Reducing Travel Time by Incident Reporting via CrowdSourcing

Leon Stenneth[2], **Waldin Stone**[1], **and Jalal Alowibdi**[2]
[1]Faculty of Engineering & Computing, University of Technology, Kingston, Jamaica
{wstone}@utech.edu.jm
[2]Department of Computer Science, College of Engineering, University of Illinois at Chicago
{lstenn2, jalowi2}@uic.edu

Abstract— *The contribution of this work is the creation of a novel system that enables motorists who witness incidents to submit reports to our system via the web. These reports are aggregated, validated and verified automatically. Then, they are used to update the road network graph. In this work, we designed and implemented an incident reporting system whereby users can report an incident such as an accident or construction on a road network. We extended the FreeSim simulator to accommodate our incident reporting system. Experimental results showed that our system is capable of reducing the travel time of users. We also presented our verification algorithms that are used to verify that reports are fact. Current approaches to congestion detection such as loop detectors probe vehicles, and video image detection may not be available on arterial streets. These technologies may not be available in countries whose transportation budget is low. Our model is less expensive, easy to implement and can work in any environment (e.g. extreme weather). This work is dependent on people's incident response input and not from sensor signals converted to traffic measurements. To the best of our knowledge, this approach is the first to consider web-based incident Crowd Sourcing with automatic incident verification. Other driver based models are telephoned based.*

Keywords: Crowd Sourcing, Shortest path, Vehicular networks, Incident reporting,Travel time reduction

1. Introduction

The aim of this research is to provide end users with the most optimal route for a particular trip. Most routing engines are based on static information and do not take congestion into route planning on arterial streets. We clearly differentiate between *recurring congestion* and *non-recurring congestion*. Recurring congestion is caused by peak travel time when most motorists are expected to be travelling. Figure 1 depicts recurring congestion during a single day in a city. Non recurring congestion is spontaneous and may result from accidents, road maintenance or extreme weather. This project is based on the latter, which is non-recurring congestion, since 2/3 of traffic delays are caused by non recurring incident based congestion [1]. This system proposed and presented in this paper enables the end users to save time and energy on a trip. Congestion information updated by motorist will not include multimedia data. Instead the information sent will be text based because of the limitations of mobile devices (e.g. battery life) and the demands of multimedia (e.g. processing power and storage). According to the 2007 Urban Mobility Report, delays due to heavy traffic are now costing Americans $78 billion in the form of 4.2 billion lost hours and 2.9 billion gallons of wasted fuel [1]. As seen in Figure 1, vehicular traffic congestion is a growing problem with more drivers experiencing severe and extreme delay where the travel time is in excess of 1.5 times of the free-flow trip time. In addition, 2/3 of traffic delays are not caused by recurring congestion. However, due to the traffic incidences, they are caused by point-based spontaneous congestion [1]. We intend to enable the latter, which is spontaneous congestion, to be taken into consideration when returning a trip to the end user.

Effective routing engines must consider both recurring traffic patterns and non-recurring spontaneous traffic events. Also, We intend to take this work further by including spontaneous causes of congestion such as accidents and construction work in our model. We refer to accidents and construction work as incidents. For a transportation graph G = (V,E)where V is a set of vertices and E is a set of edges. V corresponds to a point where two roads intersect and E is the road connecting two vertices. Congestion on the road graph results in an $edge\ e \in E$, that becomes larger for travel time or less for current travel speed and thus the characteristics of G has changed and resulted a new shortest path graph. For the congestion, it should be observed that the edge e

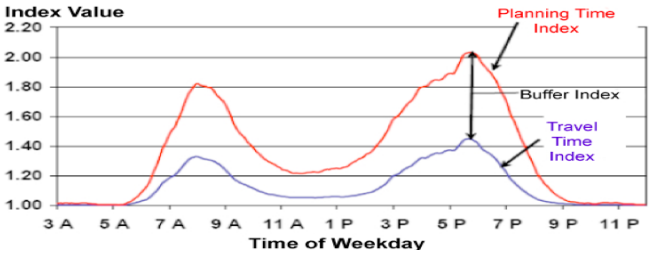

Fig. 1: Diagram Illustrates the Recurring Vehicular Traffic Congestion

will return to its original value after a period of time.

The goal is to be able to tackle incident based congestion (e.g. peak time, accidents, events such as a football game and constructions). Our application accepts congestion or incidents reports from third parties (e.g. police, ambulance drivers, highway patrol and motorist) into consideration when returning possible trip routes to end users of the system. Our goal is not to create a trip planner, instead, it is to update the routing server of trip planners with congestion information to produce more accurate trips to end users.

Accident and Construction reports will be able to be submitted by persons (e.g. police, ambulance, fire personnel and regular motorists) at the scene of the incident with regular motorist given lower priority due to trust factors. Current incident detection frameworks do not have users to submit incidents using the web and automatically verify the reports [12]. Normally, the users may make a telephone call to the transportation center and report the incident information. Using the GPS sensor on the person's mobile device maybe fruitful in identifying an incident's location. Our approach is a web-based incident Crowd Sourcing system with automatic incident verification. Some other strategies to detect incident in prior work are loop detectors, CCTV imaging, probe vehicles and microwave radar [12], [13]. These may not be available on arterial roads or in countries who's transportation budget is low. The remainder of the paper is organized as follows: Section 2 explains the prior work. Sections 3, 4, 5 and 6 introduces new edge weight computation, validation and trust, updating shortest path graph and system design respectively. Section 7 contains the simulation, algorithms, experiments and system evaluation. Section 8 and 9 contains the future work and conclusion respectively.

2. Related Works

A wide variety of sensors can be used to determine where and when an incident has manifested. Inductance loops, microwave radar, probe vehicles and video detection are some of the common sensors based approach [12], [13].

Inductance loops are inexpensive, robust and common in incident detection [12], [13]. The pitfall of inductance loop is that they require road closure for installation and maintenance. Also, these devices under perform whenever the weather condition is extreme, such as several inches of snow on the ground that cover the inductance loop. Video based approaches also have limitations [12], [13]. Since they can be installed above ground, road closure may not be necessary for installation and repair. Video detection may under perform during extreme weather such as fog or any other low visibility weather conditions. Also, we may have the scenario where vehicles hide each other from the cameras. For example, a large truck may hide several cars from the camera and the correct speed of the cars become undetected. Probe based vehicles [13] can be broken down into two categories which are *Beacon based* and *GPS or positioning based* [13]. Beacon based relies on a device called a beacon that uses Dedicated Short Range Communication (DSRC) to communicate with vehicle tags as vehicles passes the beacon. By observing the temporal and spatial components as vehicles pass from one beacon to another, the travel time can be determined. The pitfall of this approach is that all vehicles in this system are expected to have a tag that can be used to communicate with the beacons. GPS or positioning based eliminates the cost of constructing a beacon based network. This facilitates more coverage of the road networks. By taking the GPS coordinates of a probe vehicle at different time intervals, we may be able to infer the current status of the road network.

Observe that the models described above are not common on arterial roadways. Furthermore,it is unrealistically expensive to deploy any of the above systems over the entire road network of a country. The models above operate by first determining a threshold in terms of speed or travel time for a road network. Then, algorithms continue to take sensor readings and if the speed or travel time falls below the threshold an alarm is triggered[12]. Our approach is less expensive and can be used under extreme weather conditions. Based on the fact that we consider people as sensors, the entire road network of a country can be covered. Observe, our approach is not the first driver based model [12]. Other driver based models are not done via the web. Instead, they rely on phone calls for reporting. One of the challenges we faced in driver based models is the inconsistency of the reports submitted by different users.

3. New Edge Weight Computation

One of the major issues for this research is the re-computation of the weight of the road edge that has been affected by the incident. In transportation graphs, edges have weights which may be related to average time to travel an edge or the length of an edge. In our system, the weight of an edge is the current travel speed. We assume a free flow travel speed of 65 mph in this work. If there is an incident on an edge, then the edge attracts a new weight. The research issue is how to determine the new value that should be assigned to the weight of the affected edge. The approach that we use to determine the new edge weight, is dependent on the verification rule that is selected in our system.

Our two verification rules are discussed thoroughly in the algorithm section. The first rule is called "majority rule" and the second called "GPS rule". For the "majority rule", if the reporter is a "regular motorist", then the incident is treated as a fact if the total number of distinct reporters reporting the same incident exceeds some threshold. In our system, for regular motorist the threshold used is three. We then take the mean travel speed of the three reports as the new edge weight. Additionally, if the incident is reported by a uniformed personnel (e.g. police, highway patrol and

ambulance driver), then we treat the incident report as a fact immediately since we assume uniformed personnel are more trusted than regular motorists. In this case, we use the current travel speed reported by the uniform personnel as the new weight of the road edges affected by the incident

For the GPS rule, we treat an incident that is reported as a fact if and only if the reporter is close to where the incident has occurred. Closeness is determined by extracting the reporter's location coordinates (latitude and longitude) and then computing the Euclidean distance between the reporter and the incident's location point. This rule enforces the fact that persons can only reports incidents that they are a close distance from. Privacy issues concerning the location of the mobile user is addressed in our prior work [15], [16]

4. Validation and Trust

While reputation-based trust management protocols have been proposed for mobile wireless networks, the scale and potential dangers of misinformation may lead the system into chaos. The large number of motorist pose a problem, which may be reduced if we have persons willing to participate in congestion control conduct a registration and then we issue to the users them a key or unique id. When these persons witness an incident and see traffic piles, they can report online to the validation system.

We define an end user of the system as persons who submit request for trips. And a trip is defined as a route between two points, a start point and a destination point. We also define an untrusted user as a motorist who is at the scene of a congestion scenario and need to update our routing component so that other motorist can be presented with trips around the congested area. A trusted user in our system is defined as an authority with more permission or trust such as police, ambulance personnel and fire department personnel. For the untrusted users, we will require them to sign up to participate in congestion management afterwards they are issued with a key to use our system. The virtual ratings of this key may increase or decrease depending on the information they provide. We could also offer virtual incentives if the incident information that they provide is correct. The validation for this type of user (untrusted) will be more rigorous than the trusted user. The robustness of the proposed model is determined by the number of similar reports that must be received before the incident is verified as a fact. Also, the constraint that ensure that users must be close to an incident to submit a related incident report (GPS rule).For trusted parties, the validation system may be less vigorous since we assume that these persons have a higher truthfulness level.

5. Updating Shortest Path (SP) Graph

After we compute the new weight for the incident edge, recomputing the entire shortest graph may be time consuming. To compute the shortest path between two vertices in a directed graph, there are three general classes of algorithms which are Naive, Dynamic class and Pre-Computed class [14].

The *Naive* class of algorithms includes Floyd-Warshall's Algorithm [3] and Johnson's Algorithm [4]. Both of which compute the shortest path for all pairs of vertices in a graph. The complexity may be up to V^3. These algorithms, with respect to the specific problem discussed, are required to be executed every time a link update occurs, which is when a trusted/untrusted third party update has been validated. Although the other algorithms to be discussed next may be faster for updates, re-running these algorithms every time a edge update is received allows the fastest path to be determined in constant time, since the fastest paths will already have been computed.

The *Dynamic* All-Pairs Shortest Path algorithm, developed by *Demetrescu and Italiano*, attempts to improve over the naive algorithms by determining the path with minimum cost between two vertices in a graph in constant time where there is an edge update cost [5], [6].

The *Pre-Computed* class of algorithms [14] takes advantage of the fact that the graph is static. With no edge insertions, all of the paths between all pairs of nodes can be precomputed. Then, they do not require the application to take the extra step of determining all of the paths between two vertices when a request for a fastest path is being answered [14].

The *Constant Update Algorithm* [14] sacrifices the speed of retrieving fastest paths for the amount of time it takes to update the weight of an edge. The algorithm does not maintain the fastest paths at all times, but does compute the fastest path when requested by an end user. When the speed on an edge has changed by a specified threshold, the time to traverse that edge will be updated, which can be accomplished in constant time [14].

The *Constant Query Algorithm* [14] sacrifices the speed of updating an edge for the time to retrieve the fastest path. This algorithm always maintains the fastest path between all pairs of vertices by recalculating the fastest paths for all pairs of vertices that have a path containing the updated edge whenever an edge update occurs. When the speed on an edge has changed by a specified threshold, the time to traverse that edge will be updated. Also, the fastest paths for all pairs of vertices, that have a path containing that edge, will be recalculated. In our system, an edge is updated with its new weight each time an incident is verified.

6. System Design

The operation of the incident reporting system is shown in Figure 2. We clearly differentiate between the two types of users that submit incident reports in our system. First, we define a trusted user according to our system as a "uniformed personnel" such as police. The reports, that these personnel generated are more trusted than "regular

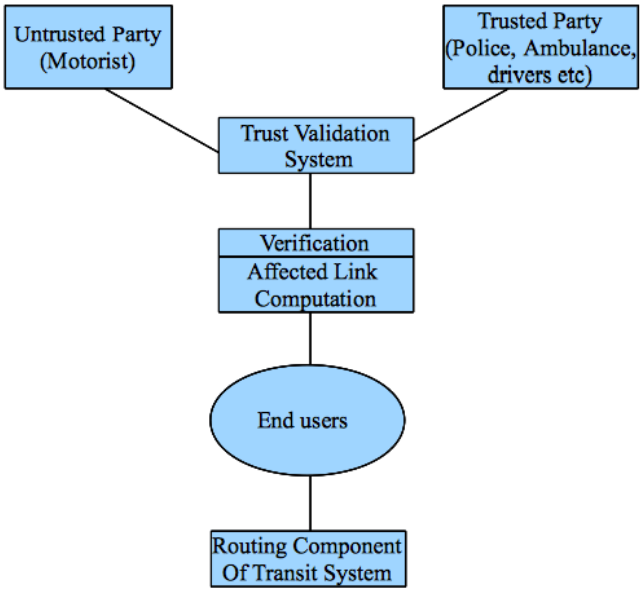

Fig. 2: Diagram Illustrates the System Design

motorist" and incident verification for these reports, are less rigorous. A single report by a "uniformed personnel" may be considered as a fact. Secondly, we classify the other motorist as untrusted parties. These reports, that are generated by untrusted parties, may be more rigorously verified. In our system, if the same incident is reported at least three times by three distinct untrusted parties and the users are close to the incident, then the report is considered a fact.

A user, either trusted or untrusted, submits a report reporting an incident that they witness on a road network. We first validate the user to ensure the user is who they claim to be and is authorized to use the system. After validation we then consider verification which is done by two rules conjunctively (1) Majority rule and (2) GPS rule. These rules are used to verify that an incident, that was reported, is a fact. When verification is completed, the next step is to compute a new weight for the incident edge. This may be done by taking an average of all the current travel speed reported for that edge by the untrusted users. The weight of the edge may also be taken directly from the incident report by the "uniformed personnel". The current travel speed of the incident edge is a parameter for report input.

Finally, the routing component, in our case FreeSim traffic simulator [2], [8] is updated with the new weight for the incident edge. In publicly available transit itinerary planning systems, the updated incident edge update would be sent to the routing engine of the transit itinerary planning system e.g Graphserver. End users, that request shortest path from an origin to a destination, may now get a path where the incidents are taken into consideration when returning the shortest or fastest path.

7. Simulation and Experiment

We extended the FreeSim freeways simulator [2], [8] to accommodate our congestion reporting system. FreeSim is a transportation simulator that enables users to insert transportation graphs and query the shortest or fastest point from an origin to a destination. The simulator implemented a variety of shortest path algorithms such as Dijkstra [9], Bellman Ford [3] and Johnson's [4]. For our experimental evaluation, we used two real world road network. The first is a road network representing the University of Illinois area of Chicago (UIC)consisting of 14 nodes and 15 edges. The second road network is represented a subsection of the Los Angeles area consisting of 50 nodes and 64 edges. A map of the UIC evaluation area in Chicago is shown in Figure 3. The simulation environment was a HP Notebook PC running Windows Vista containing a P8400 Intel DUO 2.27 GHz processor with 4GB RAM. The system is developed using JAVA and the development environment is Eclipse version 3.4.1.

7.1 Architecture and Methodology

In Figure 4, we discuss the architecture of our system. We assume that each user has a mobile device with an Internet connection. Each mobile device have access to positioning information such as GPS, cell tower triangulation or WiFi positioning.

A user u submits a request to our intelligent transportation simulator simulator FreeSim for the shortest or fastest path from an origin node to a destination node on the road network graph. FreeSim engine then runs a selected shortest path algorithm on the graph and returns the shortest or fastest path as requested by the user. All users in our system, that are traversing the road network, have access to the incident

Fig. 3: Diagram Illustrates the Evaluation Area

reporting system and report an incident that they witnessed. A typical report has the following properties *reporter name, reporter id, incident, report location*. The *reporter name* and *reporter id* is the name and identification of the person that submit the incident. The report location is the latitude and longitude of the reporter which may be accessible if the users have positioning features on the mobile device. If the mobile device is limited and cannot infer a latitude or longitude, our system still works. An incident in the report has the following properties *incident location, current travel speed, incident time, incident type , source node, destination node*. Below we explain the properties of an incident as defined in our work.

- *Incident location* represents the latitude and longitude of the incident
- *Current travel speed* represents the current travel speed of vehicles after the incident. Before the incident, we assume that vehicles travel at 65mph.
- *Incident time* represents the time of the incident
- *Incident type* represents the type of incident example accident or road construction
- *Source node and Destination node* are nodes defined by our system. The source node is a node before the incident point and the destination node is a node after the incident point.This information assist us in detecting the incident edge.

7.2 Data Structures

Three hash maps are used to track the reports and the incidents. There is an hash map called *allIncidents* that keep track of all incidents reported in the system. Another hash map called *allReports* that maintain a list of all reports in the system. The reports are stored with reporter identification as index in the hash map. If an incident is confirmed by the GPS rule or majority rule, that report is added to the*confirmedReports* hash map and removed from the *allReports* hash Map. Before we add a report to the confirmed hash map of reports, we first verify that the hash map does not already contain the possible new report.

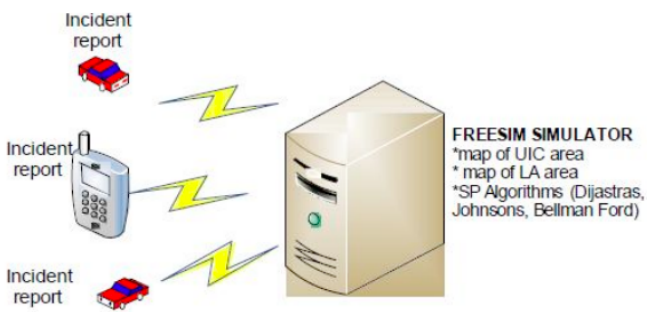

Fig. 4: Diagram Illustrates the System Architecture

7.3 Algorithms

We discuss two algorithms that we used in our system to verify an incident that was reported is a fact. The first algorithm is called *majority rule*. It considers a report or set of reports as a fact if the same incident was reported by at least three different users in the case of untrusted users. The second algorithm is called *GPS rule*. It considers a report or set of reports to be a fact if the location that a user submitted a report from is in close proximity to the report's incident location. Both rules have to be satisfied before a report can be treated as a fact. Algorithm 1 is our majority rule verification algorithm. These algorithms are used each time a new report is entered into the system. In the first step, the algorithm checks to see if the user that submitted the incident report is a "Uniformed Personnel" or not. If the user is uniformed personnel, then the data that they provide is more trusted and a single report from these personnel are considered as a fact. We then update the traffic simulator and add to the confirmed incident hash map the new report and update the road network graph in constant time with the new edge weight. Secondly, if the reporter is a "regular motorist", the algorithm compares against all other reports in the system searching for reports with a common incident source and destination. If the algorithm discovers at least three other reports from different users in the system, then that incident is confirmed. For all confirmed incidents, we update FreeSim traffic simulator with the new weight of the edge containing the incidents. The weight of the new edge is determined by taking an average of the current travel speed as reported by the three users. In line 17 of the algorithm, before updating the simulator, we ensure that this report is not duplicated the list of confirmed reports.

Algorithm 2 is our GPS rule for incident verification. For

Algorithm 1: Majority Rule

```
precondition: allReports !=null
input: Hash Map allreports,confirmedReports
                /* Hash map of all reports and confirmed reports in the system*/
       Report newReport
method:
1.          if(newReport.reporterId == "Uniformed Person")
2.                  allConfirmedReports.put(newReport);
3.                  updateTrafficSimulator()/** update FREESIM **/
4.                  return
5.          endif
6.          int count = 0;
7.          for(j=1 to allReports.size())
8.           if(allReports.get(i)!=null && newReport!=null)
9.             if(allReports.get(i).incident.sourceNode==newReport .incident.sourceNode)
10.          if(allReports.get(i).incident.DestNode==newReport .incident.DestNode)
11.                         if(allReports.get(i).reporterId!=newReport .reporterId)
12.            count++
13.         endif
14.        endif
15.       endif
16.          endif
17.         if (count==3&& newReport!c allConfirmedReports)
18.                 allConfirmedReports.put(newReport)
19.                 updateTrafficSimulator()/** update FREESIM **/
20.                 allReports.remove(j)
21.         endif
22.         endfor
23.
24    end
```

each report, the rule operates as follows. For each report, it checks to see if the incident location is close to the reporters location. This clearly means that reporters can only report incidents that you are close to. To determine if a reporter is close to an incident we compute the Euclidean distance between the reporter's location and the incident's location. Even though we refer to the ruleas GPS rule, positioning techniques used by this rule is not limited to GPS. Other positioning techniques may be considered such as cell tower triangulation or WiFi positioning.

7.4 System Evaluation

In Figure 5, we present the system during operation. The graph is a section of the Downtown area of Chicago in particularly of the UIC area along Halsted Street. In the diagram above to the left is the FreeSim traffic simulator [2], [8] and to the right is the incident reporting system that we extended the simulator to accommodate. The nodes in the graph are as follows starting from the upper left: Halsted and 20th, Halsted and 19th, Halsted and 18th, Halsted and 17th, Halsted and 16th, Halsted and 14th, Halsted and Roosevelt, Roosevelt and Morgan (South West of Halsted and Roosevelt), Roosevelt and Jefferson (North East of Halsted and Roosevelt), Halsted and Taylor (South of Halsted and Roosevelt), Halsted and Polk (South of Halsted and Taylor), Halsted and Harrison(South of Halsted and Polk) Morgan and Taylor (West of Halsted and Taylor), Harrison and Jefferson (East of Halsted and Harrison).

In the demonstration in Figure 5, the user submits a query for the fastest path from Halsted and 20th as origin and Halsted and Harrison as destination. The path recommended is highlighted in red as shown above. The total distance to be traveled is 10.30 miles and duration of travel is 9 mins and 30 seconds. The shortest path algorithm used is Dijkstra [9]. We assume that users travel at 65mph and that each block is 1/10 of a mile. Also, we discuss the situation where an incident is reported and verified and the simulator is updated with the new edge weight after a set of incidents have been reported and verified. In Figure 6, an incident is reported by a user (Alice). The incident is reported to have occurred on the edge between Halsted and Roosevelt and Roosevelt and Jefferson and the current travel speed is 1mph. Two other users before

Algorithm 2: GPS Rule

```
precondition: allReports !=null
input: Hash Map all reports,confirmedReports
                    /* Hash map of all reports and confirmed reports in the system*/
method:
1       for(j=1 to allReports.size())
2.       if(allReports.get(i)!=null)
3.        if(allReports(i).incidentLatitude closeTo allReports(i).reportLatitude)
4.           if(allReports(i).incidentLongitude closeTo allReports(i).reportLongitude)
5.              allConfirmedReports.put(allReports.get(i))
6.              updateTrafficSimulator()/** update FREESIM **/
7.              allreports.remove(i)
8.           endif
9.        endif
10.      endif
11.     endfor
end
```

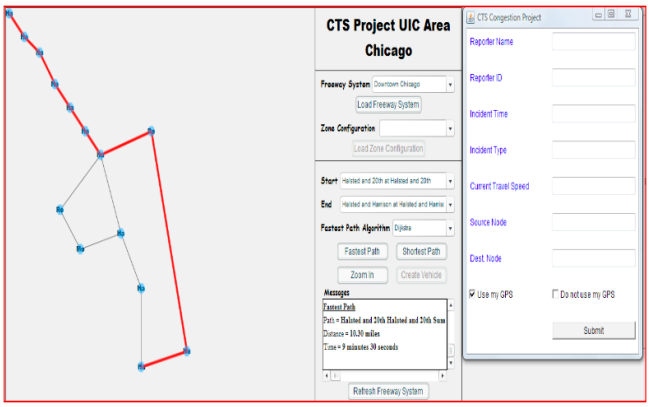

Fig. 5: Diagram illustrates the System GUI-A

Alice had submitted a similar report on the incident and was close to the incident. Based on the majority rule, the incident is verified and the updated edge speed sent to the simulator as shown in blue in the message section of the simulator. After the edge is updated, another user requested the same query as described in the first screen shot. The new fastest path is highlighted in red above and takes 10 minutes and 26 seconds and the total distance is 11.30 miles. Observe that, if the previous shortest path as in the first screen shot was taken by the user, in the event of the incident, the total travel time would have been 27 minutes and 14 seconds. This reduces the travel time of Alice by over 16 minutes. Also, there is also a reduction in fuel consumption, however we did not compute the amount of fuel that was saved.

8. Future Work

We intend to extend this work by using a mobile object generator [10], [11] to generate synthetic users moving in the streets and submitting requests to the system. These mobile objects generated by the moving object generator will report

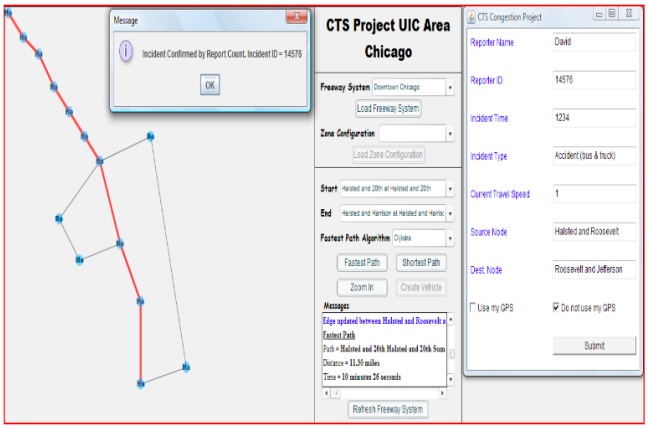

Fig. 6: Diagram illustrates the System GUI-B

incidents and the current travel speed of an edge. Incidents will be generated randomly on the road network, statistical results could then be used to show how much time can be saved by the usage of our system.

9. Conclusion

We present a system that reduces the travel time and fuel consumption of motorist whenever there is an incident on the road network. In our system, the road network graph is very dynamic and new edge weights are added to the graph when incidents are detected. The system we created is an extension of the FreeSim traffic simulator [2], [8] to accommodate our incident reporting system. Algorithms that we used for incident verification in our system are presented. Also, a demonstration example of our system during operation is provided. The demonstration shows a motorist saving over 16 minutes of travel time when our system is used.

Current sensor based models for incident detection may not be available on arterial streets. It is impractical to cover the entire road network with these sensors. Our approach uses people as witnesses and sensors. Clearly, in our model, users can perform collaborative cheating to lead the system into chaos. In subsequent work on this paper, the plan is to develop a collaborative cheating prevention algorithm for this system.

References

[1] R. Mangharam, I. Lee, and O. Sokolsky, "Real-Time Traffic Congestion Prediction," in *2nd Workshop on Experimental Evaluation and Deployment Experiences on Vehicular Networks*, WEEDEV 2009.

[2] J. Miller, and E. Horowitz, "FreeSim âĂŞ A Free Real-Time Traffic Simulator," in *10th International Intelligent Transportation Systems Conference (ITSC 2007)*, pp. 18–23, IEEE 2007..

[3] R. Floyd, "Algorithm 97: Shortest Path," in *Communications of the ACM*, vol. 5, no. 6, ACM 1962.

[4] D. Johnson, "Efficient Algorithms for Shortest Paths in Sparse Networks," in *Journal of the ACM (JACM)*, vol. 24, no. 1, ACM 1977.

[5] C. Demetrescu and G. Italiano, "A New Approach to Dynamic All Pairs Shortest Paths," in *the Thirty-Fifth Annual ACM Symposium on Theory of Computing (STOC '03)*, pp. 159-166, ACM 2003.

[6] C. Demetrescu and G. Italiano, "Experimental Analysis of Dynamic All Pairs Shortest Path Algorithms," in *ACM Transactions on Algorithms(TALG)*, vol. 2, no. 4, pp. 578–601, ACM 2006.

[7] The Bits Laboratory, "TransitGenie website" [Online]. Available: *http://www.transitgenie.com*, 2011.

[8] J. Miller, "FreeSim website" [Online]. Available: *http://www.freewaysimulator.com/index.html*, 2011.

[9] E. Dijkstra, "A Note on Two Problems in Connexion with Graph," in *Journal Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[10] B. Gedik, and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in *25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005)*, pp. 620–629, IEEE 2005.

[11] B. Bamba, L. Liu, P. Pesti and T. Wang, "Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid," in *17th International Conference on World Wide Web (WWW '08)*, pp. 237–247, ACM 2008.

[12] E. Parkany, and C. Xie, "A Complete Review of Incident Detection Algortihms & Their Deployment: What Works and What Doesn't," in *The New England Transportation Consortium*, 2005.

[13] Cambridge Systematics Inc., and Texas Transportation Institute, "Traffic Congestion and Reliability:Trends and Advanced Strategies for Congestion Mitigation," in *Federal Highway Administration*, 2005.

[14] J. Miller, "Dynamically Computing Fastest Paths for Intelligent Transportation Systems," in *IEEE Intelligent Transportation Systems Magazine, Volume 1, Number 1*, Spring 2009.

[15] L. Stenneth, P. Yu, and O. Wolfson, "Mobile Systems Location Privacy: "MobiPriv" a Robust K-Anonymous System," in *IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2010.

[16] L. Stenneth and P. Yu, " Global Privacy and Transportation Mode Homogeneity Anonymization in Location Based Mobile Systems with Continuous Queries," in *6th International ICST Conference on Collaborative Computing: Networking, Applications and Worksharing*, November 2010.