


May 2021

A Simulation Framework for Traffic Safety with Connected Vehicles and V2X Technologies

MD ABU SAYED
University of Wisconsin-Milwaukee

Follow this and additional works at: <https://dc.uwm.edu/etd>

 Part of the [Civil Engineering Commons](#), [Computer Sciences Commons](#), and the [Geographic Information Sciences Commons](#)

Recommended Citation

SAYED, MD ABU, "A Simulation Framework for Traffic Safety with Connected Vehicles and V2X Technologies" (2021). *Theses and Dissertations*. 2724.
<https://dc.uwm.edu/etd/2724>

This Thesis is brought to you for free and open access by UWM Digital Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UWM Digital Commons. For more information, please contact scholarlycommunicationteam-group@uwm.edu.

A SIMULATION FRAMEWORK FOR TRAFFIC SAFETY WITH CONNECTED VEHICLES
AND V2X TECHNOLOGIES

by
Md Abu Sayed

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the Degree of

Master of Science
in Computer Science

at
the University of Wisconsin-Milwaukee
May 2021

ABSTRACT
A SIMULATION FRAMEWORK FOR TRAFFIC SAFETY WITH
CONNECTED VEHICLES AND V2X TECHNOLOGIES

by

Md Abu Sayed

The University of Wisconsin-Milwaukee, 2021
Under the supervision of Professor Tian Zhao

With the advancement in automobile technologies, existing research shows that connected vehicle (CV) technologies can provide better traffic safety through Surrogate Safety Measure (SSM). CV technologies involves two network systems: traffic network and wireless communication network. We found that the research in the wireless communication network for CV did not interact properly with the research in SSM in transportation network, and vice versa. Though various SSM has been proposed in previous studies, a few of them have been tested in simulation software in limited extent. On the other hand, A large body of researchers proposed various communication architecture for CV technologies to improve communication performance. However, none of them tested the advanced SSM in their proposed architecture. Hence, there exists a research gap between these two communities, possibly due to difference in research domain. In this study, we developed a V2X simulation framework using SUMO, OMNeT++ and Veins for the development and testing of various SSM algorithms in run time simulation. Our developed framework has three level of communication (CV to RSU To TS) system and is applicable for large traffic network that can have mixed traffic system (CV and non-CV), multiple road side unit (RSUs), and traffic server (TS). Moreover, the framework can be used to test SSM algorithms for other traffic networks without doing much modification. Our developed framework will be publicly available for its further development and optimization.

To my parent,
my wife,
and specially my daughter

TABLE OF CONTENTS

ABSTRACT.....	ii
LIST OF FIGURES	vi
LIST OF TABLES.....	vii
LIST OF ABBREVIATIONS.....	viii
ACKNOWLEDGEMENTS.....	ix
1. INTRODUCTION	1
2. LITERATURE REVIEW	4
2.1 Communication Framework in V2X Technology.....	4
2.2 Research on Surrogate Safety Measure.....	6
2.3 Simulation Software for Safety Measure	8
2.4 Research Gap and Contribution	14
3. SYSTEM ARCHITECTURE	18
3.1 Connected Vehicle (CV) and Non-Connected Vehicle (NCV)	19
3.2 Roadside Unit (RSU)	22
3.3 Traffic Server (TS).....	23
4. IMPLEMENTATION OF SSM ALGORITHM.....	24
4.1 Rear-End Conflict (REC) TTC	25
4.2 Crossing Conflict (CC) TTC.....	29
4.3 Time Exposed Time-to-Collision (TET) and Time Integrated Time-to-Collision (TIT).....	32
5. EXPERIMENTAL SETUP AND TESTING.....	34
5.1 SUMO Parameters and Simulation Files	34
5.2 OMNeT++ Parameters.....	35
5.3 Conceptual design.....	37
5.4 Simulation Results	39
5.5 Network Communication Performance.....	41
6. CONCLUSION.....	42
REFERENCES	44
APPENDICES	48
APPENDIX A: Parameter setup for RSU in NED file	48
APPENDIX B: Parameter setup for CV in Car NED file.....	49

APPENDIX C: Algorithm of calculating Rear-End TTC for CV in CVs module	50
APPENDIX D: Algorithm of calculating Crossing TTC for CV Part A in CVs Module.....	51
APPENDIX E: Algorithm of calculating Crossing TTC for CV Part B in CVs Module	52
APPENDIX F: Algorithm of calculating aggregated TTC in RSU module	53
APPENDIX G: Algorithm of responding crossing conflict TTC request from CVs in RSU module- Part A	54
APPENDIX H: Algorithm of responding crossing conflict TTC request from CVs in RSU module- Part B	55
APPENDIX I: Store TTC data received from RSU in TS module	56
APPENDIX J: Send TTC data to CVs via RSU in TS module.....	57

LIST OF FIGURES

Figure 1 V2X communication typology	19
Figure 2 Front vehicle speed calculation from CV	20
Figure 3 TTC for Rear-End Conflict (REC).....	25
Figure 4 Pseudocode to calculate Rear-End Conflict TTC.....	27
Figure 5 Pseudocode for storing TTC at RSU.....	28
Figure 6 Crossing Conflict points of a typical four-way intersection.....	29
Figure 7 Pseudocode for calculating CTTC at RSU.....	31
Figure 8 Conceptual diagram of TET and TIT.	32
Figure 9 V2X communication framework for CVs and RSU on the road network of Milwaukee downtown, Downtown (demonstration purpose only).....	37
Figure 10 Accident scenario and route change strategy	38
Figure 11 Average TTC of the CVs.....	39
Figure 12 Safety Index (TTC) of different segment of the roads and intersections.	40
Figure 13 TET of different segment of the roads and intersections.	40

LIST OF TABLES

Table 1 Comparison of traffic simulation programs (Pell et al., 2017).....	10
Table 2 Network architecture related CV research.....	14
Table 3 SSM related CV research.....	15
Table 4 CV parameters for this study	19
Table 5 TTC message parameter for CV	26
Table 6 CTTC message parameter for CV in the proposed framework	30
Table 7 Network parameters in SUMO	35
Table 8 Simulation parameters in OMNeT++	36

LIST OF ABBREVIATIONS

CAV	Connected vehicle
CAV	Connected autonomous vehicle
CC	Crossing conflict
CF	Communication framework
FHWA	Federal Highway Administration
NCV	Non-Connected Vehicle
REC	Rear-End conflict
RSU	Roadside Unit
SPMD	Safety Pilot Model Development
SSM	Surrogate Safety Measure
TET	Time Exposed Time-to-Collision
TIT	Time Integrated Time-to-Collision
V2X	Vehicle-to-Others
V2V	Vehicle-to-Vehicle
TS	Traffic Server
TTC	Time-to-Collision

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor professor Tian Zhao of the department of Computer Science at the University of Wisconsin-Milwaukee. Prof. Zhao always extended his supporting hand whenever I ran into a trouble or had a question about my research. He always allowed me to work independently giving me a feeling of my own work, but constantly showed me the right direction whenever he thought I needed it. Without his through guidance and cooperation, it was impossible for me to complete this research. I would like to thank professor Rohit J Kate (department of Computer Science at The University of Wisconsin-Milwaukee) for his constructive remarks and guidance in conducting this research.

Most importantly, none of this could have happened without professor Xiao Qin (department of Civil and Environmental Engineering at The University of Wisconsin-Milwaukee). Without his continuous support, encouragement, and guidance, I could not think of conducting this study.

I would also like to thank Dr. Sommer, the founder of Veins simulation framework, for his virtual support when needed.

Finally, I must express my gratefulness to my parents and to my wife for providing me with enduring support and continuous encouragement throughout my years of study. This accomplishment would not have been achievable without them.

Thank you.

Md Abu Sayed

1. INTRODUCTION

With the rapid development and growing research in connected vehicle technologies, vehicles with varying levels of communication are already on the road. The information exchanged between two communication devices; Vehicle to Vehicle (V2V), and Vehicle to road infrastructure(V2X), can be used to improve traffic operation and safety. By collecting vehicle's kinematics such as position and speed from connected vehicles, roadside unit (RSU) or Traffic unit (TF) can generate necessary surrogate safety measure (SSM) to send emergency warning messages (such as basic safety message, rear end crash, intersection crash) for any impending dangers. Through optimization and high computation power, the surrogate safety measure algorithms can generate accurate results instantly. However, to evaluate the applicability and impact of various SSM algorithms in real-time application, a communication framework (CF) within transportation network is necessary that can simulate real world scenario. The CF helps exchanging information between communication devices at different level of the network in real-time. In this case, simulation-based CF is more popular in connected vehicle research field because setting up a CF and transportation infrastructure in real-world is very costly. Hence, a robust and well-designed simulated CF is required to evaluate the SSM algorithm for real time application.

The connected vehicles use various visual sensors, such as cameras, radars, and lidars, to perceive their surroundings, make driving decisions and exchange information with others. Besides, wireless communication devices are exploited as extended sensors to exchange information with nearby vehicles and infrastructure through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. The V2X communication networks are IEEE 802.11p-based vehicular ad hoc networks (VANETs). It provides low latency in V2V

communication and has achieved success in active safety applications such as cooperative collision avoidance.

Vehicle connectivity provides various safety supports such as overtaking, autonomous driving, collision avoidance, and other traffic related predictions. However, exchanging information from vehicles to remote cloud centers poses long network transmission latency, which is not applicable for real-time data dissemination and content delivery. On the other hand, CV generates tremendous amount of data, which is not possible to process on board for safety application. To address this problem, several multi-stage communication frameworks has been proposed, where in the discussions of surrogate safety measures (SSMs) are more general and they do not test the applicability of their proposed network architectures for various SSM.

For the future vehicular safety and communication applications, a customized network of V2X-technology is applicable to design network services for vehicular networks. The most common network architectures are vehicle- to-base station (roadside unit)-to-remote server. The computing capability of the base station is more powerful than that of vehicles. It assigns various tasks to associated vehicles and collects sensing data from them, whereas the base station is responsible for ensuring the accuracy, completeness, timeliness, validity, and consistency of the sensing data. Connected vehicles will offload certain heavy computing tasks to the base server (Yuan et al., 2018; Zheng & Liu, 2017).

Analysis and development of SSM applications involves two distinct components of the system: the SSM application and the communication network. Usually, they are not traditionally studied together because of the differences in domain knowledge. There are currently no opensource tools or framework that allow their simulation and analysis in a unified framework.

As a result, the mutual effects of these components are not studied, and cannot be directly modeled using existing tools.

To fill this gap, we have developed an integrated communication and traffic network framework to study the interaction between SSM applications and V2X networking component. The frameworks help to test and evaluate various SSM algorithms under wireless the communication network. The proposed framework can be used for simultaneous simulation of various safety applications and communication networks in a single framework. we explained the challenges of the design of such an integrated framework in this study and presented some preliminary results that are obtained.

2. LITERATURE REVIEW

In existing traffic safety research, safety surrogate measures have been studied from various perspectives for use in V2X technology. In this literature review, we have studied how to calculate surrogate safety measure in real-time in V2V technology, how to improve network performance by implementing different network architectures, how to improve traditional surrogate safety measures, and how to use additional information with surrogate safety measure. The following section will give a detail review of the literature.

2.1 Communication Framework in V2X Technology

With the advent of advanced computing technologies, researchers have proposed various communication architectures for V2X networks and tested their effectiveness in real world application (Avino et al., 2019; L. Li et al., 2017; J. Liu et al., 2017; Malinverno et al., 2020; K. Wang et al., 2018; Yuan et al., 2018; K. Zhang et al., 2017). In the vehicular network, traffic events can be classified into two groups: non-emergency events (e.g., route planning) and emergency events (e.g., a collision). Considering the priority of events, researchers investigated how the distribution of computational tasks can improve network performance and reduce communication latency.

V2X technology involves large datasets, time constraints and location-related, and has extreme performance requirements. (Yuan et al., 2018) proposed a two layers communication framework in which they distributed the computational tasks between vehicles and edge server to exploit the locality of service and alleviate the overload of computation. The edge servers are

located at RSUs, which monitor vehicles and coordinate the message within the transmission ranges. On the other end, the edge server is connected to cloud server for data storage. The computation at the edge and content sharing through vehicular networks significantly reduced resource utilization and improved the quality of services. (J. Liu et al., 2017) proposed software defined network (SDN) architecture consisting of communication three. The edge server located between the RSU and the cloud server communicates wirelessly with the roadside unit to reduce the overall communication delay and process the data of the cloud server. The CVs are not allowed to communicate directly with the edge server. RSU processes any request from the CV, and then RSU redirects the request to the edge server to provide the CV's needs. (L. Li et al., 2017) proposed similar but a complex cooperative network architecture for V2X. They found that by distributing tasks between the RSU and the edge, network performance can be improved. The edge server is located within the range of radio signals of the RSU to provide services to the drivers and is connected to a core server through the internet on the cloud. The edge server is also responsible for providing access to software and application providers to integrate new service.

In summary, the edge server has more powerful computational power than the CVs, and can process data efficiently and accurately. A connected vehicle will send data to the edge server to perform certain computing tasks, and the edge server will respond to the vehicle and send the data to the cloud server. The research proposed in the literature on V2X technology pays more attention to the size of packets (data/message), packet loss and communication delay. The SSM application has not been implemented or tested yet.

2.2 Research on Surrogate Safety Measure

In the past few decades, a lot of traffic safety research on surrogate safety measures (SSM) has been conducted. Among various Surrogate safety measures, Time -to Collision(TTC) is the most utilized safety surrogate which refers to “the time that remains until a collision between two vehicles would have occurred if the collision course and speed difference are maintained (Tarek & Sany, 2007). A collision is defined as an observable situation in which two or more road users are close to each other in time and space, and if their movements remain unchanged, there is a risk of collision. In our previous study, we investigated traffic safety by developing safety index for TTC, MTTC, DRAC for trip and link level using SPMD (safety pilot model development) data of real world connected vehicle (He et al., 2018). Connected vehicles have vision and LIDAR-based technologies that can be used to calculate the distance of the vehicle in front. In SPMD dataset, the speed and acceleration of the front vehicle was generated based on change of distance for each time stamp. (Laureshyn et al., 2010) provided a theoretical framework for assessing traffic safety through surrogate safety measure. They used video data to extract driver behaviors and evaluated existing safety measures such as TTC, TimeGap and TimeAdvantage for all types of approach angles. (Gettman & Head, 2003) proposed that time to collision (TTC), post encroachment time, deceleration rate, maximum speed and speed differential are the best surrogate of traffic safety. They suggested using TTC as the main severity measure. (Saunier & Sayed, n.d.) proposed an advanced surrogate safety measure at intersection that provides a probabilistic function of conflicts and can dynamically update the probability of conflict.

Surrogate Safety Measure (SSM) is one of the most widely used methods for identifying traffic conflict in V2X technology. Depending on the nature of data and available tools, various models have been proposed to calculate SSM to provide safety message to the road users in the literature. For example, (Keivani et al., 2019) proposed a conceptual model for a vision-based driver assistance system based on collaborative edge computing. The model monitors the vehicle driving in front of an ego car and notify the driver upon about any impending dangerous incidents. Their research is based on 100 simulated cars that follow the poisson distribution and use OMNET++ as a network simulator. (H. Liu et al., 2017) used MATLAB and VISSIM to determine the best parameter set of the V2V technology ADAS system in the simulation data. They focused on finding optimal value of perception reaction time (PRT) and driver headway (DH) for the application of forward collision warning (FCW). In their study, alarm is triggered once the subject driver's headway or time-to-collision (TTC) to the leading vehicle is lower than the prespecified critical headway or TTC threshold in a congested driving environment. (Gelbal et al., 2019) implemented Cooperative Collision Avoidance (CCA) for vehicles that communicate with other vehicles. They simulated four distinct collision risk on a Connected Autonomous Vehicle (CAV) Hardware-in-the-loop (HIL) simulator to test their algorithms in real-time using real electronic control and communication hardware. They applied the elastic band method that generates a modified path to avoid collisions. When multiple vehicles arrive at intersection, a dangerous zone will be created, and the vehicles will pass the intersection one by one. They used CarSim PC, dSpace SCALEXIO real-time HIL simulation computer that runs Simulink/CarSim model, an in-vehicle PC for running the cooperative collision avoidance algorithms, a laptop for monitoring and two DSRC on board unit modems for real V2V communication between the ego vehicle and the surrounding vehicles. (Y. Wu et al., 2019)

developed a warning system algorithm for drivers to avoid vehicle–bicycle crashes in the bike lane area in a connected vehicle environment. They used field data of 118 right-turning vehicle trajectories that were collected by an unmanned aerial vehicle to capture right-turning behaviors of the drivers. Their algorithm calculates post-encroachment time (PET) in different situations. When the calculated PET value is below the threshold (1.5 seconds), it generates a warning. The speed value is fixed for vehicles and bicycles. The position of right turning vehicle was calculated using OpenCV library in C++.

2.3 Simulation Software for Safety Measure

Field data are costly, time consuming and uncertain to find conflicts. Although naturalistic driving data can help develop safety surrogates, the data collection process is very time-consuming (Tarko et al., 2009; K.-F. Wu & Jovanis, 2012). Compared with traditional field observation, traffic simulation significantly reduces the workload of data collection and improves the accuracy of data in terms of eliminating human error. Traffic simulation models can provide numerical solution for traffic safety related problem. The integration of SSM and Traffic simulator provides an opportunity to study different scenarios of road conflicts and quickly evaluate possible solutions. Traffic simulators mostly generate normal and safe vehicles that hinders the development of surrogate measures. although several surrogate measures have been developed based on traffic simulation, most measure are only designed based on traditional non-intelligent vehicles. Despite these facts, it is still necessary to develop safety measures to provide traffic engineers and policy makers with useful and important information about their potential safety impacts.

In recent years, the rapid development in computational technology has improved the applicability of various commercial and open source simulation software in traffic conflict research. (Tarek & Sany, 2007) made a comparison of various commercial simulation software and presented the weakness and strength of those software. It is found that almost every software needs some modifications to provide some extent of safety measure through built-in functions or the use of external tools. (Pell et al., 2017) provided a detailed comparison of 17 traffic simulation software. They found that most simulation software can handle real-time data, but cannot provide all functions.

Table 1 Comparison of traffic simulation programs (Pell et al., 2017).

	Aimsun 7	Cube Voyager/Avenue	DRACULA	DynaMIT	FLEXYT-II	FreeSim	Integration	MITSIMLab	PELOPS	PLANSIM-T	Quadstone Paramics	S-Paramics	SITRA-B+	SUMO	TransModeler	TSIS-CORSIM	VISSIM
Key:																	
O..... No																	
X..... Yes																	
!..... (further)																	
improvement																	
[empty] not given																	
Flow model category																	
macroscopic model	X	O	O			X		O	O		O	O		O	X	O	O
mesoscopic model	X	X	O	X		O		O	O		O	O		X	X	O	O
microscopic model	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Model size restrictions																	
limit of crossings		X	O			O			X!		X	O	X	O	O	X	
limit of links		X	O			O			O		X	O	X	O	O	O	
limit of edges			O			O						O		O	O		
limit of street categories	X!		O			O			X!		X	O		O	O		
limit of lanes	X!		O			O			X		X	O		O	O	X	
limit of vehicle types		X	X			O			X		X	O		O	O		
limit of driver profiles			O			O			O		X	O		O	O		
limit of public transportation routes			O			O					X	O		O	O	X	
ITS functionalities																	
co-ordinated traffic signals	X		X		X	O	X	X	X!	X	O!	X	X	X!	X!	X	X
adaptive traffic signals	X	X	X		X	O	X	X	X!	X	X!	X	X	X!	X!	X	X

Table 2 Comparison of traffic simulation programs (Pell et al., 2017).

Key: O..... No X..... Yes !..... (further) improvement [empty] not given	Aimsun 7	Cube Avenue	DRACULA	DynaMIT	FLEXYT-II	FreeSim	Integration	MITSIMLab	PELOPS	PLANSIM-T	Quadstone Parameters	SITRA-B+	S-Parameters	SUMO	TransModeler	TSIS-CORSIM	VISSIM
	public transport priority	X	X	X		X	O	X	X	O	X	X	X	X	O!	X	X
ramp metering					X	O	X	X	X	X	X	X		O!	X	X	X
freeway flow control	X			X	X	X	X	X	X		X	X		O	X	X	X
adaptive cruise control						O			X!			O		O		X	
automated highway system						X			X!	X		O		O			
autonomous vehicles						O			X!			O		O			
v2v/v2i communication						X			X!		X!	O		X!	X!		X
automatic debiting & toll plazas					X	O	X		O		X	X		O	X	X	X
zone access control					X	O			O	X	X	X		X!	X!		
incident management	X		X	X	X	O	X		O		X	X	X	O	X	X	
variable message signs	X		X	X		O	X	X	X!	X	X	X		X!	X!		
static route guidance						X	X	X	X!	X	X	X	X	X!	X		X
dynamic route guidance	X	X		X		X	X	X	O!	X	X	X	X	X!	X		X
vehicle type specific barred turningmovement and link/lane closures						O			O		X	X		X!	X	X	
multimodal traffic	X	X				O		X	O		X	X	X	X!	X		
parking guidance						O			O	X	X	X	X	O!	X		X
public transport information						O	X		O	X	X	O		O	O		X
probe vehicles						X	X		X!	X	X	O	X	X!	X		X
street restrictions																	
speed limits	X	X	X			X		X	X			X		X	X	X	
weight	X					O			O			X		O	O		
vehicle height	X					O			O			X		O	O		
vehicle width	X					O		X	O			X		O	O		
vehicle type specific lane use (e.g. bus lane)	X											X		X	X	X	
Modelled objects and phenomena																	
cars	X!	X	X	X	X	X!	X	X	X!	X	X	X	X	X!	X	X	X
commercial vehicles/trucks		X	X		X	X!	X	X	X!	X	X		X	X!	X	X	X
(motor) cyclists		X			X	O			X!				O	X!	X		X
pedestrians		X	X		X	O			O!		X		O	X!	X	X	X
public transport vehicles on road	X	X	X		X	X!	X	X	O	X	X	X	X	X!	X	X	X
trains and streetcars/trams	X	X	O			O		O				X	X!	X	X	X	
abnormal loads/vehicles	X					O		O		X		X	O!	O			

Table 3 Comparison of traffic simulation programs (Pell et al., 2017).

	Aimsun 7	Cube Voyager/Avenue	DRACULA	DynaMIT	FLEXYT-II	FreeSim	Integration	MITSIMLab	PELOPS	PLANSIM-T	Quadstone Paramics	S-Paramics	SITRA-B+	SUMO	TransModeler	TSIS-CORSIM	VISSIM
Key: O..... No X..... Yes !..... (further) improvement [empty] not given																	
hazardous materials transportation	X					O		O				O	O	O			
parking vehicles		X	X			O	X	O			X	O	O!	X	X	X	
searching for parking space						O			O		X	X	X	O!	X	X	X
weather conditions			X	X		X!		X	X				O	O!			
traffic calming measures			X		X	X!	X	X	O		X		X	X!	X		X
congestion	X	X	X			X!			X!	X	X		X	X!	X	X	
queue length	X	X	X	X	X	O		X	O		X	X	X	X!	X	X	X
variable travel times	X		X	X	X	X!		X	O		X	X	X	X!	X	X	X
overtaking on dual carriageway roads			X			O		X	X!		X		X	X!	X	X	X
overtaking on single carriageway roads			X			O			O!				X	O!	X		
predictable incidents (e.g. roadworks)	X		X	X	X	X!	X		O		X	X	X	X!	X	X	X
incidents random-in-nature (e.g. accidents)			X	X		X!	X		X!		X		X	O!	O	X	
roundabouts		X	X		X	O	X	X	O	X	X	X	X	X!	X	X	X
queue spill back				X	X	O	X	X	O	X	X	X	X	X!	X	X	X
steep grades		X				O		X	X		X		X	X!	X	X	
cornering ability at blind corners						O			X				O	O!			
driveability at blind bend						O			X		X		O	O!			
mix of users/variety of driver profiles		X	X	X		O		X	X!		X		X	X!	X	X	
real-time data integration and analysis																	
roadside devices	X		X	X	X	X	X	X	X!	X	X!	X	X	X!	X	X	X
(x)FCD/FVD/Probe data						X!			X!		X!		O	X!			
V2I						X!		X	X!		X!		O	X!	X		
others																	

(Ejercito & Nebrija, 2009) conducted a comparison study on Matsim¹, SUMO², Aimsun³ and PTV Vissim⁴ based on open source and free use, operating system portability, creating traffic networks and associated vehicle patterns, quality of GUI and documentation, simulation output (data and files), ability to simulate very large traffic networks, additional capabilities, and CPU and Memory performance. It was found that SUMO, Aimsun, and Vissim can generate large traffic network, and have common and unique features, with different strengths and weaknesses. Hence, it is challenging to select the best simulation software from the available source. Commercial software is very costly and has many hidden algorithms that are difficult or impossible to modify to some extent. On the other hand, most open source software is not readily available to public, and they need some levels of programming. One of the great advantage of open source software is that their source code is available and there are many online user communities that support technical issues. Therefore, it is easier to extend an open source software for custom application development and validation.

SUMO (Simulation of Urban Mobility) is an open source, highly portable, microscopic multi-modal traffic simulation, which is designed to handle large traffic networks. It has been widely used in many real-world projects to answer a large number of research questions (Lopez et al., 2018). It provides TraCI⁵ API that provide great flexibility to control traffic and develop custom algorithms for real-time simulation testing. Recently, SUMO has been integrated with OMNeT++⁶ (a network simulator) through VEINS⁷ to generate real world connected vehicle

¹ <https://matsim.org/>

² <https://www.eclipse.org/sumo/>

³ <https://www.aimsun.com/>

⁴ [Traffic Simulation Software | PTV Vissim | PTV Group](#)

⁵ <https://sumo.dlr.de/docs/TraCI.html>

⁶ <https://omnetpp.org/>

⁷ <https://veins.car2x.org/>

scenarios. Veins allows online reconfiguration and rerouting of vehicles with the response of network packets. It relies on detailed models of IEEE 802.11p and IEEE 1609.4 DSRC/WAVE network layers. It can simulate vehicular network in real-time on a single workstation or computer cluster, and provides the functions of signal performance in the presence of obstacle. Veins combined with SUMO and OMNeT++ have received great attention from researchers in various fields. Compared with other open source software, their development speed is faster.

2.4 Research Gap and Contribution

The discussion above, Table 2 and Table 3 provide an overview of how research on surrogate safety measure has been developed in different studies. We found that most researchers are studying how to improve the communication latency of real-time applications in abstract form.

Table 2 Network architecture related CV research

Previous works	Summary
(J. Liu et al., 2017)	Provides a conceptual design of the V2X architecture that considers low-latency and high-reliability communication.
(Malinverno et al., 2020)	A two-layer edge computing architecture is proposed for autonomous driving services using Ns3 and SUMO simulators. Suitable for small networks with a single RSU. Proposed Collision Avoidance Algorithm (CAA), and a Collision Avoidance Strategy (CAS).
(Yuan et al., 2018)	Proposed conceptual design of V2X: a two-level edge computing architecture for V2X.
(Moradi-Pari et al., 2015)	Use the output of ns3 to develop a co-simulation tool on MATLAB for FCW and V2V technology.
(S. mehdi Iranmanesh et al., 2016)	A theoretical co-simulation tool for FCW and V2V technology is proposed, but the tool is not discussed.

Table 3 SSM related CV research

(Xie et al., 2019)	Proposed new SSM based on real-world CV SPMD data
(Wali et al., 2018)	Studied intersection level SSM based on real-world CV SPMD data
(Tawfeek & El-Basyouny, 2019)	Studied Stop-sight distance (TTC) using Historical Naturalistic driving data in MATLAB.
(S. M. Iranmanesh et al., 2018)	Proposed Adaptive FCW using SVM and Neural Networks (NN) for SPMD data
(Jahangiri et al., 2018)	Developed a Time to lane crossing (TLC) algorithm using machine learning method of random forest to classify time to lane crossing (TLC) from SPMD data.
(Kamrani et al., 2017)	Calculated location-based volatility at intersections using SPMD data
(Nadimi et al., 2020)(W. Zhang & Wang, 2019)	Calculated various TTCs using Real-world video data application in safety analysis.
(He et al., 2018)	Calculated TTC at link level and trip level using SPMD data
(Y. Wu et al., 2019)	Developed vehicle–bicycle crash warning (post-encroachment time (PET)) algorithm based on data collected form drone

Surprisingly, there are few studies that can test the applicability of SMM in CV in real time, which is only applicable to intersections or small road segments. The methods of calculating basic SSM have been improved and evaluated using offline data sets, some of which have been tested in real time simulation in the V2X environment. Although few studies used real-world connected vehicle data to calculate SSM at the link level, none of the studies calculated SSM at intersection level. Most importantly, these calculated SSMs have never been tested and evaluated for real-time applications in a V2X environment. There are many tools and techniques that can be used to calculate SSM, but most of them are commercial software and hardware involved, which are very expensive. We found that open-source software can create realistic real-time V2X simulations and has been used in research for several years. However, none of the existing studies provide a detailed programming framework for retroactive action to use information carried by connected vehicles to reduce traffic conflicts on the road.

Importantly, commercial software has been used to develop and test SSM that are not publicly available. Open-source software has been used to develop and evaluate communication network architecture. Therefore, no comparison can be made. One of the main contributions of this research is the development of the logic and V2X framework that uses information sent from connected vehicles to test SSMs on different segments of the road. The proposed framework can be used for large-scale transportation networks, multiple intersections, and multiple RSU units connected to a central server in real time. It does not need any modification in the code, except updating the intersection profile to apply for other traffic network, which gives an opportunity to test multiple traffic networks. More specifically, this study has made the following contributions:

- (1) We have developed a two-layer V2X open-source simulation framework that combines realistic DRSC network models (OMNeT++ and Veins) with realistic vehicle networks (SUMO) for real-time SSM data collection on road segments and intersections. There are many simulation software that can implement SSM. However, we chose SUMO, OMNeT++ and Veins because they are all open source and scalable.
- (2) Importantly, the framework will be publicly accessible so that other researchers can reuse the source code and enhance the application to develop new traffic safety measures on top of it.
- (3) We have developed algorithm to calculate road segments level “Rear end TTC” TTC for connected vehicle and data storage typology for RSU and Traffic server (TS), which can save SSM data for future use. The algorithm can be used to collect vehicular data from any segment of the road.

- (4) We have developed an algorithm that creates a typology for the dissipation of the safety index from the central traffic server (TS) to the car through the RSU. This algorithm can be used to respond to cars through the RSU server. For example, one of the applications might be route optimization and route distribution.
- (5) We have developed algorithm to calculate “Crossing TTC” for multiple Intersections and send warning messages. The proposed algorithms in existing studies are applicable for single intersection. The novelty of our algorithm is that it can be used to control CVs at multiple intersections in real time. The algorithm can be applied to any event on the road involving the state of following vehicles and leading vehicles, or vehicles approaching at intersections.

3. SYSTEM ARCHITECTURE

The proposed framework has been developed in OMNeT++ 5, SUMO and Veins. **Error! Reference source not found.** shows the architecture of the communication framework between nodes, which is suitable for multiple RSUs connected to a single traffic server (TS) in a mixed vehicle (CV and NCV) traffic environment. The framework consists of a Traffic Server (TS), multiple Roadside Unit (RSUs), connected vehicles (CV) and non-connected vehicles (NCV). In this framework, CVs communicate with RSU for both emergency and non-emergency events. CV vehicles do not have direct communication access to the main Traffic Server (TS). Upon receiving data and request from car, RSU decides whether the request is for an emergency event or not. If the request is an emergency, the RSU immediately runs its own SSM algorithm and sends necessary warning messages to the vehicle. In the case of non-emergency events, RSU forwards the request to the main traffic server (TS). The traffic server (TS) processes the RSU's forwarding request and sends the necessary information back to the RSU. After the RSU receives the response from the TS, it forwards the response to the vehicle. A detailed description has been given in the following section.

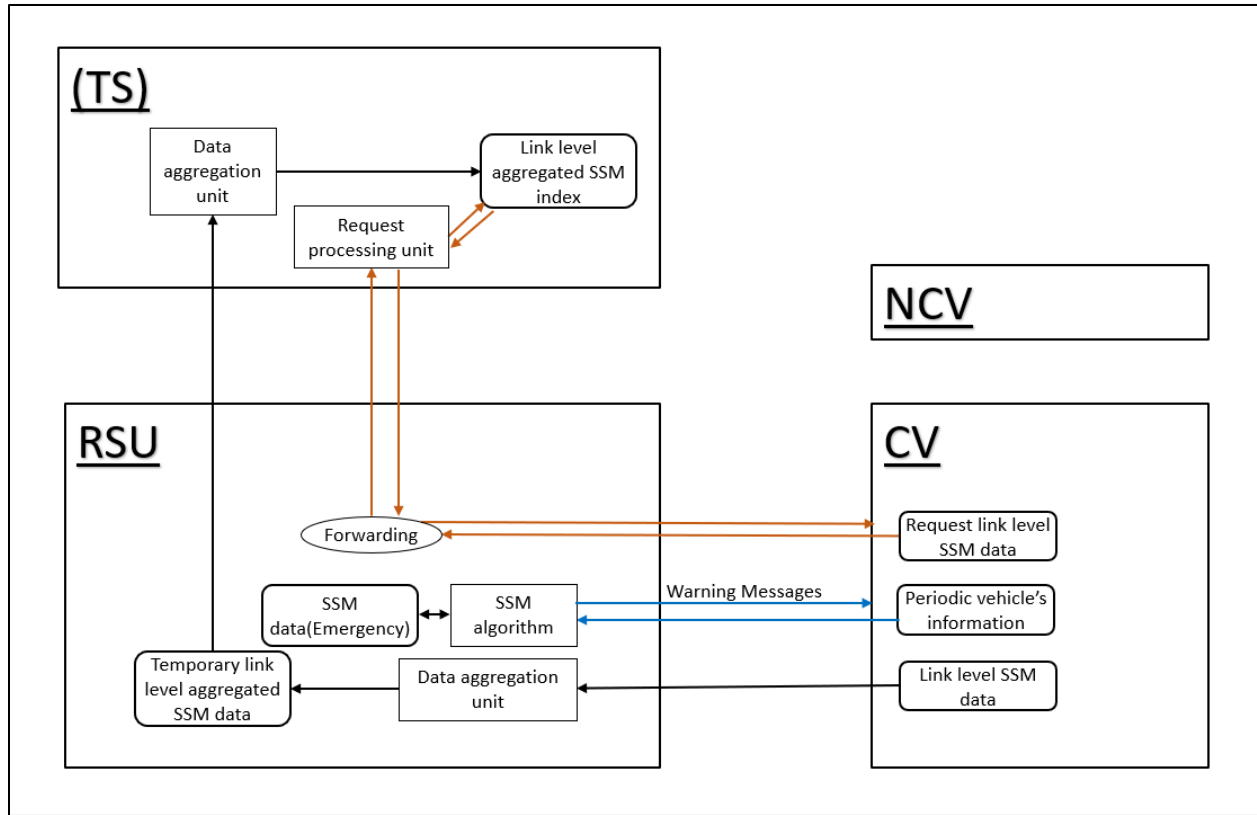


Figure 1 V2X communication typology

3.1 Connected Vehicle (CV) and Non-Connected Vehicle (NCV)

In this framework, A connected vehicle is a vehicle that have an onboard communication device through which it can communicate with RSU. It has the following information (Table 4) available at any timestamp.

Table 4 CV parameters for this study

Parameters	Unit or datatype
Speed	meter/second
Acceleration	meter/second ²

Front vehicle distance	meter
Front vehicle ID	string
Road segment ID	string
Length of Subject vehicle	meter
Width of Subject Vehicle	meter

It is assumed that a CV have a vision or LIDER base technology, which can detect the front vehicle and calculate the distance of front vehicle. CV calculates the speed of the front vehicle based on the rate of distance change. The equation (1) has been used to calculate the distance of the front vehicle.

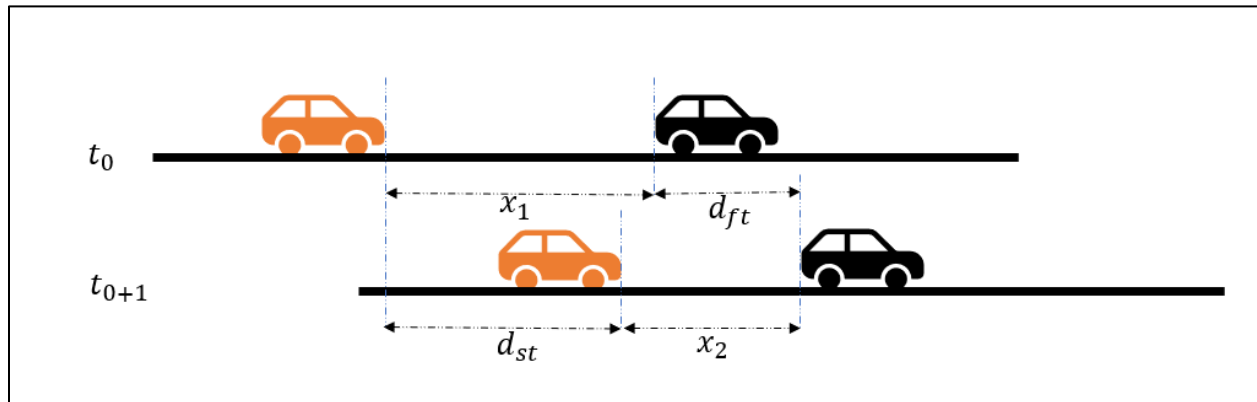


Figure 2 Front vehicle speed calculation from CV.

$$\text{Time, } t = t_{0+1} - t_0$$

$$\text{Subject vehicle driving distance after } t \text{ times, } d_{st} = \text{speed}_{st_0} \times t + \frac{1}{2} \text{acceleration}_{st_0} \times t^2$$

$$\text{Front Vehicle driving distance after } t \text{ times, } d_{ft} = x_2 - x_1 + d_{st}$$

$$\text{Front vehicle speed after } t \text{ times, } v_{ft} = \frac{d_{ft}}{t} \quad (1)$$

The connected vehicle calculates the speed of the preceding vehicle for each time stamp and calculates the SSM at the link level to send it to the RSU within its communication range. In this framework, a NCV vehicle is a vehicle that does not have any communication device on board to communicate with any of the communication nodes. Therefore, the NCV cannot be controlled from this framework. However, the state of NCV will be affected by the action of CV for example, if a CV stops on a lane, the NCV vehicle that is coming behind the CV will slow down or stops by the car following model.

We have created a module for CV that is consist of Myv2v.cc, Myv2v.h and Myv2v.NED files to control CVs through TraCi API of SUMO and transmit signal to other CV, RSU and TS in real-time. Signal transmission occurs in atomic fashion. Our module has two main functionalities; the “handlePositionUpdate” and “onWSA” function, which are described below.

The “handlePositionUpdate” provides update of every CV that are currently present in the network for every timestamp. We have implemented all the SSM algorithms and the messages for data transfer from CV in this function. We used “onWSA” functionalities to check incoming messages from other CVS, RSU and TS. After receiving the message, it will act based on the message type. Each message has a unique message name, which is used to differentiate the type of messages. Appendix C presents the main algorithm part that have been defined in “handlePositionUpdate”.

3.2 Roadside Unit (RSU)

In this research, roadside unit (RSU) is a communication and data processing unit that can directly communicate with CV and TS to exchange information within its communication range. It has three functional units: (1) data processing and storage unit, (2) SSM algorithm implementation and warning message sending unit, and (3) message forwarding unit. The data processing unit processes non-emergency CV data and stores the processed data temporarily, and finally sends the data to the traffic server. After sending the data to TS, it deletes its temporary data. The SSM algorithm implementation and warning message sending unit runs the SSM algorithms on the received kinematics of CV for each time interval and sends the warning message for impending dangerous event to the CVs only who are approaching to that dangerous event. In the message forwarding unit, the RSU forwards the non-emergency request of the CV to the TS, and in a similar manner, it forwards the response of the TS to the CV. The RSU does not encrypt any message in this unit. The main task of this unit is to immediately forward the message to the appropriate node.

A module named “Myv2vRSU” has been created for RSU to receive data from TS and CVs, send warning signal to CVs and send data to TS. Like the module of CV, it also consists of “Myv2v.cc”, “Myv2v.h” and “Myv2v.NED” files. RSU is an immobile unit, which will be activated only when it receives a signal, or an event is scheduled (such as sends data to TS every fixed time interval). We have defined all the data processing (such as storing the SSM data) and warning messages (send a warning message to CVs) algorithms under the “onWSM” function.

3.3 Traffic Server (TS)

The traffic server (TS) is the main data processing and traffic management unit with powerful computing power and large storage capacity. It is directly connected to the RSU for sending and receiving necessary information. It has an SSM data aggregation unit and a request processing unit. The SSM data aggregation unit collects data from all RSUs within its communication range at fixed time intervals, generates an aggregated SSM index and stores it in a permanent database. This unit provides data for real-time traffic management, operation and implementation of new application. The request processing unit receives forwarded requests from RSU and runs its own algorithms to generate the result of that request. After generating the result, it sends the result to the message forwarding unit of RSU. Any external data source such as weather data and historical crash data can be integrated to the TS node.

We have created a separate module for TS which is called “TS” which includes “TS.cc”, “TS.h” and “TS.NED” files to collect data from RSU and send road segment level SSM data to CV via RSU. We have defined all the algorithms under “onWSM” function. As the TS does not provide any periodic message, the “onWSM” function handles all its algorithms. Appendix I shows the algorithm of TS in C++ language.

4. IMPLEMENTATION OF SSM ALGORITHM

The surrogate safety measure (SSM) of traffic conflict are divided into three types: Rear-End Conflict (REC), Crossing Conflict (CC), and Lane Change Conflict (LCC) (Gettman et al., 2008). In this study, we have implemented TTC for Rear End collision and Crossing Conflict. In addition, we have implemented two more TTCs; Time Exposed Time-to-Collision (TET) and Time Integrated Time-to-Collision (TIT), from REC. A vehicle is considered safe if the value of TTC is more than the stopping time of the vehicle. The value of the TTC decreases with time when vehicles tend to collide at conflict points.

A REC can be calculated by sensing front vehicle from CV or by using Basic safety message of vehicle at RSU. Most of the today's vehicles already have the REC feature as inbuild system. Vehicle generates a warning message when the distance of front vehicle is reduced and gone below a threshold value. A connected vehicle with this feature can send this information with road name to the RSU that prepare aggregated SSM at road segment level. The REC can also be calculated at RSU using kinematics of two vehicles driving on the same road segment within the transmission range of the RSU. However, it has several draw backs such as

- RSU may not cover its transmission for a long road segment. Therefore, we may loss data.
- The continuous transmission of signal for RSU can create network congestion, which can result packet loss and communication delay.
- The state of a NCV cannot be captured. Under low penetration rate of CV, this process will not be effective.

For the above reasons, we have implemented the first option in our proposed framework.

For crossing conflict (CC), it is not possible to calculate TTC without exchanging vehicle kinematics among the vehicles and RSUs. Therefore, we have applied vehicle to RSU communication technologies. All the vehicles which are located at intersections and within the transmission range of RSU send their kinematics to RSU and RSU sends TTC to the car.

It is difficult to predict whether the driver of the first vehicle decelerates or accelerates. A high-speed vehicle results more dangerous conflict, which needs greater emergency braking rate and can exceed the conflict point, compared to a low speed vehicle (Carlsson, 2004; C. Wang & Stamatiadis, 2014). Therefore, this study assumes that the faster driver will maintain the speed and cross first the conflict area and other vehicles will give him right-of-way.

Time Exposed Time-to-Collision (TET) and Time Integrated Time-to-Collision (TIT) was calculated using REC. The following section gives details about the SSM algorithms.

4.1 Rear-End Conflict (REC) TTC

Time to collision (TTC) for Rear-End Conflict (REC) is calculated when two vehicles travel on the same road segment and the vehicle in front travels slower than the subject vehicle (Hayward, 1971).

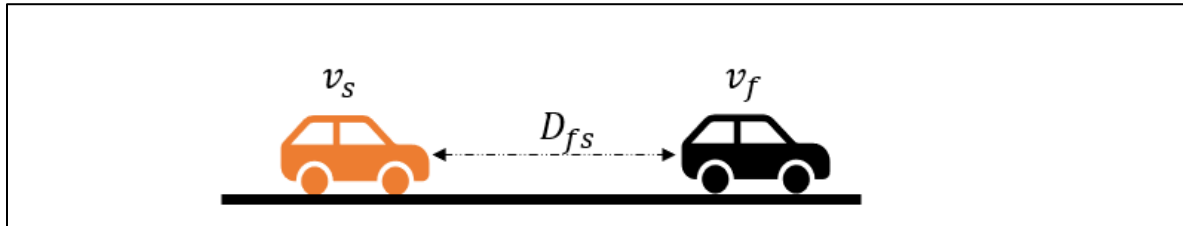


Figure 3 TTC for Rear-End Conflict (REC).

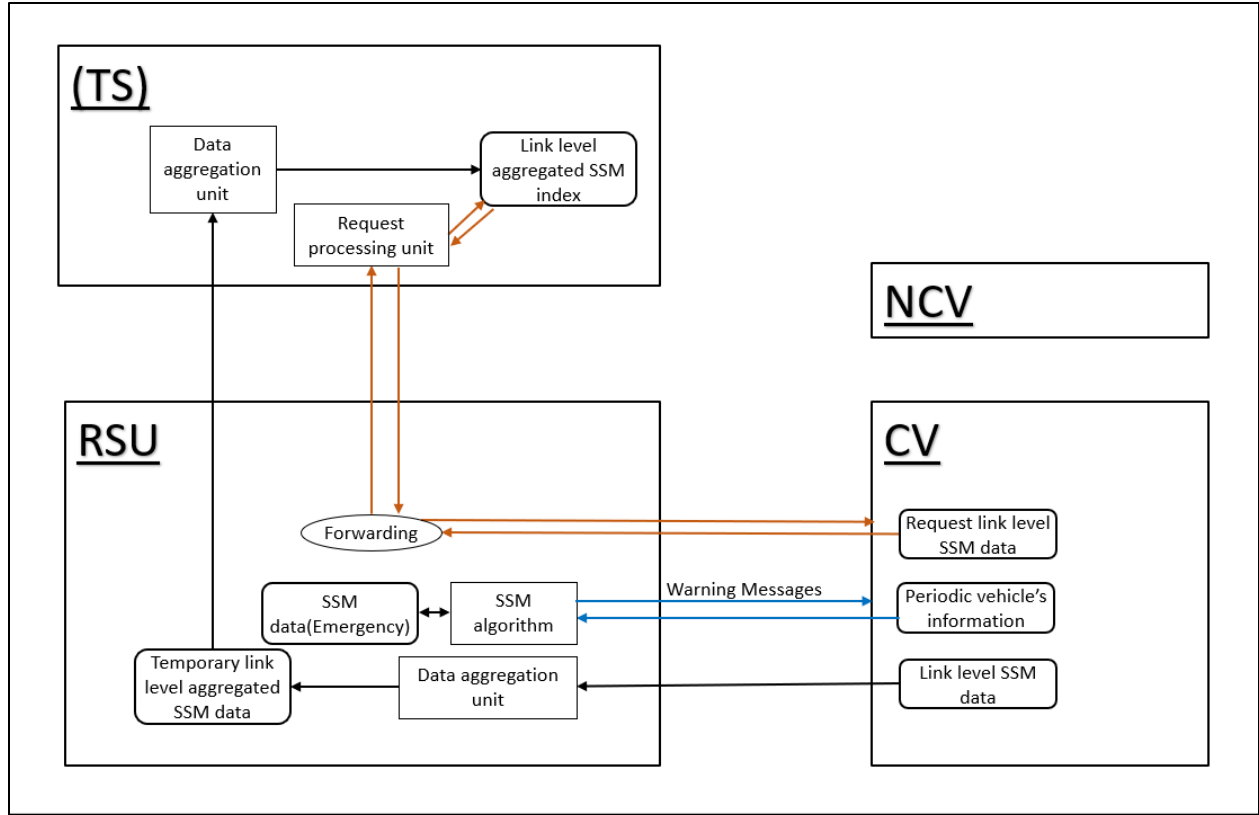


Figure 1 shows how TTC of REC is calculated form equation (2).

$$\text{Rear – End TTC} = \frac{D_{fs}}{\Delta v_{fs}} = \frac{D_{fs}}{v_f - v_s} \text{ when } v_f > v_s \quad (2)$$

D_{fs} = Distance between front and subject vehicle

v_s = Subject vehicle speed

v_f = Front vehicle speed

The Table 5 lists the content of TTC message sent by the CV to the RSU. The message contains a parameter “eventName” which is used as a unique identification key of the message. It

should be an integer number. We have used string value for our framework for easy understanding.

Table 5 TTC message parameter for CV

Parameter	Value or content	Description
Event name	TTC	This is used as unique message ID.
TTCData	Average TTC	TTC/count of TTC
TTCCountData	eventCount	Number of times TTC below the threshold value of 5second
RoadIdData	RoadID	Name of current road segment

The Figure 4 and Appendix C shows the details of the TTC algorithm for REC.

```
1 Previous lead vehicle,lp;
2 Previous lead vehicle distance, lpd;
3 Subject vehicle previous speed,sps;
4 Subject vehicle previous acceleration, spacc;
5
6 GenerateRearEndTTCfromVehicle(){
7
8     Calculate timestamp, t;
9     Find Current lead vehicle ID,lc;
10    Find current lead vehicle Distance, lcd;
11    Calculate subject vehicle current speed, scs;
12    Calculate subject vehicle current acceleration, scacc;
13    If lead vehicle found & lp == lc & t>0 do
14        Calculate lead vehicle speed, lcs;
15        Calculate temporary TTC, tempTTC;
16        If 0<=tempTTC & tempTTC<=5 then
17            TTC= tempTTC;
18            lp = lc;
19            lcd = lcd;
20            sps = scs;
21            spacc = scacc;
22
23        else do
24            lp = lc;
25            lcd = lcd;
26            sps = scs;
27            spacc = scacc;
28    else do
29        lp = lc;
30        lcd = lcd;
31        sps = scs;
32        spacc = scacc;
33    if current Road ID is not previous Road ID do
34        create and send message to all (cars and RSU):
35            TTC;
36            Road ID;
37        Reset:
38            lp;
39            cd;
40            sps;
41            spacc;
42    else do
43        continue;
44    }
```

Figure 4 Pseudocode to calculate Rear-End Conflict TTC.

As mentioned earlier, REC is calculated at the RSU level. When a CV leaves a road segment (such as link or intersection), it sends average TTC and the number of TTC event (count) of that road segment to the RSU. If no TTC event is found, the CV do not transmit any signal. We have tested our algorithm with a threshold of 5 seconds. If the TTC value is greater than 5 seconds, the event is considered a safe event, and the CV discards the TTC value. The Figure 5 shows the Pseudocode of how RSU process the TTC received from a CV. When the

```

map TTC,VTTC<roadId, <TTC,time>>;
VOID Myv2v::receiveSafetyIndexMessage () {
    received TTC from car,RTTC;
    received Road ID from car,Rid;
    received totla time of TTC from car,Rtime;
    if Rid is found in road Id do
        make a weighted average of received TTC
        +and archived TTC for that road,Rid;
        Add received time with archieved time,time
    else do
        Store received TTC;
        Store recieved time;
}

```

Figure 5 Pseudocode for storing TTC at RSU.

RSU receives TTC data from CV, it will check whether it has existing TTC value of the road segment. If not, it adds the new TTC and TTC count of the road segment to the existing list. If found, it will add the TTC and TTC count to the existing list by making a weighted average of the TTC.

4.2 Crossing Conflict (CC) TTC

A crossing conflict happens at intersection when two vehicles approach an intersection and collide to a certain location. There are four conflict points at four-way intersections of single lane road (*Figure 6*). The TTC crossing is the shortest time for two or more vehicles to approach the conflict zone at the intersection while maintaining the current state (Van der Horst, 1991)

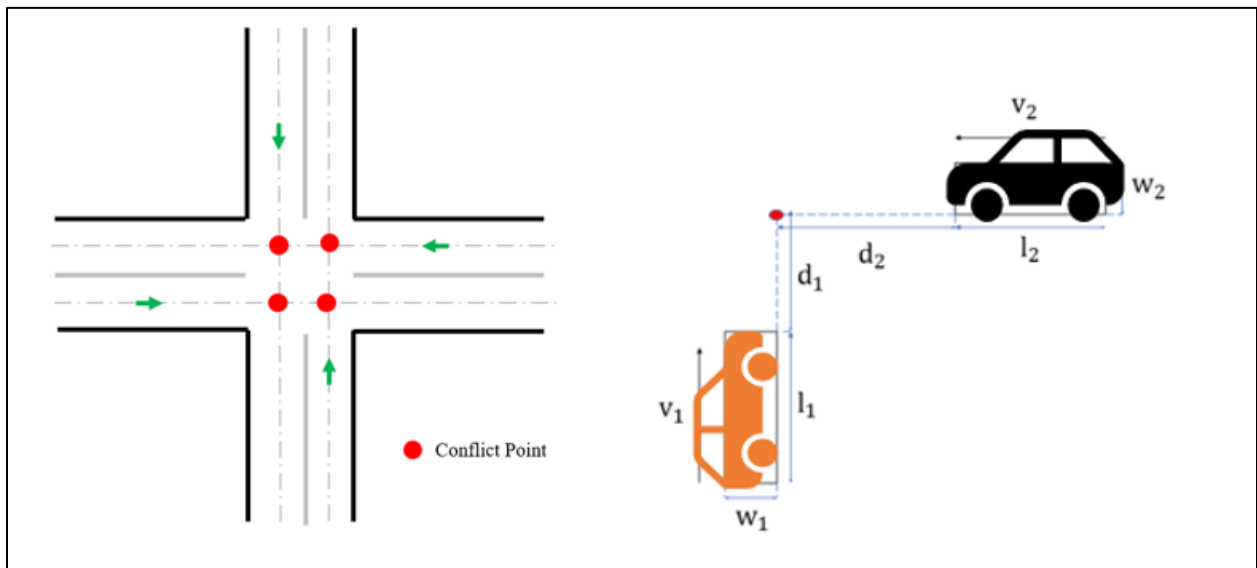


Figure 6 Crossing Conflict points of a typical four-way intersection.

The TTC of a crossing conflict is calculated by the following equation:

$$\begin{aligned}
 \text{Crossing TTC} &= \frac{d_2}{v_2} \text{ when } \frac{d_1}{v_1} < \frac{d_2}{v_2} < \frac{d_1 + l_1 + w_2}{v_1} \\
 \text{or } \frac{d_1}{v_1} &\text{ when } \frac{d_2}{v_2} < \frac{d_1}{v_1} < \frac{d_2 + l_2 + w_1}{v_2}
 \end{aligned} \tag{3}$$

d_1 = distance of vehicle 1 from front bumper to conflict point

d_2 = distance of vehicle 2 from front bumper to conflict point

l_1 = length of vehicle 1

l_2 = length of vehicle 2

w_1 = width of vehicle 1

w_2 = width of vehicle 2

In order to calculate crossing conflict, the intersection parameter (such as approaching road segment Id, Intersection ID and the coordinates of the conflict points) need to be used. The distance from the vehicle to conflict point has been calculated by using the Euclidean distance. When the CV vehicle sends its kinematics information to the RSU, the RSU first determines which intersection it is approaching.

Table 6 CTTC message parameter for CV in the proposed framework

Parameter	Value/content	Description
Event Name	“regCTTC”	Unique
RoadId	“RoadIdData”	Current road segment name
Vehicle Id	“ExtId”	Vehicle ID
speed	“SpeedData”	Current speed
Acceleration	“AccData”	Current acceleration
Vehicle length	“VehLengthData”	Length of the vehicle
Vehicle width	“Vehwidth”	Width of the vehicle
Vehicle position	“CurPosition”	Current position of the vehicle

When CV sends the “regCTTC” message to RSU, RSU first determine the approaching intersection of the vehicle and then look into the database for this vehicle information. If the vehicle exists in the list, the RSU updates the old data with the new data, and then runs the crossing conflict algorithm. If the algorithm finds a paired vehicles, the RSU immediately generates and sends a warning message to the paired vehicles. Figure 7 shows the pseudocode of the CTTC algorithm. Appendix D shows the algorithm in C++ language.

```

vehicle id, vehId;
road segment id, rdId;
vehicle x coordinate, vehposX;
vehicle y coordinate, vehposy;
dictionary of intersection profile, CTTCMap;
dictionary of vehicle profile, CTTCVehMap;
if vehId is found in CTTCVehMap do
    update existitng data of the vehicle "vehId" with new data
    for road-segment-id in CTTCMap do
        for vehicle-Id in CTTCVehMap do
            if rdID is not in road-segment-id & vehId is not in vehicle-Id do
                calculate CTTC;
                send warning message "CTTC"
else do
    for road-segment-id in CTTCMap do
        for vehicle-Id in CTTCVehMap do
            if rdID is not in road-segment-id & vehId is not in vehicle-Id do
                calculate CTTC;
                send warning message "CTTC"
    add vehId data into the CTTCVehMap

```

Figure 7 Pseudocode for calculating CTTC at RSU.

4.3 Time Exposed Time-to-Collision (TET) and Time Integrated Time-to-Collision (TIT)

(Minderhoud & Bovy, 2001) proposed Time Exposed Time-to-collision (TET) and Time Integrated Time-to-Collision (TIT) as two alternatives of the basic TTC. Several studies used TET and TIT for crash risk assessment (Y. Li et al., 2016, 2017; Z. Li et al., 2014; Mahmud et al., 2017; Nadimi et al., 2020). TET represents the total time spent in emergency situations when the TTC value is below the threshold. A traffic situation is considered as a safe situation for a road segment when the TET value is lower than the threshold of that road segment. Equation 4 can be used to calculate the TET value for a road segment. The TET does not consider the safety measure for different time-to-collision values below the threshold value. TET value is useful to distinguish the difference of scenarios. Time Integrated Time-to-Collision (TIT) represents the safety level or relative probability of conflict, which takes into account the impact of different TTC values (Nadimi et al., 2020). It is the integral of the time-to-collision profile of drivers (in s²) (Equation 6). The TIT value is used to distinguish the impact of different TTC values.

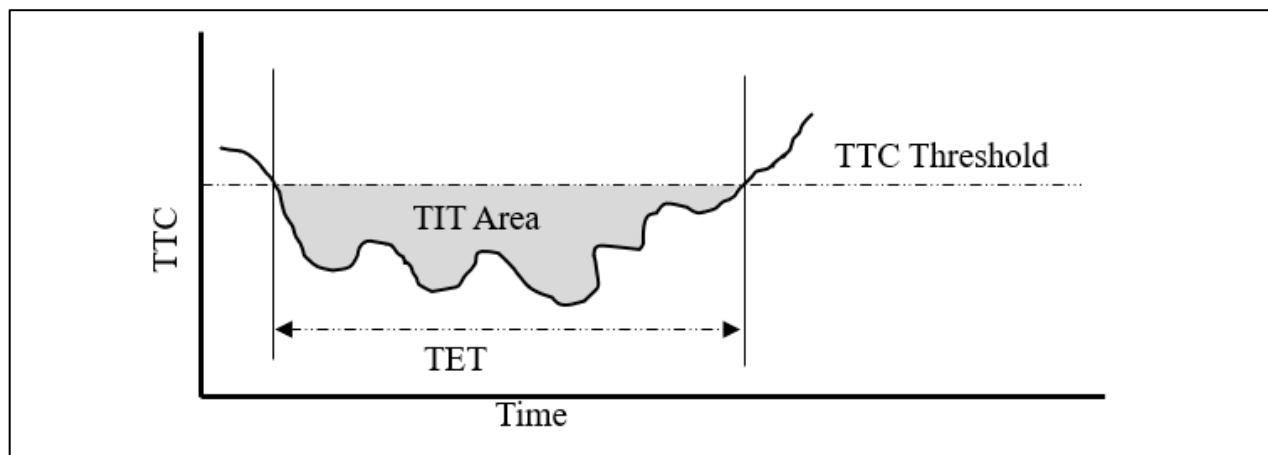


Figure 8 Conceptual diagram of TET and TIT.

$$TET = \sum_{t=0}^T \delta_i(t) \tau_{sc} \quad (4)$$

$$\delta_i(t) = \begin{cases} 1 & 0 \leq TTC_i(t) \leq \text{threshold value} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$$TIT = \int_{t=0}^T [TTC^* - TTC_i(t)] dt \quad 0 \leq TTC_i(t) \leq \text{threshold value} \quad (6)$$

$TTC_i(t)$ = TTC value of CV i at time t.

$\delta_i(t)$ i= indicator variable.

$T = \frac{H}{\tau_{sc}}$ = Time frequency where H is extended time period of study and τ_{sc} is timestamp

The TET and TIT can be extracted and prepared from the REC and CTTC algorithms.

Hence, we did not prepare any separate algorithms for these two SSMs.

5. EXPERIMENTAL SETUP AND TESTING

5.1 SUMO Parameters and Simulation Files

An experimental study was conducted to test the developed framework. We have used two traffic networks: an imaginary road network and a real-world road network. The imaginary road network was created in SUMO with some random vehicles, which contained all the traffic scenarios needed to test the SSM algorithms. We used imaginary network for testing and debugging purpose. The real-world traffic network included a part of Milwaukee downtown, Wisconsin, which was collected from OpenStreetMap⁸. The advantage of using OpenStreetMap is that most of the network parameters such as lane information, speed limit, traffic signal, etc. are included with its OSM file. After collecting the OSM file, we converted it into net.xml file to make it usable in SUMO simulation. We have created some additional files such as .rou.xml, .poly.xml, .sumo.config, launched.xml files from .net.xml files. The .rou.xml file is the routing file, which holds vehicle's profiles including route plan. The .poly.xml file provides physical infrastructure such as building, shop, natural etc. We need this file to generate network obstruction effect in OMNeT++ and Veins. The .sumo.config file is the simulation file that creates typology for all the files, define the simulation parameters and run the simulation. The launch.xml file is used in the Veins to run and control the SUMO simulation. Table 7 shows the network parameter that are used in this study. As our main goal of this study is to develop the framework, we tried to keep default parameters of SUMO and reduce human intervention. In this experimental study, we used Krauss car-following model, which is the default one in SUMO.

⁸ <https://www.openstreetmap.org/#map=4/38.01/-95.84>

The basic idea behind of a car-following model is that each vehicle speed depends on speed of leading vehicle, minimum gap, and the other static parameters such as the reaction time (Malinverno et al., 2020). Table 7 shows SUMO network parameters we used for this study.

Table 7 Network parameters in SUMO

Parameters	Value/status
Number of Trips / cars	300
Route generation method	Random Trips
Present of Traffic signal	Yes
Present of Pedestrian	No
Vehicle Type	Car
Number of Intersections	21
Number of Road Segments	23

5.2 OMNeT++ Parameters

After creating the necessary SUMO files, we defined simulation parameters of OMNet++ (Table 8). The developed framework will provide status update of traffic simulation in SUMO and network simulation in OMNeT++ for every 0.1 second. The total simulation time we defined in the SUMO configuration file is 300 seconds. We activated the obstacle model, which provides a blocking effect in packet loss and transmission. In the car module, we did not initialize periodic message so that CVs could not create unnecessary message. Periodic messages will be controlled by the car module. RSU will send temporarily aggregated TTC to TS every fixed number of received messages (for example, when the number of received messages is 10, send TTC to TS, and then reset the database). The TS module only responds when it receives any

request from the RSU. In Network Interface Card (NIC) setting, we left the default parameter as it is. However, it can be modified as needed.

Table 8 Simulation parameters in OMNeT++

Parameters	Description	Value/Status
Simulation time	Total simulation time	200s
Obstacle Module	Affect the transmission power. E.g., more high rich buildings, more packet loss.	Active
update interval	Time to collect traffic information from SUMO	0.1s
Port		9999
Car Module		
Send beacon	Send message periodically	Not active
RSU Module		
Antenna Position	The location of antenna above the ground	3m
Send beacon	Send message periodically	active
Beacon Interval	Send message after 10 second periodically	10s
TS Module		
Antenna Position	The location of antenna above the ground	3m
Send beacon	Send message periodically	Not active
NIC-Settings		
maxInterfDist	Maximum communication boundary in meter	300m
Transmission power		20mW
Bitrate		6Mbps
Minimum power level		-110dBm

Noisyfloor		-98dBm
Propagation delay		Active

5.3 Conceptual design

Error! Reference source not found. shows the functions of the V2X framework developed in the mixed transportation system on the Milwaukee, Wisconsin urban road network. In this study, we used two RSUs located at the intersections of the corridor. We set the maximum transmission distance of CV, RSU and TS to 300 meters. To define different transmission ranges for different nodes, we need to configure it internally. However, we left this part for future research. The TS was located within the range of RSUs transmission range. We set up two RSUs in this way so that they cannot communicate with each other. The CV represented by the red car can sense the vehicle in front and measure the distance between them. When the CV

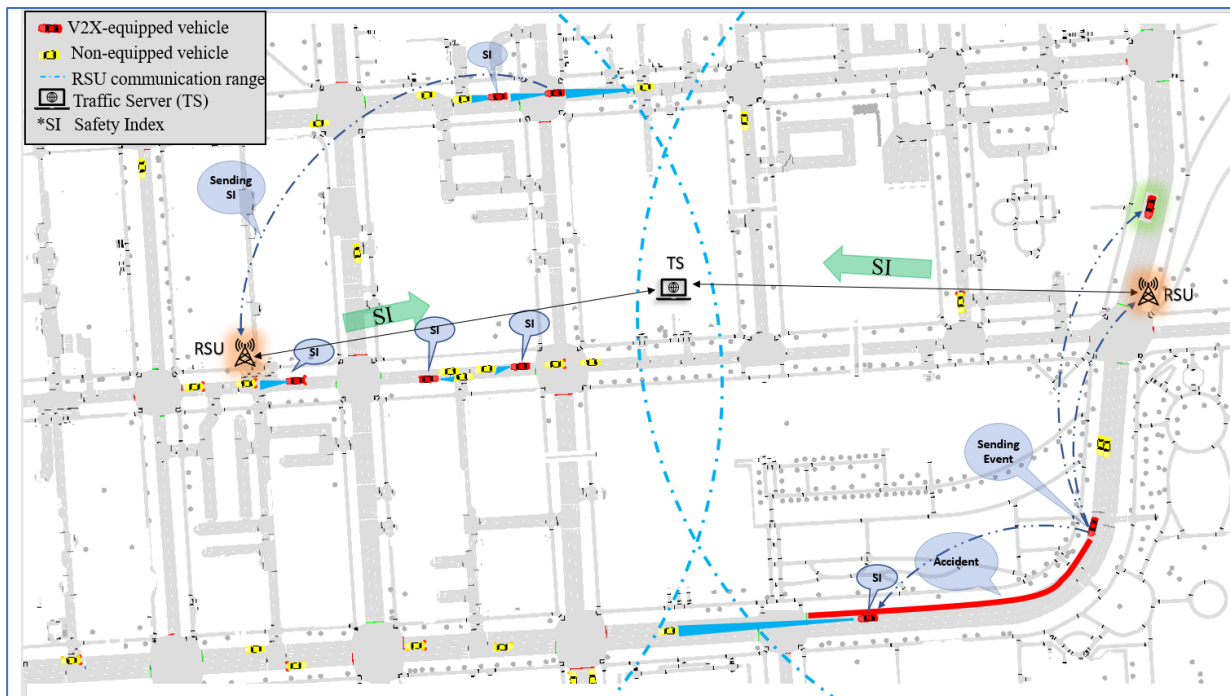


Figure 9 V2X communication framework for CVs and RSU on the road network of Milwaukee downtown, Downtown (demonstration purpose only).

(such as the red car at the top of **Error! Reference source not found.**) leaves the road segment and enters the intersection, it will immediately send the SSM to the RSU and start calculating the SSM of the intersection at the same time.

Error! Reference source not found. shows another application of the developed framework. A CV car (bottom left) observed the accident and immediately notified the other car by sending a message. The red CV car (top left) that was travelling toward that road received that message. The Car decided to change the route and sent a message to RSU to obtain a safe connecting route. The RSU forwarded that message to TS and TS sent the safest route (Green

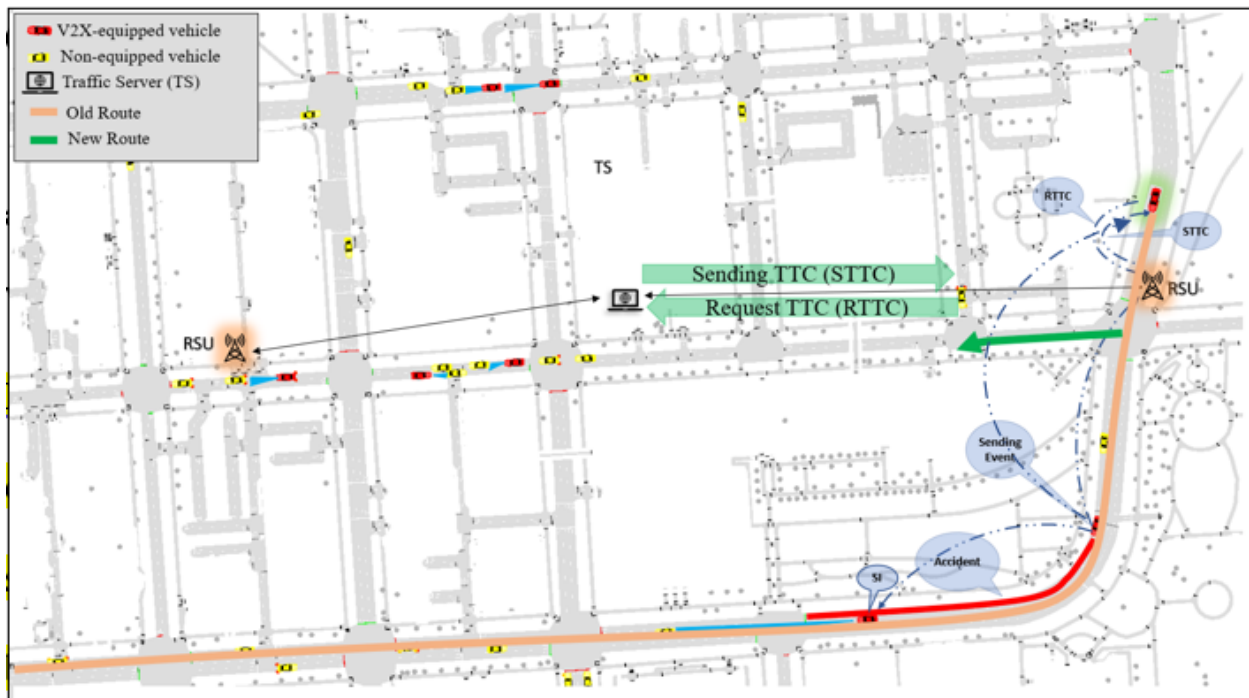


Figure 10 Accident scenario and route change strategy
arrow) to the Car via RSU.

5.4 Simulation Results

Figure 12 shows the TTC value of the CV vehicle generated in real-time simulation. After analyzing the result, we found that most of the TTCs are generated at intersection, when the vehicle was stopped at intersection.

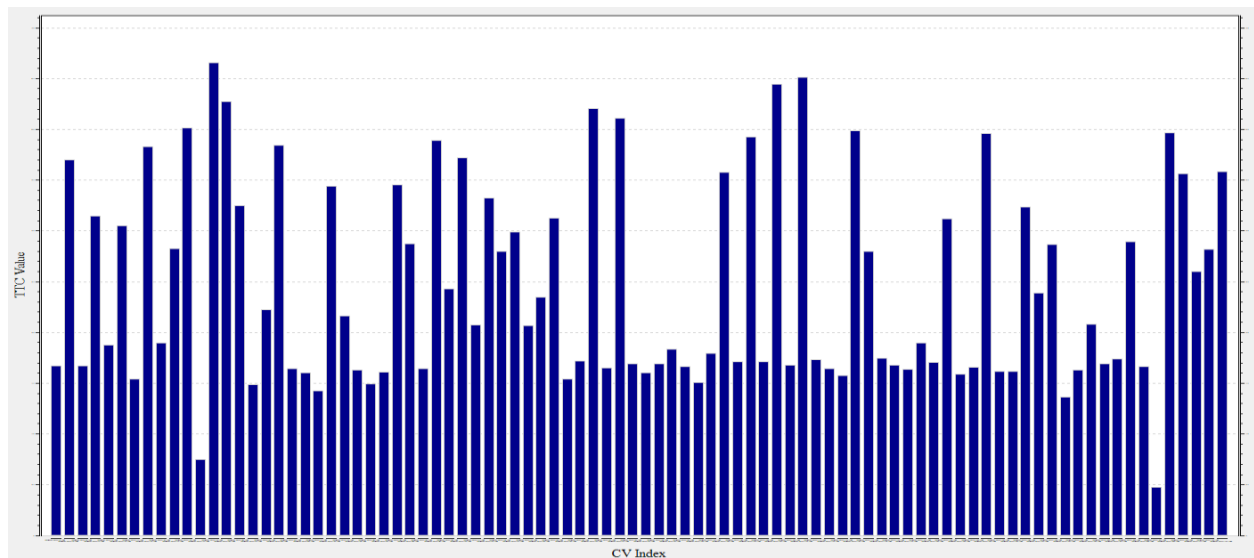


Figure 11 Average TTC of the CVs

By investigating the TTC results, we found that the driver keeps a small distance when approaching the intersection and has a small TTC value compared to when driving on the road segment. This result shows that, the driver's behavior at the intersection is riskier. Figure 11 showed the output of TTC of different road segments and intersections.



Figure 12 Safety Index (TTC) of different segment of the roads and intersections.

Figure 12 showed road segment and intersection level TTC values. We found that the average TTC is 2.79 seconds for intersection and 2.90 for road segments. Figure 13 represents the TET value of some road segments and intersections. The average TET value is 6.54 seconds for intersections and 33.7 for road segment.

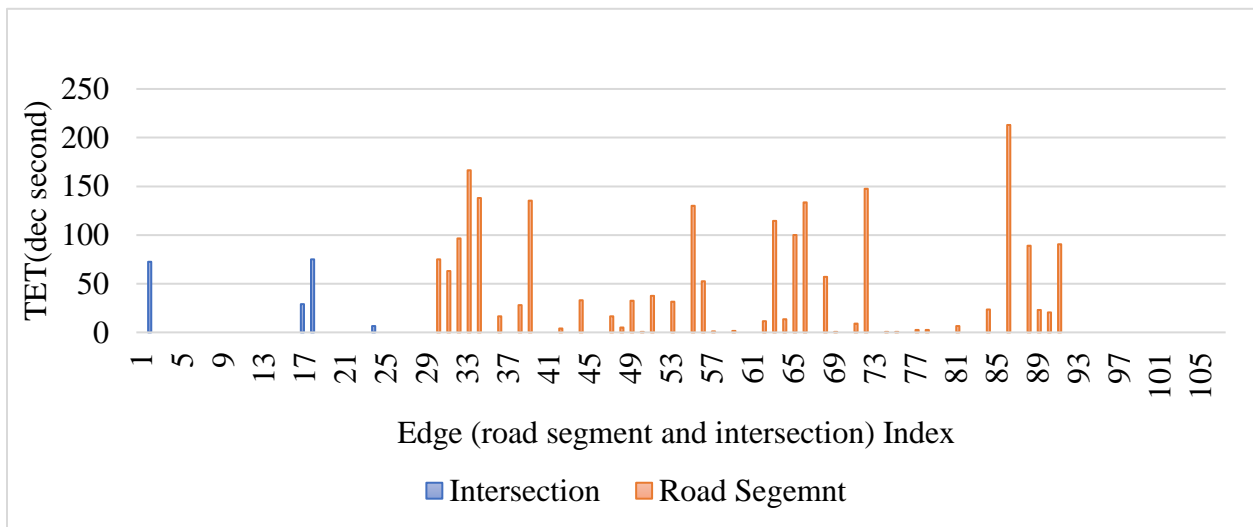


Figure 13 TET of different segment of the roads and intersections.

5.5 Network Communication Performance

We found that there is almost no packet loss when TTC messages are transmitted from Car to RSU and from RSU to TS. However, a lot of packets are lost in periodical messages. And it decreased when the car came closer to RSU. Based on this observation, we can take some measures, such as increasing the signal transmission power, establishing more RSUs, reducing the transmission range, and using a powerful communication network (such as a 5G network).

6. CONCLUSION

With the help of advance automobile technologies, the connected vehicle (CV) technology can provide surrogate safety measure (SSM) to improve traffic safety and operation. The information exchanged between Vehicle-to-Vehicle (V2V), and Vehicle-to-road infrastructure(V2X), can improve traffic operations. CV technology involves two network systems: transportation network and wireless communication network. However, research on wireless communication networks for CV and research on SSM in transportation networks do not interact adequately, and vice versa. Hence, there is a research gap between the research communities of these two network systems, which may be due to differences in research fields. Moreover, there is no research or opensource algorithm available that can be used to generate SSM for road-segments and multiple intersections in real-time. In this study, we developed a V2X simulation framework using SUMO, OMNeT++ and Veins for the development and testing of various SSM algorithms in run time simulation. SUMO is an open source, highly portable, microscopic multi-modal traffic simulation, which can handle large traffic networks. The TraCI API of SUMO provides great flexibility for vehicle control in real-time simulation. SUMO interacts with the wireless simulator OMNeT++ through the Veins framework to generate real world connected vehicle scenarios. Veins provides online reconfiguration and rerouting of vehicles with the response of network packets. It relies on detailed models of IEEE 802.11p and IEEE 1609.4 DSRC/WAVE network layers, and can simulate vehicular network in real-time.

Our developed framework has two levels of communication systems, such as CV to RSU and RSU to TS. It is suitable for simulating large-scale transportation networks with mixed transportation systems (CV and non-CV), multiple roadside units (RSU) and core transportation server (TS). Our algorithms only use the basic parameters of the vehicle (such as speed,

acceleration, distance to the vehicle ahead, etc.) from SUMO. Therefore, it will be easy to implement to other traffic simulators. Moreover, the framework can be used to evaluate new SSM algorithms in V2X communication system in real-time. Our developed framework is publicly available so that anyone can contribute to the development and optimization of the algorithm.

The developed framework is not free from limitations. The analysis and development of cooperative collision warning systems require detailed research on communication network components, collision detection algorithms and safety applications (such as SSMs). We have observed that due to weak signals, packet collisions and other environmental conditions (such as obstacles), when CV sends messages periodically, the packet loss rate is high. The framework can be improved by optimizing algorithms, including new types of messages, integrating different wireless networks (such as DSRC with 5G networks), and integrating more SSM algorithms. We left this part for the future research.

REFERENCES

- Avino, G., Bande, P., Frangoudis, P. A., Vitale, C., Casetti, C., Chiasserini, C. F., Gebru, K., Ksentini, A., & Zennaro, G. (2019). A MEC-Based Extended Virtual Sensing for Automotive Services. *IEEE Transactions on Network and Service Management*, 16(4), 1450–1463. <https://doi.org/10.1109/TNSM.2019.2931878>
- Carlsson, G. (2004). Kunskapssammanställning—Dödsolyckor och hastighet. *Knowledge Summary—Fatal Accidents and Speed. NTF-Kansliet*.
- Ejercito, P. M., & Nebrija, K. G. E. (2009). *Traffic Simulation Software Review*.
- Gelbal, S. Y., Zhu, S., Anantharaman, G. A., Aksun Guvenc, B., & Guvenc, L. (2019). Cooperative collision avoidance in a connected vehicle environment. *SAE Technical Papers, 2019-April(April)*, 1–9. <https://doi.org/10.4271/2019-01-0488>
- Gettman, D., & Head, L. (2003). Surrogate Safety Measures from Traffic Simulation Models. *Transportation Research Record, 1840*, 104–115. <https://doi.org/10.3141/1840-12>
- Gettman, D., Pu, L., Sayed, T., & Shelby, S. (2008). Surrogate Safety Assessment Model and Validation: Final Report. *Publication No. FHWA-HRT-08-051, June*, 1–324.
- Hayward, J. (1971). *Near misses as a measure of safety at urban intersections*. Pennsylvania Transportation and Traffic Safety Center.
- He, Z., Qin, X., Liu, P., & Sayed, M. A. (2018). Assessing Surrogate Safety Measures using a Safety Pilot Model Deployment Dataset. *Transportation Research Record, 2672(38)*, 1–11. <https://doi.org/10.1177/0361198118790861>
- Iranmanesh, S. M., Nourkhiz Mahjoub, H., Kazemi, H., & Fallah, Y. P. (2018). An adaptive forward collision warning framework design based on driver distraction. *IEEE Transactions on Intelligent Transportation Systems, 19(12)*, 3925–3934. <https://doi.org/10.1109/TITS.2018.2791437>
- Iranmanesh, S. mehdi, Moradi-Pari, E., Fallah, Y., Das, S., & Rizwan, M. (2016). *Robustness of cooperative forward collision warning systems to communication uncertainty*. 1–7. <https://doi.org/10.1109/SYSCON.2016.7490583>
- Jahangiri, A., Berardi, V. J., & MacHiani, S. G. (2018). Application of Real Field Connected Vehicle Data for Aggressive Driving Identification on Horizontal Curves. *IEEE Transactions on Intelligent Transportation Systems, 19(7)*, 2316–2324. <https://doi.org/10.1109/TITS.2017.2768527>
- Kamrani, M., Wali, B., & Khattak, A. J. (2017). Can Data Generated by Connected Vehicles Enhance Safety?: Proactive Approach to Intersection Safety Management. *Transportation Research Record, 2659(1)*, 80–90. <https://doi.org/10.3141/2659-09>
- Keivani, A., Ghayoor, F., & Tapamo, J. R. (2019). Collaborative Mobile Edge Computing in eV2X: A Solution for Low-Cost Driver Assistance Systems. *Wireless Personal Communications, 0123456789*. <https://doi.org/10.1007/s11277-019-06401-2>

- Laureshyn, A., Svensson, Å., & Hydén, C. (2010). Evaluation of traffic safety, based on micro-level behavioural data: Theoretical framework and first implementation. *Accident Analysis and Prevention*, 42(6), 1637–1646. <https://doi.org/10.1016/j.aap.2010.03.021>
- Li, L., Li, Y., & Hou, R. (2017). A novel mobile edge computing-based architecture for future cellular vehicular networks. *IEEE Wireless Communications and Networking Conference, WCNC*. <https://doi.org/10.1109/WCNC.2017.7925830>
- Li, Y., Wang, H., Wang, W., Liu, S., & Xiang, Y. (2016). Reducing the risk of rear-end collisions with infrastructure-to-vehicle (I2V) integration of variable speed limit control and adaptive cruise control system. *Traffic Injury Prevention*, 17(6), 597–603. <https://doi.org/10.1080/15389588.2015.1121384>
- Li, Y., Xu, C., Xing, L., & Wang, W. (2017). Integrated Cooperative Adaptive Cruise and Variable Speed Limit Controls for Reducing Rear-End Collision Risks Near Freeway Bottlenecks Based on Micro-Simulations. *IEEE Transactions on Intelligent Transportation Systems*, 18(11), 3157–3167. <https://doi.org/10.1109/TITS.2017.2682193>
- Li, Z., Li, Y., Liu, P., Wang, W., & Xu, C. (2014). Development of a variable speed limit strategy to reduce secondary collision risks during inclement weathers. *Accident Analysis and Prevention*, 72, 134–145. <https://doi.org/10.1016/j.aap.2014.06.018>
- Liu, H., Wei, H., Zuo, T., Li, Z., & Yang, Y. J. (2017). Fine-tuning ADAS algorithm parameters for optimizing traffic safety and mobility in connected vehicle environment. *Transportation Research Part C: Emerging Technologies*, 76, 132–149. <https://doi.org/10.1016/j.trc.2017.01.003>
- Liu, J., Wan, J., Zeng, B., Wang, Q., Song, H., & Qiu, M. (2017). A scalable and quick-response software defined vehicular network assisted by mobile edge computing. *IEEE Communications Magazine*, 55(7), 94–100. <https://doi.org/10.1109/MCOM.2017.1601150>
- Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., & Wießner, E. (2018). Microscopic Traffic Simulation using SUMO. *IEEE Intelligent Transportation Systems Conference (ITSC)*. <https://elib.dlr.de/124092/>
- Mahmud, S. M. S., Ferreira, L., Hoque, M. S., & Tavassoli, A. (2017). Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs. *IATSS Research*, 41(4), 153–163. <https://doi.org/10.1016/j.iatssr.2017.02.001>
- Malinverno, M., Mangues-Bafalluy, J., Casetti, C. E., Chiasserini, C. F., Requena-Esteso, M., & Baranda, J. (2020). An edge-based framework for enhanced road safety of connected cars. *IEEE Access*, 8, 58018–58031. <https://doi.org/10.1109/ACCESS.2020.2980902>
- Minderhoud, M. M., & Bovy, P. H. L. (2001). Extended time-to-collision measures for road traffic safety assessment. *Accident Analysis and Prevention*, 33(1), 89–97. [https://doi.org/10.1016/S0001-4575\(00\)00019-1](https://doi.org/10.1016/S0001-4575(00)00019-1)
- Moradi-Pari, E., Gani, S. M. O., Fallah, Y. P., Naserian, M., & Lewis, A. (2015). Co-Simulation of Cooperative Vehicle Safety Applications and Communication Networks. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, 8(2), 344–

349. <https://doi.org/10.4271/2015-01-0285>

- Nadimi, N., Ragland, D. R., & Mohammadian Amiri, A. (2020). An evaluation of time-to-collision as a surrogate safety measure and a proposal of a new method for its application in safety analysis. *Transportation Letters*, 12(7), 491–500. <https://doi.org/10.1080/19427867.2019.1650430>
- Pell, A., Meingast, A., & Schauer, O. (2017). Trends in Real-time Traffic Simulation. *Transportation Research Procedia*, 25, 1477–1484. <https://doi.org/10.1016/j.trpro.2017.05.175>
- Saunier, N., & Sayed, T. (n.d.). *Probabilistic Framework for Automated Analysis of Exposure to Road Collisions*. 96–104. <https://doi.org/10.3141/2083-11>
- Tarek, S., & Sany, Z. (2007). Traffic conflict standards for intersections. *Transportation Planning and Technology*, 22(4), 309–323.
- Tarko, A., Davis, G., Saunier, N., & Sayed, T. (2009). *Surrogate Measures of Safety*.
- Tawfeek, M. H., & El-Basyouny, K. (2019). Using naturalistic driving data to quantify driver following behavior during braking. *12th International Transportation Specialty Conference 2018, Held as Part of the Canadian Society for Civil Engineering Annual Conference 2018, July*, 260–269.
- Van der Horst, A. R. A. (1991). *A time-based analysis of road user behaviour in normal and critical encounters*.
- Wali, B., Khattak, A. J., Bozdogan, H., & Kamrani, M. (2018). How is driving volatility related to intersection safety? A Bayesian heterogeneity-based analysis of instrumented vehicles data. *Transportation Research Part C: Emerging Technologies*, 92(May), 504–524. <https://doi.org/10.1016/j.trc.2018.05.017>
- Wang, C., & Stamatiadis, N. (2014). Evaluation of a simulation-based surrogate safety metric. *Accident Analysis and Prevention*, 71, 82–92. <https://doi.org/10.1016/j.aap.2014.05.004>
- Wang, K., Yin, H., Quan, W., & Min, G. (2018). Enabling Collaborative Edge Computing for Software Defined Vehicular Networks. *IEEE Network*, 32(5), 112–117. <https://doi.org/10.1109/MNET.2018.1700364>
- Wu, K.-F., & Jovanis, P. P. (2012). Crashes and crash-surrogate events: Exploratory modeling with naturalistic driving data. *Accident Analysis & Prevention*, 45, 507–516.
- Wu, Y., Abdel-Aty, M., Zheng, O., Cai, Q., & Yue, L. (2019). Developing a Crash Warning System for the Bike Lane Area at Intersections with Connected Vehicle Technology. *Transportation Research Record*, 2673(4), 47–58. <https://doi.org/10.1177/0361198119840617>
- Xie, K., Yang, D., Ozbay, K., & Yang, H. (2019). Use of real-world connected vehicle data in identifying high-risk locations based on a new surrogate safety measure. *Accident Analysis and Prevention*, 125(May), 311–319. <https://doi.org/10.1016/j.aap.2018.07.002>
- Yuan, Q., Zhou, H., Li, J., Liu, Z., Yang, F., & Shen, X. S. (2018). Toward Efficient Content

Delivery for Automated Driving Services: An Edge Computing Solution. *IEEE Network*, 32(1), 80–86. <https://doi.org/10.1109/MNET.2018.1700105>

Zhang, K., Mao, Y., Leng, S., Maharjan, S., & Zhang, Y. (2017). Optimal delay constrained offloading for vehicular edge computing networks. *IEEE International Conference on Communications*, 1–6. <https://doi.org/10.1109/ICC.2017.7997360>

Zhang, W., & Wang, W. (2019). Learning V2V interactive driving patterns at signalized intersections. *Transportation Research Part C: Emerging Technologies*, 108(September 2018), 151–166. <https://doi.org/10.1016/j.trc.2019.09.009>

Zheng, J., & Liu, H. X. (2017). Estimating traffic volumes for signalized intersections using connected vehicle data. *Transportation Research Part C: Emerging Technologies*, 79, 347–362. <https://doi.org/10.1016/j.trc.2017.03.007>

APPENDICES

APPENDIX A: Parameter setup for RSU in NED file

```
1 package org.car2x.veins.nodes;
2
3
4 import org.car2x.veins.base.modules.*;
5 import org.car2x.veins.modules.nic.Nic80211p;
6
7 module RSU
8 {
9     parameters:
10         string applType; //type of the application layer
11         string nicType = default("Nic80211p"); // type of network interface card
12     gates:
13         input veinsradioIn; // gate for sendDirect
14
15
16
17     submodules:
18         appl: <applType> like org.car2x.veins.base.modules.IBaseApplLayer {
19             parameters:
20                 @display("p=60,50");
21         }
22
23         nic: <nicType> like org.car2x.veins.modules.nic.INic80211p {
24             parameters:
25                 @display("p=60,166");
26         }
27
28         mobility: BaseMobility {
29             parameters:
30                 @display("p=130,172;i=block/cogwheel");
31         }
32
33     connections:
34         nic.upperLayerOut --> appl.lowerLayerIn;
35         nic.upperLayerIn <-- appl.lowerLayerOut;
36         nic.upperControlOut --> appl.lowerControlIn;
37         nic.upperControlIn <-- appl.lowerControlOut;
38
39         veinsradioIn --> nic.radioIn;
40
41 }
```

APPENDIX B: Parameter setup for CV in Car NED file

```
2 package org.car2x.veins.nodes;
3
4 import org.car2x.veins.base.modules.*;
5 import org.car2x.veins.modules.nic.Nic80211p;
6
7 module Car
8 {
9
10 parameters:
11     string applType; //type of the application layer
12     string nicType = default("Nic80211p"); // type of network interface card
13     string veinsmobilityType = default("org.car2x.veins.modules.mobility.traci.TraCIMobility");
14
15
16
17 gates:
18     input veinsradioIn; // gate for sendDirect
19 submodules:
20     appl: <applType> like org.car2x.veins.base.modules.IBaseApplLayer {
21         parameters:
22             @display("p=60,300");
23     }
24
25     nic: <nicType> like org.car2x.veins.modules.nic.INic80211p {
26         parameters:
27             @display("p=60,166");
28     }
29
30     veinsmobility: <veinsmobilityType> like org.car2x.veins.base.modules.IMobility {
31         parameters:
32             @display("p=130,172;i=block/cogwheel");
33     }
34
35
36
37 connections:
38     nic.upperLayerOut --> appl.lowerLayerIn;
39     nic.upperLayerIn <-- appl.lowerLayerOut;
40     nic.upperControlOut --> appl.lowerControlIn;
41     nic.upperControlIn <-- appl.lowerControlOut;
42
43     veinsradioIn --> nic.radioIn;
44
45 }
```

APPENDIX C: Algorithm of calculating Rear-End TTC for CV in CVs module

```
122
123   curleadvehicle=traciVehicle->getLeader(TTCRange).first;
124   leaderDistance=traciVehicle->getLeader(TTCRange).second;
125   subCurSpeed=traciVehicle->getSpeed();
126   subCurAcc=traciVehicle->getAcceleration();
127   if (leaderDistance!=-1 & timeStamp>0 & traciVehicle->getRoadId()==preRoadId){
128       if (curleadvehicle==preleadvehicle){
129           leadVehSpeed=abs((PreDistance-leaderDistance-(subPreSpeed*timeStamp
130               +0.5*subPreAcc*pow(timeStamp,2.0)))/timeStamp);
131           tempTTC=leaderDistance/(subCurSpeed-leadVehSpeed);
132           if (0<=tempTTC & tempTTC<=5){ // 5 is threshold value.
133               TTC=TTC+tempTTC;
134               eventCount=eventCount+1;
135               preleadvehicle=curleadvehicle;
136               PreDistance= leaderDistance;
137               subPreSpeed=subCurSpeed;
138               subPreAcc=subCurAcc;
139               preRoadId=traciVehicle->getRoadId();
140
141           }
142       else {
143           preleadvehicle=curleadvehicle;
144           PreDistance= leaderDistance;
145           subPreSpeed=subCurSpeed;
146           subPreAcc=subCurAcc;
147           preRoadId=traciVehicle->getRoadId();
148
149       }
150   }
151   else {
152       preleadvehicle=curleadvehicle;
153       PreDistance= leaderDistance;
154       subPreSpeed=subCurSpeed;
155       subPreAcc=subCurAcc;
156       preRoadId=traciVehicle->getRoadId();
157   }
158 }
159
160
```

APPENDIX D: Algorithm of calculating Crossing TTC for CV Part A in CVs Module

```
cit = CTTCVehMap.find(vehId);
if (cit != CTTCVehMap.end()){
    CTTCVehMap[vehId].first=wsm->getIntersectionId();
    CTTCVehMap[vehId].second.first=t1;
    CTTCVehMap[vehId].second.second=t2;

    for(auto & outer_map_pair : CTTCVehMap) {
        for (auto & edge_in_list: CTTCMap){
            if (edge!=edge_in_list.first & wsm->getIntersectionId()==outer_map_pair.second.first
                & vehId!=outer_map_pair.first){
                if (t1<outer_map_pair.second.second.first
                    & outer_map_pair.second.second.first<t2 ){
                    EV <<"t1 "<<t1<<" t2 "<<t2<<std::endl;
                    TTCforRSU* CTTCWarning = new TTCforRSU("Sending Warning!");
                    populateWSM(CTTCWarning);
                    sentMessage=true;
                    CTTCWarning->setEventName("CTTC");
                    CTTCWarning->setExtId(vehId.c_str());
                    CTTCWarning->setExtIdVeh2(outer_map_pair.first.c_str());
                    sendDown( CTTCWarning);

                }
                if (t1>outer_map_pair.second.second.first
                    & outer_map_pair.second.second.first>t2 ){
                    EV <<"t1 "<<t1<<" t2 "<<t2<<std::endl;
                    TTCforRSU* CTTCWarning = new TTCforRSU("Sending Warning!");
                    populateWSM(CTTCWarning);
                    sentMessage=true;
                    CTTCWarning->setEventName("CTTC");
                    CTTCWarning->setExtId(vehId.c_str());
                    CTTCWarning->setExtIdVeh2(outer_map_pair.first.c_str());
                    sendDown( CTTCWarning);

                }
            }
        }
    }
}
```

APPENDIX E: Algorithm of calculating Crossing TTC for CV Part B in CVs Module

```
for(auto & outer_map_pair : CTTCVehMap) {  
  
    for (auto & edge_in_list: CTTCMap){  
        if (edge!=edge_in_list.first &  
            wsm->getIntersectionId()==outer_map_pair.second.first  
            & vehId!=outer_map_pair.first){  
            if (t1<outer_map_pair.second.second.first  
                & outer_map_pair.second.second.first<t2 ){  
                EV <<"t1 "<<t1<<" t2 "<<t2<<std::endl;  
                TTCforRSU* CTTCWarning = new TTCforRSU("Sending Warning!");  
                populateWSM(CTTCWarning);  
                sendMessage=true;  
                CTTCWarning->setEventName("CTTC");  
                CTTCWarning->setExtId(vehId.c_str());  
                CTTCWarning->setExtIdVeh2(outer_map_pair.first.c_str());  
                sendDown( CTTCWarning);  
            }  
  
            if (t1>outer_map_pair.second.second.first  
                & outer_map_pair.second.second.first>t2 ){  
                EV <<"t1 "<<t1<<" t2 "<<t2<<std::endl;  
                TTCforRSU* CTTCWarning = new TTCforRSU("Sending Warning!");  
                populateWSM(CTTCWarning);  
                sendMessage=true;  
                CTTCWarning->setEventName("CTTC");  
                CTTCWarning->setExtId(vehId.c_str());  
                CTTCWarning->setExtIdVeh2(outer_map_pair.first.c_str());  
                sendDown( CTTCWarning);  
            }  
        }  
    }  
}  
  
//vehicle not found in existing list, then add to the list  
CTTCVehMap.insert(make_pair(vehId,pairInt()));  
CTTCVehMap[vehId]=make_pair(wsm->getIntersectionId(),pairxy());  
CTTCVehMap[vehId].second.first=distance/wsm->getSpeedData();  
CTTCVehMap[vehId].second.second=(distance+wsm->getVehLengthData()+1.98)/wsm->getSpeedData();  
}
```

APPENDIX F: Algorithm of calculating aggregated TTC in RSU module

```
TTCforRSU* wsm = check_and_cast<TTCforRSU*>(frame);
EV << "RSU received TTC " << wsm->getTTCCountData() << std::endl;
eventnameRec=wsm->getEventName();
edge=wsm->getRoadIdData();
receivedTTC=wsm->getTTCData();
countTTC=wsm->getTTCCountData();

//.....RSU received TTC.....
if (eventnameRec=="TTC"){
    EV << "RSU received event..... " << eventnameRec << std::endl;
    findHost()->getDisplayString().setTagArg("i", 1, "green");

    it = TTCMap.find(edge);
    if (it != TTCMap.end()) {

        TTCMap[edge].first=(TTCMap[edge].first*TTCMap[edge].second
            +receivedTTC*countTTC)/(TTCMap[edge].second+countTTC);
        TTCMap[edge].second=TTCMap[edge].second+countTTC;
    }

    else {
        TTCMap[edge].first=receivedTTC;
        TTCMap[edge].second=countTTC;
    }
}
```


APPENDIX G: Algorithm of responding crossing conflict TTC request from CVs in RSU module-Part A

```

if (eventnameRec=="reqCTTC"){
    findHost()->getDisplayString().setTagArg("i", 1, "green");
    vehId=wsm->getExtId();
    vehposX=wsm->getCurPosition().x;
    vehposY=wsm->getCurPosition().y;
    intConX=CTTCMap[edge].second.first;
    intConY=CTTCMap[edge].second.second;
    double distance=dist(vehposX, vehposY, intConX,intConY);
    double t1=distance/wsm->getSpeedData();
    double t2=(distance+wsm->getVehLengthData()+1.98)/wsm->getSpeedData(); //1.98 is vehicle length
    cit = CTTCVehMap.find(vehId);
    if (cit != CTTCVehMap.end()){
        CTTCVehMap[vehId].first=wsm->getIntersectionId();
        CTTCVehMap[vehId].second.first=t1;
        CTTCVehMap[vehId].second.second=t2;
        for(auto & outer_map_pair : CTTCVehMap) {
            for (auto & edge_in_list: CTTCMap){
                if (edge!=edge_in_list.first & wsm->getIntersectionId()==outer_map_pair.second.first
                    & vehId!=outer_map_pair.first){
                    if (t1<outer_map_pair.second.second.first
                        & outer_map_pair.second.second.first<t2 ){
                        EV <<"t1 "<<t1<<" t2 "<<t2<<std::endl;
                        TTCforRSU* CTTCWarning = new TTCforRSU("Sending Warning!");
                        populateWSM(CTTCWarning);
                        sentMessage=true;
                        CTTCWarning->setEventName("CTTC");
                        CTTCWarning->setExtId(vehId.c_str());
                        CTTCWarning->setExtIdVeh2(outer_map_pair.first.c_str());
                        sendDown( CTTCWarning);
                    }
                    if (t1>outer_map_pair.second.second.first
                        & outer_map_pair.second.second.first>t2 ){
                        EV <<"t1 "<<t1<<" t2 "<<t2<<std::endl;
                        TTCforRSU* CTTCWarning = new TTCforRSU("Sending Warning!");
                        populateWSM(CTTCWarning);
                        sentMessage=true;
                        CTTCWarning->setEventName("CTTC");
                        CTTCWarning->setExtId(vehId.c_str());
                        CTTCWarning->setExtIdVeh2(outer_map_pair.first.c_str());
                        sendDown( CTTCWarning);
                    }
                }
            }
        }
    }
}

```

APPENDIX H: Algorithm of responding crossing conflict TTC request from CVs in RSU module-Part B

```

else {

    for(auto & outer_map_pair : CTTCVehMap) {

        for (auto & edge_in_list: CTTCMap){
            if (edge!=edge_in_list.first &
                wsm->getIntersectionId()==outer_map_pair.second.first
                & vehId!=outer_map_pair.first){
                if (t1<outer_map_pair.second.second.first
                    & outer_map_pair.second.second.first<t2 ){
                    EV <<"t1 "<<t1<<" t2 "<<t2<<std::endl;
                    TTCforRSU* CTTCWarning = new TTCforRSU("Sending Warning!");
                    populateWSM(CTTCWarning);
                    sendMessage=true;
                    CTTCWarning->setEventName("CTTC");
                    CTTCWarning->setExtId(vehId.c_str());
                    CTTCWarning->setExtIdVeh2(outer_map_pair.first.c_str());
                    sendDown( CTTCWarning);
                }

                if (t1>outer_map_pair.second.second.first
                    & outer_map_pair.second.second.first>t2 ){
                    EV <<"t1 "<<t1<<" t2 "<<t2<<std::endl;
                    TTCforRSU* CTTCWarning = new TTCforRSU("Sending Warning!");
                    populateWSM(CTTCWarning);
                    sendMessage=true;
                    CTTCWarning->setEventName("CTTC");
                    CTTCWarning->setExtId(vehId.c_str());
                    CTTCWarning->setExtIdVeh2(outer_map_pair.first.c_str());
                    sendDown( CTTCWarning);
                }
            }
        }
    }

    //vehicle not found in existing list, then add to the list
    CTTCVehMap.insert(make_pair(vehId,pairInt()));
    CTTCVehMap[vehId]=make_pair(wsm->getIntersectionId(),pairxy());
    CTTCVehMap[vehId].second.first=distance/wsm->getSpeedData();
    CTTCVehMap[vehId].second.second=(distance+wsm->getVehLengthData()+1.98)/wsm->getSpeedData();
}

```

APPENDIX I: Store TTC data received from RSU in TS module

```
EV << "RSU received TTC " << wsm->getTTCCountData() << std::endl;
eventnameRec=wsm->getEventName();
edge=wsm->getRoadIdData();
receivedTTC=wsm->getTTCCData();
countTTC=wsm->getTTCCCountData();
//EV << "RSU received event " << eventnameRec << std::endl;

if (eventnameRec=="TTC"){
    findHost()->getDisplayString().setTagArg("i", 1, "green");

    it = TTCMap.find(edge);
    if (it != TTCMap.end())
    {
        TTCMap[edge].first=(TTCMap[edge].first*TTCMap[edge].second
            +receivedTTC*countTTC)/(TTCMap[edge].second+countTTC);
        TTCMap[edge].second=TTCMap[edge].second+countTTC;
    }

    else
    {
        TTCMap[edge].first=receivedTTC;
        TTCMap[edge].second=countTTC;
    }
}
```

APPENDIX J: Send TTC data to CVs via RSU in TS module

```
if (eventnameRec=="sendTTC"){
EV <<"TS received request for TTC from RSU for edge: "<<wsm->getRoadIdData()<<std::endl;
findHost()->getDisplayString().setTagArg("i", 1, "green");

it = TTCMap.find(edge);
if (it != TTCMap.end()){
TTCMap.find(edge)->second.first;
TTCforRSU* wsm_sentTTC = new TTCforRSU("TTCRes TTC sent");
populateWSM(wsm_sentTTC);
sendMessage=true;
wsm_sentTTC->setTTCData(TTCMap.find(edgeName)->second.first);
wsm_sentTTC->setEventName("TTCResData");
wsm_sentTTC->setExtId(wsm->getExtId());
sendDown( wsm_sentTTC);
EV<<"TS sent requested TTC for edge: "<< traciVehicle->getRoadId().c_str()<<std::endl;
}

else {

EV <<"TS did not find TTC for the edge. "<<std::endl;
TTCforRSU* wsm_sentTTC = new TTCforRSU("TTCRes no Data");
populateWSM(wsm_sentTTC);
wsm_sentTTC->setEventName("TTCResData");
wsm_sentTTC->setExtId(wsm->getExtId());
sendMessage=true;
wsm_sentTTC->setTTCData(TTCMap.find(edgeName)->second.first);
sendDown( wsm_sentTTC);
}
}
```