.org/wiki/Pareto_principle (Accessed: 9 January 2017).

[20] Picard, R. R. and Cook, R. D., Cross-validation of regression models, Journal of the American Statistical Association,79 , 575–583, 1984.

[21] Prakhar Balasubramanian and K. Ramachandra Rao, "An Adaptive Long-Term Bus Arrival Time Prediction Model with Cyclic Variations", Journal of Public Transportation, Vol. 18, No. 1, 2015.

[22] PHP (2017) Available at: https://en.wikipedia.org/wiki/PHP (Accessed: 10 January 2017).

[23] T Bin, Y., Zhongzhen, Y. & Baozhen, Y. (2006) Bus Arrival Time Prediction Using Support Vector Machines, Journal of Intelligent Transportation Systems: Technology, Planning, and Operations,10:4, pp. 151-158.

[24] Zou, K.H., Tuncali, K. and Silverman, S.G. (2003) 'Correlation and simple linear regression1', Radiology, 227(3), pp. 617–628. doi: 10.1148/radiol.2273011499.

[25] "Adafruit GPS", https://www.adafruit.com/product/746

[26] "Arduino microcontroller", http://arduino.cc

[27] "SIM900 AT Command", https://www.espruino.com/datasheets/SIM900_AT.pdf

[28] "The Google Directions API" , https://developers.google.com/maps/documentation/directions

[29] "Adafruit GPS library", http://www.arduinolibraries.info/libraries/adafruit-gps-library

[30] "AltSoftSerial library", https://www.pjrc.com/teensy/td_libs_AltSoftSerial.html