5-2018

Building Data Visualization Applications to Facilitate Vehicular Networking Research

Noah Carter

Simulating Vehicle Movement and Multi-Hop Connectivity from Basic Safety Messages

Noah Carter*, Mohammad A. Hoque*, Md Salman Ahmed†
*Department of Computing, East Tennessee State University
†Department of Computer Science, Virginia Polytechnic Institute and State University
*{carterns, hoquem}@etsu.edu
†ahmedms@vt.edu

***Abstract*—The Basic Safety Message (BSM) is a standardized communication packet that is sent every tenth of a second between connected vehicles using Dedicated Short Range Communication (DSRC). BSMs contain data about the sending vehicle's state, such as speed, location, and the status of the turn signal [1]. Presently, many BSM datasets from various United States locations are available through the connected vehicle testbeds of U.S. Department of Transportation. However, without a proper visualization tool, it is not possible to analyze or obtain a visual overview of the spatio-temporal distribution of the data. For this purpose, a web application has been developed which can ingest a raw BSM dataset and display a time-based simulation of vehicle movement. The simulation also displays multi-hop vehicular network connectivity for DSRC. This paper gives details about the application, including an explanation of the multi-hop partitioning algorithm used to classify the vehicles into separate network partitions. A performance analysis for the simulation is included, in which it is suggested that calculating a connectivity matrix with the multi-hop partitioning algorithm is computationally expensive for a large number of vehicles.**

I. Introduction

Dedicated Short Range Communication (DSRC) offers the potential for a variety of safety-critical applications that leverage vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication modes. These applications follow standard communication protocols defined by SAE J2735 and IEEE 1609.x. One of the primary requirements for one class of safety applications is the broadcast of Basic Safety Messages (BSM) every tenth of a second through which the vehicles share their locations, speed, direction, and mobility information. When other vehicles and drivers have more information about the world around them, they can make better and safer decisions. This is the benefit of sending and receiving BSMs. Such communication is one example of Vehicle-to-Vehicle (V2V) communication, which has major implications for road safety and efficiency and is being widely studied.

DSRC is a short-range communication medium over which BSMs can be sent. However, because the range of DSRC is in many cases less than 1000 meters, it is necessary and useful to employ 'multi-hop' communication. In multi-hop communication, messages traverse a network of vehicles (that act as message relays) in order to reach their destinations. Thus, rather than going directly from the source vehicle to the targets, the messages 'hop' along an ad hoc network of closer vehicles to eventually reach as many targets as possible.

II. Related Works

A. BSM Data Analysis

Some BSM datasets are publicly available and are interesting subjects for data analysis. This includes, for example, the Safety Pilot Model Deployment project, in which tens of Michigan volunteers' vehicles were mounted with BSM-broadcasting devices. One aspect of such data analysis is visualization and simulation. Allowing an analyst to see the data in a helpful and intuitive format enables the analyst to make more informed and strategic decisions [1], [2]. Our previous research has utilized BSM data for analyzing mobility patterns and developing various safety-critical applications in connected vehicle environments [3]–[12].

The application described in this paper is intended to assist an analyst in visualizing location-related aspects of BSM data, specifically vehicle locations and vehicle connectivity partitions. To display the latter, it is necessary to calculate the partitions using a multi-hop partitioning algorithm.

B. Multi-Hop Partitioning Algorithm

Hoque et al. [11] introduced a multi-hop partitioning algorithm based on a modification of Warshall's algorithm. The purpose of the algorithm is to distinguish each vehicular ad hoc network partition–i.e., each connected component of vehicles that are within multi-hop communication range of each other. Again, vehicles can communicate directly, but they may also do so indirectly via one or more intermediate vehicles through which data and messages can pass (or 'hop'). All vehicles that can communicate with each other by one of these two means are grouped together as a partition in the output of the algorithm. The result is a collection of partitions. The current work has further extended the algorithm developed by Hoque et. al. [11] to provide a graphical interface to visualize the connected components within a metro area using an empirically obtained BSM dataset.

The algorithm begins (see Figure 1) by reading in vehicle locations and calculating a square, symmetric 'distance matrix.' Each element $a_{i,j}$ of the distance matrix is the calculated distance between the two vehicles that are represented by row i and column j. Each entry of the distance matrix is then translated into a boolean value by determining whether it is less than or greater than the predetermined DSRC range (i.e., the maximum distance a given DSRC device can broadcast

communications). Those distances that are less than the range become 1; others become 0. **Using boolean algebra,** the newly-formed boolean matrix is then multiplied by a copy of itself. This multiplication represents the first 'hop,' and the result is a 'connectivity matrix.'

The algorithm then proceeds with a series of boolean matrix multiplications, each of which represents a new hop. Every iteration has the possibility of stringing together and consolidating more vehicles into a partition. Once a matrix multiplication results in a connectivity matrix identical to the operands, then the connectivity matrix is considered finalized. Any two vehicles i and j which have a 1 at $a_{i,j}$ are said to be connected together by multi-hop communication. They share the same connectivity partition.

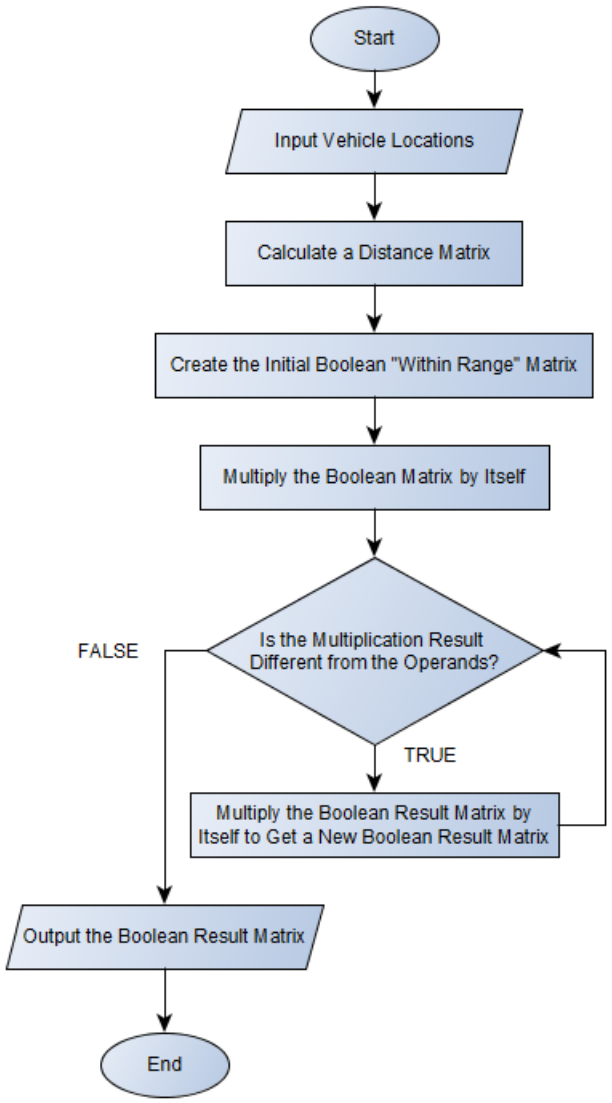


Fig. 1: Flowchart for a multi-hop partitioning algorithm, simplified from Hoque et al.'s [11] implementation.

III. Application

A. Concept and Purpose

The purpose of the application is to ingest BSM data and use it to display a simulation of vehicle activity and connectivity with respect to time. The simulator takes as input Basic Safety Messages from multiple vehicles. Ideally, these are BSMs from vehicles that were concurrently producing BSM output and that therefore have similar timestamps. Once these BSMs are uploaded as a CSV, for any given timestamp the simulator can display the position of each vehicle as a pin on a map. The user can progress from one timestamp to the next, watching the pins move along the roads on the map (see Figure 5).

In addition to its position, each pin has a color and a character(s). Pins with matching color-character combinations are in the same connectivity partition–they are able to send data to each other via multi-hop communication (see Figure 2). For example, if two pins are within the DSRC range of each other, they can communicate with a single hop (and will therefore share the same color and character for each timeframe in which they are within range). If two pins are not within the DSRC range, then they cannot communicate unless there exists a middle pin that they can both reach via DSRC. If there is at least one mutually-reachable middle pin between the two pins, the external pins can use the middle pin to help broadcast their message, creating an ad-hoc network and successfully utilizing multi-hop communication.

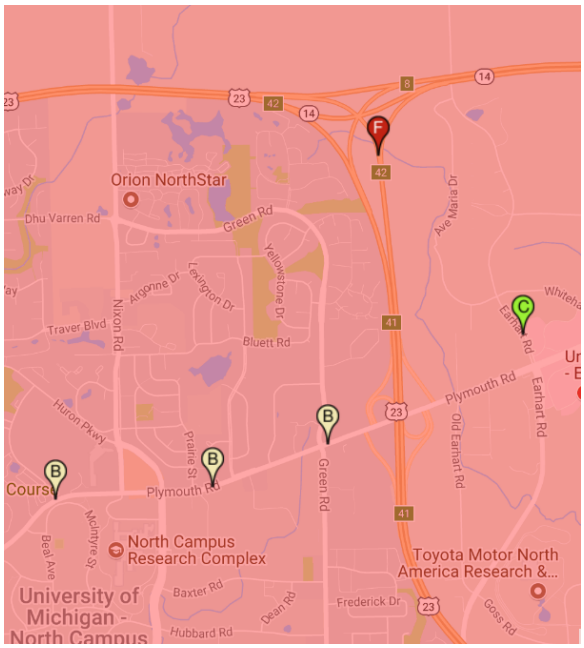


Fig. 2: Vehicle locations represented using markers of different colors

In Figure 2, vehicle locations are represented with pins. The three vehicles at the bottom left share the same connectivity partition; they have the same color and character because they can reach each other via multi-hop communication. With the passing of each timeframe, the colors and characters on the pins will change. This is because vehicles move in and out of range, alternately leaving and mingling among connectivity partitions. The range of DSRC devices–which determines the maximum distance of a single hop–can be adjusted by the user; the default DSRC range, which is based on prior research, is 1000 meters.

The simulation was created with C# and ASP.NET; Javascript calls were made to the Google Maps API.

B. Usage

An example of the expected input format is a large CSV with the structure of Figure 3. Each row should be an abbreviated form of a BSM; all BSM fields are removed except the five listed. After uploading, the user must only click "Run Simulation" or step through the timestamps one-by-one.

DEVICEID	UTC	LAT	LON	MPH_GPS
24	1.36571E+12	42.30066	-83.7999	51.24
24	1.36571E+12	42.30066	-83.7999	51.24
43	1.36571E+12	42.30285	-83.7007	46.94
51	1.36571E+12	42.30217	-83.712	34.03
42	1.36571E+12	42.31097	-83.6783	10.42
87	1.36571E+12	42.22951	-83.6141	75.29
85	1.36571E+12	42.49072	-83.3215	70.69
66	1.36571E+12	42.32052	-83.6888	66.16
42	1.36571E+12	42.31097	-83.6783	10.42
30	1.36571E+12	42.3051	-83.6924	26.45
83	1.36571E+12	42.24526	-83.6728	33.32
87	1.36571E+12	42.22962	-83.6136	75.19
87	1.36571E+12	42.22963	-83.6136	75.14
83	1.36571E+12	42.24526	-83.673	28.42
83	1.36571E+12	42.24526	-83.6729	28.2
85	1.36571E+12	42.49072	-83.3219	70.13
66	1.36571E+12	42.32083	-83.6888	66.58
51	1.36571E+12	42.3021	-83.7122	34.37
43	1.36571E+12	42.30292	-83.7004	47.58
42	1.36571E+12	42.31092	-83.6783	12.97
30	1.36571E+12	42.30515	-83.6922	28.62
24	1.36571E+12	42.30042	-83.7999	50.72
51	1.36571E+12	42.30162	-83.7131	35.97
30	1.36571E+12	42.30538	-83.6913	28.79
87	1.36571E+12	42.23056	-83.6114	76.16
66	1.36571E+12	42.32134	-83.6890	66.34
51	1.36571E+12	42.30206	-83.7121	34.37

Fig. 3: Expected CSV input format for the simulation

IV. Performance Analysis

A performance analysis of the simulation was conducted. This was done in order to study the efficiency of both the simulation as a whole and the underlying multi-hop partitioning algorithm as the number of vehicles increased within a confined space. The result was a greater understanding of some of the application's performance limitations. The multihop partitioning algorithm was identified as taking the bulk of the computation time. Future performance analysis should study the effects of keeping vehicle density constant as vehicle count changes.

A. Process

The performance testing was done by artificially generating BSM data files and measuring the simulation's performance in milliseconds when run on these files. Each generated file had a different number of vehicles per timestamp (N); some had as few as 1 vehicle and others had as many as 200 vehicles per timestamp. All vehicles' positions were constrained within a fixed rectangular area of 348.16 km in Ann Arbor, Michigan. They would appear at random points within the green rectangle in Figure 4. Rather than only appearing on road surfaces, they could appear at random anywhere within this rectangle. Also, their positions in one timestamp did not influence their positions in the next timestamp. Whereas vehicles were confined to a given territory in the generated data, as the number of vehicles increased the density rose quickly.

For test files that contained many vehicles, fewer timestamps were included. This was done to normalize the test file sizes to less than 5000 KB. It was observed during the experiment that the application incurred a significant amount of delay when uploading files larger than about 4500 KB.

The GPS coordinates of the confining rectangle for generated data are as follows, as seen in Figure 4: (42.356186, -83.522030), (42.356186, -83.816270), (42.226673, -83.522030), and (42.226673, -83.816270).

In Figure 5, the graphical interface for the simulator has few vehicles (it has a sparse distribution). In Figure 6, however, there is a dense distribution of vehicles and the corresponding partitioning is shown. In Figure 6, the time to calculate the connectivity matrix has become very noticeable following the rise in vehicle count. The number of partitions has increased to its peak and will begin to lower if additional vehicles are introduced.

B. Results and Conclusions

The completion time was measured in milliseconds and logged for various parts of the simulation process. Also logged was the number of individual input vehicles within the BSM timestamp. After numerous runs with different input CSVs, the log files were combed for the data and patterns were observed; see Figures 7, 8, 9, and 10.

The time needed to populate the display with pins was insignificant, even when the population of vehicles was large (see Figure 7). This was important to verify because it was necessary to determine whether delays were being caused by the internal multi-hop process or by the display medium.

Note that the simulation began to take considerable time at a certain point (see Figure 9). This may present a problem for those that wish to use the current tool to study very large vehicle populations (in excess of 200 vehicles). In the future, an 'export simulation' functionality could be implemented that could allow the user to run the simulation in the background and output the results to a file, to be displayed later.

From Figure 7, it appears that time required for rendering the graphic display would increase linearly as the number of vehicles increases. The display of each additional vehicle requires only the fetching of the appropriate pin image from Google's server and the display of that pin. Ultimately, time to display was insignificant (relative to the amount of time needed to perform the consecutive matrix multiplications to form the multi-hop partitions). Note however that this depends on the strength of Internet connection.

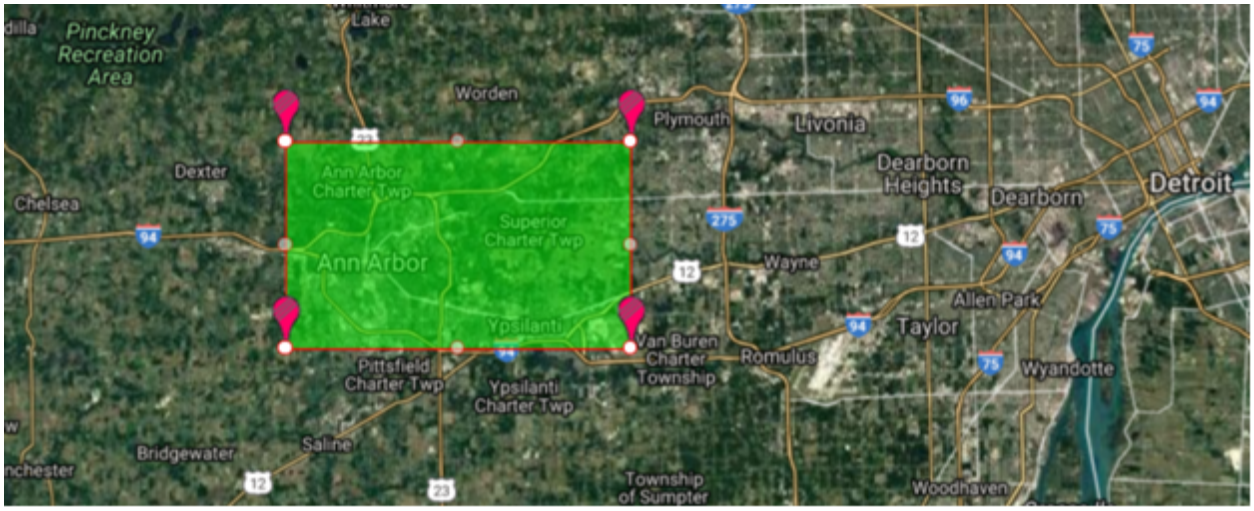


Fig. 4: Geographical location of the sample dataset used for simulation

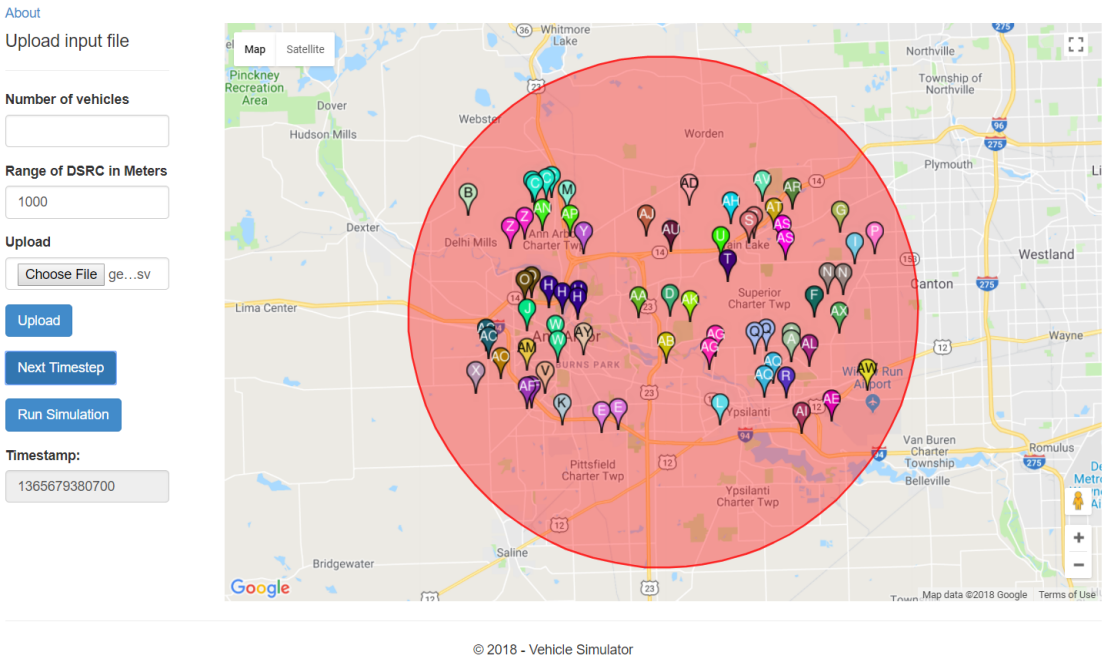


Fig. 5: Graphical interface of the simulator with sparse distribution of nodes

In Figure 8, the distance calculation has become more expensive as the number of vehicles has increased. A prerequisite to performing the partition calculations was the calculation of the distance from every vehicle to every other vehicle. These calculations needed to take place exactly once before the partitions of any particular timestamp could be determined. Calculating these distances involved approximately n^2 amount of work. (The formula for this is $n(n-1)/2$, which is close to n^2.) The results were in accordance with this expectation. Though the calculation of distances would have eventually become a problem as the number of vehicles increased, it would still be a tiny fraction of the amount of work needed to perform the partition calculations (assuming that the vehicle density was allowed to increase).

A series of consecutive matrix multiplications forms the multi-hop partitions. In Figure 9, we see that as the number of vehicles increased, the multi-hop algorithm became the most costly part of the simulation. It was concluded that this algorithm is overall the most work-intensive part of the simulation by far. At 200 vehicles (0.5 vehicles per km^2), each timestamp was taking 6 seconds on average.

The simulation was run each time assuming that each vehicle had a DSRC range of 1000 meters (1 km). As the number of vehicles within the rectangle increased, so did the number of partitions (Figure 10). The number of partitions increased as vehicle density increased, up to a saturation point. However, the rate of increase declined as the density grew. It was observed that at a density of 0.5 vehicles per km^2, the number of partitions began to decrease. If even greater densities had been tested, the number of partitions would have fallen to 1. It is expected that at a density of 1 vehicle per km^2 (where the square root of the inverse of the density equals

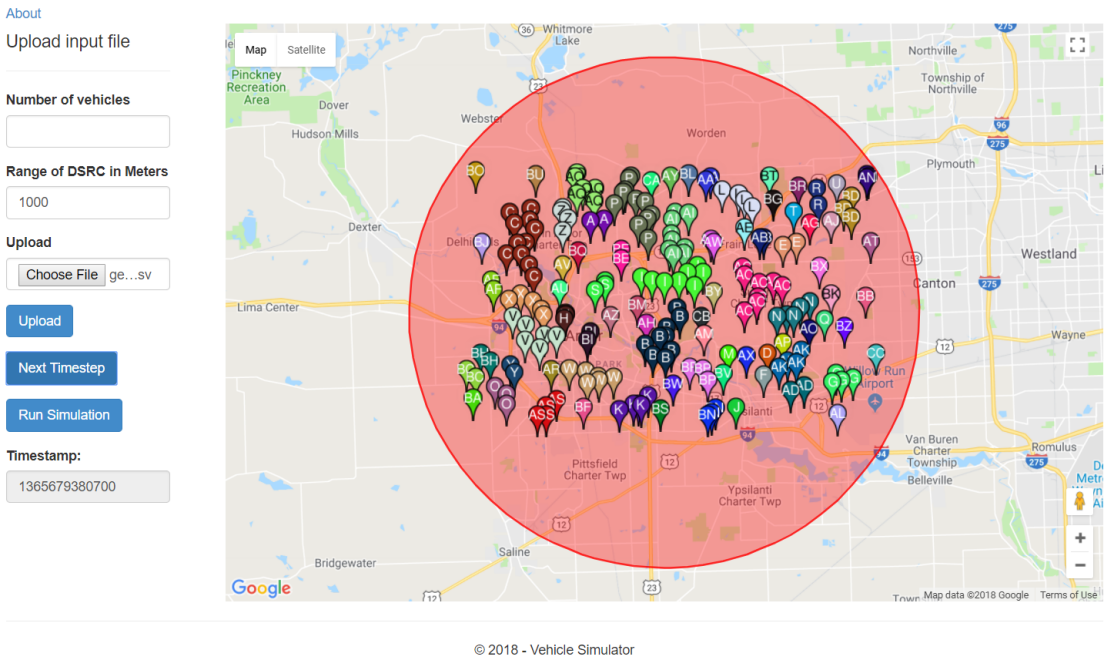


Fig. 6: Graphical interface of the simulator with high density of nodes

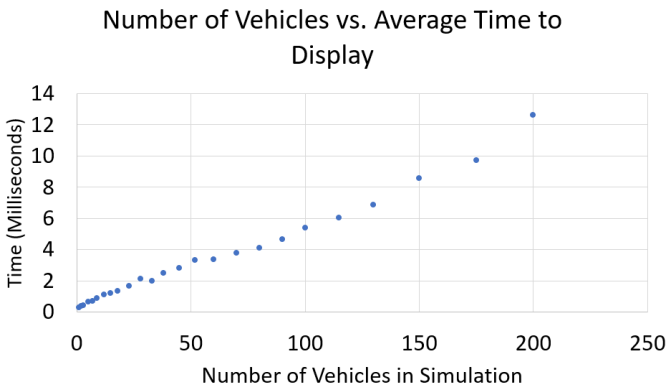


Fig. 7: Average time to render graphic display

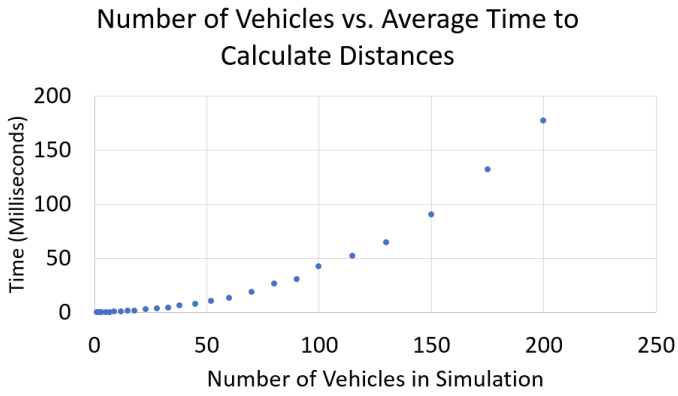


Fig. 8: Average time to calculate distances between vehicles

the transmission range) there would have been 1 partition on average. Again, however, the artificial data allowed vehicles to disperse themselves randomly. In real data, vehicles would limit themselves to the roads, effectively reducing the space between vehicles. Thus for real data the density at which there would have been only a single partition would have been considerably less than that of this artificial data. This is especially true when the roads are less like a grid and more like a highway, or when the DSRC transmission range is increased by flat land and few obstacles.

C. Limitations

The program written to artificially generate BSM data permitted vehicles to be randomly generated anywhere in the rectangular area. In reality, vehicles stay on roads and often share the same road. They do not have the freedom of going everywhere. Thus, in reality, vehicles will be less

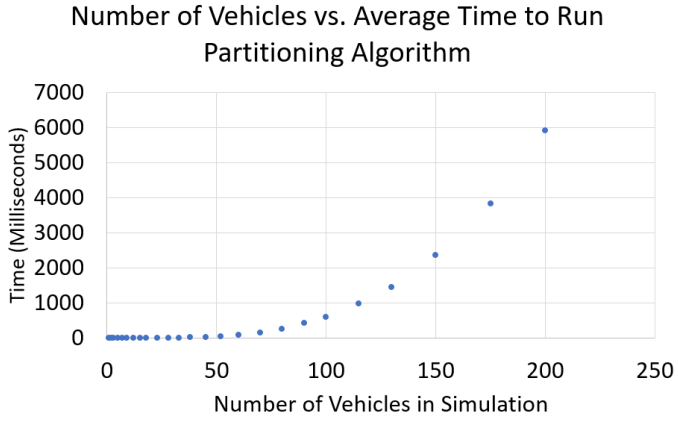


Fig. 9: Time complexity of partitioning algorithm

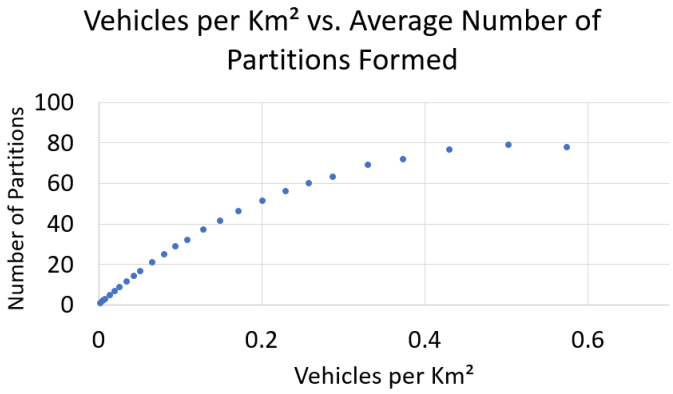


Fig. 10: Average number of partitions

sparse and less dispersed than they are in the artificially-generated data. This means that, if real BSM data had been used for performance analysis, there likely would have been more partitions than were observed. The number of partitions could have thereby had an impact on the time to calculate the partitions. There is therefore a certain degree of uncertainty regarding the extent to which the simulation will perform similarly for real data. Since this application is intended to assist transportation researchers and analysts, who generally focus on real BSM data, in the future we will run the performance analyses with real data.

V. Conclusion

An intuitive visual simulator has been developed to assist transportation researchers and analysts to study BSM data. This paper has documented the application and shared the details of its performance analysis. The application successfully generates time-ordered displays that represent vehicle positions and connectivity statuses. The complexity of the partitioning algorithm has been determined to become the a bottleneck when the number of vehicles increases within a confined space. In a future build of the tests, it will be a goal to compare the effects of vehicle population size with those of vehicle population density. For example, keeping vehicle density constant while increasing vehicle population may produce interesting results. In addition, another advancement would be to perform the performance analyses with real data instead of generated data.

References

[1] S. P. M. Deployment and members of the test conductor team, "Safety pilot model deployment one day sample data, from ann arbor, michigan," *U.S. Dept of Transportation Intelligent Transportation Systems Joint Program Office*, 2014.

[2] J. Liu and A. Khattak, "Delivering improved alerts, warnings, and control assistance using basic safety messages transmitted between vehicles," *Elsevier*, 2016.

[3] M. A. Hoque, X. Hong, and B. Dixon, "Analysis of mobility patterns for urban taxi cabs," in *Computing, Networking and Communications (ICNC), 2012 International Conference on*. IEEE, 2012, pp. 756–760.

[4] A. Elbery, H. Rakha, M. Y. ElNainay, and M. A. Hoque, "An integrated architecture for simulation and modeling of small-and medium-sized transportation and communication networks," in *Smart Cities, Green Technologies, and Intelligent Transport Systems: 4th International Conference, SMARTGREENS 2015, and 1st International Conference VEHITS 2015, Lisbon, Portugal, May 20-22, 2015, Revised Selected Papers*. Springer International Publishing, 2015, pp. 282–303.

[5] M. S. Ahmed, M. A. Hoque, and P. Pfeiffer, "Comparative study of connected vehicle simulators," in *SoutheastCon 2016*. IEEE, 2016, pp. 1–7.

[6] M. S. Ahmed, M. A. Hoque, and A. J. Khattak, "Demo: Real-time vehicle movement tracking on android devices through bluetooth communication with dsrc devices," in *Vehicular Networking Conference (VNC), 2016 IEEE*. IEEE, 2016, pp. 1–2.

[7] M. S. Ahmed and M. A. Hoque, "Partitioning of urban transportation networks utilizing real-world traffic parameters for distributed simulation in sumo," in *Vehicular Networking Conference (VNC), 2016 IEEE*. IEEE, 2016, pp. 1–4.

[8] M. S. Ahmed, M. A. Hoque, J. Rios-Torres, and A. Khattak, "Demo: Freeway merge assistance system using dsrc," in *2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*, 2017, pp. 83–84.

[9] D. Jordan, N. Kyte, S. Murray, M. A. Hoque, M. S. Ahmed, and A. Khattak, "Poster: Investigating doppler effects on vehicle-to-vehicle communication: An experimental study," in *2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*, 2017, pp. 77–78.

[10] M. S. Ahmed, M. A. Hoque, and A. J. Khattak, "Intersection approach advisory through vehicle-to-infrastructure communication using signal phase and timing information at signalized intersection," Tech. Rep., 2018.

[11] M. A. Hoque, X. Hong, and B. Dixon, "Efficient multi-hop connectivity analysis in urban vehicular networks," *Vehicular Communications*, vol. 1, no. 2, pp. 78–90, 2014.

[12] M. A. Hoque, X. Hong, and S. M. Ahmed, "Parallel closed-loop connected vehicle simulator for large-scale transportation network management: Challenges, issues, and solution approaches," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 4, 2018.

Displaying Real-Time Signal Phase and Timing Information Using a Client-Server Application Model

Noah Carter, Nusrat Chowdhury, Mohammad A. Hoque, Jacob Hoyos, Matthew Dale, JT Blevins, Nick Hodge
Department of Computing
East Tennessee State University
{carterns, hoquem, hoyosj, dalems, blevinsjt, hodgen, chowdhury,}@etsu.edu

***Abstract*—Present-day roadway intersections are characteristically dangerous and inefficient. Vehicle-to-Infrastructure communication provides a means for improving these conditions. For example, communication between a traffic controller and a human driver's mobile device can provide the driver with foreknowledge about current traffic signal states or upcoming state changes. A proof-of-concept version of such a client-server application has been developed which allows a server to ingest live data from a traffic controller and send it to a requesting web client over HTTP. The client machine displays the live intersection data in the form of an intuitive GUI. This paper presents the details of that system. The intended use is to serve as a basis for building and testing future applications which access live traffic controller data.**

I. Introduction

Intersections are generally less efficient and less safe than other parts of the roadway. More than 50% of crash-related fatalities and injuries in the United States occur at or near an intersection [1]. Sudden decelerations, idle time, and accelerations from stop lead to an increased rate of fuel consumption at these locations, thereby releasing a larger quantity of environmentally-harmful emissions and incurring higher fuel expenses. Elevations in fuel consumption and the risk of collision escalate costs to society and the environment while endangering drivers and inducing anxiety.

These suboptimal conditions are created in part by drivers' imperfect knowledge regarding the timing of future signal changes. If given foreknowledge of upcoming state changes, motorists would have a longer reaction time, allowing them to adjust their speed appropriately and/or gradually. Vehicle-to-infrastructure (V2I) communication provides an avenue for providing such foresight. To obtain real-time information about an intersection's current and upcoming state, it is necessary to access the Signal Phase and Timing (SPaT) information of a traffic controller.

A traffic controller is a device that continuously dictates the live behavior of traffic lights at an intersection. Traffic controllers can communicate with Signal Phase and Timing (SPaT) packets over UDP. These SPaT packets contain information about the current state of the intersection as well as the timing for and the configuration of the next upcoming state.

Applications can be designed to capture and parse these SPaT packets such that they become human-readable. The application described in this paper parses SPaT and displays it in an intuitive GUI. The application also submits the parsed data to a remote virtual server hosted by Amazon Web Services (AWS). The AWS server makes the data available to web clients and provides them with a web UI for displaying the current intersection status.

An important aspect of this application is that it relies on a centralized V2I communication strategy which can employ cellular technology (LTE). All personal devices with access to the World Wide Web–including mobile devices–can potentially view this web GUI representation of an intersection. Another topic of note is that this research assesses the latency of cellular technology when sending and receiving live intersection data. It was found that for this particular application there is very low latency; the client machines can depict traffic signal states in what is almost real-time.

II. Related Work

Many researchers have employed SPaT packets and the mobile LTE network for their assistive V2I applications. For example, Audi America utilized LTE as the V2I communication medium for a smart traffic light information system [2]. Audi also developed an optimal speed advisory system that suggests a traveling speed to motorists which, if followed, will allow the vehicle to reach an intersection during a green light phase. This advisory uses 3G/4G communication and a backend server [3].

An environmental research group from the University of California Riverside developed a similar application using SPaT data and LTE. It gave drivers advance notice about the upcoming traffic signal timing. Their study also contrasted the fuel consumption of an informed driver with that of an uninformed driver [4]. Yet another similar pollution-reducing application was developed with support from the USDOT. It relayed real-time SPaT information to drivers such that they were able to make informed choices [5]. Ahmed et. al proposed an advisory system that generates live traffic signal information and a speed advisory to help the driver reach a destination intersection on time [6]. Like the above works, Ahmed et. al used SPaT information to inform their advisory algorithm. However, instead of LTE/3G/4G, Dedicated Short Range Communication (DSRC)–a decentralized technology–was employed.

Each of the above Intelligent Transportation System (ITS) applications help to reduce the number of unexpected stops and starts that drivers need to perform. They thereby reduce emissions, fuel use, and safety hazards. Advances in connected vehicle technology create opportunities to develop many such applications that can boost drivers' awareness and thereby raise transportation efficiency and safety.

The system described in this paper does not yet include advanced user features, such as an advisory algorithm which can suggest a speed to drivers. It is not intended as a finished product, but as a basis for further application development. It focuses on quickly delivering real-time SPaT information to multiple mobile devices simultaneously, and it does so (like the applications of Audi and California Riverside) over the WWW. The desktop UI and the web client UI are intended to assist future researchers in testing the accuracy of their parses of SPaT communications.

III. Architecture and Implementation

A. Overall Architecture

The task of the system is to allow users to remotely monitor the status of a roadway intersection using the browser of a personal device. As a web app, this implementation does not require the installation of additional software on client machines. Figure 1 gives an overview of the system architecture. The system consists of a traffic controller, a local network switch, a local machine which serves data to AWS, the AWS virtual cloud server instance, and the clients.

First, the traffic controller (a Siemens m60) and a local Windows machine are connected over Ethernet on the same subnet. The traffic controller continuously transmits SPaT data in UDP over Ethernet with a frequency of ten times per second. As aforementioned, these SPaT packets contain information about current signal state of intersections. The local Windows machine runs a C# application which parses the SPaT data; the human-readable, parsed SPaT is displayed alongside a visual GUI representation of the intersection (see Figure 4).

Meanwhile, upon receiving a changed SPaT configuration the Windows application is triggered to make an asynchronous HTTP request containing the parsed data structure as its content. The request is made to an AWS virtual server (which in our tests happened to be in Oregon). This central Windows cloud server hosts an ASP.NET controller which receives the updated data structure within the request and saves it as text to a public HTML file (via an overwrite) on the singular AWS instance.

Next, various client machines (such as phones, tablets, or desktops) may navigate to a public URL hosted by the same AWS instance. The corresponding ASP.NET controller returns another HTML page which contains an Ajax call. The Ajax runs on the client machine, repeatedly requesting the saved, client-readable SPaT text file from the AWS environment at an interval of once per 100 milliseconds. Upon receiving a parsed SPaT file with updated information, the client's page uses JavaScript to interpret the text and then recreate a visual component of the page. This component is a GUI that represents the live intersection (see Figure 1).

B. Interpreting SPaT Packets

Note that to parse and represent the traffic status of the intersection visually, both the local Windows GUI and the web UI were coded to expect the traffic signal's control cycle to be an 8-phase cycle. This type of cycle accounts for left, through, and implied right movement within the intersection of two roads (a minor street and a major street). Figure 2 illustrates a classic 8-phase intersection. In this type of cycle, each of the 8 phases correspond to a particular direction and are assigned a phase number between 1 and 8.

The UIs account for all of the 8 phases and consist of four separate traffic signals. However, the application described here is limited to an 8-phase cycle; it cannot dynamically interpret other types of intersections. Figures 4(a), 4(b), and 4(c) depict the GUI of the Windows application. In Figure 4(a), green arrows are lit on the minor street to represent the left turns which correspond to phase 3 and 7. In Figure 4(b), left turn lights are lit green for the major street, corresponding to phase 1 and 5. Figure 4(c) shows that the intersection is permitting the through movements of phase 4 and 8. During phase changes, green lights are be replaced by yellow lights.

The desktop application utilizes the C# implementation of the UDP Client to read in SPaT data streams in raw bytes. Each SPaT packet contains 246 (or sometimes 245) bytes. See Figure 3. Bytes 210, 212, and 214–representing phases 1-8 of red, yellow, and green lights respectively–are relevant to the display. The packets are sent in UDP with the National Transportation Communications for Intelligent Transportation System Protocol (NTCIP), which is a common protocol used by many transportation devices.

C. Implementation Details

In the Windows and web UIs, the active traffic lights as determined from the SPaT are mapped to static JPG images of active lights that appear and disappear based on the activation of the appropriate bits. By default the local Windows application is configured to send data to an existing remote, public ASP.NET controller. However, it can run without a destination to which it egresses SPaT data. Or, it can transfer the data over HTTP to a local machine running the appropriate ASP.NET environment.

It was observed that if a significant programmatic pause was used as part of the Windows application or if HTTP calls were made too often (i.e., ten times per second), the local Windows machine would gradually fall behind the traffic controller. Eventually this would result in all remote clients displaying inaccurate views because of the substantial lag. Removing all programmatic pauses from the server and only performing HTTP requests for updated configurations prevented this issue. It also decreased any initial synchronization time needed between the traffic controller and the other devices.

D. Experimentation, Testing and Latency Analysis

1) Resource Usage: Testing during March 2018 involved 16 hours of cumulative program runtime; as a result, the AWS server instance reached more than 85% of the monthly free tier limit for file inputs and outputs, with 8.5 million I/Os. That is,

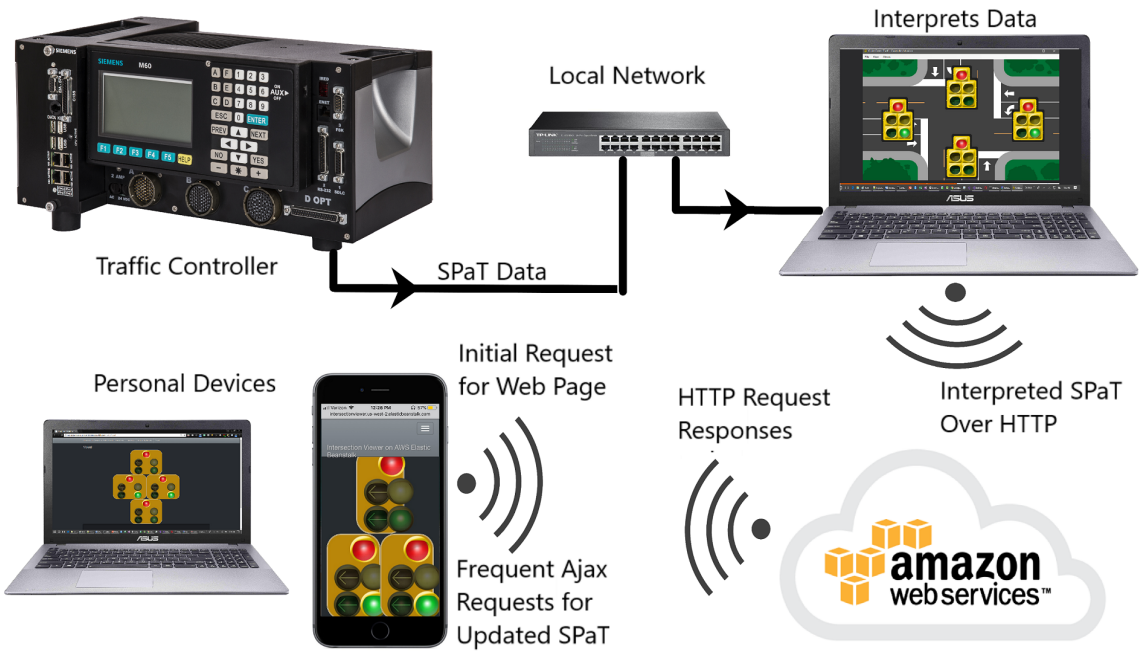


Fig. 1: Overall communication architecture for the application

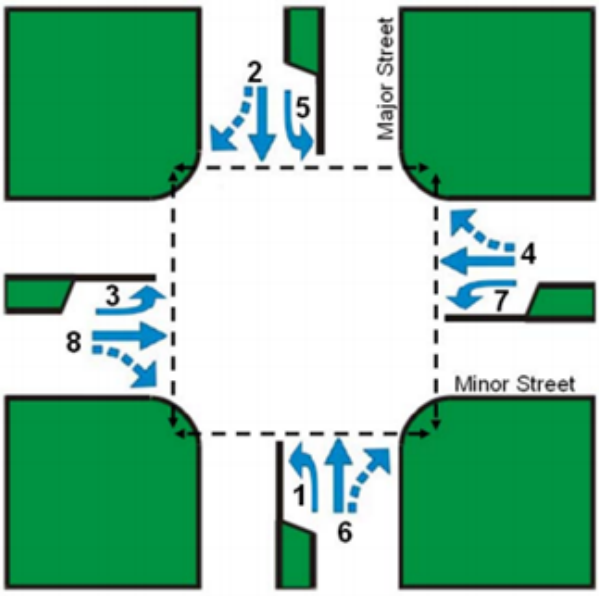


Fig. 2: A standard 8-phase intersection, as used for the application

the server read or wrote to its text files 8.5 million times. This rate is not sustainable. Writing to a file has the advantage that the file is easily available to all HTTP clients, but opening, writing to, reading from, and closing that file is expensive (computationally and financially). Another approach should be taken to allow all clients to obtain the stored information, without sacrificing the current low latency (see below). TCP sockets were suggested as an alternative to maintaining a text file on the server.

Several hours of uninterrupted testing resulted in a high temperature of the mobile device used in this experiment. It may be beneficial to revisit and observe the extent to which the client's resources are used by the Ajax calls, which currently pull text from the server and rebuild the HTML display at a rate of once per millisecond. A lower update frequency may reduce the cost of the software while not introducing noticeable latency.

2) Testing with a Live Intersection: The application was tested at the Tennessee Traffic Division of Johnson City. The traffic engineers directed the traffic controller for the intersection of Indian Ridge Rd and West Market St to send its SPaT data to the IP address of our Windows laptop. This intersection was largely a classic 8-phase intersection, with W Market being the major and Indian Ridge being the minor. The laptop received the SPaT data and, using the previously-discussed methods, sent it to the ASP.NET controller on AWS. Multiple cell phones then accessed the client web page and textual SPaT over LTE, displaying the intersection's state live.

The delays between the laptop's display and the cell phones' displays were, to the human eye, observable but not significant. It was estimated as less than 0.2 seconds. This meant that the data was traveling from the laptop to the AWS server instance in Oregon, being requested by the phones, and being transmitted to and displayed by the phones with, from a human visual perspective, practically with no latency.

As a way of comparison, the actual intersection was then observed with two methods: via the division's live-feed traffic video cameras and in-person. The video received by the divisions traffic cameras was observed to have a noticeably greater latency than that of the research application. On the phones, the display would update roughly 1.0 seconds sooner than the traffic division's live videos. (This may have been because of the way the cameras were configured or because the cameras needed to transport large video files while the

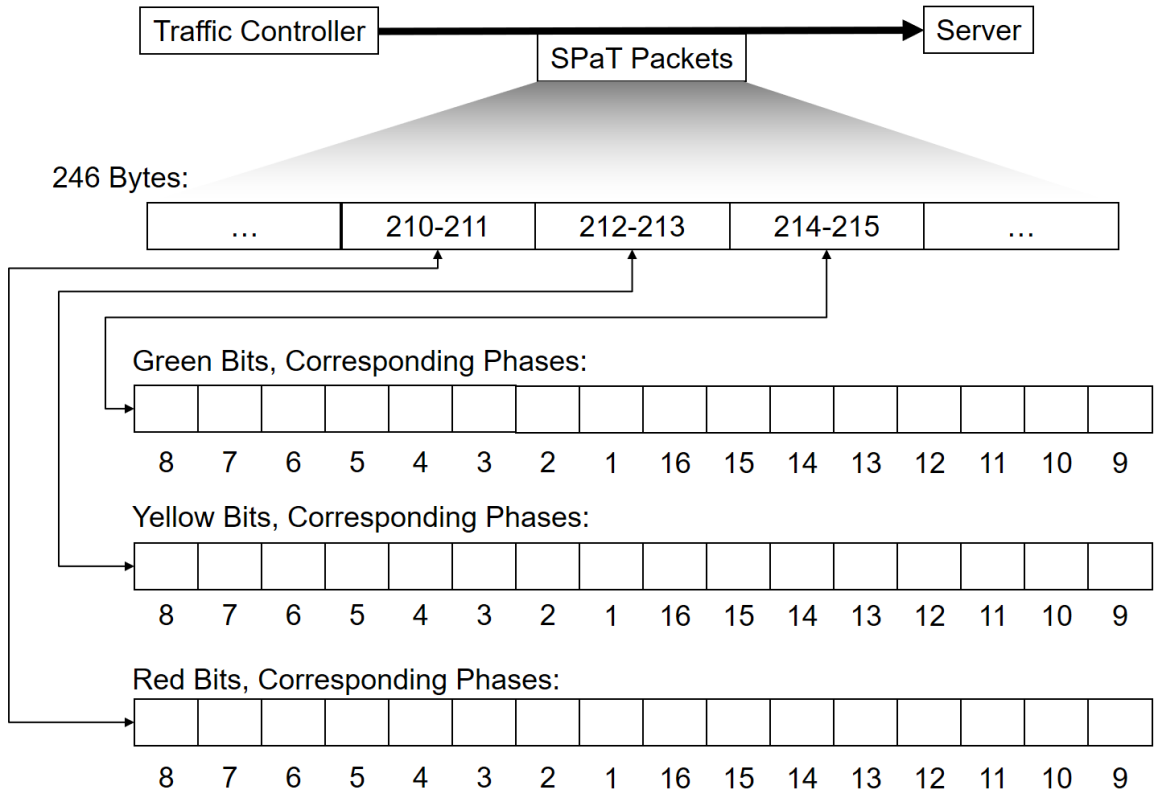


Fig. 3: SPaT packets consist of 245 or 246 bytes. Only certain bytes are relevant for this application.

research application needed only to make HTTP requests with small content loads.) When physically standing outside at the intersection, the delays between the actual traffic lights and the phones were, likewise, hardly noticeable to the human eye. The delay was approximately 0.3 seconds (as repeatedly measured with a camera and a stopwatch).

For the most part, the display was accurate, in addition to being fast. However, there were unexpected inaccuracies in the display. For example, a light would show as green on the phone when in reality both green and green left-turn were triggered. This was due, in part, to the fact that the particular intersection was using unanticipated phase logic. Though 8-phase, the intersection utilized "overlaps," in which the activation of a single light can be triggered by the activation of more than one individual phase. Removing this display bug from the research application will simply be a matter of modifying the way the SPaT data structure is interpreted.

E. Research Benefits

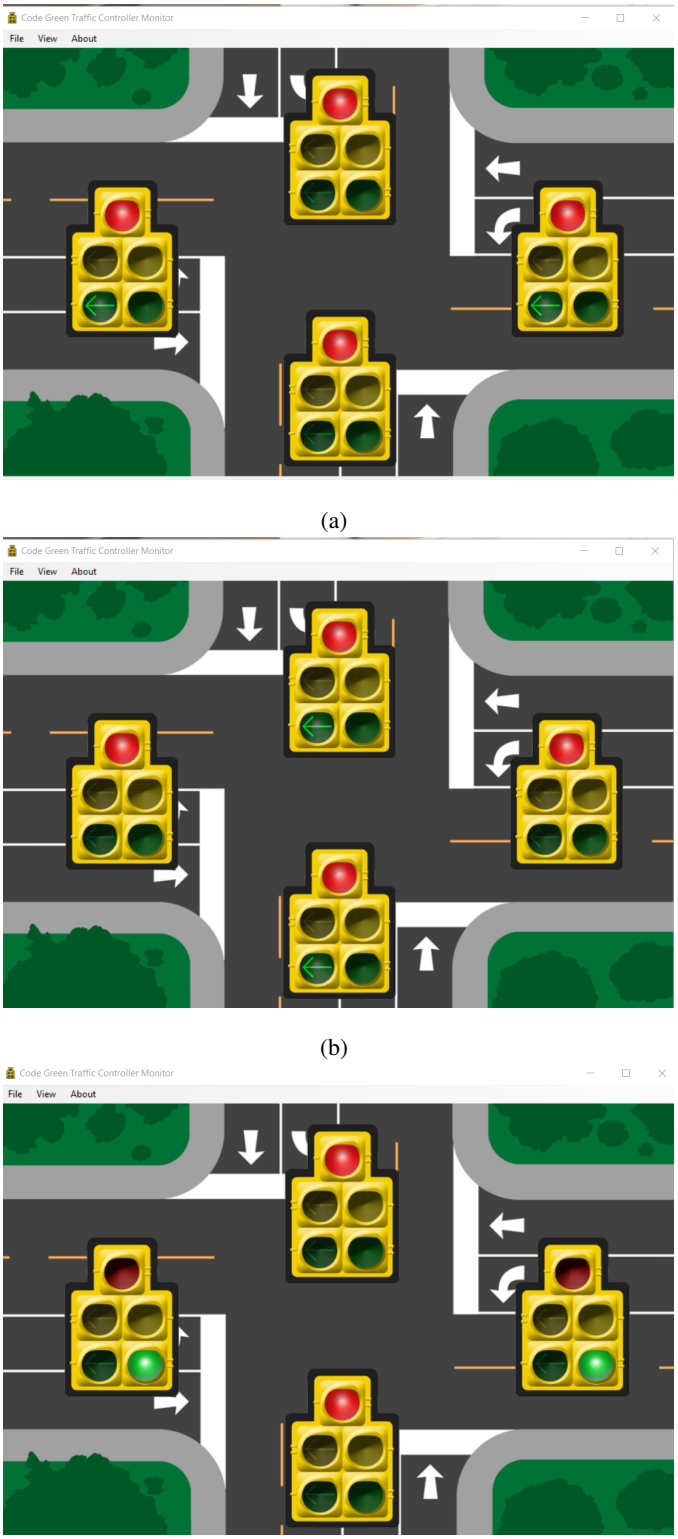
Although the web application does not include a driver speed advisory, it is a proof of concept and prototype for further development. A team of graduate students at East Tennesssee State University is currently developing an application that should give drivers foreknowledge about the upcoming status of intersections, similar to those developed in by California Riverside and others. Part of their project requires a central server to relay SPaT data to mobile devices over the wireless mobile network. AWS was discussed as the potential host for the centralized portion of their application, but there were concerns about latency. (Minimizing latency is critical to delivering timely notifications to drivers.) The low latency of the research application discussed here has now demonstrated that AWS and the mobile network could be suitable communication mediums for such applications. Hence, the researchers at ETSU's Vehicular Networking Lab plan to use proposed application as a starting point for the AWS-hosted, centralized portion of their safety and efficiency V2I application.

F. Future Development

Many issues with this application can be fixed in the short term. Currently, the application parses SPaT data from the controller into an arbitrary, homegrown data structure; the client machines are set up to interpret that format. SAE J7235 is a new standardized format that has been recently developed by the Society of Automotive Engineers to make SPaT communications uniform. A new work item is to update the message format of the this application to the new standard message format for SPaT data, SAE J7235.

Also, the local Windows application presently sends only the traffic light data from bytes 211, 213, and 215 to AWS. These contain the light statuses of an 8-phase intersection. Eventually, the application should also send the "map data," which communicates other details about the intersection such as its location.

The UIs are limited in that they can only interpret the SPaT data from 8-phase intersection configurations. Future work could include removing this limitation by enabling the ability



(a)

(b)

(c)

Fig. 4: GUI of the Windows desktop application

to display variantly phased intersections. Lastly, the CSS of the client page could be enhanced to make the HTML display more user-friendly on mobile devices. In figure 1, the mobile device screen displays the current view of the application.

G. Conclusion

The system discussed here provides an improvable architecture and a prototype for other applications which require a centralized server to host SPaT data. It can be modified and adjusted to assist with the development of centralized V2I applications. The low latency observed when testing on the mobile network with a real intersection has encouraged university graduate researchers to pursue AWS as a host for their centralized applications.

References

[1] "Intersection safety," *U.S. Department of Transportation*, date last accessed 31-July-2017. [Online]. Available: https:https://www.fhwa.dot.gov/research/topics/safety/intersections/

[2] M. Zweck and M. Schuch, "Traffic light assistant: Applying cooperative its in european cities and vehicles," *Connected Vehicles and Expo (ICCVE), 2013 International Conference on IEEE*, 2013.

[3] H. Xia, K. Boriboonsomsin, F. Schweizer, A. Winckler, K. Zhou, W. B. Zhang, and M. Barth, "Field operational testing of eco-approach technology at a fixed-time signalized intersection," *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference (pp. 188-193)*, 9 2012.

[4] Y. Zhao, S. Li, S. Hu, L. Su, S. Yao, H. Shao, and T. ... Abdelzaher, "Greendrive: A smartphone-based intelligent speed adaptation system with real-time traffic signal prediction," *In Proceedings of the 8th International Conference on Cyber-Physical Systems (pp. 229-238) ACM*, 4 2017.

[5] "Audi announces the first vehicle to infrastructure (v2i) service - the new traffic light information system." *Audi*, 08 2016, date last accessed 31-July-2017. [Online]. Available: https://www.audiusa.com/newsroom/news/press-releases/2016/08/audi-announces-first-vehicle-to-infrastructure-service

[6] M. S. Ahmed, M. A. Hoque, and A. Khattak, "Intersection approach advisory through vehicle-to-infrastructure communication using signal phase and timing (spat) information at signalized intersection," *In Transportation Research Board 97th Annual Meeting no. 18-05804.*, 2018.