Bi-Directional Traffic Management with Multiple Data Feeds for Dynamic Route Computation and Prediction System

Md. Rahatur Rahman[1], Shamim Akhter[2]
[1]*Software Engineering, Simplexhub Ltd*
[2]*Department of Computer Science and Engineering, East West University (EWU)*
Dhaka, Bangladesh

Abstract

Bi-directional communication is already in use for different real time applications including -video conferencing, chatting, telecommunication etc. However, the benefits and the results of real time bidirectional communication on moving agent applications (i.e. the road traffic system), are yet to be discovered. In this paper, a new internet-based traffic management support system with real time bidirectional communication is going to be proposed and implemented. It supports significant benefits over the existing technologies to monitor the road traffic conditions. Dynamic route computation is a vital requirement to make the proposed system more realistic. Therefore, an integrated approach- with multiple data feeds and decision tree based logic is applied to calculate the road segment weights and provide dynamic routing facility. The results indicate that the proposed traffic management support system/tool with dynamic routing (weights) is much more effective to find the optimal routes. In addition, future weight prediction module is included to utilize the structured data from the historic database and enhances the system capability to predict road weights for suggesting optimal routing in advance.

1. Introduction

Traffic jam is an on-going challenge, and is not showing any signs of improvement. It results losing valuable time (approx. 200%) being stuck in traffic jams and increases CO2 emissions (approx. 300%) due to the present traffic situation. As a result, an up-to-date technological traffic management support system/tool has become a need for metro cities. This paper aims to assist the traffic system to reduce the traffic and creates a more sustainable environment. Over the time different solutions (in [1][2][3][4][5]) have been proposed to solve this problem. The solutions vary in their core technologies, as some of them used infrared sensors, CCTV cameras, machine vision (image processing), GSM and/or cellular towers, RFID gateways, sound sensors, aerial surveillance and remote sensing etc. However, many of those solutions are either very expensive or difficult to install and maintain over the periods of time. Nevertheless, most of the existing technologies are unicast communication (client to server only). Bidirectional communication is already in use for different purposes like video conferencing, chatting, telecommunication etc. However the benefit and the results of real time bidirectional communication are yet to be discovered especially when applied to moving agents (i.e. the road traffic system). Questions are raised - whether any approach or technique would bring low operational-cost but flexible internet-based solution.

A new internet-based (WebSocket [6] over HTTP) traffic management support system with real time bidirectional communication is implemented (in our previous work [7][8]) for supporting low cost implementation, flexibility, maintainability and infrastructure security. It also provides significant benefits over the existing surveillance technologies in use to monitor road traffic condition. However, this work is based on static or manual weight updates. Optimal paths/routes are calculated from the given static weights.

The principle objective of this paper is to implement traffic management system with dynamic weights, and weight prediction modules. Weights are calculated from current road situations including-construction status, damaged status, accident status, traffic status, environmental disaster status etc. The value of the statuses are intelligently crawled by search engine, with metadata indexing (title, description, keyword etc.), directly from the multiple data feeds (like web site, RSS feed, web service etc.). Crawled data are simplified (structured) and stored in a historic table. Decision tree based logic is implemented over the simplified data to adjust the route weights in database. Dijkstra algorithm is applied to calculate the optimal path/route using the dynamic route weights. In addition, stored data is used to provide route prediction services including future weights calculation and optimal route generation.

2. Research Background

2.1. Existing traffic management technologies

Analysis of the technologies in use at present yields the followings:

Machine Vision: high quality CCTV cameras are used to capture the images of the vehicles. And then the images are processed on a terminal to create information from the data. This is a very expensive method as the CCTVs are expensive and also requires heavy installations and maintenance. Also they have limitations during heavy snowfall, sand storm, rainfall, foggy weather [4].

RFID Sensors: requires the RFID gates to be installed and the vehicles to pass through the gate. This is ineffective if the vehicles go around the gates for any reason [2][3].

Surveillance Drones: are effective if the weather is normal. However, cannot be an option during bad weather condition. Also the deployment and maintenance cost is very high for such system [1].

For the above mentioned technologies either the installation (maintenance) cost is very high or the technology consist technical limitations. In addition, they have only one directional communication channel - client to server.

2.2. WebSocket and the HTTP protocol

So far, real time web applications largely revolve around polling and other server-side push technologies. The most notable technology is Comet, which delays the completion of an HTTP response to deliver messages to the client. Comet-based push is generally implemented with JavaScript and uses long-polling [6] connection strategy. With polling, the browser sends HTTP requests at regular intervals and receives responses immediately. Obviously, this is a good solution if the exact interval of message delivery is predefined, as the occurrence of the client request can be synchronized till the necessary information is available on the server domain. However, real-time data is unpredictable and makes unnecessary requests inevitable. With long-polling, the browser sends a request to the server and the server keeps the request open for a set period. If the notification is received within that period, a response with the message is sent to the client. If a notification is not received within the preset time period, the server sends a response to terminate the open request.

It is important to understand, long-polling does not provide any substantial performance improvements over traditional polling- when a high message volume consists. In fact, it could be worse, because long-polling might spin out of control into an un-throttled, continuous loop of immediate polls [6]. Ultimately, all of these methods for providing real-time data involve HTTP request and response headers. The headers contain lots of additional, unnecessary data and introduce latency overhead.

On top of that, full-duplex connectivity requires more than just the downstream connection from server to client. In an effort to simulate full-duplex communication over half-duplex HTTP, many of today's solutions use two connections: one for the downstream and one for the upstream. The maintenance and coordination of these two connections introduce significant overhead in terms of resource consumption and create additional complexities [6].

WebSocket was developed as a part of the HTML5 with JavaScript interface. WebSocket represents the next evaluation of Comet and Ajax [6]. It reflects a full-duplex single socket connection over which messages can be sent between client and server. It also simplifies much of the complexity around bi-directional web communication and connection management. HTML5 WebSocket is not just another incremental enhancement to conventional HTTP communications; it represents a colossal advancement, especially for real-time, event-driven web applications [6]. It provides dynamic improvements over conventional full-duplex connection over HTTP. The main difference in WebSockets compared to the usual network traffic over HTTP – is the protocol. The protocol does not follow the traditional request-response convention. Once a client and a server have opened a WebSocket connection, both endpoints may asynchronously send data to each other. The connection remains open and active as long as either the client or the server closes the connection [6].

3. Proposed system architecture

In this section, a generic architecture (blending all above features and technologies) for the proposed real time system have been presented. Fig.1 illustrates the components of the proposed architecture along with their involved signaling. The vehicles referred as clients have access to GPS [9], which allows them to collect the present location in terms of latitude and longitude. The clients are also equipped with any device capable to handle the web request/response over the HTTP. The device should also have support for the IETF as RFC 6455 standards. The architecture comprises a WebSockets server. It listens for incoming connections from the client browsers which are compatible with HTML5 and WebSocket. The clients will have to relay their present location through a WebSocket server. At first an HTML5 client requests a web-page from a web server that includes the required JavaScript for WebSocket connections (or any fallback connection if required) establishment. Then, the client connects to the WebSockets server using a WebSocket

connection and starts the process of sending the location updates in predefined intervals. Any database software can be used to store client data.

The communication server(s) will be responsible to handle the incoming client connections and the database server manages the client data as well as the data of the traffic system. The routes are calculated in the database server and are served to any requesting client. The server application is built on top of open source Asp.net MVC platform. The open source implementation of SignalR for Asp.net MVC framework is utilized for the WebSocket.

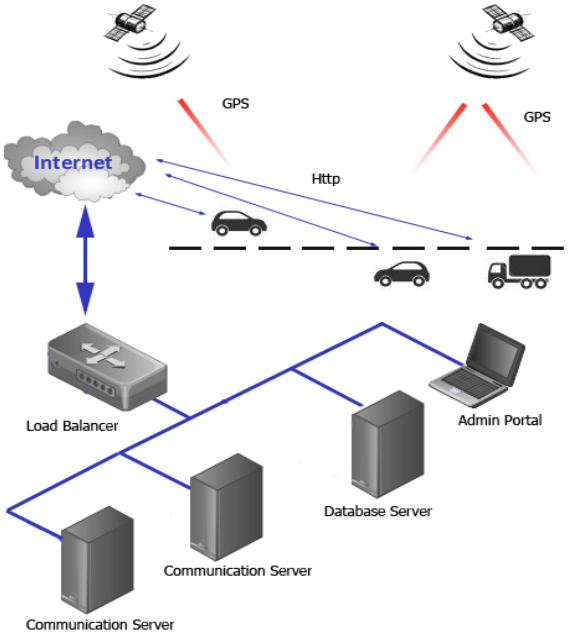

Figure 1. Architecture of the proposed system

3.1. WebSocket server

The WebSocket technology provides a bi-directional communication channel using a single TCP connection. It is designed to be implemented in applications such as web browsers. Its API is being standardized by the W3C. The connections are established over the regular TCP port 80, which ensures that the system can run behind the firewalls.

At first the client (with WebSocket protocol) requests to establish a WebSocket connection and the server responses to start the connection. The connection remains open for the whole session, until any endpoint requests to disconnect. As the WebSocket remains active, data can be transferred from server to client or vice versa regardless of the initiator. In our implementation, the WebSocket server hosts the logical parts of our web services, and responsible to maintain a list of client peers with active WebSockets and session management.

The server application has the real time location plotting system on region based available data. The server application is also responsible for data mining and business logic execution based on the collected data from various clients.

3.2. Fallback mechanism

In a real world scenario there will always be some system which will lack for the latest technology, i.e. WebSocket may not be supported by a group of devices. As a result, it is necessary to have some fallback mechanisms (WebSocket, Server Sent Events, Forever Frames and Long Polling in respective order [10]) so that system services will not be interrupted.

3.3. Connection establishment

When the application starts, the client requests server for necessary scripts. The scripts utilize the proxy connection to the server and initiate a WebSocket connection. The scripts also inform the server about the methods available on the client side. Thus, enables the server to push some data to the client side. This process has been described in detail in our previous works [7][8].

4. The traffic management support system

4.1. Test zones/study areas

Banani area of Dhaka, Bangladesh have been selected to test the vehicle routings. The roads are divided into segments depending on the junctions to test the effectiveness of the vehicle routing. The midpoint of a road is denoted by a red marker in Fig.2. The location of each region is stored in terms of latitude and longitude and placed on a Google Map [11]. Each region is marked as a rectangle and has five points – the four corners and the midpoint. Latitude and longitude for each of the points are stored in the database.

4.2. Static road weight matrix

The road weights (arbitrary numbers) are stored in a table at the database server using From-To format to support both directions. One way roads are not considered here. In real test bed the weights must be dynamically calculated (discussed in the next section) based on external data coming from weather

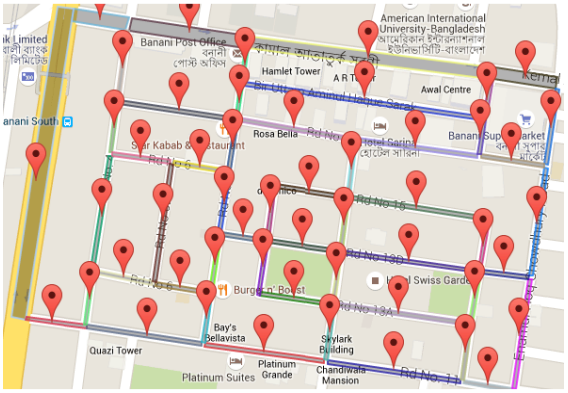

Figure 2. Test zone displayed on a Google map

Table 1. Part of the weight matrix

	Weight ID	From ID	To ID	Road Weight
1	1	2	3	1000
2	7	2	6	5
3	8	3	2	5
4	9	3	15	5
5	10	3	4	5
6	11	4	3	5
7	12	4	23	5
8	13	4	5	5

forecast, traffic congestion, road condition, and other sources. Present application is capable to work with the resulted weights for each of the roads stored in the weight table. Table.1 shows part of the weight matrix from database.

Few simulated clients are created with their test routes (as if they are on the road) and stored in a test route table. When the application is active, each test client logs in and transmits the corresponding test route to the server. The server application has the capability of mapping any transmitted location to an existing test location and plots that on an aerial map.

4.3. Dynamic road weight calculations

The application scraps data from the predefined external data sources (like web site, RSS feed, web service etc.). For test purpose three dummy weather web pages are created.

A scheduler runs in the background and scraps data for each of the road segments in a regular interval (e.g. every 6 hrs). Then, it stores the scrapped data in a historic table. At the same time, the weights for the road segments are also updated in the weight matrix by using decision tree. The decision to update the weights is decided by the decision tree. For this test purpose four attributes are considered– Rain Fall, Temperature, Road Status, and Road Works. However, the number of attributes can be changed according to the system requirements.

Table 2. Data set to calculate Entropy and Information gain using ID3 Algorithm

Rain Fall	Temperature	Road Status	Road Works	Increase Weight
No Rain	High	Closed	Normal	Yes
No Rain	High	Closed	Maintenance	Yes
Rain	High	Closed	Maintenance	Yes
Heavy Rain	High	Closed	Maintenance	Yes
Heavy Rain	Low	Open	Maintenance	No
Heavy Rain	Low	Open	Maintenance	Yes
Rain	Low	Open	Maintenance	No
No Rain	Average	Closed	Maintenance	Yes
No Rain	Low	Open	Maintenance	No
Heavy Rain	Average	Open	Maintenance	No
No Rain	Average	Open	Maintenance	No
Rain	Average	Closed	Maintenance	Yes
Rain	High	Open	Maintenance	No
Heavy Rain	Average	Closed	Maintenance	Yes

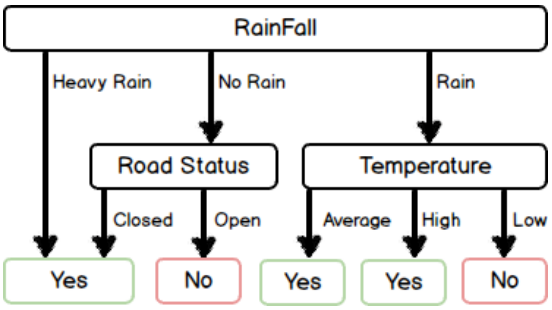

Figure 3. Visualization of the Decision Tree generated from the data set S

The attributes, class(**C**) or classifier (i.e. Yes/NO) and collection set (**S**) are displayed in Table. 2. The detail calculations of Entropy(s) and Information gain **G(S,A)** are available at [12].

External feed data (weather data) is first interpreted from numeric to an attribute in our data set **S** mentioned above. For example: for the "Rain Fall' attribute - if zero (0) for Rain Fall (mm) then it is considered as 'No Rain'. Similarly, any number between 0 and 3 is considered to be 'Rain' and anything above 3 is considered to be 'Heavy Rain'. Same goes for all other inputs.

The interpreted data is then evaluated against the generated decision tree (Fig.3) and based on the result; the application either increases or resets the weight of a road (segment) and reflects in the road weight matrix accordingly.

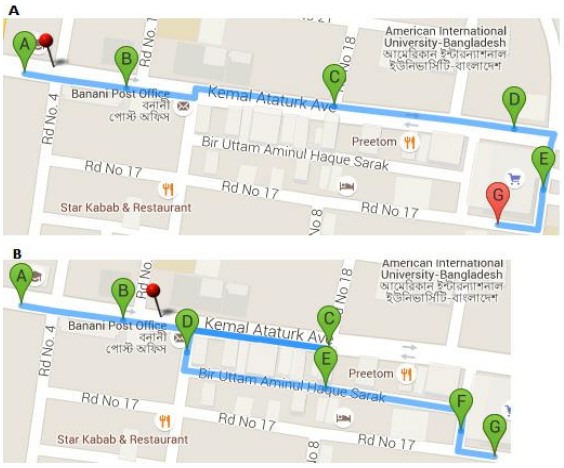

Figure 4. Route Suggestion before and after weight increase

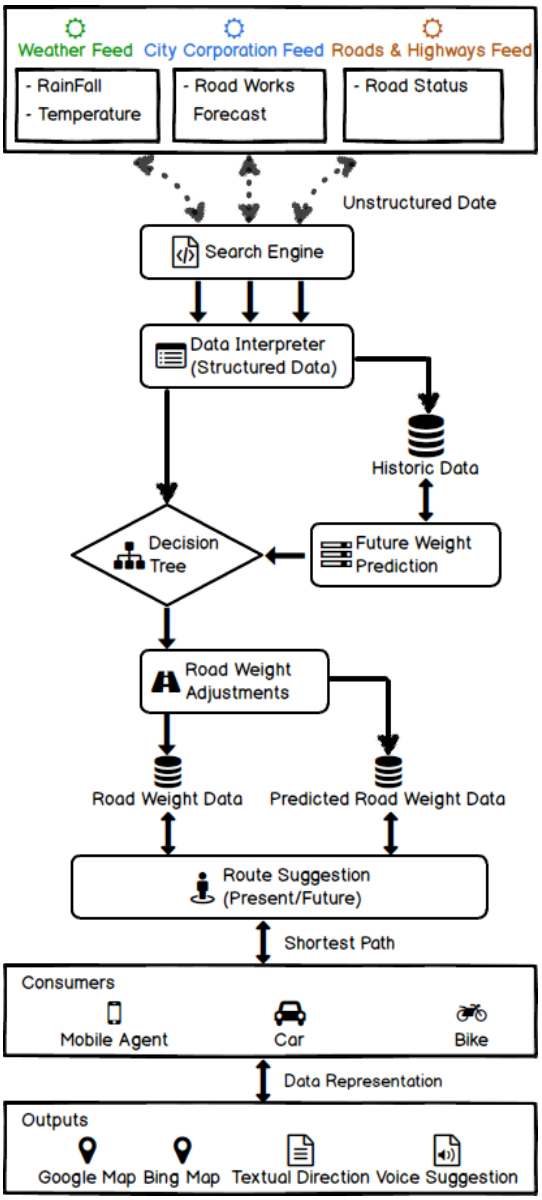

Figure 5. Work flow of the road weight processing and prediction system

Fig.4 highlights the application is changing the route suggestion dynamically based on the resulted weights from the external data sources. For example in Fig.4 (A), the point 'D' is adjacent to point 'C' and is on the right side of the point 'C'. This is the suggested route before the weight for the point 'D' is increased. Now the weight for the point 'D' has increased (by the decision tree). So this road segment is going to be omitted. Thus, Fig. 4 (B) presents alternate route with relocating the point 'D'. This is the resultant of the dynamic weight adjustments or dynamic routing.

4.4. Road weight forecast

Fig.5 shows the search engine (referred as a web crawler) crawls the feed sites. In this case the three predefined sites provide the data/forecast of the weather, City Corporation road maintenance works and road status from government Roads and Highways authority. The unstructured data is converted to structured data and stored in a historic table at the same time. Also the new data is evaluated by the decision tree and the road weights are adjusted as needed. The weight prediction module utilizes the historic data to predict the future factors and generate another road weight matrix (called the future weight matrix). This road weight matrix can be used to predict or simulate route suggestions.

4.4.1. Feed sites. Three dummy web pages created to simulate the three feed sites as mentioned above. The sites display Weather, City Corporation and Roads & Highways data in human readable format.

4.4.2. Search engine. The Search Engine referred as the web crawler visits the feed sites in regular intervals. It looks for an Html Table element with a predefined ID or Class name. It does not depend on the position or the hierarchy of the Html elements. Once the tag has been identified the crawler scraps the rows using XPath [14]. Then the data is interpreted and converted to structured data on the server side and stored in database.

4.4.3. Weight prediction. The future weight prediction module utilizes the structured data from the historic database to predict the future road weights. First and foremost the module predicts the factors like Rain Fall, Temperature, Road Status etc. and those values have direct influences to the road weight calculation. To predict the factor values Weighted Moving Average algorithm [15] is used and this algorithm allows the most recent data to have higher impact on the prediction. Different factors influence the road weights in different ways and thus, factor values are predicted and later decision tree uses them to calculate the estimated road weights.

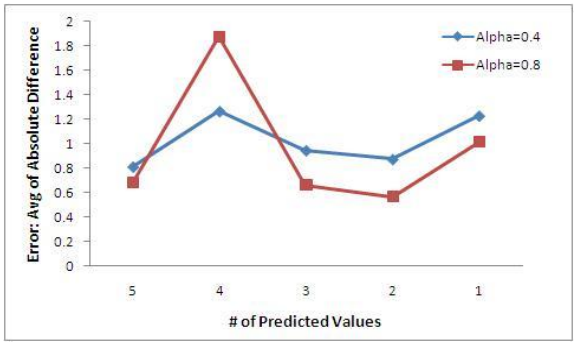

Figure 6. Number of predicted values and their corresponding errors

The estimated road weights are stored in "Predicted road weight" matrix. Thus, the application is capable to simulate and suggest future routes based on the road weights from the "Predicted road weight" matrix.

4.4.4. Prediction statistic. To analyze the performance of the prediction module, one of the weight influencing factors – Temperature is considered.

Table 3 shows the temperature data for December 10th, 2015 to December 29th, 2015 (20 data) are collected from the AccuWeather site [16].

Weighted Moving Average algorithm is used to calculate the next (future) five expected data and compared with their actual values. Average of absolute difference represents the error rate.

Fig.6 highlights the accuracy of the predicted data for two different Alpha (α) values (0.4 and 0.8) [15]. The X axis represents the # of predicted values and Y represents the corresponding error rates. The first experiment counts the fifteen (15) data as seed data and remaining are the testing data. Testing data are used to evaluate the error rate. The second experiment includes the sixteen (16th) as the most recently known data and estimates the 17th – 20th to evaluate the error rate and thus, continue for 3rd, 4th and 5th experiments. In usual condition, the higher value of Alpha includes more weight to the latest data and directs the future trends. However, for exceptional conditions- latest known data holds much smaller or higher value than the previous and thus inclusion of higher weight affects significantly on the error rates. Our experiment in Fig.6 (# of Prediction Values 4 and 1) reflects such circumstances.

4.5. Application

4.5.1. Authentication. Clients are identified when they authenticate as a user. Clients use predefined set of credential to authenticate. Upon successful authentication clients are forwarded to the client dashboard.

Table 3. Seed data for temperature

	Date	Temperature (°C)
1	12/10/2015	27
2	12/11/2015	27
3	12/12/2015	28
4	12/13/2015	28
5	12/14/2015	25
6	12/15/2015	24
7	12/16/2015	25
8	12/17/2015	24
9	12/18/2015	23
10	12/19/2015	23
11	12/20/2015	24
12	12/21/2015	24
13	12/22/2015	26
14	12/23/2015	24
15	12/24/2015	25
16	12/25/2015	23
17	12/26/2015	25
18	12/27/2015	25
19	12/28/2015	25
20	12/29/2015	26

4.5.2. Client dashboard. Clients can monitor his current location on a map. The location is marked with a marker and transmitted to the server at the same time over a Websocket connection. The location on the map updates itself according to the client's movement. The test data for each of the clients comes from the Test Route table. On the real field this data would come directly from the GPS unit instead of the Test Route table. When a client is on the dashboard a Websocket connection is established with the server and remains till the client closes the connection.

4.5.3. Server dashboard. On the server dashboard a WebSocket connection is opened and established with the server application. The server transmits the location of each of the clients (as the data is available) to the dashboard. Then the dashboard plots the locations of all the clients on a map. Both the clients and the dashboard are utilizing WebSocket based connections. Thus, the data transmission consumes much lower bandwidth.

4.5.4. Route suggestion. On the client dashboard when a client requests for a route to a destination then the system does the followings:

- Consider the current location of the user as the source.
- The selected "To Place" (selected from a drop down list) is considered to be the destination.
- Dijkstra's algorithm [13] is used to find an optimal route. Only the weights of the roads are considered as the factors.

- The weight matrix is applied to calculate an optimal route from source to destination.
- The weights inside the weight matrix are changed dynamically based on the external factors with the help of the decision tree.
- A calculated route is then displayed on a map (Fig.4)

5. Conclusion and future works

The "Bi-Directional Traffic Management with Multiple Data Feeds for Dynamic Route Computation and Prediction System" is successfully implemented with GPS and WebSocket and proves the adage of "mobility & flexibility" for traffic management. As a traffic management support system our application provides opportunities for tactical and targeted communication. Our targeted area was - easy data exchange, reduced communication cost over HTTP, dynamic routing, prediction and real time communication. Proposed application may be very helpful for routing traffics in an emergency situation where road conditions changes frequently like tornado zones, hazardous zones, and congested traffic zones.

The data from the external feeds (like web site, RSS feed, web service etc.) are stored in the database and opens opportunity to analyze or predict future traffic based on the historic data. Weighted Moving Average algorithm is used to calculate the estimated or predicted factor values. However, the performance is not so adequate.

In near future, a more robust statistical algorithm will be considered to improve road weight prediction mechanism. In addition, the decision tree based road weight calculation will be enhanced with more advance decision making system – NN, evolutionary algorithms, and/or Fuzzy Classifiers.

The proposed system can be further enhanced by embedding the software into mobile devices like micro-controller with GPS and GSM/EDGE unit to reduce the implementation costs.

6. References

[1] N.Verma, M. S. Sobhan and T. Jalil, "Novel Design Proposal For Real Time Traffic Monitoring & Management of Dhaka Metropolitan City with Rcap", Proc. of Dhaka_Infotech2012. Available at: http://www.gistconpro.com/328-Tasmirul.pdf (Access Date: 15th May, 2015).

[2] F. Aloul, A. Sagahyroon, A. Nahle, M. Abou Dehn and R. Al Anani, "GuideME: An Effective RFID-Based Traffic Monitoring System", International Conference on Advances in Computer Science and Engineering (ACSE), Phuket, Thailand, March 2012.http://www.aloul.net/Papers/faloul_acse12.pdf(Access Date: 15th May, 2015).

[3]A. Dhar, "Traffc and Road Condition Monitoring System", M.Tech report, Indian Institute of Technology, Bombay, 2008. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.518.2687&rep=rep1&type=pdf (Access Date: 15th May, 2015).

[4] T. Romão, L. Rato, P. Fernandes, N. Alexandre, A. Almada, and N. Capeta, "M-Traffic - A Traffic Information and Monitoring System for Mobile Devices", Proc. of International Workshop on Ubiquitous Computing, 2006. Available at http://dspace.uevora.pt/rdpc/bitstream/10174/1443/1/iceis06-tir-lmr-IWUC06.pdf (Access Date: 15th May, 2015).

[5] A. Angel, M. Hickman, P. Mirchandani, and D. Chandnani, "Methods of Analyzing Traffic Imagery Collected From Aerial Platforms" IEEE Transactions on Intelligent Transportation Systems, vol. 4, no. 2, June 2003. Available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.516.6998&rep=rep1&type=pdf (Access Date: 15th May, 2015).

[6]HTML5 Web Sockets: A Quantum Leap in Scalability for the Web. Available at http://www.websocket.org/quantum.html (Access Date: 10th May, 2015).

[7]Md. Rahatur Rahman and Shamim Akhter, "Real Time Bi-directional Traffic Management Support System with GPS and WebSocket". The 15th IEEE International Conference on Computer and Information Technology (CIT-2015), 26-28 Oct, 2015, Liverpool, UK, Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=7363185. (Access Date: 23rd February, 2016).

[8]Md. Rahatur Rahman and Shamim Akhter, "Bi-directional traffic management support system with decision tree based dynamic routing". 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), Available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7412080. (Access Date: 6th March, 2016).

[9] Global Positioning System. Available at http://en.wikipedia.org/wiki/Global_Positioning_System (Access Date: 10th May, 2015).

[10] Closer Look at SignalR. Available at http://signalr.blogspot.com/(Access Date: 10th May, 2015)

[11] Google Maps. Available at http://en.wikipedia.org/wiki/Google_Maps. (Access Date: 10th May, 2015).

[12] ID3 Decision Tree Algorithm - Part 1. Available at http://www.codeproject.com/Articles/259241/ID-Decision-Tree-Algorithm-Part. (Access Date: 25th August, 2015).

[13] Dijkstra's Algorithm. Available at https://en.wikipedia.org/wiki/Dijkstra's_algorithm (Access Date: 25th May, 2015).

[14] XML Path Language (XPath) at https://www.w3.org/TR/xpath/ (Access Date: 6th March, 2016).

[15] Moving average. Available at https://en.wikipedia.org/wiki/Moving_average (Access Date: 6th March, 2016).

[16] AccuWeather, http://www.accuweather.com/en/bd/dhaka/28143/january-weather/28143?monyr=1%2F1%2F2016&view=table. (Access Date: 7th March, 2016).

