A Store-and-Forward Dissemination Algorithm Using Locally Shared Vehicle Route Information

Dave McKenney and Dr. Tony White

August 31, 2013

Abstract

Data dissemination within vehicle networks is an important research area as intelligent control algorithms can greatly benefit from accurate up-to-date information. This report proposes a dissemination algorithm that relies on the local exchange of vehicle routes to select appropriate data to broadcast, which can result in a more efficient use of limited bandwidth. The proposed algorithm is implemented within the Omnet++ communication simulator and experiments are performed to evaluate the algorithm's behaviour using a bidirectional coupling with the SUMO traffic simulator. The results of the initial investigation of the algorithm's performance show that it is capable of both selecting appropriate data to broadcast from a vehicle and limiting the amount of data broadcast to prevent flooding of the network. With these promising results, future improvements to the algorithm and intelligent traffic control research are discussed.

Contents

1 Introduction

There are many proposed applications of vehicle-to-vehicle communication, such as in-vehicle entertainment, collision avoidance and automated driving. One of the most important applications from both economical and environmental viewpoints, however, is intelligent traffic control. When a large number of vehicles operate in an uninformed manner without proper control within a road network, traffic jams can form and propagate through the network causing significant increases to travel times and emissions. While intelligent control systems can lead to significant improvements in traffic flow (i.e., McKenney and White, 2013), one of the factors limiting their successful implementation is the availability of accurate real-time traffic data. In Section 3 of the previous document, a number of existing algorithms for disseminating data throughout a traffic network using vehicle-to-vehicle communication were discussed. These algorithms can be used to generate required data and distribute it for use by an intelligent traffic control system.

Three of the key desirable dissemination protocol properties outlined in Section 3.3 of the previous document were volume sensitivity, selection and aggregation of data, as well as localized operation. These are important characteristics, as they allow the dissemination protocol to adapt to widely varying traffic states and operate effectively in a system as large as a city-wide traffic network.

This document outlines a simple, localized dissemination protocol capable of selecting appropriate data to broadcast and adapting to varying proportions of active/intelligent vehicles. The dissemination algorithm presented here is the initial work in an overall research plan that will also include investigations of intelligent traffic control and adaptive vehicle routing based on the information made available by the proposed protocol. This document also provides some examples of future experimentation and development that are planned for the near future. An outline of the remainder of the document is given below.

The remainder of this document is outlined as follows. Section 2 outlines the proposed protocol by breaking the entire vehicle behaviour into a number of possible events, each of which is described in detail. Section 3 outlines the experimental setup used to investigate the basic behaviour of the algorithm, while Section 4 includes discussion of the results obtained through simulations. Section 5 presents the main conclusions from the experimentation and outlines future work to be carried out. Finally, Appendix A details the

software packages used for this work, as well as the modifications made to achieve the presented results.

2 Algorithms

Using the 802.11p protocol for wireless vehicle-to-vehicle communication (described in detail within Section 2.3 of the previous document), vehicles communicate by sending messages over the wireless medium. As the vehicles are using a wireless medium, all data sent is broadcast and available to any vehicle within range. The algorithms in this section outline what data is broadcast from a vehicle and how received data is handled.

2.1 Vehicle Events

The dissemination protocol can be divided into a number of possible vehicle events which trigger specific actions within the vehicle. The first two events considered here are those responsible for broadcasting (Send Observations, Section 2.4) and receiving (Receive Observations, Section 2.8) observations relating to traffic data. These are the most common and basic requirements for a data dissemination protocol within the intelligent traffic domain. The protocol described within this document also proposes the inclusion of route information within beacon-type messages, which allows nearby vehicles to select more appropriate data to broadcast. For this reason, two other vehicle events are included for both sending (Send Beacon, Section 2.3) and receiving (Receive Beacon, Section 2.7) these beacons. Finally, it is desirable that vehicles are capable of updating their knowledge base over time to allow for more accurate decision making, which is why there are also events included for updating the observations (Update Observations, Section 2.5) and neighbours (Update Neighbours, Section 2.6) stored in memory. Below, a summary of each possible vehicle event is given, while a more complete discussion of each event is presented later in the section.

- Send Beacon: This event causes the vehicle to construct and broadcast a beaconing message. This type of message is used by others to determine nearby vehicles, while also being used to exchange limited route-related information.

- Send Observations: This event causes the vehicle to construct and broadcast a message containing traffic state information. This is used to share important traffic observations stored in memory with nearby vehicles.
- Update Observations: This event triggers an update process on the traffic state observations stored within a vehicle's memory. This process is responsible for removing obsolete (old) observations, as well as updating the threshold (explained in detail within Section 2.4) that is used when selecting data to broadcast.
- Update Neighbours: This event triggers an update process responsible for removing neighbours who have not been heard from recently (as determined by a parameter) from memory.
- Receive Beacon: This event is triggered by the reception of a beacon message from a nearby vehicle. This allows a vehicle to acknowledge the existence of neighbours and gain information relating to their planned route.
- Receive Observations: This event is triggered by the reception of an observation message from a nearby vehicle. This allows a vehicle to add new observations to its memory.

2.2 Parameters

A number of parameters are used within the proposed algorithms, which allow the protocol's behaviour to be modified. Below, Table 1 presents existing parameters, their default values and a brief description of each. The default values were used for all experiments presented in this document; however, in later works investigation of different parameter values will be completed.

2.3 Send Beacon

When a vehicle receives a trigger specifying it should send a beacon, it simply builds a message containing its vehicle ID and planned route (list of road segments), then broadcasts the constructed message. Nearby vehicles receiving

Parameter	Default Value	Description
NeighbourUpdate	5 seconds	Determines the length of time a vehicle waits between updates to the neighbour model in memory.
ObservationUpdate	2 seconds	Determines the length of time a vehicle waits between updates to the observations stored in memory.
MinObservationInterval	1 seconds	The minimum delay between sending observations.
MaxObservationInterval	5 seconds	The maximum delay between sending observations.
MaxObservationAge	1000 seconds	The maximum age an observation stored in memory should have (older observations are removed during the update process)
MaxNeighbourAge	60 seconds	The maximum age a neighbour stored in memory should have.
BeaconInterval	2 seconds	The interval at which beacon-type messages are created and broadcast.
BandwidthLength	15 seconds	The size of the time window used to calculate the amount of data recently received.
StimulusConstant	3000	A constant used in the calculation of stimulus. Is meant to reflect the desired maximum amount of bytes received in the last BandwidthLength seconds.
MaxMessageCount	10	The maximum number of observations to be added to a single message.

Table 1: Algorithm parameters with default values and brief description.

these types of messages can use the data within to determine what vehicles are nearby (i.e., within broadcasting range), as well as the currently planned routes of those vehicles. The route data contained within a beacon message can be used by receiving vehicles to select more beneficial data to include in their own observation messages, which is discussed further in Section 2.4. After broadcasting the beacon message, the vehicle schedules the next trigger at a time in the future determined by the BeaconInterval system parameter.

The inclusion of route information in these beaconing messages differs slightly from the general use of beacons, in that beacons usually include no extra information and are only used to announce a vehicle's presence. As mentioned previously, the addition of route information here is used to prioritize data to make more efficient use of limited bandwidth. This, however, comes at the additional cost of bandwidth required to send more complex beacon messages. While it is not investigated within this report, future work could analyse the trade-off between bandwidth consumed by including route information and improved data selection. Furthermore, including other types of information in these messages could also offer an overall advantage within the system. As an example, beacon messages could be used by vehicles to request information regarding specific roads/areas outside of the currently planned route that may be useful (i.e., to be used in alternate route selection).

2.4 Send Observations

Algorithm 1 outlines the process used by vehicles to select and broadcast data. This algorithm takes a number of inputs, including a list of all observations stored in memory and the threshold values associated with those observations. The algorithm begins by calculating an estimate of the available bandwidth (line 1), which is used to determine the likelihood of sending data. The algorithm then iterates through each observation stored in memory in order of increasing threshold (line 2). This approach prioritizes information that has been deemed more important (lower threshold) based on information such as neighbouring vehicles and routes. Each observation is probabilistically added to the potential outgoing message using the calculation on line 3. If the addition of a new observation pushes the size of the message beyond the value of the MaxMessageCount parameter, the vehicle broadcasts the message, schedules the next broadcast, and terminates the process (lines 4-9). Otherwise, the vehicle continues to probabilistically add

the remaining observations, sending the message and scheduling the next broadcast once all observations have been considered. This algorithm ad-

Input: B_{rec} - Bytes of data received in the last BandwidthLength time
Input: θ_o - Thresholds for observations in memory
Input: O - Observations stored in the vehicles memory
Input: ToSend - List of observations to be sent
1 $B_{avail} = max(1 - \frac{B_{rec}}{StimulusConstant}, 0)$
2 **foreach** *o in O ordered by increasing* θ_o **do**
3 With probability $P = \frac{B_{avail}}{B_{avail}+\theta_o}$ add o to ToSend
4 **if** *Size of ToSend includes MaxMessageCount observations* **then**
5 Broadcast ToSend
6 Clear ToSend
7 Schedule next broadcast randomly in the interval [MinObservationInterval, MaxObservationInterval]
8 Exit
9 **end**
10 **end**
11 Broadcast ToSend
12 Schedule next broadcast randomly in the interval [MinObservationInterval, MaxObservationInterval]

Algorithm 1: Vehicle Build/Send Observations

dresses the two main desirable characteristics outlined in Section 3.3 of the previous document: volume sensitivity and selection/aggregation of data.

Using an estimate of available bandwidth, which is based on the amount of information received in a previous time interval, allows the algorithm to adjust to varying traffic volumes. If a large number of vehicles are sending many observations, the number of received bytes of vehicles in that area will increase, thus decreasing the likelihood of including data in a message. In the future, additional mechanisms could be added to this algorithm to further suppress the broadcasting of data by vehicles with observations that are not of use to nearby vehicles.

By selecting observations in increasing order of threshold, which is reduced based on the number of nearby vehicles that may be interested in an observation, the algorithm is capable of prioritizing certain observations over others. Data with relatively high threshold values may never be considered

due to the limit on message size. Furthermore, the probability an observation is added to the message is inversely related to the threshold of that observation. This has the effect of increasing the likelihood of adding observations with low thresholds compared to those with higher threshold values. It should also be noted that the set O of observations used in Algorithm 1 contains only the most recent observations for each road segment in memory. This is a very crude form of data aggregation, in which it is assumed that the most recent observation is the only one of importance to other vehicles. While most would agree that this is true, future extensions to this work could include other data within a message, such as the rate at which traffic volumes are changing for a road segment. This rate of change information is already used within the current system, but is only used for calculation of thresholds and is not explicitly broadcast to other vehicles.

2.5 Update Observations

The observation update is responsible for two things: removing expired observations and recalculating thresholds. Algorithm 2 shows the steps taken by vehicles to update the observations stored in memory. The first step

```
1 Remove all expired (old) observations
2 Reset threshold of all road segments (default to 1)
3 foreach Road segment r do
4     foreach Neighbour vehicle n do
5         if r is in n's route then
6             Set threshold of r = current value × (1-ROC)
              /* ROC explained in discussion */
7         end
8     end
9 end
```

Algorithm 2: Vehicle Update Observations

taken is to remove all observations that are beyond a certain age, as determined by the MaxObservationAge system parameter (line 1). The thresholds of all road segments are then reset to their default value (line 2) and decreased based on the number of neighbouring vehicles that have that road segment within their planned route (lines 3-9). To determine the magnitude of the decrease, rate of change (ROC) information is used, with higher rate of

change resulting in more significant threshold decreases. This is an intuitive decision, as information relating to road segments with rapidly varying state should be more desirable/useful than information relating to road segments with very little variation in traffic state. Currently, the ROC value for a road segment is calculated by dividing the percentage change between the last two measurements by the difference in time between those measurements (as done in Equation 1). The absolute value is then taken to ensure the ROC is always positive. In future updates to the algorithm, more advanced methods of rate of change analysis can be implemented which allow for a more accurate calculation of the rate of change, and thus a more accurate prediction of future values (i.e., alpha-beta filtering using a time window).

$$\begin{aligned} Change &= NewMeasurement - OldMeasurement \\ PercentChange &= \frac{Change}{OldMeasurement} \\ TimeChange &= NewTime - OldTime \\ ROC &= \frac{PercentChange}{TimeChange} \end{aligned} \tag{1}$$

2.6 Update Neighbours

Within the current protocol, vehicles simply remove expired neighbours. Each vehicle maintains a list of nearby neighbours, which is updated with the most recent time a beacon has been received from each neighbour. If a specified amount of time has passed since receiving a beacon from a neighbour, that neighbour is removed from the list. Future improvements to this mechanism may adjust the neighbour model to be capable of capturing different estimates of 'closeness' for each neighbour. This could allow neighbours who are slightly out of range to affect the data selection process in a lesser way than neighbours whose beacons have been received recently.

2.7 Receive Beacon

When a vehicle receives a beaconing message from a nearby vehicle, the sending vehicle's ID is stored in the receiving vehicles memory, along with the current time. If the sending vehicle's ID is already stored in memory, the stored time is modified to reflect that the sending vehicle is still within range. The planned route of the sending vehicle is also taken from the message and stored into memory so it can be used in the data selection process.

2.8 Receive Observations

Currently, when a vehicle receives an observation message it simply stores any new observations contained within the message in memory, discarding any duplicate messages. Future development of the protocol, however, could augment the behaviour in several ways. As a first example, relatively new observations about a specific road segment extracted from a message could result in an increase of the threshold associated with that road segment, thus making observations about that segment less likely to be included in messages. This can be a desired effect, as it can be assumed that a large percentage of nearby vehicles would also receive this information and, therefore, would delete the duplicate information if it were to be sent again. Another modification could use the message's sender ID to modify road segment thresholds. For example, if vehicle A sends a very recent observation about road segment R, vehicle B may chose not to send information about segment R, as vehicle A already has a relatively recent view of segment R's traffic state.

3 Experimental Setup

To investigate the performance of the proposed dissemination protocol, the algorithms outlined within this document were implemented within the Omnet++ communication simulator. SUMO traffic simulation scenarios were then created and coupled with the communication simulator to evaluate the behaviour of the algorithm. This section outlines the software used for the project, the traffic scenarios developed for experimentation, as well as three simple experiments that were performed to analyse the algorithm and demonstrate its capabilities.

3.1 Software

Within Section 4 of the previous document, bidirectionally coupled simulators were chosen as the best current solution for vehicle-to-vehicle communications research. This approach couples a communication simulator with a traffic simulator, allowing for more advanced functionality such as real-time modification and update of vehicle routes/positions.

From the review of available simulators presented in Section 4.3 of the previous document, the choice was made to use the ns-2 simulator coupled

with the SUMO traffic simulator, as these are both widely used and existing frameworks already existed to perform bidirectional coupling. However, it was found that the necessary frameworks were no longer under development and did not support the most recent versions of the ns-2 or SUMO simulators. After additional research, the Omnet++ communication simulator was selected due to its wide support base and ability to be coupled with the SUMO traffic simulator using the Veins vehicle-to-vehicle communication framework.

The Veins framework consists of a set of models for vehicle-to-vehicle communication that can be compiled into the Omnet++ communication simulator, as well as the tools necessary to couple the communication simulation with SUMO traffic simulations. From the base models provided by the Veins framework, users can create additional custom functionality, such as the data dissemination protocol described in this document.

While this subsection provides a brief description of the software used, Appendix A includes detailed descriptions of the software setup and required modifications that were performed to implement the proposed data dissemination protocol.

3.2 Traffic Scenario

For this project, two traffic networks were implemented within the SUMO traffic simulation environment. The first, shown in Figure 1 consists of a simple 2-way grid network with simple traffic flow (exact traffic flows described within Section 3.3). This network is the one used within the experiments presented in this document. The second network, shown in Figure 2, is a much more complex network based on real-world road infrastructure from the Pembroke, Ontario and Petawawa, Ontario areas. This more complex traffic network was also assigned theoretical vehicle volumes/routes to simulate possible daily traffic flows (i.e., morning and evening rush hours separated by a lower volume period). The more complex network will be used in planned experiments to evaluate the large-scale behaviour of the algorithm and compare its behaviour to other existing dissemination algorithms.

While the more complex traffic network offers a closer representation of real-world traffic, the simpler traffic network is used in the initial experiments presented in this document. Using this simple example allows desired behaviour of the algorithm to be logically outlined and compared to the actual results. This allows the experimenters to verify that the protocol is

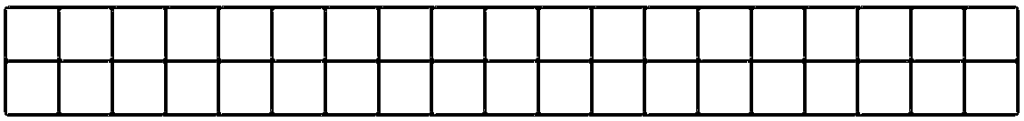

Figure 1: A simple grid network of two-way roads.

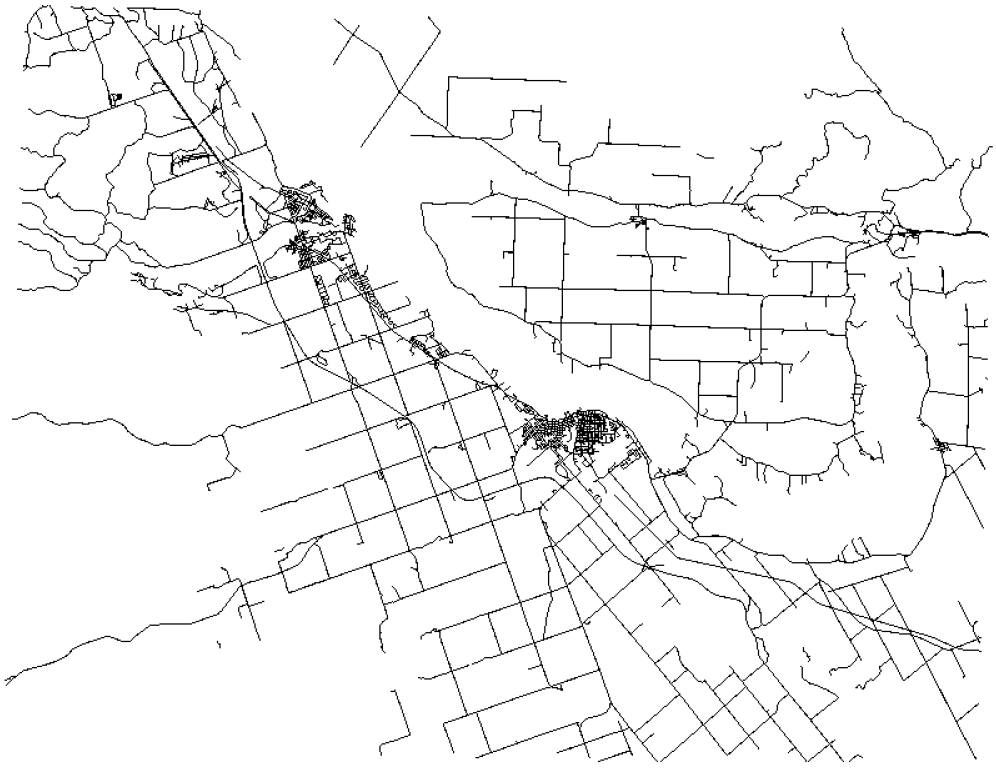

Figure 2: A more complex road network based on real-world data.

working as intended, which is important in the initial development stages of the algorithm and is made prohibitively difficult when using a larger and more complex scenario.

3.3 Experiments

Three simple experiments were completed to investigate the behaviour of the algorithm. The first two of these experiments considered the threshold values associated with road segments within the network over time as vehicles passed each other while travelling in opposite directions. The third test was designed to analyse the ability of vehicles to self-regulate the amount of data being broadcast based on the amount of data being received over a short time interval. A more in-depth description of each of these three tests is given below.

3.3.1 Two Vehicles

This experiment consisted of two vehicles entering the network shown in Figure 1 from opposite sides and travelling in opposite directions. Each vehicle would then travel for some time before crossing the path of the other vehicle, at which point beaconing messages would be received and road observations could be exchanged. As the vehicles are travelling on opposite sides of the main road, their routes do not share any common road segments. For this reason, additional information was added to the routes and initial observations of the two vehicles, as described below.

- Vehicle 1: Road segments with the following IDs were appended to the end of the vehicle's route: R1, R2, R3, R4, R5.
- Vehicle 2: The observations listed in Table 2 were added to the vehicle's memory. This simulated the vehicle initially travelling on road segments R1-R10, while also receiving additional observations from another vehicle regarding road segments R4, R5, R7 and R8 (boldface entries).

From the additions, we can see that observations relating to road segments R1-R5 would be of most use to Vehicle 1, as they are part of its planned

Road ID	Observation Time	Travel Time
R1	10	10
R2	20	10
R3	30	10
R4	**20**	**10**
R4	45	15
R5	**30**	**10**
R5	60	15
R6	70	10
R7	**50**	**10**
R7	85	15
R8	**60**	**10**
R8	100	15
R9	110	10
R10	120	10

Table 2: Initial vehicle observations for Vehicle 2

route. Also, the rate of change information associated with segments R4 and R5 should result in these two segments being prioritized over R1-R3.

3.3.2 Three Vehicles

This experiment was much the same as the one explained above; however, a third vehicle was added to the network shortly after the first two. This vehicle was assigned a route very similar to that of Vehicle 1. The only difference was that road segments R6, R7, R8, R9, R10 were appended to the end of Vehicle 3's route, instead of R1-R5. Based on this information, it would be expected that observations would be prioritized identically to the scenario above until Vehicle 2 receives a beacon from both vehicles. It would then be expected that observations for segments R4, R5, R7 and R8 would be prioritized, due to the rate of change measurements and the fact that an equal number of neighbours are interested in each piece of data.

3.3.3 Many Vehicles

This experiment involved inserting 50 vehicles into each end of the network in short succession (1 second between each pair). The percentage of vehicles

capable of broadcasting data was then varied between 10, 20, 30, 40 and 50 percent. It is expected that the total amount of data being broadcast should increase as the percentage of broadcast-capable vehicles increases. This increase should continue until the threshold is reached, at which point vehicles should begin to self-regulate the amount of data they are sending.

4 Results and Discussion

This section presents the results of the experiments outlined in Section 3. Along with discussing the results, this section also includes brief discussion of possible future algorithmic improvements.

4.1 Two Vehicles

To evaluate the algorithm's ability to prioritize the desired observations, the thresholds of certain sets of observations were plotted over time. Based on Algorithm 2, observations are considered in order of increasing threshold. Therefore, observations that have the lowest threshold are prioritized over those with higher thresholds. Also, a lower threshold results in a larger probability of being included in the message to be broadcast. Figure 3 shows the thresholds for 3 sets of observations stored in Vehicle 2's memory:

- All: The average threshold of all observations in memory.
- Changing Route: The average threshold of observations relating to road segments that have a positive rate of change and are included in Vehicle 1's planned route.
- Steady Route: The average threshold of observations that are included in Vehicle 1's planned route but have a current rate of change of 0.

Before any beacons have been exchanged between vehicles, all sets of observations maintain the default threshold value of 1. After approximately 200 seconds of simulation, the two vehicles come within range of each other and beacon messages with route information are exchanged. After this point, the threshold of both sets of observations relating to Vehicle 1's route maintain below-average threshold values. Also, the thresholds of observations with a positive rate of change drop to 0, meaning they will be considered first when building a message and maintain the highest probability of being added. This

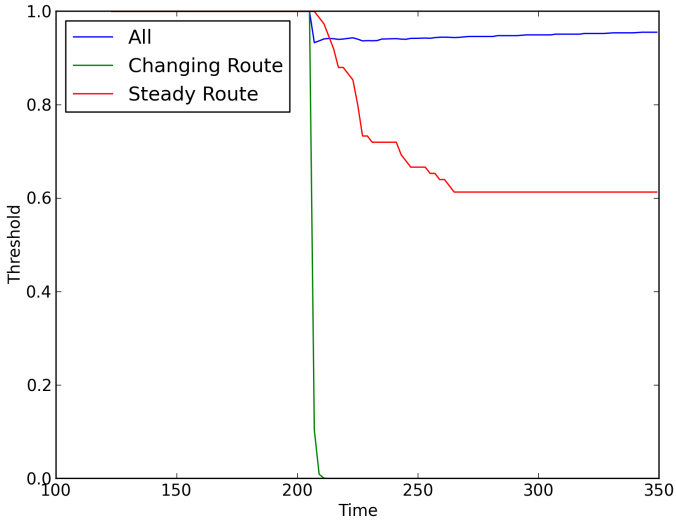

Figure 3: Thresholds over time for several sets of observations.

result shows that the algorithm is capable of selecting data that is important to a nearby neighbour, which can allow for more effective use of the available bandwidth.

It should be noted, however, that once the threshold values drop, they do not return to normal levels for the remainder of the simulation. A future improvement to the algorithm should consider how these thresholds could be modified after their associated observations have been included in a broadcast message. For example, the thresholds of included observations could be raised to a level higher than the default for a short period of time after being broadcast. This would have the effect of including a variety of messages over short time intervals, instead of sending the same 'important' message multiple times to the same subset of vehicles. This will result in a more efficient use of the available bandwidth when compared to simply broadcasting the same data repeatedly.

4.2 Three Vehicles

While the above experiment considered a scenario in which a single vehicle crossed paths with another, this experiment examines how data is prioritized

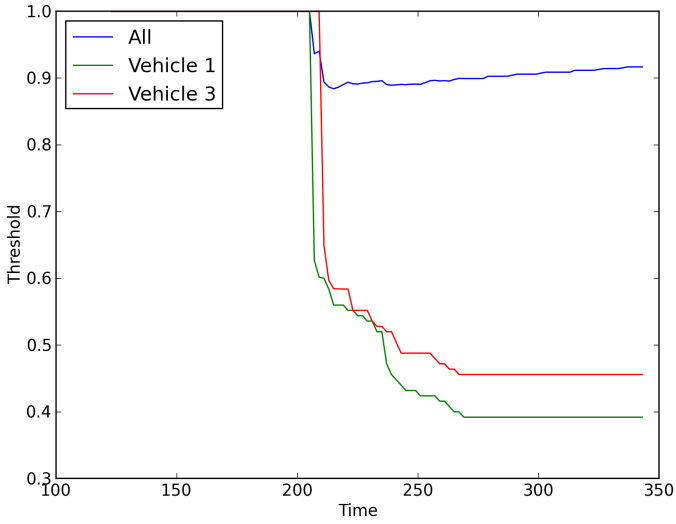

Figure 4: Thresholds over time for several sets of observations.

when two passing vehicles are present. Figure 4 plots the thresholds for 3 sets of data: the overall average, the threshold of observations for segments in the first passing vehicle's route (Vehicle 1), and the threshold of observations for segments in the second passing vehicle's route (Vehicle 3). The resulting graph is similar to that of Figure 4, but it is shown that once the beacon from Vehicle 3 has been received, observations for both passing vehicles are prioritized over other observations. This demonstrates that the algorithm is capable of addressing a situation where there are a number of neighbours. It is hoped, then, that this will scale effectively when considering a larger problem size where there may be a high number of neighbours in the vicinity of a vehicle.

4.3 Many Vehicles

To analyse the ability of vehicles to self-regulate transmission using the proposed algorithm, Figure 5 plots the average bytes sent over a number of 10 second intervals for varying percentages of active vehicles. This figure shows that as the percentage of active vehicles increases, the average number of bytes sent is increasing significantly slower. In fact, there is little difference

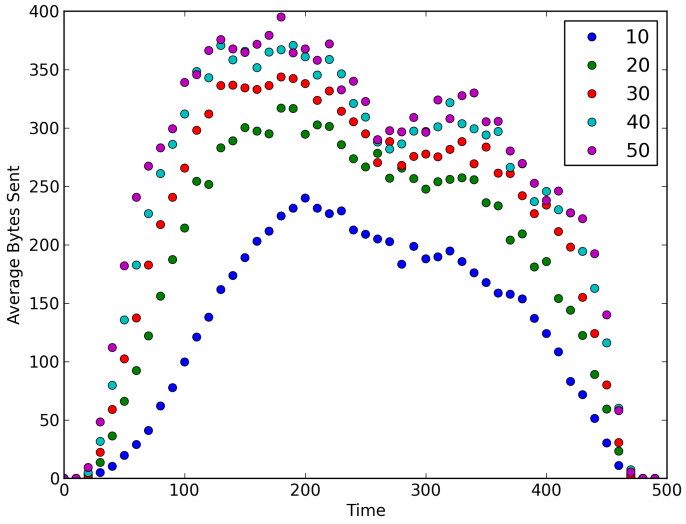

Figure 5: Average bytes sent over 10 second intervals with varying percentage of active vehicles.

between the amount of bytes sent when using 40% and 50% active vehicles. This shows that the algorithm is capable of limiting the amount of data sent by all vehicles, allowing the protocol to perform in varying conditions without overloading the network and causing a large number of failed packet deliveries.

One issue of note with this approach is that the bandwidth can be underutilized when a low number of active vehicles are present. As an example, assume that the maximum desirable average bytes sent per ten second interval is 400. At the peak period, using 40 and 50 percent active vehicles comes close to this threshold. At other times and with a lower percentage of active vehicles, however, significantly more data could be sent without coming close to the threshold. In the future, it may be worth exploring adaptive broadcasting rates to address this problem. Currently, estimated bandwidth usage is only utilized in limiting the probability of adding an observation to a broadcast message and broadcasts are scheduled in a random interval. By allowing adaptive broadcast intervals, vehicles could identify when there is a relatively high amount of available bandwidth and decrease the delay between broadcasting attempts.

5 Conclusion

Without accurate measurements of traffic state, intelligent traffic control is made extremely difficult. Vehicle-to-vehicle communication has been proposed as an effective method for generating and propagating accurate traffic state information. This document outlines a data dissemination protocol to be used by wireless-enabled vehicles to select and broadcast appropriate data to neighbouring vehicles. By supplying other vehicles with information relative to their travel plans, more intelligent control decisions can be made by those vehicles (i.e., with dynamic route modification). Furthermore, this data can also be communicated with other control infrastructure, such as intelligent traffic signal controllers, allowing for improved decision making abilities.

Using bidirectionally coupled communication and traffic simulators, this work performed initial experiments to verify the behaviour of the proposed protocol. It was found that the algorithm is capable of selecting desirable data to share with neighbouring vehicles. It was also shown that vehicles using the proposed protocol are capable of self-regulating the amount of data broadcast within the network to ensure that the wireless medium does not become flooded, resulting in performance degradation.

With the framework used here in place (i.e., vehicle-to-vehicle communication, data dissemination algorithm, traffic network implementations), it will now be possible to easily investigate further improvements to the proposed protocol, as well as other intelligent control methods. The following subsection describes the future work to be carried out in more detail.

5.1 Future Work

The initial focus of future work will be the implementation of the basic algorithm improvements mentioned in Sections 2 and 4. These basic improvements are easy to implement and should result in improved overall protocol performance.

With the positive results from the initial analysis of the protocol's performance presented here, future work will investigate the overall performance of the protocol within a realistic traffic scenario where there are many vehicles and largely variable routes. This performance evaluation will involve comparing the proposed protocol to an (or several) existing data dissemination algorithms (see Section 3 of the previous document for a detailed description

of several existing protocols). This comparison will use a number of metrics to determine the ability of each algorithm to propagate desirable information through the network, resulting in more accurate traffic state knowledge.

Finally, the framework and tools used here will be further extended to investigate possible intelligent traffic control methods. One of the main focuses of the future work in this area will be to consider real-time route adaptation by vehicles within the network based on the information they receive while travelling through the network. Another focus of intelligent traffic control research will involve the improvement of existing traffic signal control algorithms using information made available using vehicle-to-vehicle communication (i.e., through advanced notification of incoming vehicles and planned routes).

A Software Setup

The simulation environment used for this research project consisted of several pieces of freely available software, the basic architecture of which is outlined in Figure 6. This appendix briefly describes the software packages used, and describes in more detail the additions/modifications required to implement user-defined behaviour for vehicle agents within the simulation.

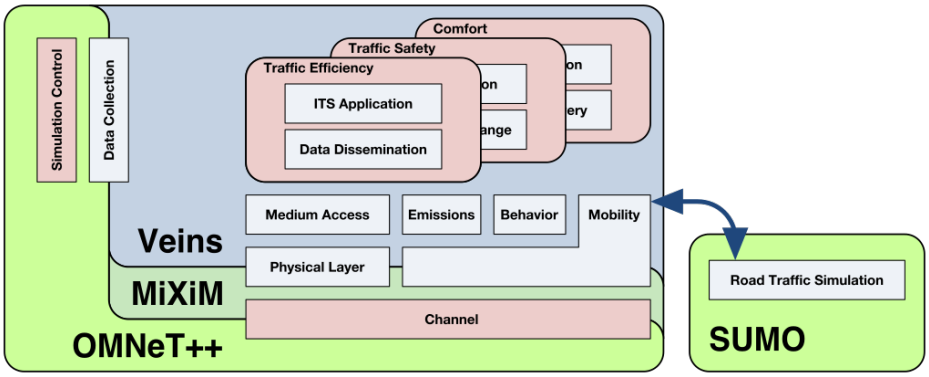

Figure 6: Architecture of a Veins-based bidirectionally coupled simulator (Veins Architecture, 2013).

Traffic simulation within this project is achieved using the SUMO traffic simulator, which is "an open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks"(Simulation of Urban Mobility - SUMO, 2013). The current release version of SUMO is 0.17; however, version 0.15 was used for this work as it was supported by the version of Veins and Omnet++ that were used. The software included within the SUMO software package allows for easy generation of several simple networks (i.e., the grid network used in this work), while also offering tools to import networks from sources such as OpenStreetMap (OpenStreetMap, 2013), which was used to create the more detailed traffic network in Figure 2. The main page of the SUMO wiki (`http://sumo-sim.org/wiki/Main_Page`) contains links to installation guides, tutorials, downloads/tools, as well as other documentation.

The base communication simulator used for the project was Omnet++, which is an "extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators" (Omnet++ Network Simulation Framework, 2013) usable on most popular operating sys-

tems (i.e., Windows, Linux, Mac OS). Omnet++ is a well-documented software package with an active mailing list-based support community. The most recent version of Omnet++ is 4.3, however version 4.2 of Omnet++ was used during this project (version 4.2 was the most recent at the time model development began) and can be downloaded from the Omnet++ website. Those who are interested in getting started with Omnet++ simulations should consult the main documentation page for Omnet++, which includes basic introduction materials, as well as a detailed user manual and API reference. While Omnet++ provides support for general purpose network simulation, the Mixim framework provides models for wireless networks, including wireless sensor networks, vehicular networks and radio interference (MiXiM, 2013).

Veins (Veins, 2013) is an open-soured framework responsible for coupling the Omnet++ communication simulator with the SUMO traffic simulator. Two of the most important features offered by Veins are the supplied models of 802.11p and DSRC/WAVE network layers, along with the mobility models which trace vehicles from a SUMO simulation within Omnet++ (SUMO $\rightarrow$ Omnet++) and allow for real-time route modifications (Omnet++ $\rightarrow$ SUMO). A tutorial for installing and building Veins is provided at `http://veins.car2x.org/tutorial/` and is a good initial step for any researchers interested in using bidirectionally coupled simulation of communication and traffic.

One of the most important classes to consider within the Veins framework is the BaseWaveApplLayer class, which is a base class modelling the behaviour of a WAVE-enabled vehicle. This class can be extended by a user to include custom defined behaviour for vehicles within a simulation. The class offers base implementations of several important functions, shown in Figure 7. Several of these methods may be overridden in a user-defined application class to offer customized functionality. An explanation of the role each of these functions is responsible for within an application, as well as the modifications that were made for this project (the DynVehApp class available at `http://www.scs.carleton.ca/~davidmckenney/RequiredFiles.zip`) is the implemented extension of the BaseWaveApplLayer class) are included below.

handleSelfMsg

Omnet++ is a message-driven simulator and the handleSelfMsg function is responsible for handling messages a vehicle sends to itself (internally). These messages are scheduled to be sent to trigger specific

```
virtual void handleSelfMsg(cMessage* msg);
virtual WaveShortMessage* prepareWSM(std::string name, \
        int dataLengthBits, t_channel channel, \
        int priority, int rcvId, int serial=0);
virtual void sendWSM(WaveShortMessage* wsm);
virtual void onBeacon(WaveShortMessage* wsm) = 0;
virtual void onData(WaveShortMessage* wsm) = 0;
virtual void handlePositionUpdate(cObject* obj);
```

Figure 7: Important functions within the BaseWaveApplLayer class of Veins (Veins, 2013).

desired behaviour within the vehicle. As an example, the Update and Send functionality discussed in Section 2.1 are self-scheduled events that are triggered by a self-message within the vehicle. The role of handleSelfMsg, then, is to parse the message type and data from the cMessage input variable and trigger the required behaviour (i.e., call a function to update the neighbour model).

prepareWSM

This function is used to prepare a message to be broadcast across the wireless medium by the vehicle. The function allows a user to specify required parameters of the message that will be used in initializing and building the message to be sent. This work uses the default values provided by Veins for the last four parameters required; however, the message name (beacon or data) and size are specified during message creation. This function returns a WaveShortMessage, which can then be sent using the sendWSM function. This function is used by the proposed protocol, but no modifications were made to alter the functionality.

sendWSM

This function is responsible for sending a data message in the form of a WaveShortMessage object. Like prepareWSM, this function was used within the project but was not modified.

onBeacon

This function is responsible for handling the reception of a beacon

message. It should be noted, however, that since this work proposes an alternate use of 'beacon' messages, which include additional data within a beacon, this function was not used to handle the beacon messages within this work (instead, beacon messages were treated as a type of data message, described below). Other implementations, though, may require this function to be modified to perform the required actions when a beacon is received.

onData
: This function is responsible for handling the reception of data messages. Within the implementation outlined here, the function is required to identify whether the data message contains observations or a beacon with route information from a neighbour. It is then responsible for updating either the observation or the neighbour model with the new information contained within the message.

handlePositionUpdate
: This function is called automatically by Veins when the position of the vehicle is updated from the SUMO traffic simulation. Within this work, the main behaviour required when the position of the vehicle is updated is to store the travel time within the set of observations if the vehicle has reached the end of the current road segment. There are a wide range of vehicle attributes available through the Traffic Control Interface (TraCI) offered by the SUMO simulator, which can be accessed using the mobility model offered by Veins. This allows a user to retrieve a vehicle's current road segment, position, speed, and other state variables.

These functions define the main behaviour of an intelligent vehicle application; however, the user is free to define any state and functionality outside of these functions as well. This allows for a fully customized vehicle behaviour to achieve the user's goals for the system. A full listing of the code used to implement the proposed protocol is available for download at `http://www.scs.carleton.ca/~davidmckenney/RequiredFiles.zip`. This codebase can be used to further guide the creation of new user-defined protocols using the Veins framework.

References

Dave McKenney and Tony White. Distributed and adaptive traffic signal control within a realistic traffic simulation. *Engineering Applications of Artificial Intelligence*, 26(1):574 – 583, 2013. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2012.04.008. URL `http://www.sciencedirect.com/science/article/pii/S0952197612000966`.

MiXiM. `http://mixim.sourceforge.net/`, August 2013.

Omnet++ Network Simulation Framework. `http://www.omnetpp.org/`, July 2013.

OpenStreetMap. `http://www.openstreetmap.org/`, July 2013.

Simulation of Urban Mobility - SUMO. `http://sumo-sim.org/wiki/Main_Page`, July 2013.

Veins. `http://veins.car2x.org/`, July 2013.

Veins Architecture. `http://veins.car2x.org/documentation/veins-arch.png`, July 2013.