5-2017

An Investigation into the Performance Evaluation of Connected Vehicle Applications: From Real-World Experiment to Parallel Simulation Paradigm

Md Salman Ahmed
East Tennessee State University

An Investigation into the Performance Evaluation of Connected Vehicle Applications: From Real-World Experiment to Parallel Simulation Paradigm

A thesis

presented to

the faculty of the Department of Computing

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Master of Science in Computer and Information Sciences

by

Md Salman Ahmed

May 2017

Keywords: CV Technology, Parallel Simulator, V2X Communication, ITS Application, DSRC, Merge Control Algorithm, Network Partitioning, Communication Protocols

ABSTRACT

An Investigation into the Performance Evaluation of Connected Vehicle Applications:

From Real-World Experiment to Parallel Simulation Paradigm

by

Md Salman Ahmed

A novel system was developed that provides drivers lane merge advisories, using vehicle trajectories obtained through Dedicated Short Range Communication (DSRC). It was successfully tested on a freeway using three vehicles, then targeted for further testing, via simulation. The failure of contemporary simulators to effectively model large, complex urban transportation networks then motivated further research into distributed and parallel traffic simulation. An architecture for a closed-loop, parallel simulator was devised, using a new algorithm that accounts for boundary nodes, traffic signals, intersections, road lengths, traffic density, and counts of lanes; it partitions a sample, Tennessee road network more efficiently than tools like METIS, which increase inter-process communications (IPC) overhead by partitioning more transportation corridors. The simulator uses logarithmic accumulation to synchronize parallel simulations, further reducing IPC. Analyses suggest this eliminates up to one-third of IPC overhead incurred by a linear accumulation model.

DEDICATION

This thesis is dedicated to my lovely wife, Nasrin Sultana, for her continuous inspiration and sacrificial care for me. I would also like to dedicate this thesis to my parents for making me who I am and supporting all the time.

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my thesis advisor Dr. Mohammad A. Hoque for his continuous support through knowledge and sincere appreciation. He always discussed research problems with great care and appreciated my opinions. I really appreciated his promptness and guidance in the moments whenever I had a question about my research.

I would also like to thank the faculties who helped me to organize this thesis by providing their valuable input and suggestions. Without their valuable comments and feedback, it would have been a hurdle for me organize this thesis.

I would also like to thank the School of Graduate Studies at East Tennessee State University for supporting this research financially by providing me the Graduate Student Research Grant.

Finally, a special thanks goes to Mr. Mohsen Kamrani of the Department of Civil and Environmental Engineering at the University of Tennessee, Knoxville for helping us to conduct our experiments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Motivation

As automobile usage has increased, so have highway congestion, incidents, fatalities, and greenhouse gas emissions. In 2012 USA TODAY reported that Americans annually waste 1.9 billion gallons of gasoline in traffic on congested roads and pay more than $100 billion in wasted fuel and lost time [1]. Road safety has become another concern worldwide. WHO reported that around 1.25 million people die in over 180 countries per year due to road fatalities and crashes [2]. These fatalities and crashes cause a huge loss of property, along with injuries, disabilities, and deaths. The U.S., for example, experienced 32,744 and 35,092 traffic fatalities in 2014 and 2015, respectively. Fatalities, moreover, increased by 10.4% over the first half of 2016 as compared to 2015 [3].

These adverse effects of automobile usage can potentially be lessened with connected vehicle (CV) technology. CV technology aims to connect all vehicles in networks of roads using infrastructure support and wireless communication. Ideally, CV will allow vehicles to exchange alert, warning, and safety-critical information with other vehicles and communities to collect real-time traffic data for transportation engineers to plan and design efficient transportation systems, while improving transportation systems' overall throughput. CV employs Dedicated Short Range Communication (DSRC), a newer wireless protocol for inter-vehicle communication (IVC), developed for the automobile industry. CV applications typically assume the presence of a DSRC-based Intelligent Transportation System (ITS) that supports the exchange of information obtained through vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-pedestrians (V2P) communications. This exchange of information, collectively known as V2X communications, is being explored for opportunities to assist drivers in avoiding congestion, reducing vehicle stops, choosing a best route, and optimizing fuel efficiency.

CV research is being promoted by U.S. Department of Transportation's Federal Highway Administration through its Open Source Application Development Portal (OSADP) [4]. The OSADP initiative divides V2X research into roughly sixteen categories, including Collision Avoidance, Crash Prevention and Safety, Information Management, Road Weather Management, Transit Management, Driver Assistance, and Freeway Management. The most popular of the applications that have been uploaded to the OSADP include systems for assisting in lane-changes [5], changing the signal phase timing for emergency and transit vehicles [6], avoiding collisions [7], disseminating signal phase and timing information [8], reducing traffic congestion [9], and preventing crashes [10]. Much of the research into Driver Assistance and Freeway Management System has focused on automating the freeway merge assistance system [11, 12, 13, 14] to mitigate fatalities incurred by freeway merge conflicts, reduce overall driving time, and improve traffic throughput and road safety measures.

Before CV applications can be deployed in real-world settings, they will need to be validated and perfected in laboratory settings in order to prevent possible damage and loss of lives. This validation can include the use of realistic simulations of traffic networks. Simulating ITS and CV systems will require the integration and synchronization of two tightly coupled domains: transportation and communication. The transportation domain models vehicular mobility, including traffic routing, lane-changing, vehicle dynamics, and traffic signal controls. The communication domain models mechanisms for data-traffic-related communications, including packet routing, end-to-end message delivery, and V2X-related cross-layer protocols. These two domains directly affect each other's operation. For example, high speed traffic networks with high vehicle density may delay V2X communications and degrade communication quality [15]. On the other hand, communication delay and data loss may degrade the modeling of vehicular operation. Such degradations, even if minute, could adversely affect the ability of V2X-based applications to assure their users' safety.

Statement of the Research Problem

Initially, this research focused on developing and evaluating a novel CV application for freeway merge assistance. The research was subsequently expanded to the creation of parallel and distributed simulators, due to the need to stress-test the application in a safe environment before deploying it in a real-world setting. Efforts to develop a complete feedback loop based transportation simulator with a wireless network simulator for modeling and evaluating V2X-based ITS applications have been ongoing for the past decade. Unfortunately, current state-of-the-art simulators are sequential in nature and require huge amounts of memory and execution time to simulate large-scale urban transportation networks. These limitations have created a need for a parallel and distributed simulation environment to simulate safety-critical ITS applications.

Accordingly, the work described in this document focused on four related challenges:

1. *Modeling and developing a data-driven V2X-based ITS application using drivers' data and vehicular mobility traces.* This application, a freeway merge assistance system, uses IVC to exchange mobility traces between vehicles in the form of basic safety messages (BSMs). These traces, which include vehicles' positions, directions, speed, acceleration, and deceleration, provide advisory information to drivers while they are merging into a freeway from an entrance ramp (see chapter 2). This research was in collaboration with researchers from The University of Tennessee, Knoxville.

2. *Analyzing sequential simulators to identify the critical issues and proposing a parallel simulation framework.* A comprehensive literature review of state-of-the-art sequential simulators identified the limitations of contemporary simulators (see chapter 3) and requirements for a parallel simulation platform (see chapter 4). The proposed parallel simulation framework simulates vehicular mobility using the SUMO traffic simulator [16] and IVC using OMNET++ [17].

3. *Partitioning a large transportation network into smaller components in order to reduce the cost of modeling events.* An effective algorithm for network partitioning— one that yields a high ratio of computation to communication for a requested number of simulated subnets— is needed to minimize the time and memory resources needed to simulate a traffic network. These resources are exponential in the number of vehicles in the network's roads. For example, in one experiment involving a sequential simulator [18], the simulation of a 200 node (vehicle) network created 4,600,000 events and required 16 minutes to process the events. The goal of this work was to improve a simulation's speed by reducing the cost of modeling events (see chapter 5). Since finding an optimal partitioning of a network is an NP-hard problem, practical heuristics are required to ensure an even distribution of a workload while minimizing communication between the components.

4. *Reducing inter-partition communication overhead.* A parallel simulator runs each partition's simulation on a different processor and accumulates each processor's partial simulation results to produce the final simulation result. As a result, the inter-partition communication overhead can greatly impact the total execution time of the parallel simulator. This part of the research involved analyzing the inter-process communication overhead using two accumulation patterns (see chapter 6).

Results

The research described in this thesis produced four novel results.

1. The first result was a decentralized freeway merge assistance system that provides advisory information to drivers solely depending on DSRCs to reduce freeway merge conflicts and increase overall traffic throughput. The system's performance and accuracy were evaluated on 8 exits along Interstate I-26 in northeast Tennessee (see chapter 2).

2. The second was a partitioning algorithm for real-world transportation networks based on real world traffic parameters, including system boundary nodes, traffic signals, intersections, road lengths, traffic density, and counts of lanes. The partitioning mechanism proved better than the traditional tools like METIS for reducing the division of transportation corridors.

3. The third was a logarithmic accumulation algorithm that accumulates partial simulation results from processors and produces the final simulation result. Since the actual simulation of transportation networks was beyond the scope of this research, a matrix-matrix multiplication algorithm was used to evaluate the performance and communication overhead of the logarithmic communication overhead. Knightrider [19], one of two high-performance computing clusters at East Tennessee State University's High Performance Computing Center, was used to run the distributed matrix-matrix multiplication algorithm using MPI communications.

4. The fourth was a design for an envisioned Integrated Distributed Connected Vehicle Simulation platform incorporating hardware-in-the-loop simulation together with the closed-loop coupling of SUMO and OMNET++. The simulation platform addresses two

major limitations of current sequential simulators, reducing execution times and resource usage, using the partitioning and accumulation algorithms described above.

These results, together with other considerations like avenues for future research, are described in more detail in this thesis's remaining chapters.

CHAPTER 2

FREEWAY MERGE ASSISTANCE SYSTEM

Freeway Merge Assistance System using DSRC

Md Salman Ahmed, Mohammad Hoque, Brian Bennett
Department of Computing
East Tennessee State University
Johnson City, TN 37614
{ahmedm, hoquem, bennetbt}@etsu.edu

Asad Khattak
Department of Civil and Environmental Engineering
University of Tennessee
Knoxville, TN 37996
akhattak@utk.edu

***Abstract*—With the advent of the connected vehicle (CV) technology, researchers have started to re-engineer the design of automated highway systems from different aspects, such as queuing analysis, ramp metering, and merge control algorithms. Previous development of freeway merging algorithms were mainly relied on the infrastructure support requiring ramp metering with the aid of the inductive loop detectors or transponders mounted at several reference points on the highway. Recently, with the aid of CV technology, some researchers have theoretically modeled the freeway merging algorithms using longitudinal control mathematical models. However, those models were simply evaluated using simulators without any actual implementation of the system. Our current work presents a complete implementation of a novel decentralized algorithm for a freeway merging assistance system using the Dedicated Short Range Communication (DSRC) technology. The freeway merge assistance system has been tested on 8 ramps on a real-world freeway. As of now, this is the first attempt to develop and implement a fully decentralized freeway merging algorithm that does not require any infrastructure support.**

***Keywords*—Connected Vehicle, DSRC, Merge Assistance System, Freeway, Ramps, Advisory, Decentralized System, Merge Control Algorithm.**

I. Introduction

Road safety and congestion have become growing concerns around the globe over the past few years. WHO reported that around 1.25 million people die over the 180 countries per year due to road fatalities and crashes [1]. These fatalities and crashes cause not only a huge loss of property, but also injuries and disabilities, or even deaths. The rate of fatalities is also alarming for developed countries like the United States and the United Kingdom. For example, the total number of fatalities in 2014 and 2015 in the U.S. were 32,744 and 35,092 respectively, and the fatality percentage in the first half of 2016 was also 10.4% greater than the first half percentage of 2015 [2]. Also, Americans waste more than 1.9 billion dollars due to the congestion problem in roads [3]. Among the road networks, the entrance from ramps to a freeway for merging is one of the major reasons for accidents, low traffic throughput, and delays. A study shows that merge conflicts from ramps to freeways incur 20-30% of truck accidents [4].

Researchers are working vigorously to automate the freeway merge assistance system from the early '90s to mitigate fatalities incurred by the freeway merge conflicts, reduce overall driving time, and improve traffic throughput and road safety measures. However, the automation of the freeway merge assistance system is neither an easy nor a straightforward task. An effective freeway merge assistance system must implement three related applications, such as ramp metering, lane-change advisory mechanisms, and merge control mechanisms. Two main reasons for merge conflicts are the lack of an appropriate gap (between a lead and a lag vehicle) on the freeway, and the freeway drivers' unawareness about vehicles in ramps. Many researchers tried to study the optimal gap requirement for the merging in early '80s [5], [6]; however, they relied mostly on simulation tools.

Recently, researchers and automotive companies have begun working on vehicles that include On-Board Units (OBUs) to increase road safety and provide assistive services to drivers. A vehicle communicates and passes information to other vehicles on the road using the OBU's Dedicate Short Range Communication (DSRC) mechanism. Developers around the world are developing various applications (lane-change assistants [7], signal phase timing for emergency and transit vehicles [8], collision avoidance systems [9], signal phase and timing information [10], traffic congestion [11], crash prevention [12], etc.) for CV environments and uploading the applications into the Open Source Application Development Portal (OSADP) [13] of the Federal Highway Administration of the U.S. Department of Transportation. In this paper, we utilize the concepts of the current state-of-the-art algorithms [14]–[17] and the DSRC technology to implement a freeway merge assistance system that uses a three-way handshaking communication protocol to provide advisory information to drivers. The freeway merge assistance system will be released in the OSAD Portal after a comprehensive testing.

The rest of the paper is organized as follows. Section II summarizes the stat-of-the-art algorithms for the freeway merge management; section III discusses the challenges that one must solve while implementing a freeway merge assistance system; section IV-B describes the preliminary data collection procedure; section IV presents our technical approaches for implementing the freeway merge assistance system; section VI discusses the results from the field experiments; finally, we conclude in section VII stating the limitations, alternatives, and future research plans.

II. Related Work

From the early 80s, researchers have made significant progress in Automated Highway Systems (AHS) areas. However, the research has been transformed into a new dimension when vehicle and infrastructure were enabled to run Intelligent Transportation System (ITS) applications. Similarly, the automation of the freeway merging research has followed a new direction with the introduction of connected-vehicle environment. Before the introduction of Dedicated Short Range Communication (DSRC) technology for ITS applications, researchers used infrastructure supports or Internet-based vehicles to develop their merge control algorithms ([17], [18]). For example, Lu et al. discussed an adaptive closed-loop merging algorithm that determines the speed requirement for the merging vehicles using several reference points (indicated by external hardware like coded magnets or transponders) and the speed of the main lane vehicles [18]. The main limitation of such design is that the hardware and infrastructure modifications take a lot of resources and time. Wang et al. discussed a cooperative merging control algorithm using inter-vehicle communication where each vehicle acts as an intelligent agent. The merging control algorithm uses the existing longitudinal control mathematical models. The authors used the Internet-based cars to perform V2V communication; however, no details were given about the communication protocol. Rather they focused more on the mathematical models.

With the advent of the DSRC technology, researchers have begun to rethink the design of the AHSes in different areas such as the queuing analysis, ramp metering, and merge control algorithms. For example, authors in [19]–[21] discuss the queuing analysis for the ramp and freeway using entrance capacity and entry priority while taking the DSRC technology into account. To achieve good freeway traffic throughput, a queue control algorithm must take the capacity of an entrance ramp into account. Hall et al. analyzed the important and critical components of exit/entrance ramps (e.g., capacity of an entrance ramp, distance between two entrance ramps, distance between vehicles on the ramps, average time vehicles spend on an entrance ramp, number of lanes of a ramp, etc.) of AHSes [21]. Gap-responsiveness, variable speed, and coordinated ramp metering techniques were also discussed in [22], [23]. Lu et.al described a longitudinal control ramp metering algorithm using different reference points with the help of infrastructure support [18].

Now, researchers have been working on freeway merge control algorithms considering different points of view, such as whether the system is centralized or decentralized, whether the system is designed for autonomous vehicles, whether the system provides good advisory information, and whether the system considers virtual platooning in their mathematical models. For example, Wang et al. discussed a proactive and decentralized merging control algorithm that makes the advisory decisions at some point before the actual merging point [14]. Based on the advisory decisions, vehicles on ramps and freeways can adjust their speed. However, they assumed that their algorithm knows the decision points and merging points beforehand. Some researchers focused on providing advisory information for changing lanes in freeways using the vehicular dynamics. For example, Park et al. improved their previous fixed length safe gap lane changing advisory algorithm to variable length safe gap with respect to the speeds and vehicle dynamics to improve the freeway traffic and reduce merge conflicts [15]. They collected the vehicle data for a length of 2500ft near the merging point, starting a length of 1500ft before the merging point and 1000ft after the merging point, and provided the lane changing advisory messages. For calculating the safety gap, they considered the vehicle type, vehicle length, acceleration, deceleration, and constant speed. Another important factor of designing the freeway merge assistance system is the analysis of driver response to the advisory messages. Hayat et al. described the driver reactions to the advisory messages in different roadway scenarios and traffic conditions and presented a survey about the factors that the drivers consider while responding the advisory messages [16].

Almost all the merge control algorithms in [14], [22], [24]–[29] were based on the position, speed, acceleration, and time to reach the merging point (the time to reach the merging point is also known as time to crash or ttc). However, Davis presented a merge control algorithm using an adaptive cruise control technique to improve the traffic throughput in [30]. Some researchers also discussed the impact of cooperative driving to the merge control algorithms in [31] and [32].

Most of the researchers evaluated the performance of their algorithms using simulation tools due to the unavailability of DSRC enabled OBU in the market. Very few researchers attempted to use roads in a very controlled environment (typically inside a test bed facility). No one has attempted to use the actual freeway to evaluate merge assistance systems or merge control algorithms. Thus, the systems or the algorithms lack proper evaluation and overlook some unanticipated challenges.

III. Problem Description and Challenges

In this section, we summarize the research challenges to implement the freeway merge assistance system. We also discuss some unanticipated challenges while collecting the preliminary data (data collection procedure is discussed in section IV-B) in actual freeways.

*1) **Gap length**:* Prediction or generation of the safe gaps on the rightmost lane in a freeway is a crucial factor for providing good advisory suggestions by a freeway merge assistance application. Depending on the traffic, a merge assistance system can detect or generate the safe gaps in several ways. For example, if a vehicle is traveling on the rightmost lane, the assistance system can suggest the vehicle to change the lane (if possible) to create a gap. However, the application should advise such well in advance. On the other hand, during heavy traffic when there is no room for lane changes, the application should advise multiple vehicles traveling in the rightmost lane to form platoon and drive cooperatively (the group vehicle can either slow down or speed up) to create a safe gap for cooperative merging. This estimation of the gap must be very precise and accurate.

*2) **Time to merge**:* It is necessary to calculate the time to reach the merging point by measuring the distance between the merging point and a vehicle. However, the linear distance approximation methods may not work for some entrance ramps, such as the cloverleaf interchanges (Fig. 1), due to the complex geometrical shape.

*3) **Advisory start time**:* The freeway merge assistance system should disseminate the advisory information to all participating vehicles so that the vehicles in the entrance ramp and freeway can merge smoothly. Typically, a vehicle in the ramp needs around 20-35 seconds to merge completely onto the freeway. Hence, there should be a finely tuned time line for displaying the advisory information to the drivers. Displaying

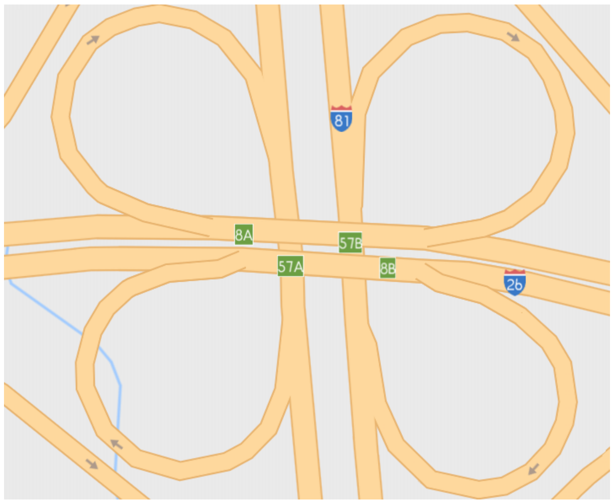


Fig. 1: Circular entrance ramps in cloverleaf interchanges

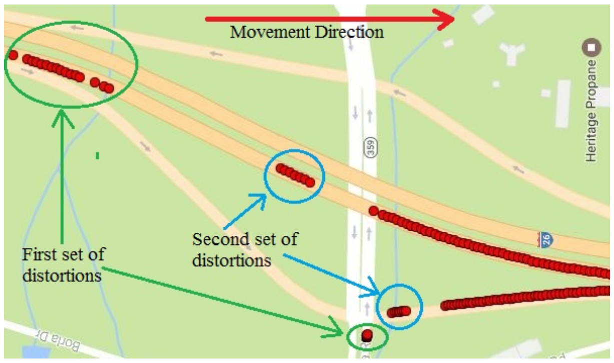


Fig. 2: Distortions in the DSRC signals

advisories too early may lead to confusion regarding merging decision. On the other hand, showing the advisory information too late will leave insufficient response time for the drivers.

4) ***Driver response time and behavior****:* The effectiveness of any highway advisory application depends on the drivers' response time and willingness to cooperate with the suggested advisory messages. Since there has been no DSRC application implemented and tested for freeway merging, driver behavior is still an open research issue.

5) ***Vehicle lane detection****:* Only vehicles on the rightmost lane in a freeway will conflict with those merging from entrance ramps. Hence, the merging assistance system needs a mechanism to identify vehicles on the rightmost lane, and to discard non-conflicting vehicles on other lanes to reduce the computational complexity the system. Vehicles in the opposite direction, vehicles that cross the merging point, and the vehicles in opposite entrance or exit ramps are the non-conflicting vehicles.

6) ***Fog computing****:* DSRC enabled On-Board Units spend most of the computational resources for disseminating various kinds of safety packets. Additional computations might be burdensome for the OBUs. Hence, it might be efficient to offload some of the complex computations to a connected smart device using the concept of fog computing. However, the communication latency between the OBU and a smart device should be studied to find out how much computation can be offloaded without hampering the real-time execution of the system.

7) ***Distorted signals****:* DSRC signals can be distorted or lost because of nearby buildings, bridges, steep highways, differences of altitude, etc. We experienced distortions when collecting preliminary data in Fig. 2. It might be necessary to continue extrapolating the trajectory until further signal is received or discarding the distortions.

IV. System Development

In this section, we describe the step by step technical approaches used to develop the algorithms for implementing the assistance system. The system uses the DSRC enabled OBUs to communicate between vehicles and an Android device to display the advisory alerts and information to the drivers. The communication between an OBU and an Android device is established using the Bluetooth connectivity.

A. Assumptions

For the initial version of the freeway merge assistance system to work properly, we assume the following criteria.

1) The system assumes that all the vehicles are running in the connected-vehicle environment.
2) Since the communication delay is in the third order of a second, the DSRC communication delay is negligible.
3) Similarly, the system's computation cost is negligible.
4) The assistance system knows when a vehicle enters on the entrance ramp.
5) The system orders the vehicles based on the time to reach the merging point. The system also determines the safe gaps in the freeway based on the speeds and the differences of the freeway vehicles' time to reach the merging point.
6) The entrance ramps are not circular and not significantly bended.
7) Only the ramp vehicles observe the mobility traces of other vehicles including themselves and make the advisory decisions. The freeway vehicles only transmit the BSM and replies to the control messages using synchronization messages (more details about the control and synchronization messages appear in section IV-C).
8) The freeway merge assistant system provides the advisory messages; however, the compliance of the advisory messages is sole depended on the drivers.

B. Preliminary Data Collection

To analyze the many parameters of the merge assistance system, we collected preliminary data. We designed and developed the assistance system based on the analysis of

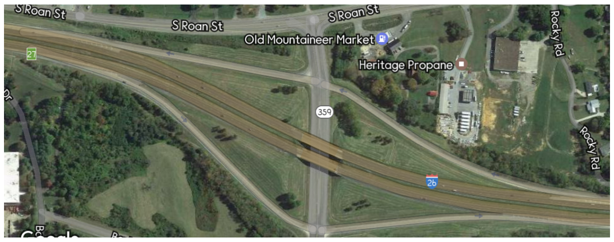


Fig. 3: Primary location of the preliminary field-test (Interstate 26 exit 27)

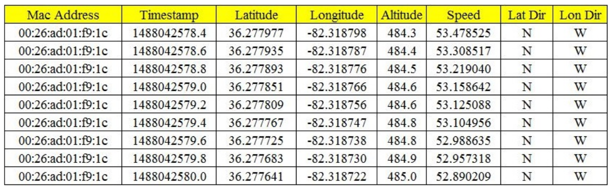
Fig. 4: Secondary location of the preliminary field-test (entrance ramp to US 321)

the preliminary data. Detailed steps of the preliminary data collection procedure include the following:

*1) **Location of the field-test experiments**:* To collect the preliminary data, we conducted our pilot experiment on the interstate I-26 (Fig. 3) and US Highway-321 (Fig. 4). We used two vehicles equipped with DSRC aftermarket On-Board Units (Arada Locomate Classic OBU [33]) to collect the preliminary data. One driver drove the first car in the ramp and another driver drove the second car in the freeway. Since timing was a crucial factor to collecting appropriate data, we synchronized our timings by phone. We took a total of six samples in three exits (Exit 27 on I-26 West bound, Exit 27 on I-26 East bound, and an Exit on US-321 North bound). We discarded three samples because of the poor timings of the two drivers.

*2) **Data storage and format**:* Our live trajectory data collected during the preliminary field tests were stored real-time on a USB drive as space separated values in text files. The USB drive was attached to the OBU. The data elements comprised of the transmitting device ID, GPS positions (latitude, longitude, altitude), GPS time, speed, and direction of vehicle heading. The in-built GPS unit attached with the DSRC device calculated the speed in m/s, which was converted to mph by our communication protocol before transmitting.

*3) **Preliminary data analysis**:* The table in Fig. 5 shows a sample of our collected data and the data format. We plotted the positions of the two vehicles (one in the ramp and another in the freeway) on Google Maps (Fig. 6) with the interval of every fifth of a second. We can determine from the preliminary data that the speed of the vehicles in the freeway is almost constant; however, the speed fluctuates when a driver sees any vehicles on the entrance ramp. The fluctuation in the speed potentially leads the merge conflicts. Fig. 7 and Fig. 8 show the speed fluctuation. We can also determine the average acceleration time, average distance covered by a vehicle on



Mac Address	Timestamp	Latitude	Longitude	Altitude	Speed	Lat Dir	Lon Dir
00:26:ad:01:f9:1c	1488042578.4	36.277977	-82.318798	484.3	53.478525	N	W
00:26:ad:01:f9:1c	1488042578.6	36.277935	-82.318787	484.4	53.308517	N	W
00:26:ad:01:f9:1c	1488042578.8	36.277893	-82.318776	484.5	53.219040	N	W
00:26:ad:01:f9:1c	1488042579.0	36.277851	-82.318766	484.6	53.158642	N	W
00:26:ad:01:f9:1c	1488042579.2	36.277809	-82.318756	484.6	53.125088	N	W
00:26:ad:01:f9:1c	1488042579.4	36.277767	-82.318747	484.8	53.104956	N	W
00:26:ad:01:f9:1c	1488042579.6	36.277725	-82.318738	484.8	52.988635	N	W
00:26:ad:01:f9:1c	1488042579.8	36.277683	-82.318730	484.9	52.957318	N	W
00:26:ad:01:f9:1c	1488042580.0	36.277641	-82.318722	485.0	52.890209	N	W

Fig. 5: Sample data

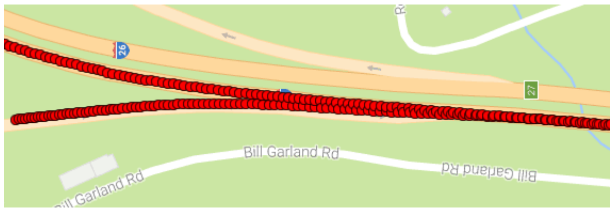


Fig. 6: Positions of the two vehicles on exit 27 of I-26 west bound

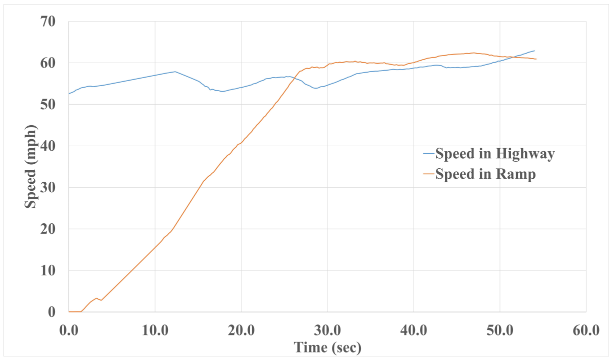


Fig. 7: Speeds of the freeway vehicle and the ramp vehicle on exit 27 of I-26 west bound

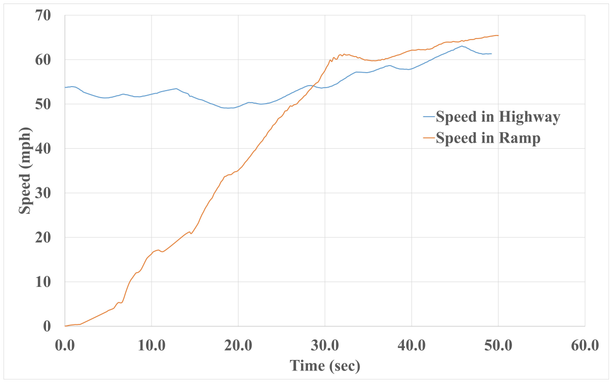


Fig. 8: Speeds of the freeway vehicle and the ramp vehicle on exit 27 of I-26 east bound

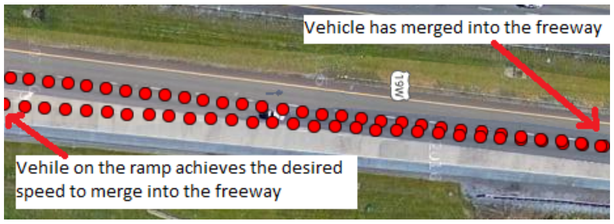

Vehicle has merged into the freeway
Vehile on the ramp achieves the desired speed to merge into the freeway

Fig. 9: Merging of the ramp vehicle into the freeway

the entrance while accelerating, average merging time, and average merging distance. The average acceleration time and the merging time for the vehicle merging into the freeway from the ramp on exit 27 of I-26 west bound were 15 seconds and 3.6 (average 4) seconds respectively. To calculate the average merging time, we sampled the timestamps when the vehicle on ramp achieves the desired speed to merge into the freeway (1483727093.8) and completely merges into the freeway (1483727097.4), shown in Fig. 9. The ramp vehicle on the exit 27 of I-26 west bound covered 285 meters for achieving 60 mph speed and 96 meters for merging into the freeways. This data indicates that any freeway merge assistance system must start its operations 15-20 seconds before reaching the merging point (or 300-400 meters from the merging point).

C. Communication Protocol

The merge assistance system uses a single hop communication protocol. The step-by-step details of this protocol is described in one of our prior works [34]. The system also uses a 3-way handshaking protocol (Fig. 10) for synchronizing the timings of the connected vehicles. In the 3-way handshaking protocol, a vehicle can make a synchronization request to other vehicles by transmitting a control message (the format of the control message is given in Table. I). The other vehicles can reply to the control message using a synchronization message (the format of the control message is given in Table. II). The recipient of the synchronization messages can also acknowledge the synchronization message by transmitting an acknowledge message. We describe more on how and when to use the 3-way handshaking protocol in the making advisory decisions step described in section V-C.

TABLE I: Format of a control message

MAC	TTC	TDM

MAC	=	Address of the OBU
TTC	=	Time to reach the crash/merging point
TDM	=	Timestamp of making advisory decisions

TABLE II: Format of a synchronization message

IER	RT	IDM	TTC

IER	=	Is the vehicle on the entrance ramp?
RT	=	Amount of time spent by a vehicle on the ramp
IDM	=	Is the advisory decision made?
TTC	=	Time to reach the crash/merging point

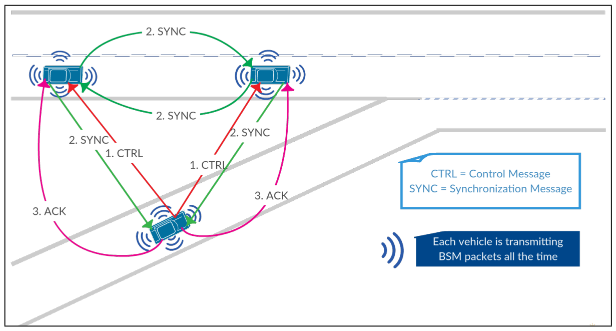

2. SYNC
2. SYNC
2. SYNC
2. SYNC
1. CTRL
1. CTRL
3. ACK
3. ACK
CTRL = Control Message
SYNC = Synchronization Message
Each vehicle is transmitting BSM packets all the time

Fig. 10: 3-way handshaking protocol

We also implemented Bluetooth communication between OBU and Android device. For the preliminary data, each OBU transmitted its identifier, position, speed, and direction in every fifth of a second through a transmitter program. The OBU also received and logged the mobility traces of the vehicles within its range. Since the GPS timestamp was updated every fifth of a second, the transmitter program transmitted data five times in a second.

V. Work-Flow of the System

The freeway merge assistance system goes through several steps before providing advisory messages to drivers. Algorithm 1 describes the pseudo code of the core algorithm. Addition details are described in the following subsections.

A. Transmission and reception of BSM packets

The freeway merge assistance system transmits and receives the BSM packets with the customized payload fields described in the Table III. The system only receives the BSM packets from vehicles within its range. The system then sends data to the connected smart phone to plot each vehicle on a map (Fig. 11 & line 14 in the algorithm). If no vehicle is within the DSRC range of the current vehicle (current vehicle means the vehicle where the assistance system is running), then the merging assistance system plots only the current vehicle on the map and provides no advisory messages. The merging system marks the current vehicle with the blue color and all the connected/neighbor vehicles with the red color. The system also has a ramp tracker system that repeatedly tracks if vehicles are entering in the entrance ramp.

TABLE III: Payload fields of a BSM packet

MAC	TS	Lat	Lon	Alt	S	Lat Dir	Lon Dir

MAC	=	Address of the OBU
TS	=	Timestamp
Lat	=	Latitude
Lon	=	Longitude
Alt	=	Altitude
S	=	Speed
Lat Dir	=	Latitude Direction
Lon Dir	=	Longitude Direction

Algorithm 1: CalculationTTC

```
Data: $ramps \longleftarrow$ list of ramps with start positions
Result: $ttc \longleftarrow$ time to reach the merging point for each vehicle
1  begin
2      while true do
3          $packet$ = receiveDSRCPacket()
4          if packet == CTRL then
5              $m \longleftarrow$ unwrap(packet)
6              if m.getMAC() == MY_MAC_ADDRESS then
7                  transmitSYNCMessage($ttc$)
8          else if packet == SYNC then
9              $m \longleftarrow$ unwrap($packet$)
10             $ttc$.add($m$.getTTC())
11         else
12             $myData \longleftarrow$ getMyData() /* data of the vehicle that runs the algo */
13             $vehicleData \longleftarrow$ unwrap($packet$)
14             $NeighborTracker$.track($vehicleData$)/* track neighbors on the map */
15             $isEnteredRamp$ = $RampTracker$.track($ramps$, $myData$.getPosition())
16             if !isEnteredRamp then /* Only ramp vehicle observes the dynamics */
17                 continue
18             if isDecisionMade then /* One time decision only */
19                 continue
20             $constAccel \longleftarrow$ CalculateRampVehicleConstAcceleration()
21             if $constAccel <$ APP_THRESOLD_ACCEL then
22                 continue
23             $Map$.insert($vehicleData$.macaddress)
24             $vehicleNo \longleftarrow Map$.find($vehicleData$.macaddress)
25             $DataQueue[vehicleNo]$.push($vehicleData$)
26             $observedTime \longleftarrow$ getObservationTime()
27             if $observedTime$ <APP_OBS_TIME then
28                 continue
29             for each freeway vehicle do
30                 $sampledData \longleftarrow$ sampleVehicleData($DataQueue$, $numOfSamples$)
31                 for each sample in $sampledData$ do
32                     $mergPoint$ = calcuateMergePoint($sample$)
33                     $mergePoints$.add($mergePoint$)
34                     $speeds$.add($sample$.speed)
35                 $avgSpeed \longleftarrow$ calculateAvgSpeed($speeds$)
36                 $avgSpeeds$.add($avgSpeed$)
37             $finalMergePoint \longleftarrow$ calculateAvgMergePoint($mergePoints$)
38             for each freeway vehicle do
39                 $d \longleftarrow$ calculateDistance($sampleData$.getLastSample().getPosition(), $finalMergePoint$)
40                 $t \longleftarrow d/avgSpeeds$.getSpeed()
41                 $ttc$.add($t$)
42             $rampVehicleAvgAccel \longleftarrow$ calculateAvgAcceleration($DataQueue$)
43             $d \longleftarrow$ calculateDistance($DataQueue$.getLast().getPosition(), finalMergePoint)
44             $t \longleftarrow$ solveQuadraticEqn($rampVehicleAvgAccel$, $DataQueue$.getLast().getSpeed(), $d$)
45             ttc.add(t)
46             for each freeway vehicle do
47                 $m \longleftarrow$ generateCTRLMessage($ttc$)
48                 transmitCTRLMessage($m$)
49             $isDecisionMade \longleftarrow true$
```

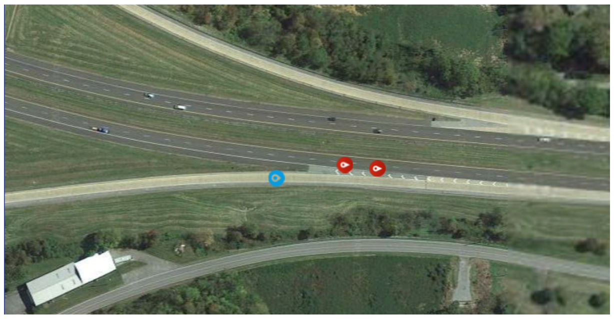


Fig. 11: Three connected vehicles on the map indicated by the markers

TABLE IV: Advisory and alert messages of the system

Advisory and Alert Messages	Description
Entered the ramp	This alert message is sent to all the connected vehicles in both in the ramp and freeway when a vehicle entered in an entrance ramp.
Keep the speed	This advisory message is sent to that vehicle which takes the lowest time to reach and cross the merging point.
Merge behind	This advisory message is sent to a vehicle which should merge behind another vehicle.
Slow down	This advisory message is sent to a vehicle that should slow down to make a longer gap and let another vehicle to merge in front of it.

B. *Observation of vehicular trajectories*

If the merging assistance system detects any vehicle entering in an entrance ramp, the system notifies the presence of the ramp vehicle to all the connected vehicles using the ***"Entered the ramp"*** alert message as described and depicted in Table IV and Fig. 12 respectively. This alert message alerts drivers for an oncoming merging and helps them to know the presence of vehicles on the entrance ramp without bending their necks (or in the cases when their vision is blocked by bushes or altitude differences). Once the ramp vehicle achieve a constant acceleration, the merge assistance system of the ramp vehicle triggers the core algorithm and starts keeping tracks the mobility traces of the connected vehicles including its own trajectory for t_1 seconds. From the tracked traces, the system calculates t_1 crash/merging points. The reason for calculating t_1 merging points is to reduce the error incurred by the approximation of the merging point. The system then calculates the final merging point by averaging the t_1 merging points. The distance of a merging point greater than $400m$ from the current position of the ramp vehicle is discarded while calculating the average. The detailed techniques of the approximation of a merging point is discussed in section V-E.

C. *Generation of advisory messages*

Once the merge assistance system finds the final merging point, the ramp vehicle determines the required time to reach the merging point (which is also known as the *time to crash*) for each vehicle. Then the system requests for synchronization from other vehicles by transmitting a CTRL message to each vehicle. The CTRL message contains the timing information of vehicle for whom the CTRL message transmitted. Each vehicle replies to the CTRL message by transmitting a SYNC message to every vehicle. The ramp vehicle then acknowledges the SYNC messages by sending an ACK message to each vehicle. The ACK message contains the timing information of the ramp vehicle. Once all vehicles receive the ACK messages from the ramp vehicle, that means all vehicles are synchronized and each vehicle has the timing information of all other vehicles, including itself. Then the assistance system of each vehicle generates the appropriate advisory message for itself using the timing information. For example, if a vehicle requires the least time to reach the merging point than other vehicles, the assistance system generates an advisory message called "***Keep the speed***". If the time of a freeway vehicle to reach the merging point is longer than a ramp vehicle, then the assistance system generates an advisory message called "***Slow down***". If the time of a ramp vehicle to reach the merging point is longer than a freeway vehicle, the assistance system generates an advisory message called "***Merge behind***". However, the freeway merge assistance system does not consider a safe gap before providing the "***Merge behind***" advisory message because the determination of safe gaps is out of the scope of this study. But the freeway assistance system makes a reasonable assumption based on the speed and time differences of freeway vehicles. For example, if the time difference between two freeway vehicles to reach the merging point is 3 seconds and their average speed is 60mph, then the gap length would be around 80 meters. However, no minimum gap requirement was enforced in this study.

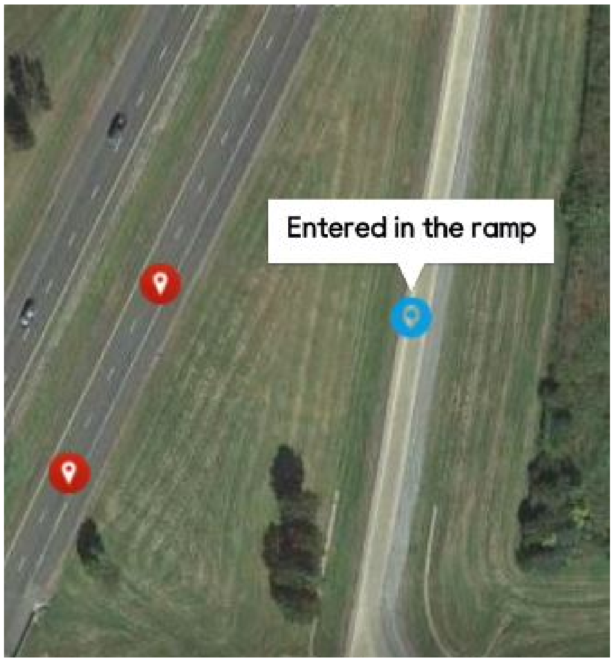


Fig. 12: A vehicle entered on the ramp

D. Visualization of advisory message

Once the merge assistance system generates the advisory message, the system sends the information to the map application on an Android device using Bluetooth connectivity. The application displays the advisory message over the marker as a text message. The application also highlights the reference marker. For example, a freeway vehicle may need to slow down for a ramp vehicle; in this case, the application displays the "***Slow down***" message to the freeway vehicle and highlights the ramp vehicle. Several scenarios on the advisory visualization are depicted in the Fig. 13. For example, Scenario 2 showed that a ramp vehicle is being advised to merge behind the freeway lead vehicle using the "***Merge behind***" advisory message over the ramp vehicle marker and the marker of the freeway lead vehicle was highlighted.

E. Calculating the Time to Crash

To calculate the *time to crash*,

1) First, we calculate the bearing of the trajectories for both freeway and ramp.
2) Then, we find the intersection of the two extrapolated great circles using the most recent lat-lon coordinates and the associated bearings. This gives us the approximate merging point.
3) From the merging point, we calculate the distances from both the approaching vehicles along the ramp and on the freeway.
4) Once the distance is known, we find the time to crash for both freeway and ramp vehicles using the kinematic equations.

Below we describe the mathematical equations involved in each of the aforementioned steps.

1) ***Finding the Bearing:*** The bearings for both the ramp and freeway vehicles were calculated using two subsequent recent GPS coordinates from respective trajectories. The following equation was used for this purpose:

$$\theta = \operatorname{atan2}(\sin\Delta\lambda\cos\phi_2, (\cos\phi_1\sin\phi_2 - \sin\phi_1\cos\phi_2\cos\Delta\lambda)) \tag{1}$$

where,

$$\begin{aligned} \phi_1, \lambda_1 &: \text{latitude and longitude of first reference point} \\ \phi_2, \lambda_2 &: \text{latitude and longitude of second reference point} \\ \Delta\lambda &: \text{the difference in longitude between the two points} \end{aligned}$$

2) ***Finding the intersection point:*** This was calculated from Edward Williams' aviation formulary [35] using series of GIS equations as follows:

$$\begin{aligned} \delta_{12} &= 2\arcsin\left(\sqrt{(\sin^2(\frac{\Delta\phi}{2}) + \cos\phi_1\cos\phi_2\sin^2(\frac{\Delta\lambda}{2}))}\right) \\ \theta_a &= \arccos(\sin\phi_2 - \frac{\sin\phi_1\cos\delta_{12}}{\sin\delta_{12}\cos\phi_1}) \\ \theta_b &= \arccos(\sin\phi_1 - \frac{\sin\phi_2\cos\delta_{12}}{\sin\delta_{12}\cos\phi_2}) \end{aligned}$$

if $\sin(\lambda_2 - \lambda_1) > 0$

$$\begin{aligned} \theta_{12} &= \theta_a \\ \theta_{21} &= 2\pi - \theta_b \end{aligned}$$

else

$$\begin{aligned} \theta_{12} &= 2\pi - \theta_a \\ \theta_{21} &= \theta_b \end{aligned}$$

$$\begin{aligned} \alpha_1 &= (\theta_{13} - \theta_{12} + \pi)\%2\pi - \pi \\ \alpha_2 &= (\theta_{21} - \theta_{23} + \pi)\%2\pi - \pi \\ \alpha_3 &= \arccos(-\cos\alpha_1\cos\alpha_2 + \sin\alpha_1\sin\alpha_2\cos\delta_{12}) \\ \delta_{13} &= \operatorname{atan2}(\sin\delta_{12}\sin\alpha_1\sin\alpha_2, \cos\alpha_2 + \cos\alpha_1\cos\alpha_3) \\ \phi_3 &= \arcsin(\sin\phi_1\cos\delta_{13} + \cos\phi_1\sin\delta_{13}\cos\theta_{13}) \\ \Delta\lambda_{13} &= \operatorname{atan2}(\sin\theta_{13}\sin\delta_{13}\cos\phi_1, \cos\delta_{13} - \sin\phi_1\sin\phi_3) \\ \lambda_3 &= (\lambda_1 + \Delta\lambda_{13} + \pi)\%2\pi - \pi \end{aligned}$$

where,

$$\begin{aligned} \phi_1, \lambda_1, \theta_1 &: \text{1st starting point \& bearing} \\ \phi_2, \lambda_2, \theta_2 &: \text{2nd starting point \& bearing} \\ \phi_3, \lambda_3 &: \text{intersection point} \\ \% &: \text{(floating point) modulo} \end{aligned}$$

3) ***Calculation of the distance:*** For calculating the distance, we can use the Haversine Formula.

$$\begin{aligned} a &= \sin^2(\frac{\Delta\phi}{2}) + \cos\phi_1\cos\phi_2\sin^2(\frac{\Delta\lambda}{2}) \\ c &= 2\operatorname{atan2}(\sqrt{a}, \sqrt{1-a}) \\ d &= Rc \end{aligned}$$

where, ϕ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km);

4) ***Using Kinematic equations to calculate the time:*** Once the system finds the distances to the final merging point, it calculates the time required to reach that point using two kinematic equations: (i) $d = u * t + \frac{1}{2}a * t^2$ and (ii) $d = vt$. We need the first equation to calculate the time for the vehicles in the ramps and the second equation for the vehicles in the freeways. We calculated the acceleration for the freeway vehicles from our preliminary data and found that the acceleration is on average $0.15m/s^2$. Since $0.15m/s^2$ acceleration is negligible enough to consider the freeway speed as constant, we consider the second equation for the freeway vehicles to calculate the time required to reach the merging point.

VI. Results

To evaluate our model, we conducted our pilot experiment on interstate I-26 for exits 27, 32, 34, and 36 in both East and West bound lanes. Three drivers participated in the experiment who have valid US driver licenses and are accustomed to driving in interstates. Before the experiment in the interstate, we trained the drivers about how the system works and how to interpret the alert and advisory messages. Among the three drivers, two drove in the freeway and one drove in the ramp. The driver who drove in the ramp synchronized his timing by phone with the lead driver in the freeway to merge at relatively the same time so the merge assistance system could detect a potential merge conflict. The second driver in the freeway followed the first driver, keeping a distance of around 50-100 meters. These drivers drove to generate three scenarios as described in Table V. Fig. 13 illustrates the visual representations of the advisory messages that drivers receive through the Android device. The distance covered by a vehicle from the decision point to the merging point (Distance), average acceleration of the ramp vehicle (Avg Accel), average speed of the freeway vehicles (Avg Speed), time to crash (TTC) values, and associated merge advisory messages of the three vehicles are described in Table VI . The associated merge advisory messages are also shown in the table. For example, the ramp vehicle in the exit 27 (EB) covered 166.208 meters from the decision point to reach the merging point. The freeway merge assistance system couldn't generate merge advisories for exit 34 (both East bound and West bound) due to the error in merging point approximation. The significant bend in the two ramps resulted the approximation error.

TABLE VI: Distance traveled and time passed to reach the merging point from the decision point with the associated merging advisory messages

Entrance Ramp No of I-26	Ramp vehicle			Freeway vehicle 1			Freeway vehicle 2		
	Distance (m)	TTC (sec)	Avg Accel (m/s^2)	Distance (m)	TTC (sec)	Avg Speed (mph)	Distance (m)	TTC (sec)	Avg Speed (mph)
27 (WB)	136.328	3.603	3.519	132.006	2.789	47.334	107.377	2.317	46.342
	Merge behind (Freeway vehicle 1)			Advisory not generated			Keep the speed		
32 (WB)	279.746	6.958	3.543	304.243	5.741	52.994	406.324	8.140	49.915
	Merge behind (Freeway vehicle 1)			Keep the speed			Slow down for (Ramp vehicle)		
34 (WB)	-1	-1	3.688	-1	-1	44.702	-1	-1	44.966
	Merge advisory not generated (details reason in section VI)			Advisory not generated			Advisory not generated		
36 (WB)	249.177	5.525	3.670	265.440	4.279	62.024	324.747	5.449	59.593
	Merge behind (Freeway vehicle 2)			Keep the speed			Advisory not generated		
36 (EB)	272.701	6.336	4.779	252.588	4.525	55.819	349.361	6.674	52.343
	Merge behind (Freeway vehicle 1)			Keep the speed			Slow down for (Ramp vehicle)		
34 (EB)	-1	-1	2.842	-1	-1	64.283	-1	-1	64.757
	Merge advisory not generated (details reason in section VI)			Advisory not generated			Advisory not generated		
32 (EB)	268.663	7.179	3.080	318.373	4.968	64.081	301.152	4.386	68.647
	Merge behind (Freeway vehicle 1)			Advisory not generated			Keep the speed		
27 (EB)	166.208	5.599	2.073	168.284	2.987	56.321	193.347	3.319	58.239
	Merge behind (Freeway vehicle 2)			Keep the speed			Advisory not generated		

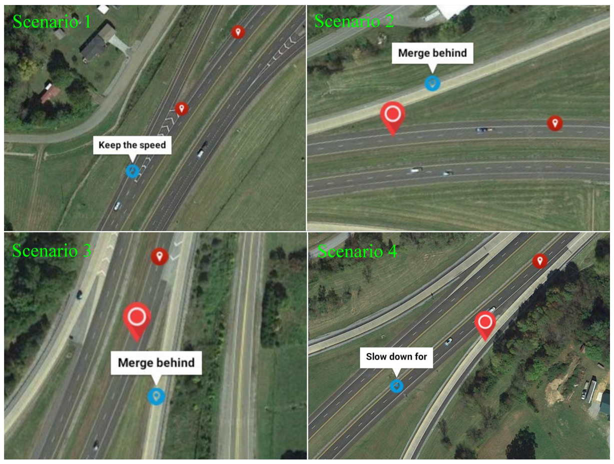


Fig. 13: Advisory visualization on the map

TABLE V: Advisory scenarios

Scenario 1	In this scenario, the assistance system suggests the ramp vehicle to merge in front of the freeway lead vehicle.
Scenario 2	In this scenario, the assistance system suggests the ramp vehicle merge behind the freeway lead vehicle.
Scenario 3	In this scenario, the assistance system suggests the ramp vehicle merge behind the freeway lag vehicle.

VII. Conclusions and Future Plans

Progress in CV technology has created opportunities for researchers and automakers to develop applications that provide vehicles with new safety, alert, and assistive features. This paper described the necessity of connected vehicle technology for detecting and avoiding merge conflicts on the freeways. The research described a novel decentralized freeway merge assistance system. To the best of our knowledge, this is the first attempt to develop and evaluate a freeway merge assistance system using real-world vehicular mobility traces and an actual interstate. We described the step-by-step technical approaches of a freeway merge assistance system. We also evaluated the merge assistance system for eight exits along interstate I-26. Experiments demonstrate that the system can provide accurate advisory information for straight ramps. However, the initial version of the merge assistance system has some limitations, such as detection of the conflicting vehicles in the right most lane in freeways, detection of non-conflicting vehicles, and incorporation of circular ramps. In the next version of the merge assistance system, we will try to address the limitations and minimize their impacts. In our algorithm, we have used the Haversine formula which is more precise than the Equirectangular approximation. However, to increase the scalability, we may use the Equirectangular approximation since it is computationally less expensive.

Another important issue of the freeway merge assistance system is the driver compliance. Driver compliances significantly impact the overall accuracy and performance of the merge assistance system. However, good driver compliances depend on the visual system of the advisory information. However, providing good visualization of advisory information that makes less distraction to the drivers is challenging. Therefore, in the future, we plan to incorporate the cruise control feature in our merge assistance system so every vehicle can act as a level one semi-autonomous vehicle. More specifically, every vehicle will trigger the cruise control at the decision point and maintain its current speed until it crosses the merging point. This way, we can minimize the impacts of the driver compliance issue. Finally, we will upload the source code of our merge assistance system to the Open Source Application Development Portal after rigorous testing.

References

[1] WHO. (20) Global status report on road safety, 2015. Retrieved: 2016-01-09. [Online]. Available: http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/

[2] NHTSA. (2007) Traffic safety facts, crash and stats. washington, dc: Nhtsas national center for statistics and analysis. Retrieved: 2016-01-09. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812332

[3] G. Stoller. (2007) Road congestion wastes 1.9 billion gallons of gas. Retrieved: 2015-11-5. [Online]. Available: http://usatoday30.usatoday.com/money/industries/energy/story/2012-03-25/wasted-fuel-report/53776164/1

[4] B. N. Janson, W. Awad, J. Robles, J. Kononov, and B. Pinkerton, "Truck accidents at freeway ramps: data analysis and high-risk site identification," *Journal of Transportation and Statistics*, vol. 1, no. 1, pp. 75–92, 1998.

[5] C. Yang and K. Kurami, "Homing guidance of on-ramp vehicles for safe merging," in *American Control Conference, 1992*. IEEE, 1992, pp. 1773–1776.

[6] C. Yang, M. Milacic, and K. Kurami, "A longitudinal control concept for merging of automated vehicles," in *Intelligent Vehicles Symposium (1993: Tokyo, Japan). Proceedings of the Intelligent Vehicles' 93 Symposium*, 1993.

[7] LANE-CHANGE-1.2. Lane changing using adaptive cruise control. Retrieved: 2017-01-10. [Online]. Available: https://www.itsforge.net/index.php/community/explore-applications#/36/91

[8] MMITSS-AZ. Signal phase and timing for emergency and transit vehicles. Retrieved: 2017-01-10. [Online]. Available: https://www.itsforge.net/index.php/community/explore-applications#/30/63

[9] CaA-Speed-Harmonization-v1.0. Collision warning and avoidance. Retrieved: 2017-01-10. [Online]. Available: https://www.itsforge.net/index.php/community/explore-applications#/35/111

[10] SPaT-1.2. Signal phase and timing through smart phone. Retrieved: 2017-01-10. [Online]. Available: https://www.itsforge.net/index.php/community/explore-applications#/30/76

[11] TCSPT-v1.0. Traffic congestion information. Retrieved: 2017-01-10. [Online]. Available: https://www.itsforge.net/index.php/community/explore-applications#/30/117

[12] RESCUME-CA-IASIM-1.0. Crash prevention. Retrieved: 2017-01-10. [Online]. Available: https://www.itsforge.net/index.php/community/explore-applications#/36/45

[13] OSADP. (2017) Federal highway administration of the u.s. department of transportation. Retrieved: 2017-01-10. [Online]. Available: http://www.itsforge.net/

[14] Z. Wang, L. Kulik, and K. Ramamohanarao, "Proactive traffic merging strategies for sensor-enabled cars," in *Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*. ACM, 2007, pp. 39–48.

[15] H. Park, C. Bhamidipati, and B. Smith, "Development and evaluation of enhanced intellidrive-enabled lane changing advisory algorithm to address freeway merge conflict," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2243, pp. 146–157, 2011.

[16] M. T. Hayat, H. Park, and B. L. Smith, "Connected vehicle enabled freeway merge assistance system-field test: Preliminary results of driver compliance to advisory," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 1017–1022.

[17] Y. Wang, E. Wenjuan, W. Tang, D. Tian, G. Lu, and G. Yu, "Automated on-ramp merging control algorithm based on internet-connected vehicles," *IET Intelligent Transport Systems*, vol. 7, no. 4, pp. 371–379, 2013.

[18] X.-Y. Lu, H.-S. Tan, S. E. Shladover, and J. K. Hedrick, "Implementation of longitudinal control algorithm for vehicle merging," *Ann Arbor*, 2000.

[19] R. Cowan, "The uncontrolled traffic merge," *Journal of applied probability*, pp. 384–392, 1979.

[20] R. W. Hall and C. Li, "Evaluation of priority rules for entrance to automated highways," *Journal of Intelligent Transportation Systems*, vol. 6, no. 2, pp. 175–193, 2001.

[21] R. W. Hall, A. Nowroozi, and J. Tsao, "Entrance capacity of an automated highway system," *Transportation Science*, vol. 35, no. 1, pp. 19–36, 2001.

[22] X.-Y. Lu, P. Varaiya, R. Horowitz, D. Su, and S. E. Shladover, "A new approach for combined freeway variable speed limits and coordinated ramp metering," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, 2010, pp. 491–498.

[23] C. Huang, M. Jiang, and G. Chai, "Fuzzy control for ramp metering and variable speed limitation of freeway," *Computer Technology and Development*, vol. 12, no. 12, pp. 38–41, 2010.

[24] X.-Y. Lu, H.-S. Tan, S. E. Shladover, and J. K. Hedrick, "Automated vehicle merging maneuver implementation for ahs," *Vehicle System Dynamics*, vol. 41, no. 2, pp. 85–107, 2004.

[25] S. Kato and S. Tsugawa, "Cooperative driving of autonomous vehicles based on localization, inter-vehicle communications and vision systems," *Jsae Review*, vol. 22, no. 4, pp. 503–509, 2001.

[26] Q. Xu and R. Sengupta, "Simulation, analysis, and comparison of acc and cacc in highway merging control," in *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*. IEEE, 2003, pp. 237–242.

[27] T. F. Golob, W. W. Recker, and V. M. Alvarez, "Safety aspects of freeway weaving sections," *Transportation Research Part A: Policy and Practice*, vol. 38, no. 1, pp. 35–51, 2004.

[28] T. R. Connolly and J. K. Hedrick, "Longitudinal transition maneuvers in an automated highway system," *Journal of dynamic systems, measurement, and control*, vol. 121, no. 3, pp. 471–478, 1999.

[29] V. Milanés, J. Godoy, J. Villagrá, and J. Pérez, "Automated on-ramp merging system for congested traffic situations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 500–508, 2011.

[30] L. Davis, "Effect of adaptive cruise control systems on mixed traffic flow near an on-ramp," *Physica A: Statistical Mechanics and its Applications*, vol. 379, no. 1, pp. 274–290, 2007.

[31] A. Kesting, M. Treiber, M. Schönhof, F. Kranke, and D. Helbing, "Jam-avoiding adaptive cruise control (acc) and its impact on traffic dynamics," in *Traffic and Granular Flow05*. Springer, 2007, pp. 633–643.

[32] T. Sakaguchi, A. Uno, and S. Tsugawa, "An algorithm for merging control of vehicles on highways," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (1997: Grenoble, France). Innovative robotics for real-world applications... Vol. 3*, 1997.

[33] Arada-Systems-Inc. Arada locomate classic on board unit. Retrieved: 2017-01-20. [Online]. Available: http://www.aradasystems.com/locomate-obu/

[34] 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.

[35] E. Williams. Aviation formulary. Retrieved: 2017-01-20. [Online]. Available: http://williams.best.vwh.net/

CHAPTER 3

STATE-OF-THE-ART SEQUENTIAL SIMULATORS

Comparative Study of Connected Vehicle Simulators

Md Salman Ahmed, Mohammad Asadul Hoque, Phil Pfeiffer
Department of Computing
East Tennessee State University
ahmedm@goldmail.etsu.edu, hoquem@etsu.edu, phil@etsu.edu

***Abstract*—Contemporary studies of Intelligent Transportation Systems (ITS) use simulations of vehicular and communications traffic, due to the ethical and practical infeasibility of conducting experiments on real transportation networks. Different simulators have been developed for modeling real-time vehicular mobility and inter-vehicular communication under varying traffic and roadway conditions. While most model the effect of mobility on communications, only a few simulate the impact of inter-vehicular communication on vehicular mobility. None, moreover, are implemented as parallel or distributed frameworks: an essential requirement for the study of ITS applications in large-scale urban environments. As a starting point for developing such a framework, one contemporary simulator, VNetInetSim, was tested to determine its behavior under large loads. Testing determined that VNetInetSim's memory usage and execution time increase exponentially in the number of simulated vehicles while remaining relatively constant under increased communication traffic.**

Keywords—Intelligent Transportation System (ITS), Inter-vehicle Communication, Simulator, Vehicle dynamics, Vehicular Ad Hoc Network.

I. Introduction

Over the past few decades, a substantial increase in automobile usage has led to increases in highway congestion, incidents, fatalities and greenhouse gas emissions. In 2012 USA TODAY reported that Americans annually waste 1.9 billion gallons of gasoline in traffic on congested roads and pay more than $100 billion in wasted fuel and lost time [1]. These adverse effects of automobile usage impact peoples' lives and degrade the quality of the Earth's environment.

Currently, automakers and technology developers like Google, Ford, and General Motors are making concerted efforts to improve surface transportation through Automated Vehicle (AV) technology [2], [3]. While AV can potentially reduce the stress of navigating traffic, its focus in most of cases is limited to the operation of vehicles in isolation from one another. This limitation is addressed by Connected Vehicle (CV) technology, which seeks to apply inter-vehicular communication to the development of safe, driver-friendly, and energy efficient assistive technologies for vehicle operation. One of the primary goals of CV research is the optimization of traffic flow across an entire transportation network through the exchange of information obtained through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. This exchanging of information, collectively known as V2X communications, could assist drivers in avoiding congestion, reducing vehicle stops, choosing a best route, and optimizing fuel efficiency.

978-1-5090-2246-5/16/$31.00 ©2016 IEEE

The large scale deployment of CV technologies faces several challenges, particularly for urban environments. Evaluating the performance of CV-based safety-critical real-time applications in large-scale urban environments under varying traffic and roadway conditions is difficult, since these conditions can't be generated in practice. Additionally, failures of CV-based applications may result in loss of lives. These issues can be addressed by using simulations to study and test ITS applications. Simulating ITS and CV systems, however, requires the integration and synchronization of two tightly coupled domains. The one, the transportation domain, models vehicular mobility, including traffic routing, car-following, lane-changing, vehicle dynamics, driver behavior, and traffic signal controls. The other, the communication domain, models mechanisms for data-traffic-related communications, including packet routing, end-to-end message delivery, and V2X-related cross-layer protocols. These two domains directly affect each other's operation. For example, high speed traffic networks with high vehicle density may delay V2X communications and degrade communication quality [4]. On the other hand, communication delay and data loss may degrade the modeling of vehicular operation. Such degradations, even if minute, could adversely affect the ability of V2X-based applications to assure their users' safety.

Efforts to develop a complete transportation simulator with a wireless network simulator for modeling and evaluating V2X-based ITS applications have been ongoing for the past decade. Older simulators fed fixed mobility trajectories to a communication network simulator. Many researchers [5]–[8] have studied the various mobility models developed for state-of-the-art simulators. However, a comparative modular analysis of different simulator components has yet to be written. Our current research, which focuses on the capabilities and limitations of existing sequential simulators in terms of their modular organization and architecture, has identified the need for a parallel simulation platform to support large-scale simulations of urban surface transportation systems [9].

The rest of the paper is organized as follows. Section II surveys the state of the art in CV simulators. Section III summarizes this survey's findings in tabular form. Section IV presents the results of preliminary load tests of VNetInetSim, a contemporary ITS simulator, and what they reveal about the simulator's scalability. Section V concludes with considerations related to the implementation of parallel simulators for evaluating large scale urban vehicular networks.

II. State of the Art VANET Simulators

Current Vehicular Ad Hoc Network (VANET) simulators can simulate the impact of vehicular communication on transportation systems. Some simulators can also create dynamic mobility trajectory traces and mobility models. Examples of these simulators include ASH, STRAW, Veins, VnetIntSim, TraNS, iTETRIS, GrooveSim, and Automesh.

A. *ASH*

Application-aware SWANS with Highway mobility (ASH) [10] provides an application-aware mobility model using two-way communication between a vehicular mobility model and a network simulator. Ibrahim and Weigle use the term "application-aware" to emphasize ASH's simulation of safety considerations such as alert information and lane-changing through two-way communication.

ASH extends work by various authors. Its supporting modules include the Scalable Wireless Ad hoc Network Simulator (SWANS) [11], which ASH uses as its network model; the Intelligent Driver Model (IDM) [12] module, which models how cars follow other cars; the Minimizing Overall Braking decelerations Induced by Lane changes (MOBIL) [13] module, which uses an incentive criterion for lane attractiveness and a safety criterion to model lane changes; and a node model for its mobility model. ASH also uses the Inter-Vehicle Geocast (IVG) [14] and probabilistic IVG (p-IVG) [15] protocols to broadcast messages.

ASH extensions to SWANS include the following:

- *Modeling two-way communication between the mobility and networking models*. ASH implements two-way communication by using its application layer to override IDM/MOBIL's normal behavior through acceleration, deceleration, and lane-change mobility control primitives.
- *Modeling highway topology*. ASH's configuration file specifies road segment characteristics such as segment length, number of directions, number of lanes, and the number and locations of exits and entries.
- *Modeling mobility states*. ASH's node model represents a participating vehicle as a mobile communicating node, a non-participating vehicle as a mobile silent node, a roadside unit as a static communicating node, and a road obstacle as a static silent node. Participating vehicles run user-defined applications at simulation time whereas non-participating vehicles run a null application.
- *Intelligent broadcast*. In place of flooding-based broadcasting, ASH uses the IVG algorithm with a timer for node broadcast. IVG reduces network traffic by using a timer to expire broadcast messages.
- *Logging and statistical facilities*. ASH supports logging utilities at different levels including the simulation, lane, vehicle, and message type levels. It also maintains the statistical simulation data of every vehicle in order to answer statistical queries.

B. *OVNIS*

Pigne et al. describe OVNIS as a realistic vehicular network management platform that can adjust node mobility and generate vehicular traces at runtime [16]. OVNIS manages an interconnection between the Simulation of Urban Mobility (SUMO) traffic simulator [17], a vehicular mobility simulator that supports programmed interaction through Application Program Interfaces (APIs), and network simulator 3 (ns-3) [18], a wireless network simulator that can simulate about 20000 nodes in a network. OVNIS also embeds a tool that generates vehicular traces based on real traffic data.

OVNIS's Traffic Aware Network Manager, the network management platform's main component, maintains a feedback-based interconnection with its traffic simulator and nodes applications modules. The Traffic Aware Network Manager module does the following during simulation:

- Starts, initializes and operates the network simulator.
- Starts the traffic simulator.
- Allows the nodes applications module to query the traffic simulator about every node's speed, position, speed limit, and lane number.
- Iteratively pulls mobility information from the traffic simulator.
- Manages node mobility according to the pulled mobility information.

Pigne et al. evaluated OVNIS using two experiments. The first tested OVNIS's overall computation performance based on its radio signal ranges. The experimental data shows that "the smaller the range, the faster the computation." The second experiment evaluated OVNIS's correctness, based on the extent to which simulated vehicles changed routes as the volume of vehicles increased. Their experiments showed that the vehicles' average speed decreases and inter-vehicular communication increases with an increase in the volume of vehicles. Then the vehicles start finding alternative routes and managing their routes.

C. *STRAW*

Choffnes and Bustamante's STreet RAndom Waypoint (STRAW) [19] application supports the modeling of vehicular motion in urban roads. STRAW can model road segments, intersections, traffic control mechanisms, and individual vehicles, including high speed vehicles and inter-vehicular communication. STRAW's support for modeling individual vehicles, according to its authors, distinguishes it from earlier VANET simulators.

STRAW treats a vehicle as a node with a set of properties, including maximum speed, reaction time and acceleration rate. Road segments, or portions of roads between two intersections, are modeled according to their shape, length, width, name, speed limit, class and address attributes. Traffic control mechanisms provide deterministic admission control protocols for vehicles at each intersection.

STRAW is architected as a system of three interacting component models. They include an intra-segment mobility model, an inter-segment mobility model, and a route management and execution model.

The intra-segment mobility model simulates vehicle motion within individual road segments. Motion is simulated using a car-following mechanism that accounts for the speed of the vehicle that a simulated vehicle is following and the distance to that vehicle. Vehicles use this model to accelerate to a maximum limit and decelerate on encountering speed limits, stop signs and stoplights.

The inter-segment mobility model determines how vehicles behave at intersections. The model applies a deterministic admission control protocol to determine how vehicles accelerate and decelerate. It also determines a vehicle's waiting time at stop signs and stop lights.

The route management and execution model determines the road segment that a vehicle will enter when it crosses an intersection. The model can choose this segment using a deterministic or a stochastic strategy. The deterministic strategy selects the next segment based on the fastest time and shortest distance to a preassigned destination, as calculated by the A* search algorithm. The stochastic strategy assigns probable road choices to a vehicle based on its trajectory. It then uses a probability value at each intersection to select the next segment.

STRAW supports two strategies for modeling driver response to vehicular collisions. In the particle system approach, a vehicle detects and reacts to collision events. In the vehicular approach, a vehicle detects collisions and avoids them when it can.

According to Choffnes and Bustamante, STRAW's mobility model is general enough to integrate into any wireless network simulator. The model performs well in terms of memory usage, but the computation cost is high for large numbers of vehicles. The model also fails to support the dynamic allocation and deallocation of vehicle nodes and lane changing.

D. Veins

The Vehicles in Network Simulation (Veins) [20] is a hybrid framework for evaluating the impact of inter-vehicular communication (IVC) protocols on road traffic mobility. Veins consists of a network simulator, a road traffic simulator, and a communication channel that supports the active exchange of control and data between the two simulators.

Veins' network simulator, OMNeT++ [21], is an event based simulator that simulates VANET protocols with the help of Veins' INET Framework extension. OMNeT++ represents VANET scenarios as hierarchical modules and stores the relationship and communication links between modules in network description files. Connectivity protocols such as TCP, UDP, IPv4, and ARP are added to OMNeT++ as extensions by the INET Framework.

Veins' road traffic simulator extends SUMO with Krauß's (1998) car-following mobility model. According to Sommer et al. [22], combining SUMO with the IVC protocols provides better simulation results than SUMO alone.

Veins uses dedicated modules to support bidirectional communications between OMNeT++ and SUMO. These modules use a TCP connection to exchange simulation commands and mobility traces. Each simulator buffers commands as it receives them and processes commands in the order received.

Commands are processed in rounds, as follows. At each time step, OMNeT++ sends all buffered commands to SUMO. SUMO simulates a round of traffic, then replies with a series of commands and generated mobility traces. OMNeT++ uses the traces to reconfigure the movement of nodes (vehicles). OMNeT++ allows nodes to alter their speeds and routes according to IVC, if all commands are processed and nodes reconfigured before next scheduled time step.

Sommer et al. used Veins to evaluate the impact of two IVC protocols on VANET scenarios. In the one protocol, vehicles communicate directly to a dedicated centralized Traffic Information System (TIS) using TCP connections and standard MANET (Mobile Ad Hoc Network) protocols. Vehicles exchange incident warnings with the TIS at intervals of 60s or 180s depending on road traffic. The TIS also maintains connections with roadside units in order to improve IVC. In the other protocol, vehicles maintain inter-vehicle communications by distributed or self-organized TIS using UDP broadcast communication. Incident warnings are flooded through VANET by UDP broadcast. When a vehicle gets a warning message, it queries the originating vehicle to determine if the warning is current.

The authors evaluated the protocols' impacts on vehicular mobility using a Manhattan grid and a real street map. In both cases, the authors ran four sets of simulations:

- One where vehicles were free to move without any interruption, with no IVC.
- One where the leading vehicle was stopped for a short duration with no IVC.
- Two where the vehicles' average speeds were calculated based on small and large scale simulations with the support of IVC. The small scale and large scale simulations used 5 hops and 25 hops to disseminate information, respectively.

Stationary vehicles in these experiments reported incidents using timestamped warning messages. Upon identifying these incidents, the network simulator stored the incident information and adjusted travel time for the stationary vehicles. The simulation then used Dijkstra's shortest path algorithm to compute new routes that bypassed the incident for the segment's other vehicles.

In both sets of experiments, the average speed of the first, third and fourth runs was greater than the second run. This indicates that stopping the leading vehicle in the second set of simulations caused congestion that increases other vehicles' travel time. During the third and fourth runs, those vehicles used inter-vehicle communication to get congestion information, then change their routes and increase their average speed.

E. VNetIntSim

Vehicular Network Integrated Simulator (VNetIntSim) [23] provides a modeling and simulation framework for VANETs and Intelligent Transportation System (ITS) applications. VnetIntSim consists of linker modules that integrate the INTEGRATION traffic simulator [24] with the OPNET communication network simulator [25]. These modules provide a two-way communication channel between INTEGRATION and OPNET.

Four modules drive VnetIntSim's operation. VnetIntSim's configuration reader module specifies an XML topology file containing vehicle specifications for configuring OPNET. VnetIntSim's communication module creates a shared memory region for the INTEGRATION and OPNET simulators, which then exchange information through shared memory. INTEGRATION's location module calculates vehicular locations and sends them to OPNET's driver module. Finally, its driver module checks simulation time from the received information, identifies simulation time mismatches, fixes inconsistencies and updates the vehicles' information.

When VnetIntSim starts execution, it establishes a communication channel between INTEGRATION and OPNET. First, the two simulators exchange hello messages to create the connection. The simulators then synchronize their simulation attributes, interval, and duration; the number of vehicles; and network size.

After successful synchronization, VnetIntSim enters its simulation loop. The VnetIntSim simulator primarily does movement-based simulation. It provides updates on the number of moving vehicles in a network, their locations, and traffic density. Though the simulator can simulate simple vehicle-to-vehicle and vehicle-to-infrastructure scenarios consistently, it fails to simulate large-scale scenarios.

F. TraNS

The Traffic and Network Simulation Environment (TraNS) simulator [26] simulates VANETs, accounting for vehicular mobility. TraNS supports two modes of simulation. In network-centric simulation, TraNS simulates statically determined traffic flows (e.g. music or travel information) [27]. The traffic simulator generates a simulation trace and the network simulator simulates the trace file. In application-centric simulation, TraNS allows dynamically generated exceptional events (e.g. abrupt braking and collision avoidance) to alter traffic [28]. Since the traffic and network simulators can run concurrently in application-centric simulation, no trace file is generated. As a result, this approach reduces the memory consumption for large-scale simulation.

G. iTETRIS

The Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions (iTETRIS) [29] simulates ITS applications on large-scale vehicular networks. iTETRIS supports WiMAX, UMTS, and DVB-H wireless and radio access technologies. iTETRIS is the first simulator to support the European Telecommunications Standard Institute (ETSI) ITS G5A standard.

According to Rondinone et al., iTETRIS achieves accurate simulations for realistic and complex traffic scenarios. Its modular architecture supports the integration of external modules. iTETRIS proper is a front-end for ns-3 and SUMO. It accepts input on roads and traffic in a SUMO-compatible format. The iTETRIS Controlling System interacts with SUMO and ns-3 and synchronizes simulation data with ITS applications using push-pull command mechanisms.

iTETRIS's accuracy for simulations of low- and mid-density traffic is better than its simulations of high-density traffic. Its features include providing information on fuel consumption and traffic congestion along with suggesting speed and route changes accordingly.

H. GrooveSim

GrooveSim [30] simulates inter-vehicular communication and vehicular mobility in a road traffic network using the authors' communication and mobility model and the GrooveNet routing protocol. GrooveNet, a hop-based communication protocol, uses a dedicated short range communication based transceiver, a global positioning system, a cellular modem, and audio/video devices to broadcast data and information over multiple hops.

GrooveSim represents a vehicular network as a planar graph whose edges represent road segments and whose vertices represent intersections. Road segments are modeled using Topologically Integrated Geographic Encoding and Referencing (TIGER) [31] records that contain the segments' names, types, locations (latitude and longitude), addresses, and speed limits. The graph abstraction is used for the shortest path calculation and region partitioning.

GrooveSim supports an on-road driving mode, a virtual traffic network simulation mode, a playback mode, a hybrid simulation mode, and a test scenario generation mode. In its driving mode, a real vehicle sends warning messages to other real vehicles using the GrooveNet portable networking kit and sends warning messages. In simulation mode, GrooveSim simulates a virtual road traffic network based on vehicular mobility and communication models. In playback mode, it replays simulations of vehicular movement and communication using drive and simulation mode logs. In hybrid simulation mode, it simulates real and virtual vehicles on a road traffic network. In test generation mode, it generates parameterized simulation scenarios using models that include vehicles' IDs, speed models, origins, destinations, routes, and waypoints along the route.

GrooveSim defines its own mobility and communication models. The mobility model determines vehicular mobility based on a minimum and maximum speed, the number of vehicles on road segments, road segment speed limits, and a four-state Markov-based probabilistic model. The probabilistic model uses two states for city roads and two for highway roads. The communication model uses a two-state Gilbert-Elliot Markov model, a collision model, and a channel model to guarantee concurrent inter-vehicular communications.

GrooveNet's communication protocol uses a message diffusion mode to periodically exchange non-critical data such as congestion information. It uses a message directed mode to immediately exchange time-critical data such as alert messages. The protocol uses region based multi-hop routing in order to speed the communication and reduce message flooding overhead.

GrooveSim provides on-road crash warnings, sudden braking alerts, congestion information, traffic updates, and location based commercial services.

I. Automesh

The Automesh [32] simulation framework for ITS applications integrates five modules with three plug-in modules, as follows:

- *Driving simulator module*. Automesh generates a dynamic mobility model for individual vehicles using an environmental model that supports speed limits and traffic signals. Automesh also accounts for vehicle dynamics including rates of acceleration and deceleration. This ability to dynamically generate mobility models distinguishes Automesh from other network and traffic simulators.
- *Network simulator module*. Automesh's network simulator simulates inter-vehicle communication by using received data from the driving simulator's dynamic mobility models to change driving behavior.
- *Propagation simulator module*. To evaluate the correctness and performance of communication protocols, Automesh provides a propagation simulator that simulates propagation calculation algorithms.
- *Geographic database server module*. This module provides geographic information such as road network information, a digital elevation model, and real 3D building information.
- *Automesh graphical user interface module*. This module provides a graphical user interface for configuring simulations and playing simulations' animations.
- *Vehicle control plug-ins*. This module allows the driving simulator to attach custom driving behavior algorithms and custom mobility models to itself.
- *Propagation plug-ins*. This module allows custom wireless propagation models to interface to the network simulator.
- *Communication protocol plug-ins*. This module allows customized communication protocol stacks to interface to with network simulator.

III. Comparative Summary

All of these simulators are implemented as sequential programs, though some could be modified to run in distributed and parallel computing environments. OVNIS, TranNS, GrooveSim, and Automesh model vehicular mobility dynamically using vehicle trajectory traces whereas ASH and STRAW use the car-following model. VnetIntSim and iTETRIS use linker modules to communicate between transportation and network modules whereas ASH, Veins and OVNIS use two-way communication. GrooveSim and Automesh also support the modeling of communication protocols. Table I summarizes these simulators' names, their mobility models and their communication models.

IV. Need for Parallel Simulation Platform

Scalability is the most important limitation for all current platforms. VANETs and ITS simulation require high levels of scalability. Sequential simulations lack the processing resources to simulate urban transportation networks in real time. For example, in one experiment involving a sequential simulator [33], the simulation of a 200-node network created 4,600,000 events and required 16 minutes to process the events.

TABLE I: Summary of the above mentioned simulators

Simulators	Two simulation models of a simulator	
	Mobility model	**Network model**
ASH	IDM/MOBIL, IVG	SWANS
OVNIS	SUMO	NS-3
STRAW	Developed their own model	SWANS
Veins	SUMO, IVC	OMNET++
VnetIntSim	INTEGRATION	OPNET
TraNS	SUMO	NS-2
iTETRIS	SUMO	NS-3
GrooveSim	Developed their own model	Their own network model
Automesh	Customizable to add any mobility model	NS-2 or Qualnet

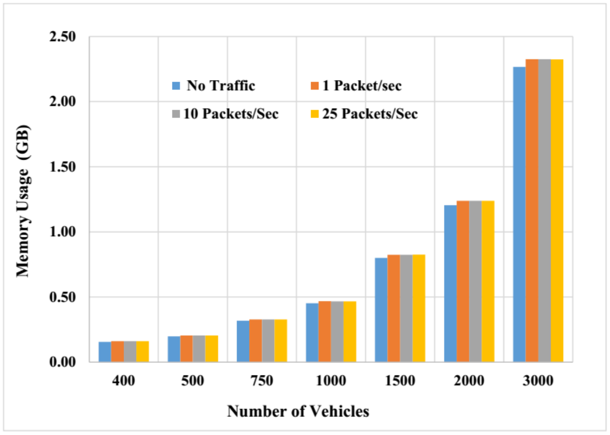


Fig. 1: The memory usage (GB) vs the number of nodes

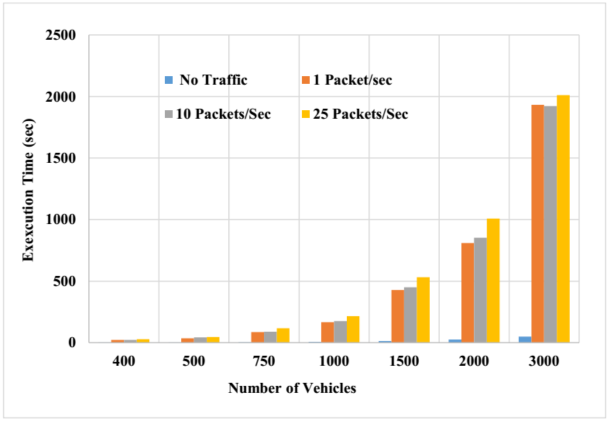


Fig. 2: The execution time (sec) vs the number of nodes

These levels of simulated traffic flow impose time, resource, and scalability constraints on sequential simulations of large-scale urban environments.

These observations motivated us to use VNetInetSim to analyze those factors that had the greatest impact on VANET scalability. We found that the number of wireless nodes (vehicles) and the data traffic rate per node were the primary

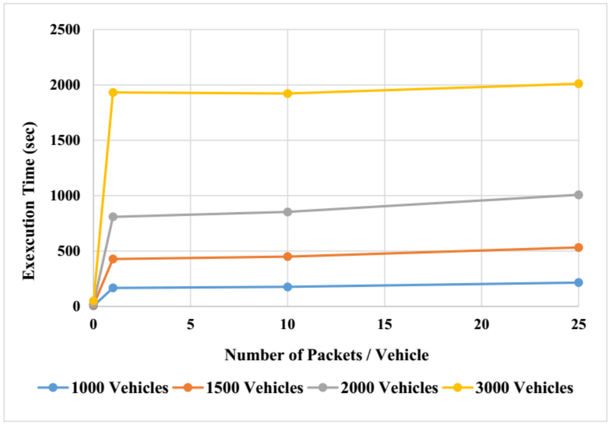


Fig. 3: The execution time (sec) vs traffic rate per vehicle

impediments to scalability. Our preliminary results show that memory usage and execution time increase exponentially with the number of vehicles in the system (Fig. 1 and 2). As shown in Fig. 1, increasing the data traffic rate for a given number of nodes has no significant effect on the memory usage. This is because OPNET, VNetInetSim's network simulator, discards packets when they reach their destinations, releasing their memory. These increases, however, do produce significant increases in simulation execution time (Fig. 2). This is to be expected. Fig. 3 shows a log-increase in the simulation time with respect to the traffic rate. These results were obtained on a machine of Intel Core-i7 Quad-core processor, 4 GB of memory, and running windows 7 Ultimate.

V. Conclusions

Most of the VANET simulators we surveyed can effectively simulate small-scale transportation networks. However, the simulation of large-scale urban environments will require parallel and distributed simulation. A parallel and distributed simulation platform must address the issues of optimal network partitioning, accurate parallel architecture, and synchronization between simulators. Graph-theoretical approaches and sparse matrix-based techniques could be used to achieve the necessary partitioning [34], while a parallel architecture that synchronizes separate communication and simulation modules could be used to structure this platform. We plan to investigate the challenges and issues pertaining to implementing parallel simulation platforms for the large-scale evaluation of CV-based urban transportation network.

References

[1] G. Stoller. (2007) Road congestion wastes 1.9 billion gallons of gas. Retrieved: 2015-11-5. [Online]. Available: http://usatoday30.usatoday.com/money/industries/energy/story/2012-03-25/wasted-fuel-report/53776164/1

[2] Google. (2016) Google self-driving car project. Retrieved: 2015-02-09. [Online]. Available: https://www.google.com/selfdrivingcar/

[3] B. CAREY. (2012) Shelley, stanford's robotic racecar, hits the track. Retrieved: 2016-02-09. [Online]. Available: http://news.stanford.edu/news/2012/august/shelley-autonomous-car-081312.html

[4] M. Alam, M. Sher, and S. A. Husain, "Vanets mobility model entities and its impact," in *Emerging Technologies, 2008. ICET 2008. 4th International Conference on*. IEEE, 2008, pp. 132–137.

[5] F. J. Martinez, C. K. Toh, J.-C. Cano, C. T. Calafate, and P. Manzoni, "A survey and comparative study of simulators for vehicular ad hoc networks (vanets)," *Wireless Communications and Mobile Computing*, vol. 11, no. 7, pp. 813–828, 2011.

[6] S. A. Hussain and A. Saeed, "An analysis of simulators for vehicular ad hoc networks," *World Applied Sciences Journal*, vol. 23, no. 8, pp. 1044–1048, 2013.

[7] M. K. Patel, "Comparative study of vehicular ad-hoc network mobility models and simulators," *International Journal of Computer Applications*, vol. 47, no. 6, pp. 38–43, 2012.

[8] S. Khandelwal, "Comparative analysis of network simulator for vehicular adhoc networks (vanet) communication," *Journal of Advanced Computing and Communication Technologies*, vol. 2, no. 2, 2014.

[9] 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.

[10] K. Ibrahim and M. C. Weigle, "Ash: Application-aware swans with highway mobility," in *INFOCOM Workshops 2008, IEEE*. IEEE, 2008, pp. 1–6.

[11] R. Barr, "Swans-scalable wireless ad hoc network simulator," *March, URL¡ http://jist. ece. cornell. edu/docs. html*, 2004.

[12] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, p. 1805, 2000.

[13] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record: Journal of the Transportation Research Board*, 2007.

[14] A. Bachir and A. Benslimane, "A multicast protocol in ad hoc networks inter-vehicle geocast," in *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, vol. 4. IEEE, 2003, pp. 2456–2460.

[15] K. Ibrahim and M. C. Weigle, "Cascade: Cluster-based accurate syntactic compression of aggregated data in vanets," in *GLOBECOM Workshops, 2008 IEEE*. IEEE, 2008, pp. 1–10.

[16] Y. Pigné, G. Danoy, and P. Bouvry, "A platform for realistic online vehicular network management," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. IEEE, 2010, pp. 595–599.

[17] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–simulation of urban mobility," in *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, 2011.

[18] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, 2008.

[19] D. R. Choffnes and F. E. Bustamante, "An integrated mobility and traffic model for vehicular wireless networks," in *Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*. ACM, 2005, pp. 69–78.

[20] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved ivc analysis," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 1, pp. 3–15, 2011.

[21] A. Varga *et al.*, "The omnet++ discrete event simulation system," in *Proceedings of the European simulation multiconference (ESM2001)*, vol. 9, no. S 185. sn, 2001, p. 65.

[22] S. Krauß, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," Ph.D. dissertation, Universitat zu Koln., 1998.

[23] A. Elbery, H. Rakha, M. Y. ElNainay, and M. A. Hoque, "Vnetintsim: An integrated simulation platform to model transportation and communication networks."

[24] M. Van Aerde, B. Hellinga, M. Baker, and H. Rakha, "Integration: An overview of traffic simulation features," *Transportation Research Records*, 1996.

[25] J. Prokkola, "Opnet-network simulator," *URL http://www. telecomlab. oulu. fi/kurssit/521365A tietoliikennetekniikan simuloinnit ja tyokalut/Opnet esittely*, vol. 7, 2006.

[26] M. Piorkowski, M. Raya, A. L. Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux, "Trans: realistic joint traffic and network simulator for vanets," *ACM SIGMOBILE mobile computing and communications review*, vol. 12, no. 1, pp. 31–33, 2008.

[27] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 15–34.

[28] S. Rampfl, "Network simulation and its limitations," in *Proceeding zum Seminar Future Internet (FI), Innovative Internet Technologien und Mobilkommunikation (IITM) and Autonomous Communication Networks (ACN)*, vol. 57, 2013.

[29] M. Rondinone, J. Maneros, D. Krajzewicz, R. Bauza, P. Cataldi, F. Hrizi, J. Gozalvez, V. Kumar, M. Röckl, L. Lin *et al.*, "itetris: a modular simulation platform for the large scale evaluation of cooperative its applications," *Simulation Modelling Practice and Theory*, vol. 34, pp. 99–125, 2013.

[30] R. Mangharam, D. S. Weller, D. D. Stancil, R. Rajkumar, and J. S. Parikh, "Groovesim: a topography-accurate simulator for geographic routing in vehicular networks," in *Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*. ACM, 2005, pp. 59–68.

[31] U. S. C. Bureau. (2015) Us geological survey (usgs) topographic maps. Retrieved: 2015-10-25. [Online]. Available: http://www.census.gov/geo/maps-data/data/tiger.html

[32] R. Vuyyuru, K. Oguchi, C. Collier, and E. Koch, "Automesh: Flexible simulation framework for vehicular communication," in *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*. IEEE, 2006, pp. 1–6.

[33] H. Wu, R. M. Fujimoto, and G. Riley, "Experiences parallelizing a commercial network simulator," in *Simulation Conference, 2001. Proceedings of the Winter*, vol. 2. IEEE, 2001, pp. 1353–1360.

[34] M. 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.

CHAPTER 4

PARALLEL SIMULATOR: ISSUES AND CHALLENGES

Parallel Closed-Loop Connected Vehicle Simulator for Large-Scale Transportation Network Management: Challenges, Issues, and Solution Approaches

Mohammad A. Hoque, *Senior Member, IEEE*, Xiaoyan Hong, *Member, IEEE*, Md Salman Ahmed, *Student Member, IEEE*

***Abstract*— The augmented scale and complexity of urban transportation networks have drastically increased the execution time and resource requirements of vehicular network simulations, exceeding the capabilities of sequential simulators. The need for a parallel and distributed simulation environment is inevitable from a smart city perspective, especially when the entire city-wide information system is expected to be integrated with numerous services and ITS applications. In this paper, we present a conceptual model of an Integrated Distributed Connected Vehicle Simulator (IDCVS), which can emulate real-time traffic in a large metro area by incorporating hardware-in-the-loop simulation together with closed-loop coupling of SUMO and OMNET++. We also discuss the challenges, issues, and solution approaches for implementing such a parallel closed-loop transportation network simulator addressing partitioning problems, synchronization, and scalability issues. One unique feature of the envisioned integrated simulation tool is that, it utilizes vehicle traces collected through multiple roadway sensors—DSRC on-board unit, magnetometer, loop detector, and video detector. Another major feature of the proposed model is the incorporation of hybrid parallelism in both transportation and communication simulation platforms. We identify the challenges and issues involved in IDCVS to incorporate this multi-level parallelism. We also discuss the approaches to integrate hardware-in-the-loop simulation, addressing the steps involved in preprocessing sensor data, filtering and extrapolating missing data, managing large real-time traffic data, and handling different data formats.**

***Keywords*—Connected Vehicle, Parallel Simulation, Network Partitioning, Scalability, Communication Overhead, Dedicated Short Range Communication, hardware-in-the-loop simulation.**

I. Introduction

With the advent of big data and connected vehicle (CV) technologies, the parameters and requirements for simulating metro-scale urban transportation networks with heterogeneous vehicles have evolved substantially. Today's transportation engineers at the Traffic Management Centers (TMCs) feel the necessity of a parallel CV simulation tool that would allow them to visualize the immediate system-wide effect of any change in traffic parameters—signal timing, detour, lane closure—before making the decisions. Unfortunately, current state-of-the-art traffic simulators (VISSIM [1], CORSIM [2], SUMO [3] etc.) are not capable of modeling future transportation scenarios involving connected vehicles since traffic simulators only model vehicular traffic. Simulation of a transportation network with CV requires a bi-directional coupling mechanism between a transportation simulator and a communication simulator. This mechanism has led to the concept of the closed-loop CV simulator, which has recently drawn a significant amount of research interests within the community. However, the computational capacity of such a bi-directionally coupled (closed-loop) simulator is significantly limited by the number of CVs equipped with on-board units (OBUs) and the number of roadside units (RSUs) deployed within the metro-wide transportation network, since these DSRC devices transmit millions of basic safety messages (BSMs) packets every minute requiring massive computational resources. Existing sequential closed-loop simulators can barely handle one thousand vehicles simulated on a scenario involving no more than a few intersections. Thus, incorporating parallelism in both transportation and communication simulation platforms will enable efficient management of large-scale transportation network and control of traffic parameters involving connected vehicles. In addition, integrating roadway sensor data through hardware-in-the-loop simulation with the closed-loop software simulator will enable the traffic engineers to make informed decisions by evaluating the system-wide impact of changing traffic parameters in real-time.

A vast amount of research effort has been recently directed towards improving surface transportation through self-driving autonomous vehicles as well as connected vehicles (CVs) using the 5.9 GHz Dedicated Short Range Communication (DSRC) technology. Automakers and technology developers like Google, Ford, and General Motors etc. are working to improve the controllability features of autonomous or semi-autonomous vehicles. While self-driving cars can potentially reduce the stress of navigating through congested traffic, CVs can optimize the traffic flow across an entire transportation network through the exchange of information among vehicles and infrastructure. CV applications use information obtained through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications—collectively known as V2X communications—to assist drivers in avoiding congestion, reducing vehicle stops, choosing the best route, and optimizing fuel efficiency. Hence, CV-based emerging Intelligent Transportation Systems (ITS) applications can result in transformative changes to the overall surface transportation system.

To accurately simulate ITS applications on a scenario involving connected vehicles, it is necessary to integrate a full-fledged transportation simulator with a wireless network simulator, resulting in the need for a closed-loop simulator. This kind of closed-loop simulator requires a tight synchronization between two stand-alone simulation modules: a transportation module and a communication module. The transportation module is responsible for the modeling of vehicle mobility applications including traffic routing, car-following, lane-changing, vehicle dynamics, driver behavior modeling, and traffic signal control modeling etc. On the other hand, the communication module accounts for data traffic network modeling including packet routing, end-to-end delivery of messages using V2X communication, wireless media access, cross-layer protocols, information security, and authentication mechanisms.

In a CV simulation environment, the two simulation modules operate as a real-time feedback control loop with tight synchronization requirements. These two modules highly influence the operation of one another. For example, vehicle dynamics, mobility, speed, and density affect the communication links between vehicles as well as the data packet routing; hence, they also affect the communication quality, i.e., reliability, throughput, and delay. Conversely, the data communication parameters—for example, the number of packet losses between vehicles and the end-to-end delivery delay—can adversely affect the mobility decisions made by the transportation simulator, particularly when a V2X message carries detour information due to an accident. For a V2X-based safety application, it is important to realize that the slightest delay in communication, even about a fraction of a second, can have serious consequences and may even be fatal. Considering the complexity of each system (transportation and communication) in addition to the high level of interdependency between them, it is easy to perceive how challenging the simulation of an integrated CV system can be.

II. Related Work

Most of the previous efforts to simulate vehicular networks were based on fixed mobility trajectories that were fed to the network simulator. Several mobility generator frameworks (VANETMOBISIM [7], SUMO [3], MOVE [8], STRAW [9], FREESIM [10], CITYMOB [11]) have been developed to produce the vehicular trajectories that are fed into various network simulators (NS2 [12], NS3 [13], OMNET++ [14], OPNET [15], JIST/SWANS [16], QualNET [17], etc.) without incorporating the effect of ITS applications on the mobility of the vehicles. Lee and Park [18] used the NCTUns communications simulator to examine the effects of communications using VISSIM trajectory data offline with no feedback loop for traffic simulation. GrooveSim [19] simulates inter-vehicular communication and vehicular mobility in a road traffic network using a customized mobility model and the GrooveNet [20] routing protocol. MobiREAL [21] incorporates mobility support on the Georgia Tech Network Simulator (GTNetS [22]). The capabilities of these type of open-loop simulators are limited to studying only unidirectional effects between the two domains. For example, studying the effect of various mobility models on the performance of end-to-end data communication using these simulators could characterize the dependency of the communication module on the transportation module, but it would be impossible to study the impact of data communication on the transportation system by incorporating changes in vehicle route, speed, signal timings, and mobility patterns based on newly received messages. Hence, this approach cannot be used to study bidirectional effects between the two tightly coupled domains.

Recently, there has been a significant amount of interests and efforts to design closed-loop CV simulators by coupling two types of simulators. Traffic and Network Simulation Environment (TraNS [23]) links the traffic simulator SUMO and the network simulator ns-2. Multiple Simulator Interlinking Environment for IVC (MSIE [24]) integrates ns-2, VISSIM traffic simulation, and application simulation (MATLAB) into a simulation environment for vehicular ad hoc networks (VANETs). Veins [4] is a tool that provides a closed-loop integration using SUMO and OMNeT++ as traffic and communications simulator respectively. Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions (iTETRIS [25]) integrates SUMO with ns-3 through IP-based sockets and allows implementation of several ITS applications in various programming languages. VNetIntSim [6] couples OPNET and INTEGRATION. None of these closed-loop simulators integrate with hardware-in-the-loop simulation technique. Very recently, Songchitruksa et. al. developed a closed-loop CV simulator (CONVAS [5]) by coupling VISSIM and ns-3 with the support for hardware-in-the-loop simulation technique, which is the first closed-loop simulator incorporating roadway sensor data. However, these tools lack in providing support for simulating large-scale transportation scenario using parallel and distributed computing. Another major limitation is that there is no mechanism available for collecting roadway sensor data from individual intersections and feeding them to the simulation environment to facilitate real-time traffic decision support at the TMCs.

III. Parallel Simulation of Connected Vehicle Applications

Unfortunately, none of the simulation tools described in the previous section provide any mechanism for parallel or distributed simulation of connected vehicle applications for large-scale management of transportation network. The augmented scale and complexity of urban transportation networks have drastically increased the execution time and resource requirements of vehicular network simulations, exceeding the capabilities of sequential simulators. The need for a parallel and distributed CV simulation environment is inevitable from a smart city perspective where the entire city-wide information system will be integrated with numerous services and ITS applications, particularly when the metro-wide multimodal transportation systems get connected to the smart city infrastructure through DSRC. Currently, the New York City connected vehicle pilot project sponsored by the United States Department of Transportation (USDOT [28]) aims at the integration of multimodal transport (including subway, transit bus, and taxis) with the smart city infrastructure. One of the use cases of our envisioned parallel simulation tool is to provide very precise information about the traffic change consequences—such as transit bus delays or tentative queue length considering the preemptive detour advisory disseminated through DSRC—enabling a TMC official to make informed decision when a major corridor needs to undergo closure of lanes due to maintenance.

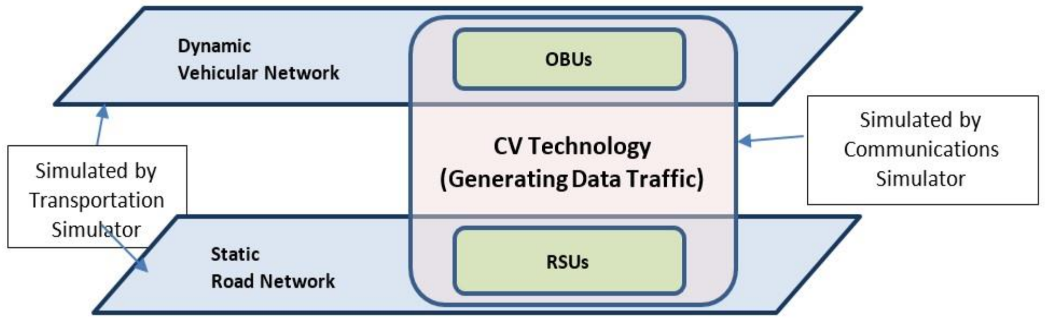


Figure 1: Bi-layer complex transportation network

A. Challenges and Issues

In this section, we identify some of the challenges and issues associated with implementing a parallel closed loop simulator for large-scale transportation network management. Later we provide insights to the solution approaches that can address these problems.

1) Partitioning of Bi-layer Complex Transportation Network

The fundamental research problem involved in this parallel simulator design is to determine a near-optimal partitioning heuristic using a bi-layer network model—a static road network overlaid with a dynamic vehicular network—connected by the CV technology which spans across both the networks (Figure 1). Earlier research mainly focused on partitioning static road networks for distributed simulation without considering the data traffic generated by DSRC communications. The bi-layer model will address partition issues in both the dynamic vehicular network involving CVs (OBUs) as well as the static infrastructure (RSUs) and the interactions between the two levels. The solution approaches in section IV will provide a guideline to incorporate real-world roadway traffic parameters with the data traffic parameters within the partitioning heuristic for connected vehicle environments.

The biggest challenge in partitioning vehicular networks is that the partitions cannot be fully separated. In fact, due to the communication and high mobility, partitions have a high level of interdependency and interactivity (i.e., a message or a vehicle moves from one partition to another) that necessitates communication between partitions to achieve consistency and accuracy. Inefficient partitioning of such networks can produce high communication volume between the different partitions and high processing overhead in each partition, consequently resulting in low simulation speeds. So, it is necessary to create partitions in such a way that reduces the interactivity and interdependence between them. Another proven NP-hard problem is the load-balancing problem. Due to the interdependency between events in different partitions, the simulation must be synchronized between the partitions; i.e., low-load (high-speed) partitions must wait for high-load (low-speed) ones to finish. This means that the maximum overall simulation speed is limited to the minimum speed among all the partitions.

2) Reducing Inter-Simulator Communication Overhead

A major problem for parallelizing a closed-loop CV simulator is that it not only requires decomposing the two standalone simulators (the transportation and communication simulators) and synchronizing the components within each simulator, but it also requires tight synchronization between the two simulators. DSRC technology requires that the vehicles broadcast their current locations every 0.1 seconds, meaning that the two simulators must synchronize ten times per second. This synchronization process adds extra overhead if the two simulators are running on separate computing nodes in a distributed computing environment requiring them to communicate over MPI. With a shared memory interface between the two simulators running in the same partition, this Inter-Simulator Communication overhead is expected to be reduced. However, using shared memory also creates a race condition between multiple processes running on the same computing node. Hence, there is always a trade-off between contention (shared memory) and latency (distributed memory), which is a major research problem. In addition, in a CV environment, the closed-loop interactions between communication and transportation systems must be executed in real-time to accurately model the impact of one system on its counterpart. For instance, the real-time interactions between SUMO and OMNET++ should facilitate dynamic speed control for the vehicles in the vicinity of traffic signals, where vehicles and signal controllers can exchange information to compute the optimal signal timing and vehicle trajectory.

3) Existence of heterogeneous vehicles

Another challenging aspect of simulating transportation network involving CVs is due to the slow market penetration rate of connected vehicles, which implies that during the transition period there will always be two types of vehicles on the road—one that is connected through DSRC (CV) and the other that is not connected (non-CV). It is expected that CV technologies will penetrate the market slowly over the next few years. Hence, until the time comes when all the cars on the road are equipped with factory-built or after-market DSRC devices, there will always be two types of vehicles on the road: one that has DSRC on-board unit (OBU) and the other that does not have OBU. CVs broadcast their actual GPS positions and speed every 0.1 seconds through the basic safety messages (BSMs) while the non-CVs can only be detected through roadway sensors and traffic light cameras. At present, there is no closed loop simulator that supports heterogeneous types of vehicles for simulation of CV applications. The only closed-loop simulator (CONVAS) that incorporates hardware-in-the-loop simulation allows the connected vehicles to communicate with the simulator through RSU and does not account for the non-connected vehicles.

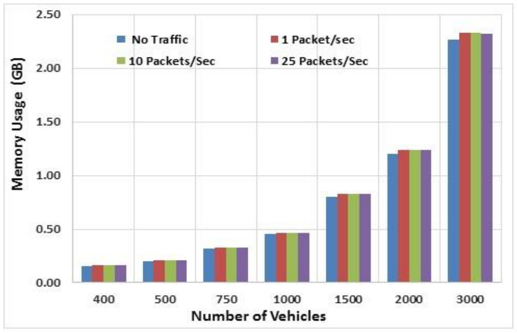
Using the state-of-the-art closed-loop simulator with the support for hardware-in-the-loop simulation technique, only the vehicles with OBU will be able to participate in the network-wide communication, while the vehicles without OBU will not be able to be detected in the simulator. Incorporating non-CVs in the hardware-in-the-loop simulation mechanism is quite challenging because the closed-loop CV simulator needs to be fed from several different sources of sensor data—CV traces through BSM messages and non-CV traces from roadway sensors (loop detectors and video detectors). Taking input from these sources, the simulator needs to be able to generate realistic mobility traces for the non-CVs, in addition to mapping the actual positions of the CVs where the simulator should graphically represent the CVs and non-CVs differently to distinguish between the actual position and speed vs. projected position and speed.

4) Synchronization problem

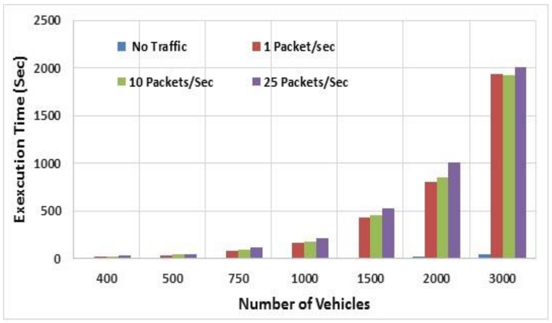
Simulation of data traffic is computationally more resource intensive than simulating vehicular traffic [6]. This makes the closed-loop simulation of CVs more challenging because of the imbalance of computation resource requirement causing synchronization problem between the transportation simulator and communication simulator. This is because of the huge amount of DSRC basic safety messages (BSM) disseminated from each vehicle every 0.1 seconds, where each BSM message needs to go through several layers of encapsulation and de-capsulation steps within the wireless network protocol stack at both ends. Some of the services in the data communication protocols, e.g. error detection, routing, and connection establishment, are computationally more expensive compared to the services from the vehicular traffic simulator that do not require passing through multiple layers of protocols. In fact, simulation of vehicular traffic only involves trace generation using microscopic mobility models. Hence, the data traffic simulator primarily causes the bottleneck. Typically, the data traffic is simulated using network simulators such as OPNET, OMNET++, Qualnet, NS-2, or NS-3. One experiment [29] demonstrated that the simulation of a 200-node network for only one minute generated more than 4,600,000 events and required 16 minutes of CPU time. The increasing complexity of the protocol stacks in communication end systems further aggravates this problem and has spurred efforts to develop parallel network simulators.

5) Scalability of Parallel Simulation

The scalability of parallel systems depends on the ratio of time spent in computation vs. communication. For any parallel system, the fraction of time spent in inter-process communication increases with the number of processors while the fraction of time spent in actual computation decreases. Initially, for the lower number of processors, the computational time is much greater than the communication time. With the increase of the number of processors, the computational time decreases with respect to communication time. At some point, for a specific number of processors (*p*), the communication time starts dominating over computation time. This value of *p* essentially determines how better the system is scalable. The higher the value of *p* the better the scalability. Therefore, scalability is one the most important problems in any parallel system, especially when it involves both distributed and shared memory architecture. Hence the architecture of such hybrid parallel system needs to be designed in such a way that reduces the inter-process communication overhead and increases the scalability. It is noteworthy to mention that this inter-process communication could take place between the transportation simulator and network simulator using shared memory (using OpenMP) or between the instances of the same simulator running different partitions on distributed cluster nodes (through MPI). Without achieving a certain level of scalability, the system will not be able to simulate a city-wide scenario with several hundred thousand vehicles and millions of BSM messages every minute. To study the scalability requirements for parallel implementation, we have evaluated the performance of a sequential CV simulator, VNetIntSim [33, 34], in terms of memory usage and execution time. The preliminary results showed that the number of wireless nodes (vehicles) and the data traffic rate per vehicle are the primary reasons behind the scalability issue. Figure 2 shows that both the memory usage and execution time increase exponentially with the number of vehicles in the system.



(a)



(b)

Figure 2. (a) Memory usage and (b) execution time vs. number of vehicles

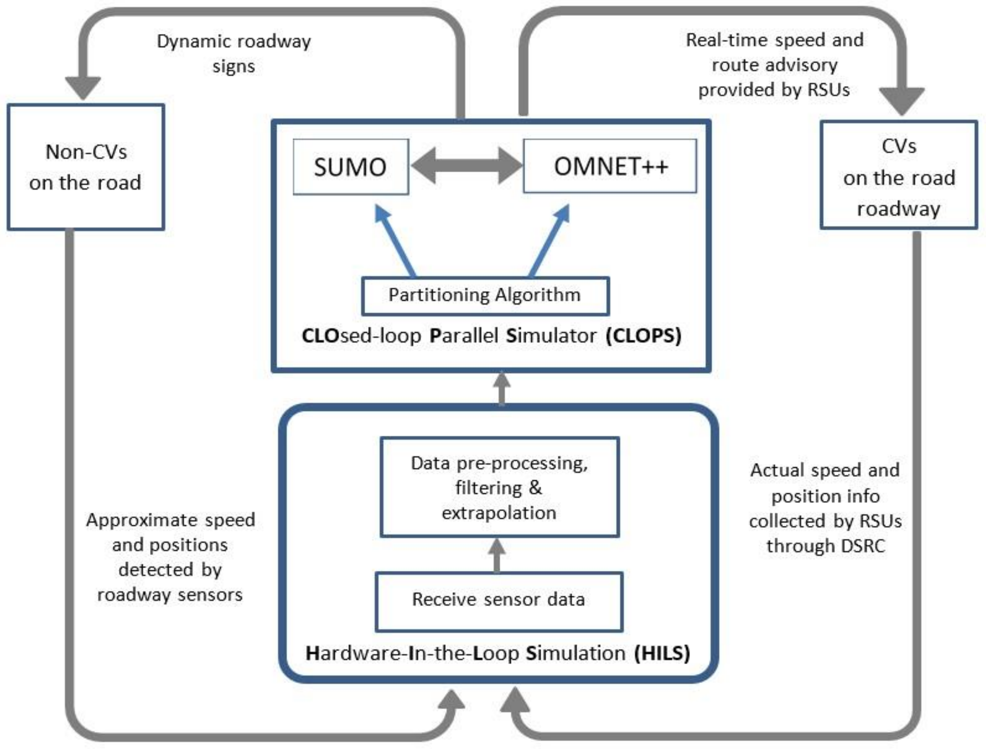

Dynamic roadway signs
Real-time speed and route advisory provided by RSUs
Non-CVs on the road
SUMO
OMNET++
CVs on the road roadway
Partitioning Algorithm
CLOsed-loop Parallel Simulator (CLOPS)
Data pre-processing, filtering & extrapolation
Receive sensor data
Hardware-In-the-Loop Simulation (HILS)
Approximate speed and positions detected by roadway sensors
Actual speed and position info collected by RSUs through DSRC

Figure 3: Conceptual Model of Integrated Distributed CV Simulator (IDCVS)

IV. Conceptual Model

In this section, we present a conceptual model of an **I**ntegrated **D**istributed **C**onnected **V**ehicle **S**imulator **(IDCVS)** and in section V we discuss the technical approaches for implementing such a robust simulation tool. Figure-3 below shows our hypothetical model of IDCVS system that includes hardware-in-the-loop simulation techniques for both CVs and non-CVs. IDCVS will incorporate two basic modules—a **CLO**sed-loop **P**arallel **S**imulator (**CLOPS**) and a **H**ardware-**I**n-the-**L**oop **S**imulation (**HILS**) module.

A. Hardware-In-the-Loop Simulation (HILS)

HILS will have an interface to receive the sensor data from both CVs and non-CVs through multiple sources. For non-CVs, the approximate location and speed can be detected through video detectors and inductive loop detectors, and this information will be passed as input to the HILS module. We can use the video detection software from ITERIS and the loop detector software from SIEMENS that can supply the sensor data to the HILS receiver component. On the other hand, the CV's can be detected more easily through the BSM messages received by the RSUs. Once the sensor data is received, additional data-preprocessing, filtering and extrapolation will be needed before the data can be used by CLOPS. This will require developing filtering algorithms for loop-detection and video-detection data to isolate the CV traces from the non-CV traces.

B. CLOsed-loop Parallel Simulator (CLOPS)

A CLOsed-loop Parallel Simulator (CLOPS) can be developed through coupling between SUMO and OMNET++, both of which are open source simulators. An efficient partitioning heuristic will decompose the complex transportation network into two separate sets of partitions—where each set of the partition will be sent to the individual simulator (SUMO and OMNET++). It might appear that CLOPS could be developed as a parallel and distributed framework on top of Veins since Veins also utilizes a coupling between SUMO and OMNET++. However, since Veins does not support heterogeneous vehicles, it is not possible to extend Veins for the simulation scenarios involving both CVs and non-CVs. In addition, CLOPS may have the capability to vary the ratio of CVs to the non-CVs as per the market penetration rate. This requires a non-uniform partitioning between SUMO and OMNET++.

C. Modes of Operation for IDCVS

One important feature of this conceptual IDCVS system is that it will have the option to simulate in two different modes—closed-loop simulation (CLSim) mode and HILS-mode. The CLSim mode will simulate without sensor data, in which case the entire simulation will be run within CLOPS. To incorporate both DSRC-equipped and non-equipped vehicles on CLSim scenario, we can randomly distribute the vehicles with OBU within the road network based on a user specified technology penetration rate. On the other hand, the HILS-mode will enable simulation based on real-time sensor data.

V. Implementation Approaches

In this section, we discuss the possible technical approaches to address the challenges pertaining to implementation of the integrated simulator.

A. Developing Network Partitioning Heuristic

A crucial challenge for the partitioning problem described in section III is that, due to the imbalance of computational resource requirements between transportation simulator (SUMO) and network simulator (OMNET++), a single partitioning scheme may not work for both the simulators. Apart from that, the number of vehicles will also vary among the two simulators where SUMO needs to simulate the traces for all vehicles (both CVs and non-CVs) whereas OMNET++ only simulates data traffic generated from the CVs. If a single partitioning heuristic is used, the synchronization problem will be further aggravated. Hence, it is necessary to have two separate partitioning schemes for SUMO and OMNET++.

In our recent work [30], we have identified the following issues and parameters that play vital roles in designing an efficient partitioning heuristic:
1) *System boundary nodes of each partition:* The total number of inter-process communication or messaging depends on the number of system boundary nodes of each partition.
2) *The number of partitions:* Almost every graph partitioning algorithm is based on a pre-specified number of partitions, which may not always generate the optimal solution in practice. Instead of specifying an exact number of partitions, an upper bound and lower bound can be provided as input to the algorithm to determine the best partitioning solution within the specified range.
3) *Intersection cut:* If an intersection is considered as a boundary node for a partition, then a significant amount of vehicle mobility data must be communicated between the partitions. In this context, an important factor—whether to prioritize signalized intersection over un-signalized intersection as a candidate for boundary node—remains open for further research, which should be investigated.
4) *Link/Edge cut:* When a link or edge is selected to be cut then the traffic volume along the cut link is directly proportional to the amount of information exchanged between the two partitions along the link. In this case, a good strategy would be to cut the links with minimum traffic to reduce the communication overhead between partitions.

B. Partitioning Approach for SUMO

To create the network graph, the OSM file of the experimental city can be downloaded from the www.openstreet.org website. To avoid unnecessary complexities, residential street, service path, footway, cycleway, motorway, and unclassified roads can be excluded from the graph. Table 1 shows some suggested parameters that can be incorporated to generate the weighted graph matrix. It could be easily possible to extend an existing graph partitioning software like METIS [37] for this generating the partitions of the transportation network. METIS is a very stable partitioning package implementing the popular Kernighan-Lin heuristic. METIS performs the partition of a graph in three phases: coarsening, partitioning, and uncoarsening. In coarsening phase, the heavy edge matching scheme can be used, whereas in the uncoarsening phase, the Kernighan-Lin graph refinement algorithm can be used. The coarsest graph can be bisected using graph growing followed by boundary Kernighan-Lin algorithm with graph partitioning using recursive bisection technique. The input for METIS can be provided using the generated graph matrix and weight parameters.

Table 1: List of parameters considered for partitioning heuristics

Parameter	Technique
Node weight	All signalized intersections in the OSM data will be identified. These types of intersections or nodes will be assigned a higher weight. Un-signalized intersections will be assigned the sum of the number of incoming and outgoing lanes as the weight.
Link length	The length between two nodes will be calculated using the Haversine formula: $d = 2r\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right)+\cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$ where, d=Distance between two points/nodes; r=Radius of Earth (6367 km); φ_1,φ_2=Latitude of point 1 and 2; λ_1,λ_2= Longitude of point 1 and 2
Number of lanes	The number of lanes of a road segment or a link will be extracted from the OSM data.
Link density	The density of a road segment or link will be extracted from the Google Map Application's newly introduced traffic layer [36]. The density is expressed in three categories: low, medium, and high.
Link priority	The road segment will be assigned a priority index based on the weighted summation of link length, the number of lanes, and link density.

C. Partitioning Approach for OMNET++

Some of the key factors concerning partitions for OMNET++ in the context of closed-loop parallel simulation for reduced interactivity and interdependence include vehicle mobility, communication events and external stimuli from the simulated transportation applications. These factors directly influence the previously mentioned challenges pertaining to network partitioning. The application stimuli are the drive for CV communications, which can be sporadic or proactive. The transportation network information such as the road network (road links, road nodes), car density on each link, and car speeds and distribution determine the vehicular mobility. This information can be further utilized to quantify the number of communications events. The approach to optimize partitions is to consider the number of discrete events in the communication network as the basis for drawing the boundary between the connected components. For example, one way to incorporate this approach is to employ the vehicle density and the length of each link as link weights in partitioning techniques (such as the minimum cut or minimum k-cut algorithms) to partition the network and minimize the interactivity between different portions. The lower the density and the longer the length of a link, the higher the possibility that the link is a cut link in the network. The rationale is that the density and length represent the continuity of the communication route on this link. Therefore, the lower this ratio (density/length), the less communication between the ends of the link. In addition, the partitions need to be adaptive to the dynamics of the application stimuli and the mobility. To address this issue, we can consider the simulation granularity and duration of the current partition time. The goal is to develop an intelligent algorithm to schedule the partitioning job.

D. Design of Closed-loop Parallel Simulator (CLOPS)

The closed-loop parallel simulator (CLOPS) integrates SUMO and OMNET++ as two standalone simulators. OMNET++ has the flexibility to dynamically create and delete nodes; this capability is necessary for a parallel simulation environment since the wireless vehicular nodes will be distributed in multiple network partitions based on geographic location. In addition, OMNET++ provides support for both distributed and shared memory computing which is needed for this project. The PHY and MAC layers of DSRC (IEEE 802.11p and IEEE 1609.4) have already been implemented in the OMNET++ platform by the open-source research community, which can be utilized in our research. This is a big advantage compared with OPNET since OPNET does not currently include the DSRC protocol stack.

CLOPS will incorporate hybrid parallelization schemes for both the traffic simulator and network simulator that will allow the integrated platform to run in parallel on clusters of computers within a supercomputing facility. The hybrid inter-process communication will be incorporated using both MPI and OpenMP. Figure 4 illustrates the envisioned architecture for parallelization. Both the transportation and communication simulators will have master controllers (the Transportation Simulation Controller (TSC) and Network Simulation Controller (NSC)) that will coordinate the computational load distribution among the parallel sub-processes. Each of these sub-processes is supposed to simulate a portion of the transportation network defined by the network partitioning. The controller will communicate with the sub-processes using MPI, while a transportation simulator sub-process corresponding to a specific partition communicates with its network simulator counterpart using OpenMP.

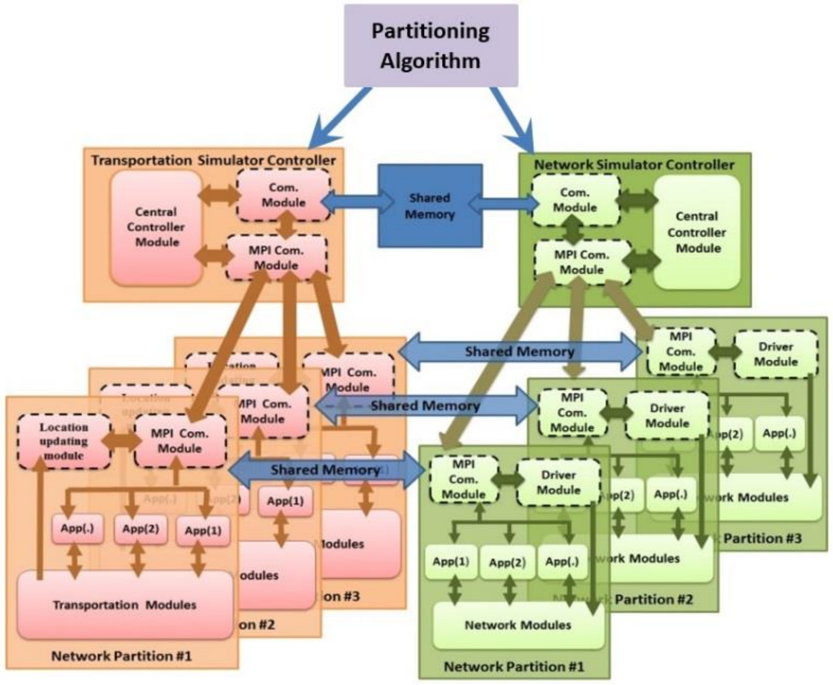


Figure 4. Envisioned distributed simulation architecture incorporating hybrid parallelism

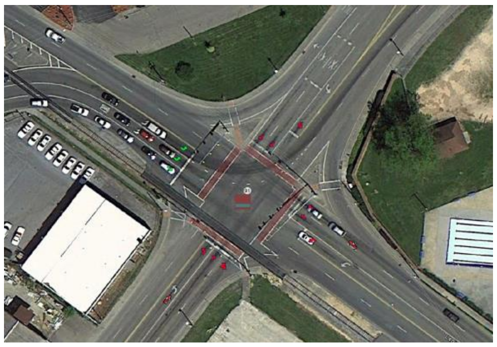
It could be beneficial to utilize two levels of parallelization: network and event levels. At the network level, the overall network can be divided into multiple partitions for both SUMO and OMNET++, each of which will run on a different machine. The TSC and NSC are responsible for managing the loads and synchronizing the partitions within the transportation and communication domains, respectively. At the event level, events can run in parallel within a pre-calculated look-ahead interval. The calculation of the optimum look-ahead interval is crucial in the event-level parallelization. In fact, the look-ahead interval involves a tradeoff between the simulation speed and output accuracy. In the event-level parallelization, utilizing parameters such as node locations and number of hops between two nodes can increase the scalability of the parallel simulation. For instance, nodes that are spatially separated by long distances can run events in parallel within longer look-ahead intervals without affecting the output.

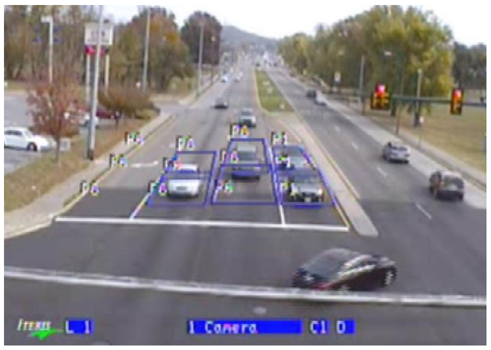
The communication between the TSC and NSC can be achieved by using shared memory. The vehicles' locations will be calculated and sent to the NSC periodically through the shared memory, and any required application information between the TSC and NSC will be exchanged through the shared memory. Compared to TCP/IP message passing, shared memory has the advantages of reliability and the highest possible speed of information exchange. In contrast, the message size in TCP/IP message passing is limited; thus, in the case of large network size, a large number of messages are needed for each location update. Consequently, TCP/IP message passing may create a communication bottleneck, resulting in the degradation of simulation speed.

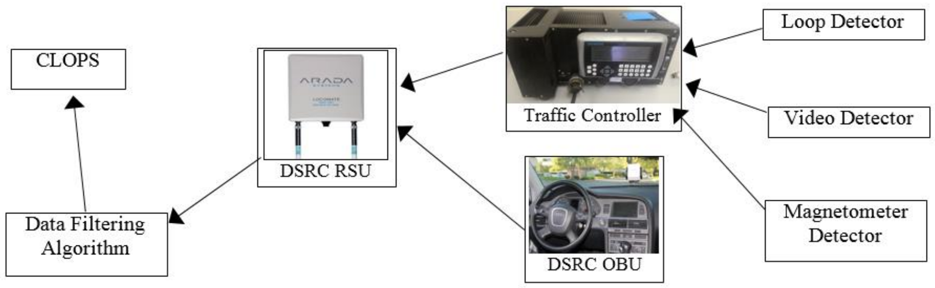
E. Incorporating Hardware-In-the-Loop Simulation (HILS)

To capture the movement of the non-CVs, several types of detectors can be used such as the magnetometer, inductive loop detection (ILD) and Video detection etc. Loop detection is also capable of counting traffic. But it is not 100% reliable for actual traffic counts because the loops in the adjacent through lanes are often tied together for one output for the movement. To solve this problem, the latest video detection technology capable of counting actual traffic can act as a complement for the loop detector. Figures 5(a) and 5(b) shows how the two software detect vehicles at the intersection through software.

Since the target is to simulate both CVs and non-CVs, it is necessary to feed the vehicles' information to the traffic and communication simulators. The RSU can automatically detect the CVs from the BSM packets, but the loop detection and video detection techniques are necessary for detecting the non-CVs. Once the RSU gets the data from all the sources (e.g. BSM packets, inductive loop, video, and magnetometer), a filtering algorithm separates the non-CVs from the CVs using the BSM packets. However, detection of the non-CV is not sufficient for the hardware in loop simulation. We need the mobility trace of a non-CV vehicle between two intersections. A car-following model between one/two CVs and a non-CV can be used to extrapolate the missing trace of a non-CV vehicle. For example, the missing mobility trace of a non-CV vehicle can be extrapolated using two CVs' mobility traces where one CV precedes the non-CV and one CV that follows the non-CV. Figure 5c shows the flow of sensor data for hardware-in-the-loop simulation.


(a) Loop detection software


(b) Video Detection software



(c) Flow of data between sensors

Figure 5. Integrating hardware-in-the-loop simulation techniques

Some of the challenges associated with integrating hardware-in-the-loop simulation are described below:

1) Isolating CV traces from loop-detection and video detection data

The hardware-in-the-loop simulation (HILS) technique can capture roadway sensor data from four different sources—DSRC broadcast messages, inductive loops, video detectors, and wireless magnetometers. Unfortunately, the roadway sensors at intersections cannot differentiate between a CV and non-CV. So, a reliable filtering mechanism is needed to identify the CVs among all the traffic by filtering out the CV data from other two sensors' data based on the GPS position and loop detection timings.

2) Missing traces between two intersections

It is very challenging to emulate non-CVs based on sensor data because of the missing traces between two intersections since they can only be detected at the intersections. Also, the SUMO generated mobility traces between two intersections are the only sources to fill up the missing traces. However, this approach may give some margin of error since some vehicles may arrive at their destination before reaching the next intersection while some other vehicles may start from a mid-point between the two intersections. Since the goal is to approximate the expected traffic between two intersections at a given time, some established statistical models are necessary to validate the simulation results between two intersections.

3) Inaccurate traffic count by loop detectors

Loop detection can detect traffic but is less reliable for actual traffic counts because the loops in the adjacent through lanes are often tied together for one output for the movement. Also, due to the length of the loop (40 to 50 feet) at the stop bar, multiple vehicles may be over the same loop or the loops tied together at the same time which reduces vehicle count accuracy.

4) Different data formats

Typically, data loggers' records include events at an intersection, including a light turning green, a light turning yellow, a vehicle detector turning on, a vehicle detector turning off, and pedestrian walk phase active. While CV data follows DSRC beacon format, loop detector, and video detector inputs are again in a different format. Thus, different pre-processing algorithms are needed.

CONCLUSION

In this paper, we have discussed a conceptual model that can simulate system-wide changes in traffic parameters on roadways involving both connected vehicles and regular vehicles. We have identified the major challenges and issues for implementing the hardware-in-the-loop simulation and incorporating parallelism in the closed-loop simulation. We have also discussed the solution approaches for the challenges and issues involved in implementing the conceptual model. However, only a few solutions have been actually implemented. We have discussed possible technical approaches to address the challenges and implementation issues. Our ongoing efforts are directed towards implementation of this model and evaluation of the scalability for emulating metro-wide transportation network.

REFERENCES

1. Fellendorf, Martin. "VISSIM: A microscopic simulation tool to evaluate actuated signal control including bus priority." 64th Institute of Transportation Engineers Annual Meeting. Springer, 1994.
2. Halati, Abolhassan, Henry Lieu, and Susan Walker. "CORSIM-corridor traffic simulation model." Traffic Congestion and Traffic Safety in the 21st Century: Challenges, Innovations, and Opportunities. 1997.
3. Krajzewicz D, Rossel C. Simulation of Urban MObility (SUMO). German Aerospace Centre, 2007. Available at: http://sumo.sourceforge.net. Last Accessed: 14 November 2016.
4. Sommer, C., R. German, and F. Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. IEEE Transactions on Mobile Computing, Vol. 10, No. 1, 2011, pp. 3-15.
5. Praprut Songchitruksa, Srinivasa Sunkari, Ines Ugalde, Juan Aparicio Ojea, and Justinian Rosca. Integrating Vissim and ns-3 for Connected/Automated Vehicle Simulation: A Case Study of Intelligent Dilemma Zone Avoidance, Proceedings of the ITS America Annual Meeting, 2016.
6. Ahmed Elbery, Hesham Rakha, Mustafa Y. ElNainay and Mohammad A. Hoque, VNetIntSim - An Integrated Simulation Platform to Model Transportation and Communication Networks. Proceedings of International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS), 2015.
7. Haerri J, Fiore M, Fethi F, Bonnet C. VanetMobiSim: generating realistic mobility patterns for VANETs. Institut Eurécom and Politecnico Di Torino, 2006. Available at: http://vanet.eurecom.fr/. Last Accessed: 14 November 2016.
8. MOVE (MObility model generator for VEhicular networks): Rapid Generation of Realistic Simulation for VANET, 2007. Available at: http://lens1.csie.ncku.edu.tw/MOVE/index.htm. Last Accessed: 14 November 2016.
9. STRAW - STreet RAndom Waypoint - vehiclar mobility model for network simulations (e.g., car networks), 2008. Available at: http://www.aqualab.cs.northwestern.edu/projects/144-straw-street-random-waypoint-vehicular-mobility-model-for-network-simulations-e-g-car-networks, Last Accessed: 14 November 2016.
10. FreeSim, 2008. Available at: http://www.freewaysimulator.com/
11. Martinez FJ, Cano JC, Calafate CT, Manzoni P. Citymob: a mobility model pattern generator for VANETs. In IEEE Vehicular Networks and Applications Workshop (Vehi-Mobi, held with ICC), Beijing, China, May 2008.
12. Ns-2 Network Simulator. http://nsnam.sourceforge.net/wiki/index.php/Main_Page , Accessed November 11, 2016.
13. ns-3 Network Simulator. https://www.nsnam.org/ , Accessed November 11, 2016.
14. A. Varga, "The OMNeT++ discrete event simulation system," Proceedings of the European simulation multiconference (ESM'2001), vol. 9, p. 65, 2001.
15. Riverbed Technologies, 2008. Available at: https://www.riverbed.com/products/steelcentral/opnet.html, Last Accessed: 14 November 2016.
16. JiST/SWANS: Java in Simulation Time/Scalable Wireless Ad hoc Network Simulator, 2004. Available at: http://jist.ece. cornell.edu/
17. Scalable Network Technologies. Qualnet. Scalable Network Technologies, Inc., 2006. Available at: http://web.scalable-networks.com/qualnet-network-simulator, Last Accessed: 14 November 2016.
18. Lee, J., and B. B. Park. Investigating Communications Performance for Automated Vehicle-Based Intersection Control under Connected Vehicle Environment. IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea, June 28 - July 1, 2015.
19. Mangharam R, Weller D, Rajkumar R, Mudalige P, Bai F. GrooveNet: A Hybrid Simulator for Vehicle-to-Vehicle Networks. Carnegie Mellon University, 2006. Available at: https://github.com/mlab-upenn/GrooveNet, Last Accessed: 14 November 2016.
20. R. Mangharam, D. S. Weller, R. Rajkumar, Priyantha Mudalige and Fan Bai, "GrooveNet: A Hybrid Simulator for Vehicle-to-Vehicle Networks", Second International Workshop on Vehicle-to-Vehicle Communications (V2VCOM), San Jose, USA. July 2006.
21. MobiREAL, 2008. Available at: http://www.mobireal.net/, Last Accessed: 14 November 2016.
22. The Georgia Tech Network Simulator (GTNetS), 2008. Available at:

http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/, Last Accessed: 14 November 2016.

23. Piorkowski, M., M. Raya, A. Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux. TraNS: Realistic Joint Traffic and Network Simulator for VANETs. ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 12, No. 1, 2008, pp. 31-33.
24. Lochert, C., A. Barthels, A. Cervantes, and M. Mauve. Multiple Simulator Interlinking Environment for IVC. 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2005), Cologne, Germany, September 2, 2005, pp. 87-88.
25. Rondinone, M., J. Maneros, D. Krajzewicz, R. Bauza, P. Cataldi, F. Hrizi, J. Gozlvez, V. Kumar, M. Rockl, L. Lin, O. Lazaro, J. Leguay, J. Haerri, S. Vaz, Y. L. M. Sepulcre, M. Wetterwald, R. Blokpoel, and F. Cartolano. iTETRIS: A Modular Simulation Platform for the Large Scale Evaluation of Cooperative ITS Applications. Simulation Modelling Practice and Theory, Vol. 34, Elsevier, 2013.
26. K. a. W. M. C. Ibrahim, "ASH: Application-aware SWANS with highway mobility," INFOCOM Workshops 2008, IEEE, pp. 1-6, 2008.
27. Y. a. D. G. a. B. P. Pigné, "A platform for realistic online vehicular network management," GLOBECOM Workshops (GC Wkshps), 2010 IEEE, pp. 595-599, 2010.
28. U.S. Department of Transportation, https://www.transportation.gov, Last Accessed: 13 November 2016.
29. W. Hao, R. M. Fujimoto, and G. Riley, "Experiences parallelizing a commercial network simulator," in *Simulation Conference, 2001. Proceedings of the Winter*, 2001, pp. 1353-1360 vol.2.
30. Ahmed, M.S & Hoque, M., "Partitioning of Urban Transportation Networks Utilizing Real-World Traffic Parameters for Distributed Simulation in SUMO," IEEE Vehicular Network Conference, Columbus, Ohio, 2016.
31. Ahmed, M.S & Hoque, M., "Partitioning of Urban Transportation Networks Using Evolutionary Algorithm for Distributed Simulation in SUMO," ACM-Mid Southeast Conference, Gatlinburg, Tennessee, 2016.
32. Ahmed, M.S.; Houser, J.; Hoque, M. & Pfeiffer, P., "Reducing Inter-process Communication Overhead in Parallel Sparse Matrix-Matrix Multiplication," ACM-Mid Southeast Conference, Gatlinburg, Tennessee, 2016.
33. Ahmed Elbary, Hesham Rakha, Mustafa ElNainay, Mohammad A Hoque, "An Integrated Architecture for Simulation and Modeling of Small- and Medium-Sized Transportation and Communication Networks", Smart Cities, Green Technologies, and Intelligent Transport Systems, Springer Lecture notes in Communications in Computer and Information Science, ISBN: 9783319277530
34. Ahmed Elbary, Hesham Rakha, Mustafa ElNainay, Mohammad A Hoque, "VNetIntSim: An Integrated Simulation Platform to Model Transportation and Communication Networks," International Conference on Vehicle Technology and Intelligent Transport Systems 2015.
35. Do Not Pass Warning Application. Available at: http://www.its.dot.gov/infographs/DoNotPass.htm (*Accessed Nov 13, 2016*)
36. https://productforums.google.com/forum/#!topic/maps/byNxu_lT0do
37. Karypis, George, and Vipin Kumar. "METIS--unstructured graph partitioning and sparse matrix ordering system, version 2.0." (1995).

CHAPTER 5

NETWORK PARTITIONING

Partitioning of Urban Transportation Networks Utilizing Real-World Traffic Parameters for Distributed Simulation in SUMO

Md Salman Ahmed, Mohammad A. Hoque
Department of Computing
East Tennessee State University
{ahmedm, hoquem}@etsu.edu

***Abstract*—This paper describes a partitioning algorithm for real-world transportation networks incorporating previously unaccounted parameters like signalized traffic intersection, road segment length, traffic density, number of lanes and inter-partition communication overhead due to the migration of vehicles from one partition to another. We also describe our hypothetical framework for distributed simulation of the partitioned road network on SUMO, where a master controller is currently under development using TraCI APIs and MPI library to coordinate the parallel simulation and synchronization between the sub-networks generated by our proposed algorithm.**

Keywords—OSM, Network Partition, METIS, SUMO, TraCI, MPI, Parallel Simulation

I. Introduction

For parallel network simulation, network partitioning is an effective method for speeding up the simulation process as well as maintaining the compatibility with machines with low resources that can run each partition. Since the simulation time and memory usage exponentially increase with the network size (number of vehicles and traffic volume), efficient network partitioning can greatly improve the scalability of parallel simulation. However, network partitioning is proven to be an NP-hard problem. Hence, an optimal partitioning may not be feasible. Practical partitioning heuristics are required that account for road networks and vehicular density and mobility to ensure an even division of the workload while minimizing communication between the partitioned elements.

While effective partitioning is crucial for speeding up the simulation of large-scale transportation network, this partitioning is very challenging due to many reasons. First, with connected vehicles emerging on the roads, the partitions could not be fully separated. In fact, due to communication and high mobility, partitions may have high level of interdependency and interactivity (i.e. a message or a vehicle moves from one partition to another) that demands communication between partitions to achieve consistency and accuracy. Second, inefficient partitioning of such networks can produce high communication volume between the different partitions, and imbalanced computing load in each partition, consequently results in low simulation speeds. So, it is necessary to create partitions in such a way that reduces the interactivity and interdependence between them. Thirdly, the synchronization and partitioning equity. Due to the interdependency between events in different partitions, simulation should be synchronized in all the partitions, i.e. low load (high speed) partitions must wait for high load (low speed) ones to finish. This means that the maximum overall simulation speed is limited to the minimum speed among all the partitions. Thus, the best speed is achieved when partitions have approximately equal loads. In fact, solutions for these three reasons may contradict one another i.e. creating independent partitions may result in huge load differences that can eventually degrade the speed. It is an optimization problem between a set of tradeoffs such as number of partitions, result accuracy, simulation speed, memory requirements etc.

In this context, the transportation network information such as road network (road links, road nodes), vehicle density on each link, vehicle speeds and distribution can be effectively utilized to optimize the partitioning techniques. For example, the vehicle density and the length of each links can be employed as link weight in partitioning techniques (such as minimum cut or minimum k-cut algorithms) to partition the network and minimize the interactivity between different portions. The lower the density and the longer length for a link, the higher the probability of being a cut link in the network. The rationale is that the density and length represents the continuity of the communication route on this link. Therefore, the lower this ratio (density/length) the lower the communication between the ends of the link.

Our current research contributions include the development of a novel partitioning algorithm for large scale urban transportation network incorporating previously unaccounted parameters like traffic volumes, signalized intersections, number of lanes, length of links etc. to balance the load for distributed simulation using SUMO. This would allow the large-scale evaluation of any innovative connected vehicle application or algorithm in a cluster-computing environment.

The rest of the paper is organized as follows. Section II describes the existing work on partitioning of transportation network. In section III, we identify the important parameters needed for partitioning. Section IV describes the actual steps involved in our proposed partitioning scheme with some preliminary results followed by a high-level overview of the work-in-progress distributed simulation platform on SUMO that can simulate individual partitions in parallel. Finally, we conclude with our future work leading to the development of distributed simulation platform enabling the simulation of large scale urban transportation network with connected vehicles.

II. Related Work

Many researchers have attempted to develop efficient partitioning schemes for large-scale transportation networks to simulate the scenarios in distributed environments using clusters. A well-designed partitioning scheme can greatly reduce the number of inter-process communication because vehicles frequently move from one partition to another in which case all the information corresponding to the mobility of those migrating vehicles need to be transferred to the new partition. Johnson *et. al.* [1] generated partitions using the shortest distance domain decomposition algorithm utilizing the standard label correcting technique with the objective of minimization of system boundary nodes to reduce inter partition communication cost. A significant amount of research effort has been dedicated for load-balancing among the partitions. For example, Meshkat *et. al.* [2] used genetic algorithm to divide a road network into two equally balanced partitions and repeated the process recursively to further divide the two generated partitions. Hyper-graph based partitioning algorithms using hmetis [5] have been discussed in [3] and [4], considering two-heuristics based hypothetical partitioning techniques. However, all the above partitioning techniques lack of the context of real transportation road networks—traffic density, number of lanes.

A complete road map for parallel road traffic simulator is discussed in [6] and [7]. In [6], the authors provide their own road network partitioning scheme and distributed version of SUMO. In [7], the transportation networks are partitioned by spatial decomposition [8] and simulated using JUTS, TRANSIMS, and AIMUSN. However, the former lacks of the parameters of actual road networks that affect the partition significantly. The later one considers only grid like road network. MOVES [9] also provides a distributed simulation platform on top of ARTIS simulation software. MOVES focuses on mainly the modularity and integrity of its layered software architecture, but does not focus on the real-world transportation networks and partitioning techniques. The distributed versions of SUMO are also discussed in [10] and [11]. The authors discuss about border edge management of a partition in [10] whereas in [11], the authors focus on the implementation of the distributed version in clusters. In both cases, the authors assumed that the network is already partitioned.

III. Issues for Road Network Partitioning

We have identified the following issues and parameters that are crucial for consideration while designing a heuristic for the partitioning of urban transportation network for parallel simulation.

A. *System boundary nodes of each partition*

System boundary nodes of a partition are responsible for communicating with other partitions to transfer and receive data and control of vehicles. The system boundary nodes pack several transfer requests and transfer the packed request to other partitions. So, the minimum number of system boundary nodes in a partition ensures the separation of responsibility and low communication cost.

B. *The number of partitions*

Deciding the number of partitions in the transportation network is a crucial factor for balancing the loads and minimizing the communication cost. Almost every graph partitioning algorithm determines the partitioning based on a pre-specified number of partitions which may not always generate the optimal solution in practice. Instead of specifying an exact number of partitions, an upper bound and lower bound can be provided as input to the algorithm to determine the best partitioning solution within the specified range.

C. *Load balancing*

As mentioned before, load balancing issue has been studied extensively for network partitioning since this directly impacts the overall simulation time. However, the metrics considered for load-balancing are not sufficient from the context of real transportation networks involving variable traffic densities and lane distributions. Hence, the weights for the nodes and links should be carefully assigned to address this issue.

D. *Intersection cut*

If an intersection is kept in a partition for the sake of one high density road and left all the links incident to the intersection in other partitions, it introduces a huge communication overhead. In other words, if an intersection is considered as a boundary node for a partition, then a significant amount of vehicle mobility data must be communicated to and from each partition that contains the intersection as a boundary node due to large number of vehicles migrating from one partition to another. In this context, an important factor—whether to prioritize signalized intersection over un-signalized intersection as a candidate for boundary node—remains open for further research.

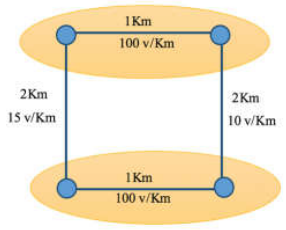


Fig. 1: Sample partitioning illustrating link cut minimizing inter-partition information exchange

E. *Link/Edge cut*

When a link or edge is selected to be cut then the traffic volume along the cut link is directly proportional to the amount of information exchanged between the two partitions along the link. In this case a good strategy would be to choose the links with minimum traffic for cut to reduce the communication overhead between partitions. For example, the road network in Fig. 1 shows four links with the lengths and average car densities. These two partitions have the minimal interaction between them due to the lower traffic densities (10 vehicles/km and 15 vehicles/km), thus their discrete simulation events can safely run in parallel.

IV. Proposed Partitioning Approach

Below we describe the steps involved in our partitioning scheme along with some preliminary results obtained for the road network of Johnson City, TN.

A. Creating graph

To create the graph, the OSM file of Johnson City, TN is downloaded from the openstreet.org website. A python script was written for extracting the intersections, road segments, traffic signals, and number of lanes. Since a road segment or a road is a combination of two or more nodes in OSM file, the degree of all nodes is calculated to find out the intersections. To keep the graph clean, many road types such as living street, service path, foot way, cycle-way, motorway and unclassified roads are excluded from the graph. The nodes that have only one degree (e.g. dead end) is also excluded from the graph. The Fig. 2 depicts the generated graph of Johnson City, TN where Google map API is used to overlay the graph vertices and edges.

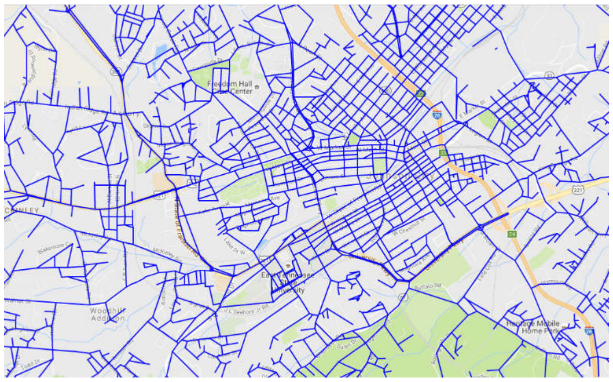


Fig. 2: Graph of Johnson City, TN is generated using OSM file and overlaid on the Google Map

B. Generating graph matrices

Each node or vertex of the generated graph has the latitude and longitude values along with a unique number identifier assigned by OSM file data structure. Along with the latitude, longitude, and the node identifier—an index ranged from 1 through $|V|$, where $|V|$ is the number of vertices in the graph—is assigned to each vertex. The length of the links between nodes are calculated using the Haversine formula that takes the latitude and longitude of two nodes and returns the distance between them. The following equation was used for calculation of link lengths based on Haversine formula.

$$d = 2r\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right)+\cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

where,
d=Distance between two points/nodes
r=Radius of Earth (6367 km)
φ_1=Latitude of point 1
φ_2=Latitude of point 2
λ_1= Longitude of point 1
λ_2= Longitude of point 2

Table 1: List of parameters considered for partitioning

Parameter Name	***Extraction Technique***
Node weight	All signalized and un-signalized intersections in the OSM data are identified using the above-mentioned python program. An un-signalized intersection is assigned a weight by multiplying its degree with the average of incoming and outgoing link densities. A signalized intersection is assigned a higher weight than un-signalized intersections.
Link length	The length between two nodes is calculated using the Haversine method.
Number of lanes	The number of lanes of a road segment or link is extracted from the OSM data.
Link density	The density of a road segment or link is extracted from the Google Map Application's newly introduced traffic layer [13]. The traffic volume is sampled in each of the 24 hours in a day and calculated the average density. For simplicity, the density is expressed in three categories: low, medium, and high.
Link priority	The road segment is assigned the summation of link length, the number of lanes, and link density as the priority.

The above table (Table 1) shows all the parameters that have been extracted from the OSM data to generate a weighted graph.

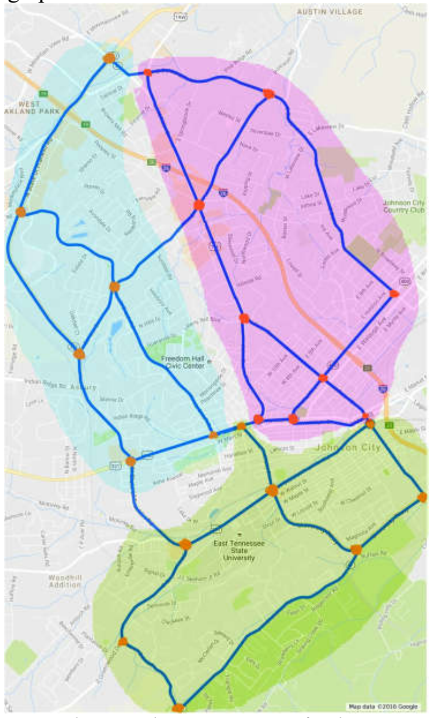


Fig. 3: Road network partitioning of Johnson City, TN

C. *Partitioning the graph by METIS*

Since METIS [5] is the most stable unstructured graph partitioning package, we partition the generated graph using METIS. The input data for METIS is provided using the generated graph and weight parameters. Since METIS only supports the node and link weight, a node and link weighted graph is generated. METIS performs the partition of a graph in three phases: the coarsening, partitioning, and uncoarsening phase. In coarsening phase, the heavy edge matching scheme is used, whereas in the uncoarsening phase, the Kernighan-Lin graph refinement algorithm is used. The coarsest graph is bisected using graph growing followed by boundary Kernighan-Lin algorithm with graph partitioning using recursive bisection technique.

Fig. 3 shows a sample partitioning of the road network of Johnson City, TN considering the parameters as described in the previous section. For simplicity, here we have only provided the multi-lane signalized corridors as the input road network to the METIS-based partitioning algorithm.

V. Distributed Simulation using SUMO

A distributed simulation platform on SUMO is currently under development that simulates each partition of the graph in a separate processor node. A master program is responsible for starting the simulation in all partitions and synchronizing the simulation results. The master program is written in C++ using the Traffic Control Interface (TraCI) [12] and Message Passing Interface (MPI) libraries. The communication and synchronization between processors are done using MPI. Each processor node has also the information of the complete graph information along with its own partition information. The SUMO input and configuration files are generated dynamically for each partition. The master program communicates and starts the SUMO simulator using TraCI which is packaged along with the SUMO source tree. SUMO simulator can be operated as a server. TraCI is performed as a middle-ware between the master program and SUMO where the TraCI is connected with SUMO as a client. Vehicles and routes are dynamically created by the master program and added to the SUMO simulator using TraCI. The routes are calculated from a source node to a destination node using the Dijkstra's algorithm. When a vehicle leaves a boundary node of a partition, the master program determines the next partition the vehicle will enter, removes the vehicle from current partition, and passes the whole vehicular dynamics of the vehicle to the entering partition. The master program also tracks the time needed to transfer the vehicle and its dynamics to the new partition.

VI. Conclusion

In this paper, we proposed our network partitioning approach for large-scale transportation network considering some important parameters like signalized traffic intersection, road segment length, traffic density, number of lanes and inter-partition communication overhead. Most of these factors were not accounted for in earlier work. We also discussed the critical issues involved in partitioning of a typical road network. Finally, we described our hypothetical framework for distributed simulation of the partitioned road network on SUMO, where a master controller is being developed using TraCI APIs and MPI library to coordinate the parallel simulation and synchronization between the partitions generated by our current algorithm. Our future work includes incorporating all the identified weight parameters in the graph partitioning technique by modifying the four algorithms) used in METIS (heavy edge matching, Kernighan-Lin graph refinement, graph growing followed by boundary Kernighan-Lin, and recursive bisection) to meet the needs of real-world transportation network.

References

[1] Johnson, Paul, Duc Nguyen, and ManWo Ng. "Large-scale network partitioning for decentralized traffic management and other transportation applications." Journal of Intelligent Transportation Systems (2016): 1-13.

[2] Meshkat, Amir, and J. L. M. Vrancken. "Multi-Objective Road Network Partitioning." Procedia-Social and Behavioral Science (2014).

[3] Xu, Yan, and Gary Tan. "An offline road network partitioning solution in distributed transportation simulation." In Distributed Simulation and Real Time Applications (DS-RT), 2012 IEEE/ACM 16th International Symposium on, pp. 210-217. IEEE, 2012.

[4] Etemadnia, Hamideh, and Khaled Abdelghany. "On the Network Partitioning of Large Urban Transportation Networks."

[5] Karypis, George, and Vipin Kumar. "METIS--unstructured graph partitioning and sparse matrix ordering system, version 2.0." (1995).

[6] Ventresque, Anthony, Quentin Bragard, Elvis S. Liu, Dawid Nowak, Liam Murphy, Georgios Theodoropoulos, and Qi Liu. "SParTSim: a space partitioning guided by road network for distributed traffic simulations." In Proceedings of the 2012 IEEE/ACM 16th international symposium on distributed simulation and real time applications, pp. 202-209. IEEE Computer Society, 2012.

[7] Potuzak, Tomas. "Distributed-parallel road traffic simulator for clusters of multi-core computers." In Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, pp. 195-201. IEEE Computer Society, 2012.

[8] Nagel, Kai, and Marcus Rickert. "Parallel implementation of the TRANSIMS micro-simulation." Parallel Computing 27, no. 12 (2001): 1611-1639.

[9] Bononi, Luciano, Marco Di Felice, Gabriele D'Angelo, Michele Bracuto, and Lorenzo Donatiello. "MoVES: A framework for parallel and distributed simulation of wireless vehicular ad hoc networks." Computer Networks 52, no. 1 (2008): 155-179.

[10] Acosta, Andrés, Jairo Espinosa, and Jorge Espinosa. "1 Distributed Simulation in SUMO Revisited: Strategies for Network Partitioning and Border Edges Management."

[11] Loolaid, Indrek. "Using SUMO in a distributed manner to generate road traffic data."

[12] Wegener, Axel, Michał Piórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. "TraCI: an interface for coupling road traffic and network simulators." In Proceedings of the 11th communications and networking simulation symposium, pp. 155-163. ACM, 2008.

[13] https://productforums.google.com/forum/#!topic/maps/byNxu_1T0do

CHAPTER 6

INVESTIGATION OF THE INTE-PROCESS COMMUNICATION OVERHEAD

Reducing Inter-Process Communication Overhead in Parallel Sparse Matrix-Matrix Multiplication

Md Salman Ahmed · Jennifer Houser · Mohammad A. Hoque · Rezaul Raju · Christopher Tymczak · Phil Pfeiffer

ABSTRACT

Parallel sparse matrix-matrix multiplication algorithms (PSpGEMM) spend most of their running time on inter-process communication. In the case of distributed matrix-matrix multiplications, much of this time is spent on interchanging the partial results that are needed to calculate the final product matrix. This overhead can be reduced with a one-dimensional distributed algorithm for parallel sparse matrix-matrix multiplication that uses a novel accumulation pattern based on the logarithmic complexity of the number of processors (i.e., $O(log(p))$ where p is the number of processors). This algorithm's MPI communication overhead and execution time were evaluated on an HPC cluster, using randomly generated sparse matrices with dimensions up to one million by one million. The results showed a reduction of inter-process communication overhead for matrices with larger dimensions compared to another one dimensional parallel algorithm that takes $O(p)$ run-time complexity for accumulating the results.

Keywords: MPI communication, communication overhead, parallel computing, performance analysis, scalability, sparse matrix-matrix multiplication.

INTRODUCTION

The widespread use and importance of matrix applications have created a compelling need for efficient algorithms for matrix-matrix multiplication. Matrix representations of real-world phenomena have numerous applications in science and technology, in fields that include electrical engineering, medical science, physics, quantum chemistry (VandeVondele et al., 2012), mathematics, and computer science. Matrix-matrix multiplication is indispensable for almost every research field that involves scientific computation and numerical methods like optimization, linear algebra, algebraic multigrid (Briggs et al., 2000), finite element analysis, and tensor contraction (Gilbert et al., 2008). In computer science, areas such as graphics, networking, wireless communication, video and audio analysis, image processing, graph theory (Dongen, 2008), big data analysis and language processing use matrix-matrix multiplication. Networks, for example, are commonly modeled with adjacency matrices: two-dimensional matrices whose elements represent connections and weights between a network's nodes. Repetitive multiplication of adjacency matrices can determine multi-hop reachability, transitive closure and dynamic partitioning within a mobile ad hoc network.

Researchers have worked for several decades to devise matrix-matrix multiplication algorithms that outperform the traditional, $O(n^3)$ algorithm. The need for such algorithms is driven by the processing of very large matrices, often with trillions of elements. Currently the fastest matrix-matrix multiplication algorithm, the Coppersmith-Winograd algorithm, has a run time complexity of $O(n^{2.375477})$(Williams, 2012). In computations involving matrices of larger dimensions, the main challenge for the matrix multiplication algorithm is a scarcity of computational resources. Increasingly, parallel processing is being used to address this challenge.

In one important special case, the nature of the data being processed creates particular opportunities for fast multiplication. Sparse matrices, or matrices whose elements consist largely of zeros, are commonly used to model real-world phenomena. Algorithms for sparse matrix-matrix multiplication improve on classic algorithms by focusing solely on products of nonzero elements. These algorithms' performance depends on factors that include the number and distribution of nonzero elements in the matrices to multiply, the structures used to store the matrices, the number of processors allocated to a computation, and the efficiency of inter-processor coordination. In particular, the use of efficient communication models and data structures can greatly speed up parallel multiplication.

Over the past few decades, researchers have extensively studied the Parallel Sparse Generalized Matrix-Matrix multiplication problem, hereafter referred to as PSpGEMM (Buluc et al., 2008). Numerous algorithms have been designed that apply a variety of distribution models, storage mechanisms, and communication models to PSpGEMM. These approaches have been incorporated into standard libraries and tools such as BLAS. Despite all these efforts, however, the impact of inter-process communication cost on the overall speedup and scalability has received relatively little attention. The scalability of any PSpGEMM algorithm depends largely on its strategy for inter-process communication, due to the amount of communication needed to exchange partial results between processors during the compilation of the final product matrix.

This paper describes a comparison of two one-dimensionally distributed PSpGEMM algorithms in terms of the impact of inter-process communication cost. The first one, developed previously by Hoque et al. (2015), uses an algorithm with $O(p)$ run-time complexity to accumulate partial results. It is hereafter referred to as the Original version of PSpGEMM, the other uses a novel $O(log(p))$ algorithm to accumulate results. This comparison focuses on how communication overhead, particularly MPI overhead, impacts these algorithms' performance, relative to randomly generated sparse matrices with dimensions up to one million by one million. These preliminary results indicate a significant reduction of inter-process communication overhead for matrices with larger dimensions compared to the Original PSpGEMM algorithm (Hoque et al., 2015). Section II reviews published communication models for PSpGEMM. Section III presents the algorithms' method of matrix-matrix multiplication. Section IV presents the details of the two algorithms (Original and Logarithmic) in terms of the communication patterns. Section V presents the results of performance analysis. Section VI concludes by summarizing these findings and discussing avenues for future work.

RELATER WORK

The scalability and performance of parallel SpGEMM algorithms are highly depended on inter-process communication, where most of these algorithms' execution time is spent. Most algorithmic designs, however, focus more on computation techniques rather than optimizing communications. Very few classical algorithms describe the communication cost of sparse matrix-matrix multiplication. Ballard et al. discussed a unified communication analysis of existing and new algorithms that provide an optimal lower bound for communication cost

(2013). In this paper, optimal communication costs of three 1D algorithms such as Naive Block Row (Buluc et al., 2008), Improved Block Row (Challacombe, 2000) and Outer Product (Kruskal et al., 1989) were outlined in terms of bandwidth costs and latency costs.

Ballard et al. (2012) described CAPS, a parallel, communication-optimal algorithm for matrix multiplication. Their algorithm seeks to efficiently balance the load among participating processors while minimizing interprocessor communication. It recasts Strassen's sequential algorithm as a recursive tree, dividing the multiplication algorithm into 7 subproblems, based on whether the dimensions of the matrices to multiply are large (unlimited memory scheme with BFS traversal) or small (limited memory scheme with DFS traversal).

Ballard et al. (2015) also described a hypergraph partitioning approach for parallel sparse matrix-matrix multiplication. They modeled SpGEMM using a hypergraph and reduced the communication cost by communicating between processors along with a critical path of the multiplication algorithm.

Utrera et al. (2015) discussed SpGEMM-related communication imbalances caused by the communication library and the interconnection network. The authors characterized this imbalance as a major source of performance degradation for sparse matrix-vector multiplication. They also analyzed their characterization using the fork-join and task based implementations and MPI protocols.

Most PSpGEMM algorithms assume that an efficient communication model is a natural consequence of an effective computation model. Only a very few papers describe the specific overhead due to the distribution and accumulation of partial results between processors: the source of most communication overhead. In what follows, the authors attempt to address the need for a better understanding of these overheads by providing a theoretical framework for an efficient partial results accumulation pattern; an implementation the pattern; and an analysis of the implementation's efficiency.

OUTER PRODUCT MATRIX MULTIPLICATION

Both algorithms studied use outer product matrix multiplication to solve $AB = C$, where A and B are sparse matrices of size $N \times N$. The authors assume that both A and B are symmetric matrices.

Both algorithms parallelize a serial method for matrix multiplication that begins by computing the outer product of A and B. This method takes the i^{th} column of matrix A and multiplies it by the j^{th} row of matrix B to produce a sub matrix C_i of dimension $N \times N$. This is continued such that each column of A and each row of B is multiplied together, which produces a total of N sub

matrices: $C_1, \ldots, C_N$. The resulting sub matrices are summed element-wise to produce the final result, matrix C, as shown in the following equation:

$$\sum_{i=1}^{N} C_i = C$$

In the following description of this algorithm's parallel implementations, the authors let p denote the total number of processors, N/p the number of rows or columns of the input matrix sent to each processor P_i and α the average number of nonzero elements in each row or column of an input matrix. Initially, the algorithms divide input matrices A and B into blocks of size N/p, distributing them over p processors. Each processor computes the outer product on its part of the matrix by multiplying each column in the block with each row in the block to produce a sub matrix C_i. The average number of non-zero elements in each row or column of a sub matrix C_i is at most α^2. Figure 1 illustrates the distribution of a matrix over four processors to produce four sub matrices.

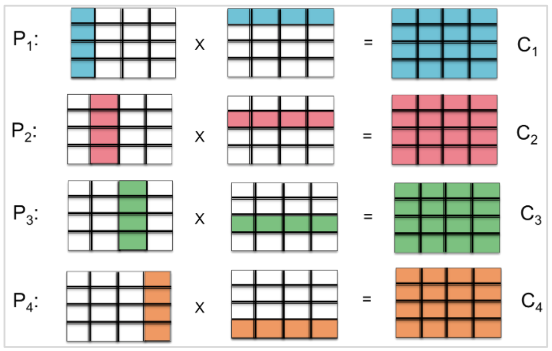


Figure 1: Distribution of an input matrix using outer product multiplication on four processors.

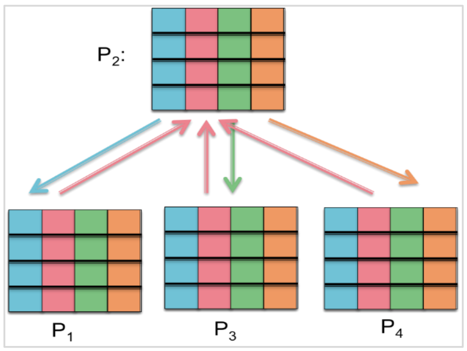


Figure 2: Merging results onto process two using four processes in total.

Once each processor computes the sub-matrix that contains its portion of the results, the partial results are merged through the sending and receiving of data to corresponding processors. This merging is done based on the patterns outlined in the next section. Because of the resulting matrix's size (on the order of 10^{12} elements for the largest input size 10^6), the final matrix C is left distributed over the processes.

IMPLEMENTATION OF PSPGEMM ALGORITHM

The authors present two versions of parallel sparse matrix-matrix multiplication algorithms with distinct merging scheme to illustrate a reduction in complexity created by a reduction in communication overhead. Both versions use the same storage mechanism and hashing techniques as described by Hoque et al. (2015). The algorithms differ only in a number of times data is sent and received between nodes during the merging of partial results that follows the computation of the sub-matrices. The authors also present the mandatory and auxiliary storage mechanism for the two algorithms to exchange data.

Original Merging Pattern

The first merging pattern accumulates its partial results as follows. After each sub-matrix is calculated, it is repartitioned into p column-wise blocks and then redistributed. Each process sends the i^{th} block of its sub matrix to the corresponding i^{th} processor to be merged with the partial results being received from the other processes. Figure 2 illustrates processor P_2 merging its results with the remaining three processors: processors P_1, P_3, and P_4 send partial results from their second block to P_2, and processor P_2 sends the partial results in the first, third, and fourth block to P_1, P_3, and P_4, respectively.

Based on the distribution process described in the outer product matrix multiplication section, if each processor receives $\left\lceil \frac{N}{p} \right\rceil$ columns upon the distribution of the input matrices, the total number of non-zero elements each process contains after computing its sub matrix C_i is equal to $\alpha^2 \left\lceil \frac{N}{p} \right\rceil$. Because each process exchanges data with $p - 1$ processes, every process communicates an average of $\frac{p-1}{p} \alpha^2 \left\lceil \frac{N}{p} \right\rceil$ elements. Accordingly, the amount of data that a process transfers to other processes using this communication pattern has complexity of $O\left(\frac{\alpha^2 N}{p}\right)$.

The total communication overhead is determined by the number of processes that send and receive data, the amount of data transferred, and delays created by the irregular distribution of non-zero elements throughout the input matrices and the resulting variation in the number of computations each process needs to calculate its portion of the partial result. Let the largest of these delays, the synchronization delay, be denoted by δ. The total communication overhead is then given as $(p - 1)\left(\left\lceil \frac{N}{p} \right\rceil + \delta\right)$.

Logarithmic Merging Pattern

In the proposed Logarithmic merging pattern, each process P_i sends its partial results to another process in $log(p)$ number of stages where p is the total number of processes involved in calculating the partial results. In each of these stages, the process P_i divides its total partial result matrix into two bins. The first bin contains the elements of the partial matrix whose column indexes are less than a mid-value. The second contains the elements whose column indexes are greater or equal to this mid-value. The mid-value is calculated in each stage for a particular computing process from the number of column-wise blocks per process. This calculation also determines a low index (l) and a high index (h), based on the number of processes (p) and a process's rank: a unique number assigned to each processor. These indices determine which bin to send and which to receive.

After dividing the partial result matrices into two bins, process P_i calculates the rank (r) of another process P_j with which to interchange bins. P_i then exchanges half of its partial results with P_j by sending one of the two bins and receiving the other.

Figure 3 illustrates the merging pattern for 8 processes where each process communicates with other processes in 3 (i.e., $log_2(8)$) stages. In each stage, a processor P_i determines another processor P_j to send to, along with the bin to send. For example, in the first stage P_1 sends its second bin to P_5, while P_5 sends its first bin to P_1. Each process P_i distributes half of the partial results to P_j and discards the contents of the bin that was sent while appending the contents that it receives to its other bin. For example, P_1 appends the contents received from P_5 to its first bin and removes the contents from its second bin. Similarly, P_5 appends the contents received from P_1 to its second bin and removes the contents from its first bin. The gray areas in Figure 3 indicate the removed contents.

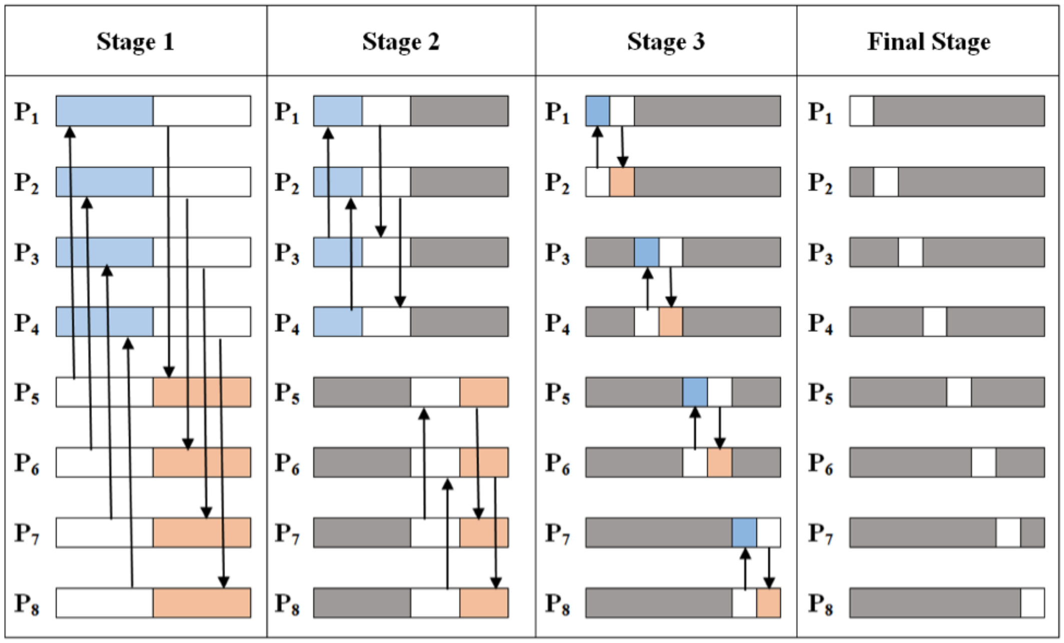


Figure 3: Logarithmic communication between processes.

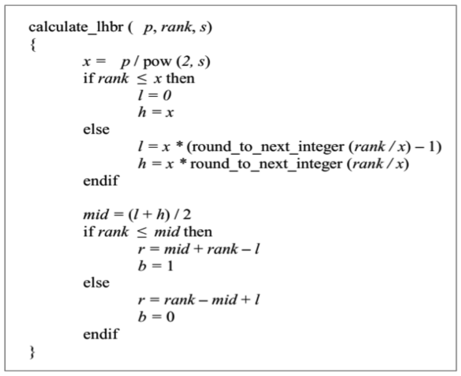
Since each process divides its partial results into two bins at each stage, a process creates a total of p bins after completing the $log(p)$ number of stages. In the final stage, each process contains partial results from each of the p processes including itself. For example,

- In stage 1, results are exchanged between process pairs P_1 and P_5; P_2 and P_6; P_3 and P_7; and P_4 and P_8. In this exchange, each process acquires one additional set of partial results, generated by the other. Following stage 1, processes pairs P_1 and P_5; P_2 and P_6; P_3 and P_7; and P_4 and P_8 share each others' results.
- In stage 2, results are exchanged between P_1 and P_3; P_2 and P_4; P_5 and P_7; and P_6 and P_8. In this exchange, each process acquires two additional sets of partial results: one set generated by the exchange's other process and a second this other process acquired during stage 1. Following stage 2, processes P_1, P_3, P_5, and P_7 share results, as do processes P_2, P_4, P_6, and P_8.
- In stage 3, results are exchanged between P_1 and P_2; P_3 and P_4; P_5 and P_6; and P_7 and P_8. In this exchange, each process acquires the remaining four sets of partial results. Following stage 3, all processes have one another's partial results.

At each stage, each process must determine a low value, a high value, the rank of another process with which to exchange data, and the bin (one of two) to send to the other process. Let

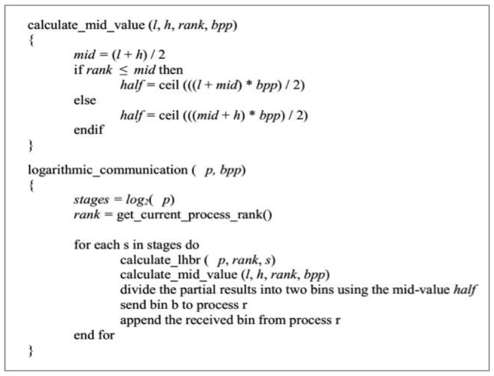
$rank$ = the computing process's rank
s = the current stage
bpp = number of column-wise blocks per process
$half$ = the mid-value for dividing the partial results

Each process then uses the algorithm from Figure 4 (left part) to calculate l, the process's low value for this stage; h, the process's high value for this stage; b, the index of the bin to send; and r, the other process's rank.



```
calculate_lhbr ( p, rank, s)
{
        x = p / pow (2, s)
        if rank ≤ x then
                l = 0
                h = x
        else
                l = x * (round_to_next_integer (rank / x) – 1)
                h = x * round_to_next_integer (rank / x)
        endif

        mid = (l + h) / 2
        if rank ≤ mid then
                r = mid + rank – l
                b = 1
        else
                r = rank – mid + l
                b = 0
        endif
}
```



```
calculate_mid_value (l, h, rank, bpp)
{
        mid = (l + h) / 2
        if rank ≤ mid then
                half = ceil (((l + mid) * bpp) / 2)
        else
                half = ceil (((mid + h) * bpp) / 2)
        endif
}

logarithmic_communication ( p, bpp)
{
        stages = log2( p)
        rank = get_current_process_rank()

        for each s in stages do
                calculate_lhbr ( p, rank, s)
                calculate_mid_value (l, h, rank, bpp)
                divide the partial results into two bins using the mid-value half
                send bin b to process r
                append the received bin from process r
        end for
}
```

Figure 4: Logarithmic merging algorithm.

Figure 4 (right part) shows the Logarithmic algorithm's procedure for managing overall inter-process communication. In this algorithm, the mid-value is calculated in order to divide the partial results into two bins.

Because the Original and Logarithmic algorithms implement identical methods for computing each matrix's partial results, each process's computations on each of its submatrices will average $\alpha^2 \left\lceil \frac{N}{p} \right\rceil$ operations resulting in $O\left(\alpha^2 \left\lceil \frac{N}{p} \right\rceil\right)$ complexity. Based on the merging schema in the proposed communication pattern, the partial results are accumulated in $log_2(p)$ stages where p is the number of processes. On each stage, any one process of the p processes transfers on average $\left(\frac{1}{p}\right) th$ of the total data, i.e., on average the amount is $\alpha^2 \left\lceil \frac{N}{p} \right\rceil$. Since the accumulation of partial results is done in $log_2(p)$ stages, the total amount of data transferred between processes is $log_2(p)\alpha^2 \left\lceil \frac{N}{p} \right\rceil$, which results in a complexity of $O\left(log_2(p)\alpha^2 \left\lceil \frac{N}{p} \right\rceil\right)$. Similarly, to the delay in communication caused by varying computation times between nodes, the inclusion of the synchronization delay between nodes causes the total overhead communication to have complexity of $O\left(log_2(p)\alpha^2 \left\lceil \frac{N}{p} \right\rceil + \delta\right)$.

DATA STRUCTURES

Storing just the non-zero data elements of a sparse matrix greatly reduces the amount of space that such matrices consume. The two algorithms use lists (e.g., vectors) to store a matrix's data elements. This list pairs each data element with its row and column index.

The matrices generated by the outer product computations are stored in a hash table. Each element's hash key is generated from its row and column indices as its hash key. Hash keys are uniform over the size of the hash table. Collisions resulting from the hashing of multiple elements to the same key are managed using external hashing: i.e., with a key-indexed linked list. Each hash table stores partial results as well as a portion of the final result in the end. In order to exchange a block of data with other processors, partial results must be copied from the hash table to a contiguous chunk of sequential memory (e.g., an array).

Figure 5: ETSU HPC Clusters.

PERFORMANCE ANALYSIS

The performance of the two PSpGEMM algorithms was analyzed on Knightrider, one of two high-performance computing clusters at East Tennessee State University's High-Performance Computing Center (Figure 5). Knightrider, which the university obtained in 2011, consists of 48 HP ProLiant BL280c G6 compute nodes and 1 HP DL380 G7 master node. The cluster totals 588 processors with 2.3 terabytes of memory, where each node contains a dual Xeon X5650 2.66 GHz processor, 12 cores, and 48 gigabytes of memory. The nodes are connected using a 4x QDR InfiniBand interconnect and Voltaire 36-port InfiniBand switches. The cluster hosts a total of 30 terabytes of central storage on its hard drives and 160 gigabytes of local storage on its compute nodes (High, 2007).

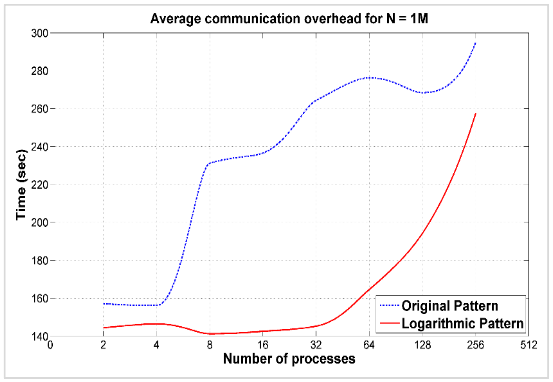


Figure 6: Average communication overhead for N = 1M.

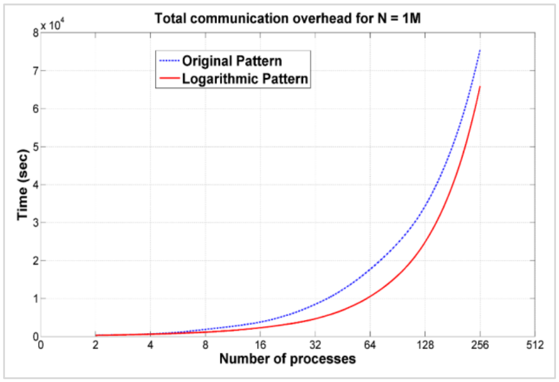


Figure 7: Total communication overhead for N = 1M.

Each of the PSpGEMM algorithms was evaluated in terms of its total execution time, total distributed computing time, average computation time per process, total MPI communication overhead, and average communication overhead per process. The experimental parameters that were varied include the input matrix's dimension (up to one million) and the number of computing processes (up to 256). The total number of processes excludes a separate, master process, which both algorithms use to load the input file into memory: only the computation nodes are included in the calculations.

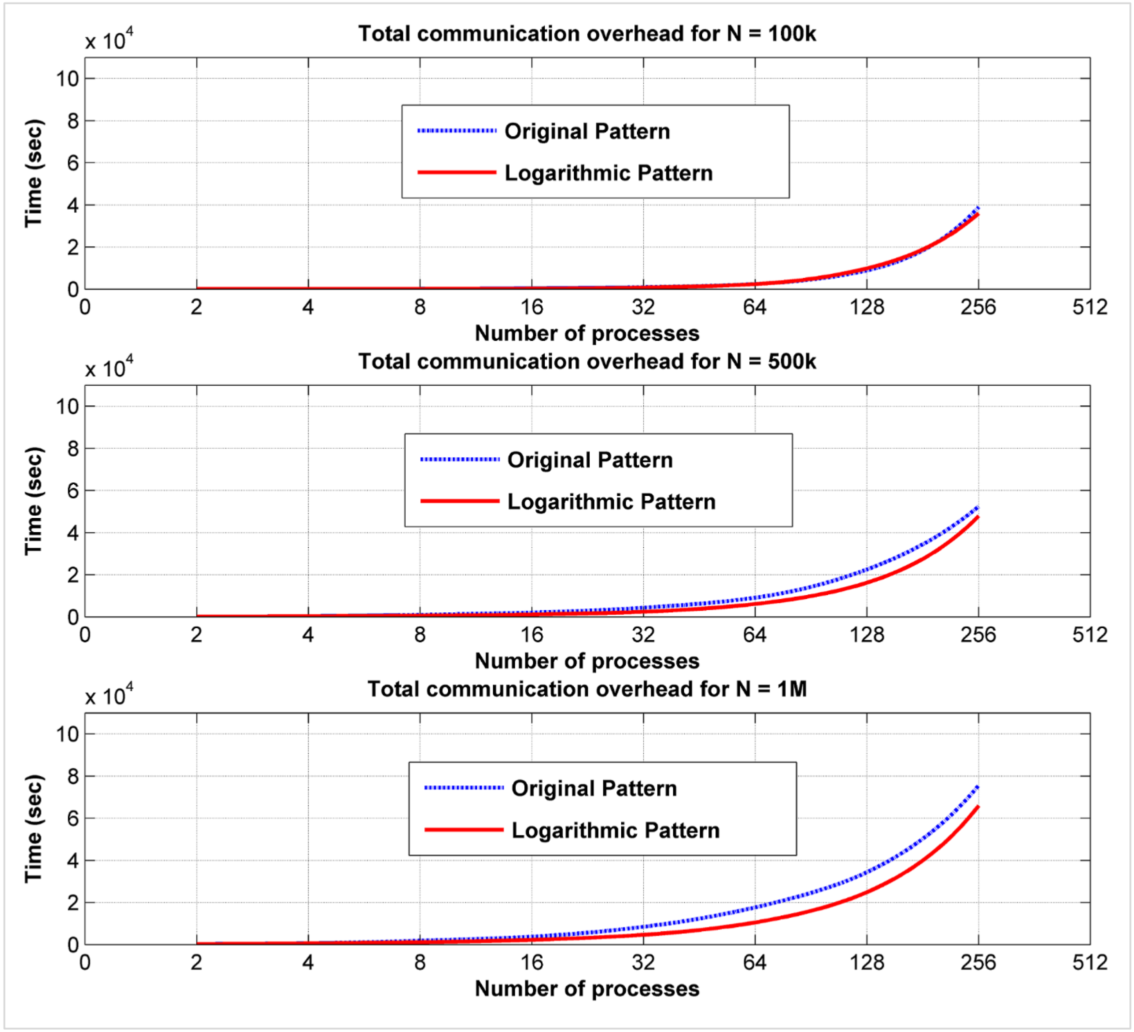


Figure 8: Total overhead communication for N = 100K, N = 500K, and N = 1M.

As indicated by Figures 6 and 7, the Logarithmic merging pattern reduces the average communication overhead and total communication overhead incurred by the Original merging pattern**.** Figure 8 shows that Original merging algorithm and the Logarithmic merging algorithm exhibit almost equal total overhead communication for input file $N = 100{,}000$. For the larger input sizes of $N = 500{,}000$ and $N = 1{,}000{,}000$, the proposed merging algorithm exhibits lower total overhead communication. This may suggest that the greatest benefits from the Logarithmic algorithm occur for larger matrices, which is precisely what the algorithm is designed for. Likewise, for the smallest input size, the Original merging pattern and the Logarithmic pattern achieved almost equal total execution time (Figure 9).

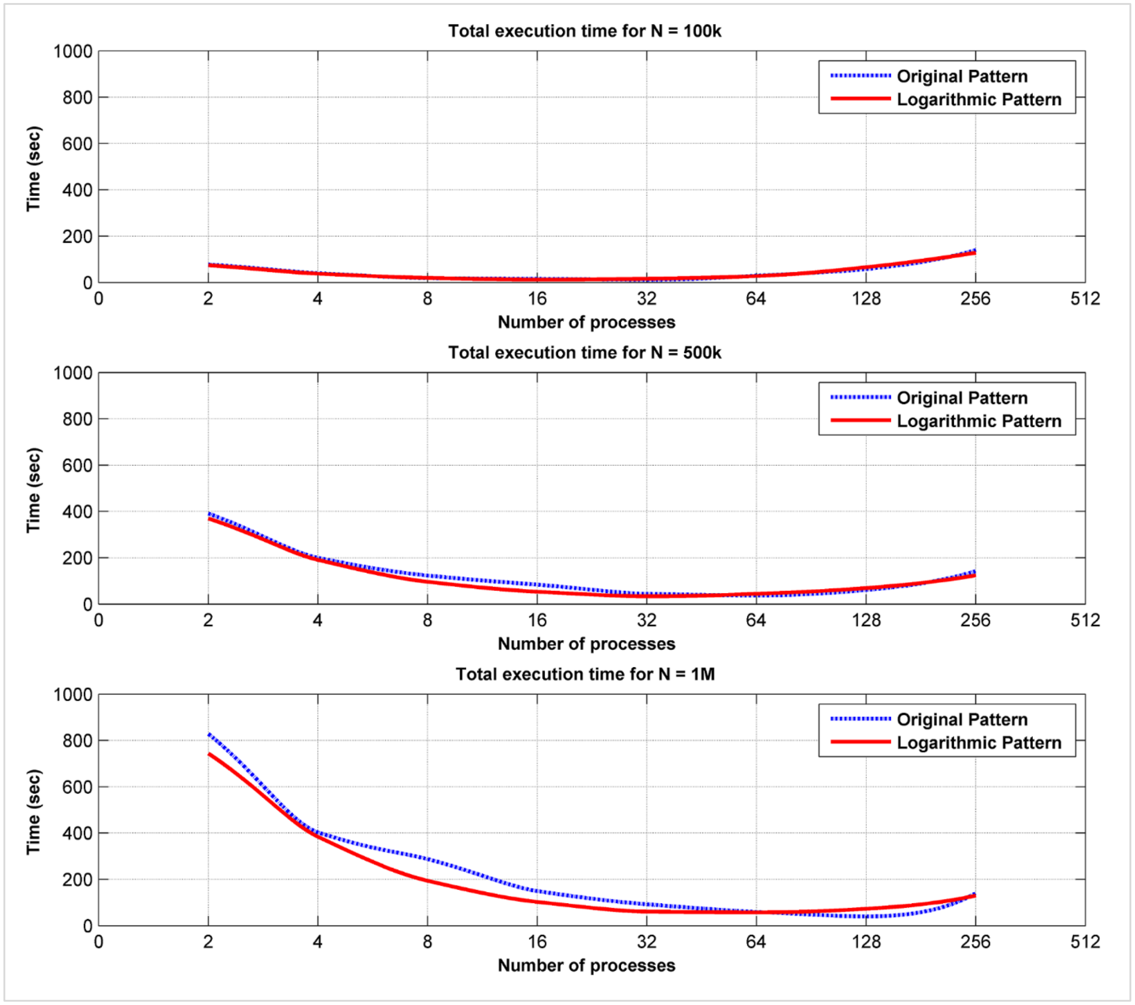


Figure 9: Total execution time for N = 100K, N = 500K, and N = 1M.

CONCLUSION AND FUTURE WORK

In this paper, the authors have explored two merging patterns for accumulating the partial results of sparse matrix-matrix multiplication in parallel. A theoretical framework and supporting implementation have been developed for a merging pattern where each node sends and receives half of its data in $log_2(p)$ iterations, resulting in total communication overhead of $O\left(log_2(p)\alpha^2\left\lceil\frac{N}{p}\right\rceil+\delta\right)$. Based on the performance on the high-performance computing cluster Knightrider, the data collected for three input sizes (100K, 500K, 1M) shows that the proposed

Logarithmic pattern, as predicted, incurs lower communication overhead, which in turn reduces the total execution time.

Several issues related to the algorithms' relative performance still need to be addressed. Currently, the Logarithmic merging algorithm assumes that the number of processors in use is an exact power of 2. This restriction will be removed in a forthcoming version of this algorithm, which will allow it to run on any number of processors. One particular issue of the Logarithmic merging pattern is its failure to yield as great of an improvement over the Original linear merging pattern as anticipated. Our analysis attributes this failure to the overhead incurred by copying data from a processor's hash table into a contiguous package for transmission. Our future study will focus more on the optimization of the data packaging overhead.

Another topic of particular interest is the Logarithmic algorithm's scalability. This can be assessed by running the algorithm at a more powerful facility like Oak Ridge National Lab (Oak, 1943) for a larger number of processors. Exploring the performances based on different sizes and implementations of the hash table and varying the sparsity and distribution of non-zero elements in the input matrices can help obtain additional information concerning the scalability and characteristics of the Logarithmic merging algorithm.

REFERENCES

Ballard, G., Buluc, A., Demmel, J., Grigori, L., Lipshitz, B., Schwartz, O., & Toledo, S. (2013, July). Communication optimal parallel multiplication of sparse random matrices. In Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures (pp. 222-231). ACM.

Ballard, G., Demmel, J., Holtz, O., Lipshitz, B., & Schwartz, O. (2012, June). Communication-optimal parallel algorithm for strassen's matrix multiplication. In Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures (pp. 193-204). ACM.

Ballard, G., Druinsky, A., Knight, N., & Schwartz, O. (2015, June). Brief announcement: Hypergraph partitioning for parallel sparse matrix-matrix multiplication. In Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures (pp. 86-88). ACM.

Briggs, W. (2000). van E. Henson, and S. McCormick. A Multigrid Tutorial.

Buluc, A., & Gilbert, J. R. (2008, September). Challenges and advances in parallel sparse matrix-matrix multiplication. In Parallel Processing, 2008. ICPP'08. 37th International Conference on (pp. 503-510). IEEE.

Challacombe, M. (2000). A general parallel sparse-blocked matrix multiply for linear scaling SCF theory. Computer physics communications, 128(1-2), 93-107.

Gilbert, J. R., Reinhardt, S., & Shah, V. B. (2008). A unified framework for numerical and combinatorial computing. Computing in Science & Engineering, 10(2).

High Performance Computing Center: East Tennessee State University (2007). Retrieved February 20, 2016, from http://www.etsu.edu/hpcc/.

Hoque, M. A., Raju, M. R. K., Tymczak, C. J., Vrinceanu, D., & Chilakamarri, K. (2015). Parallel sparse matrix-matrix multiplication: a scalable solution with 1D algorithm. International Journal of Computational Science and Engineering, 11(4), 391-401.

Kruskal, C. P., Rudolph, L., & Snir, M. (1989). Techniques for parallel manipulation of sparse matrices. Theoretical Computer Science, 64(2), 135-157.

Oak Ridge National Laboratory (1943). Retrieved February 20, 2016, from https://www.ornl.gov/.

Utrera, G., Gil, M., & Martorell, X. (2015, March). Evaluating the Performance Impact of Communication Imbalance in Sparse Matrix-Vector Multiplication. In Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on (pp. 321-328). IEEE.

VandeVondele, J., Borstnik, U., & Hutter, J. (2012). Linear scaling self-consistent field calculations with millions of atoms in the condensed phase. Journal of chemical theory and computation, 8(10), 3565-3573.

Van Dongen, S. (2008). Graph clustering via a discrete uncoupling process. SIAM Journal on Matrix Analysis and Applications, 30(1), 121-141.

Williams, V. V. (2012, May). Multiplying matrices faster than Coppersmith-Winograd. In Proceedings of the forty-fourth annual ACM symposium on Theory of computing (pp. 887-898). ACM.

CHAPTER 7
CONCLUSIONS AND FUTURE PLAN

Progress in CV technology has created opportunities for researchers and automakers to develop applications that provide vehicles with new safety, alert, and assistive features. Due to the ethical and practical infeasibility of conducting experiments on real transportation networks, these applications will need to be validated in laboratory settings before being deployed in real-world settings. If simulations are to find practical use in validating ITS applications, approaches like those described in this thesis will need to be devised for creating efficient parallel simulations of ITS applications in large-scale transportation networks.

To this end, the research described in this thesis developed a novel decentralized freeway merge assistance system. To the best of my knowledge, this is the first attempt to develop and evaluate a freeway merge assistance system using real-world vehicular mobility traces and an actual interstate. Though experiments demonstrate that the system can provide accurate advisory information for straight ramps, additional work will be needed to support merging on curved ramps.

Another important research issue of the freeway merge assistance system is driver compliance. Currently, the freeway merge assistance system assumes that every driver will comply with its advisories. In real world settings, drivers might ignore these advisories, which could have a major impact on the system's accuracy and performance. While a good advisory visualization could improve driver compliance, designing such a visualization system, will prove challenging. As an alternative, future versions of the merge assistance system will treat merging vehicles as semi-autonomous entities, triggering their cruise control mechanisms at the decision point and maintaining their current speeds until they complete the merge.

The research described a network partitioning strategy that extended METIS with complex node and edge weighting functions that account for a network's traffic parameters. Future research will focus on developing a customized version of METIS that uses customized versions of its four partitioning algorithms: heavy edge matching, Kernighan-Lin graph refinement, graph growing followed by boundary Kernighan-Lin, and recursive bisection.

This research explored two merging patterns for accumulating partial results to produce the final output using a sparse matrix-matrix multiplication. Though the logarithmic merging pattern performs better than the linear pattern in most experiments, the merging pattern still needs to address several issues related to the algorithms' relative performance. Currently, the logarithmic merging algorithm assumes that the number of processors in use is an exact power of 2. This restriction will be removed in a forthcoming version of this algorithms. Another topic of interest is the algorithm's scalability. This can be assessed by running the algorithm at a more powerful facility like Oak Ridge National Lab [20] for larger numbers of processors. Exploring performance based on different matrix sizes and processor numbers should yield better characterizations of the algorithm's scalability.

Finally, the research described in thesis represents an initial attempt to develop a complete feedback-loop based parallel simulator. Future work will include the actual implementation a parallel simulation framework using TraCI APIs where a master controller will manage the partitioning of transportation networks, simulating individual partitions, and synchronizing the partial simulation results.

REFERENCES

[1] G. Stoller. Road congestion wastes 1.9 billion gallons of gas. http://usatoday30.usatoday.com/money/industries/energy/story/2012-03-25/wasted-fuel-report/53776164/1, 2007. [Online; Retrieved: 2015-11-5].

[2] WHO. Global status report on road safety, 2015. http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/, 20. [Online; Retrieved: 2016-01-09].

[3] NHTSA. Traffic safety facts, crash and stats. washington, dc: Nhtsa's national center for statistics and analysis. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812332, 2007. [Online; Retrieved: 2016-01-09].

[4] OSADP. Federal highway administration of the u.s. department of transportation. http://www.itsforge.net/, 2017. [Online; Retrieved: 2017-01-10].

[5] LANE-CHANGE-1.2. Lane changing using adaptive cruise control. https://www.itsforge.net/index.php/community/explore-applications#/36/91. [Online; Retrieved: 2017-01-10].

[6] MMITSS-AZ. Signal phase and timing for emergency and transit vehicles. https://www.itsforge.net/index.php/community/explore-applications#/30/63. [Online; Retrieved: 2017-01-10].

[7] CaA-Speed-Harmonization-v1.0. Collision warning and avoidance. https://www.itsforge.net/index.php/community/explore-applications#/35/111. [Online; Retrieved: 2017-01-10].

[8] SPaT-1.2. Signal phase and timing through smart phone. https://www.itsforge.net/index.php/community/explore-applications#/30/76. [Online; Retrieved: 2017-01-10].

[9] TCSPT-v1.0. Traffic congestion information. https://www.itsforge.net/index.php/community/explore-applications#/30/117. [Online; Retrieved: 2017-01-10].

[10] RESCUME-CA-IASIM-1.0. Crash prevention. `https://www.itsforge.net/index.php/community/explore-applications#/36/45`. [Online; Retrieved: 2017-01-10].

[11] Ziyuan Wang, Lars Kulik, and Kotagiri Ramamohanarao. Proactive traffic merging strategies for sensor-enabled cars. In Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks, pages 39–48. ACM, 2007.

[12] Hyungjun Park, Chiranjivi Bhamidipati, and Brian Smith. Development and evaluation of enhanced intellidrive-enabled lane changing advisory algorithm to address freeway merge conflict. Transportation Research Record: Journal of the Transportation Research Board, (2243):146–157, 2011.

[13] Md Tanveer Hayat, Hyungjun Park, and Brian L Smith. Connected vehicle enabled freeway merge assistance system-field test: preliminary results of driver compliance to advisory. In Intelligent Vehicles Symposium Proceedings, 2014 IEEE, pages 1017–1022. IEEE, 2014.

[14] Yunpeng Wang, E Wenjuan, Wenzhong Tang, Daxin Tian, Guangquan Lu, and Guizhen Yu. Automated on-ramp merging control algorithm based on internet-connected vehicles. IET Intelligent Transport Systems, 7(4):371–379, 2013.

[15] Muhammad Alam, Muhammad Sher, and S Afaq Husain. Vanets mobility model entities and its impact. In Emerging Technologies, 2008. ICET 2008. 4th International Conference on, pages 132–137. IEEE, 2008.

[16] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation. ThinkMind, 2011.

[17] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[18] Hao Wu, Richard M Fujimoto, and George Riley. Experiences parallelizing a commercial

network simulator. In Simulation Conference, 2001. Proceedings of the Winter, volume 2, pages 1353–1360. IEEE, 2001.

[19] High Performance Computing Center. East tennessee state university. `http://www.etsu.edu/hpcc/`, 2012. [Online; Retrieved: 2016-02-20].

[20] Oak ridge national laboratory (ornl), 2007. URL `https://www.ornl.gov/`. Retrieved: Feb 20, 2016.

VITA

MD SALMAN AHMED

Education:	B.S. in Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, 2013
	M.S. in Computer and Information Sciences, East Tennessee State University, Johnson City, Tennessee, 2017
Professional Experience:	Graduate Teaching Assistant, East Tennessee State University, Department of Computing, August 2016 – Present
	Software Developer Intern, BancIntranets, Johnson City, Tennessee, May 2016 – August 2016
	Graduate Research Assistant, East Tennessee State University, Department of Computing, August 2015 – April 2016
	Senior Software Engineer, Samsung R&D Institute, Dhaka, Bangladesh, February 2015 – August 2015
	Software Engineer, Samsung R&D Institute, Dhaka, Bangladesh, March 2013 – January 2015
Publications:	Ahmed, M. S., Hoque, M. A., & Khattak, A. J. (2016, December). Demo: Real-time vehicle movement tracking on Android devices through Bluetooth communication with DSRC devices. In Vehicular Networking Conference (VNC), 2016 IEEE (pp. 1-2). IEEE.
	Ahmed, M. S., & Hoque, M. A. (2016, December). Partitioning of urban transportation networks utilizing real-world traffic parameters for distributed simulation in SUMO. In Vehicular Networking Conference (VNC), 2016 IEEE (pp. 1-4). IEEE.
	Ahmed, M. S., Hoque, M. A., & Pfeiffer, P. (2016, March). Comparative study of connected vehicle simulators. In SoutheastCon, 2016 (pp. 1-7). IEEE.

Honors and Awards: Bangladesh University of Engineering and Technology

Dean’s list for outstanding results in 4th year

Samsung R&D Institute

Icon of the Month award for March 2015

IEEEXtreme Programming Contest 10.0

Our team (Combucs) positioned 1st in Tennessee and 18th in the USA among over 2100+ worldwide teams

IEEEXtreme Programming Contest 9.0

Our team (Combucs) positioned 1st in Tennessee and 51st in the USA among over 2100+ worldwide teams

ACM-Mid Southeast Conference, Gatlinburg, TN

Won 3rd place in the graduate student presentation competition

East Tennessee State University, Johnson City, TN

Received Outstanding Computing Graduate Student award from the Department of Computing