Adaptive message passing scheme in VANET

Prepared By

NIKUNJ DOSHI

10MCEC06

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May-2012

Adaptive message passing scheme in VANET

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Nikunj Doshi

(10MCEC06)

Guided By

Prof. Vijay Ukani

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2012

DECLARATION

I, **Nikunj Kiranbhai Doshi**, **10MCEC06**, give undertaking that the Major Project entitled **"Adaptive message passing scheme in VANET"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in Institute of Technology of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Nikunj Doshi

Certificate

This is to certify that the Major Project entitled "Adaptive message passing scheme in VANET" submitted by Nikunj K Doshi (10MCEC06), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Vijay Ukani
Guide,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. S.N. Pradhan
Professor and P.G. Co-ordinator M.Tech. CSE
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D. J. Patel
Professor and Head,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr K Kotecha
Director,
Institute of Technology,
Nirma University,
Ahmedabad

Abstract

A Vehicular Ad-Hoc Network, or VANET, is a form of Mobile ad-hoc network, to provide communications among nearby vehicles and between vehicles and nearby fixed equipment, usually described as roadside equipment. In VANET, or Intelligent Vehicular Ad-Hoc Networking, defines an Intelligent way of using Vehicular Networking. VANET integrates multiple ad-hoc networking technologies such as WiFi IEEE 802.11 b/g, WiMAX IEEE 802.16, Bluetooth, IRA, ZigBee for easy, accurate, effective and simple communication between vehicles on dynamic mobility. A vehicle in VANET is considered to be an intelligent mobile node capable of communicating with its neighbors and other vehicles in the network. The goal of the projects is to create new network algorithms or modify the existing for use in a vehicular environment. In the future vehicular ad hoc networks will assist the drivers of vehicles and help to create safer roads by reducing the number of automobile accidents. In the current project we will focus on hybrid model of cluster based VANET and infrastructure based VANET.

In adhoc mode, only clusters are present with each cluster having cluster head. Also all cluster members of each cluster communicate with cluster head. These cluster head communicate with other cluster heads.

In infrastructure mode, Road Side Units (RSUs) are already present and each and every vehicle communicate with these RSU.

In this report we will propose hybrid model, where by in city areas, infrastructure mode will be used and when vehicle moves to highway where RSUs may not be available then in that case, infrastructure mode will switch to ad-hoc mode.

Index Terms - VANET, Cluster, RSU, Infrastructure, Adhoc

Acknowledgements

With immense pleasure, I would like to present this report on the dissertation work related to "Adaptive message passing scheme in VANET". I am very thankful to all those who helped me for the successful completion of Major project and for providing valuable guidance throughout the project work.

I would first of all like to offer thanks to **Prof. Vijay Ukani** (Guide), Institute of Technology, Nirma University, Ahmedabad whose keen interest and excellent knowledge base helped me to finalize the topic of the dissertation work. His constant support and interest in the subject equipped me with a great understanding of blacks and whites of the required architecture for the project work.

Along with him I cannot miss to thank **Dr. S.N.Pradhan**, **Prof. Zunnun Narmawala**, **Prof. Manish Chaturvedi**, and Dharmendra Sutharia whose motivation and work are my driving forces. My sincere thanks and gratitude to **Prof. D.J. Patel**, Professor and Head, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his continual kind words of encouragement and motivation.

I would like to thanks **Dr. Ketan Kotecha**, Director, Institute of Technology, Nirma University, Ahmedabad for providing basic infrastructure and healthy research environment.

I am thankful to Nirma University for providing all kind of required resources. I would like to thank The Almighty, my family, for supporting and encouraging me in all possible ways. I would also like to thank all my friends who have directly or indirectly helped in making this work successful.

- Nikunj Doshi

10MCEC06

Contents

List of Figures

List of Tables

Abbreviation Notation and Nomenclature

VANET ..... Vehicular Ad-hoc NETwork
MANET ..... Mobile Ad-hoc NETwork
OBU ..... On-Board Unit
RSU ..... Road Side Units
ITS ..... Intelligent Transport System
V2I ..... Vehicle to Infrastructure
V2V ..... Vehicle to Vehicle
WAVE ..... Wireless Access in Vehicular Environment
DSRC ..... Direct Short Range Communication
AODV ..... Ad hoc On demand Distance Vector
E2E ..... End-to-End
MOVE ..... MObility model generator for VEhicular networks
NAM ..... Network AniMator
NS ..... Network Simulator
PDR ..... Packet Delivery Ratio
SUMO ..... Simulation for Urban MObility
TCL ..... Tool Command Language
TIGER ..... Topologically Integrated GEographic Encoding and Referencing

Chapter 1

Introduction

1.1 VANET

Vehicular Ad-hoc NETworks are a subset of the Mobile Ad-hoc Networks MANETs which provide wireless communication capabilities between devices in a certain range. In a VANET environment this devices are either stations (STAs) installed in vehicles as on board units (OBUs) or access points (APs) strategically located in fixed points along the road, and hence usually referred to as road side units (RSUs). Vehicular Ad-Hoc Networks have been a very active research topic in the last years due to the very positive impact of their implementation in road safety. For security communications, which require short message exchange, there is no need to implement a seamless handover scheme, as it would imply a rather significant signaling overhead, delaying the communications to a point where the security message may be invalid or the communications to be enabled when the vehicle is close to leaving the service area. However, security communications are not the only goal of VANETs: there are a number of applications that require continuous communications during periods of time that exceed the span available when the mobile node (MN) is traveling through the service area of a given RSU, such as voice over IP (VoIP), video streaming, etc [5]. The aim is the study of the available simulation environments for VANETs, the modeling of the simulation environment and the analysis of the results in order to optimize the IEEE 802.11p protocol parameters in order to provide fast and reliable seamless transmission of message - when the system is not working on what will be later defined as WAVE mode - for the mobile devices while moving between the transmission ranges of different infrastructure nodes.

1.2 Applications

There are four main applications:

- Safety Warning: The VANET promises more safety features.

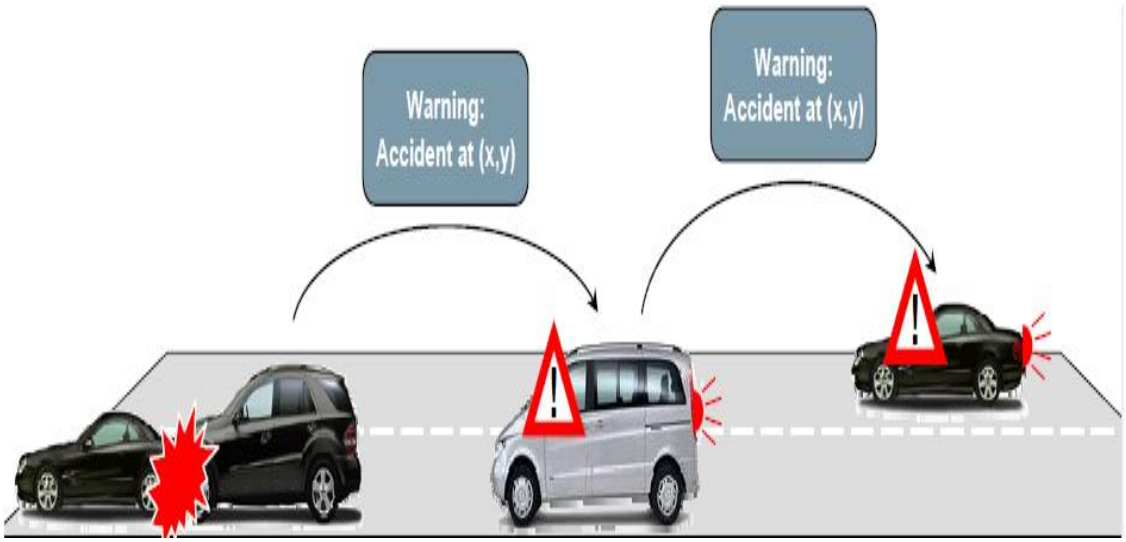

Figure 1.1: Safety Warning [4]

- Traffic Management: The VANET offers more efficient driving.

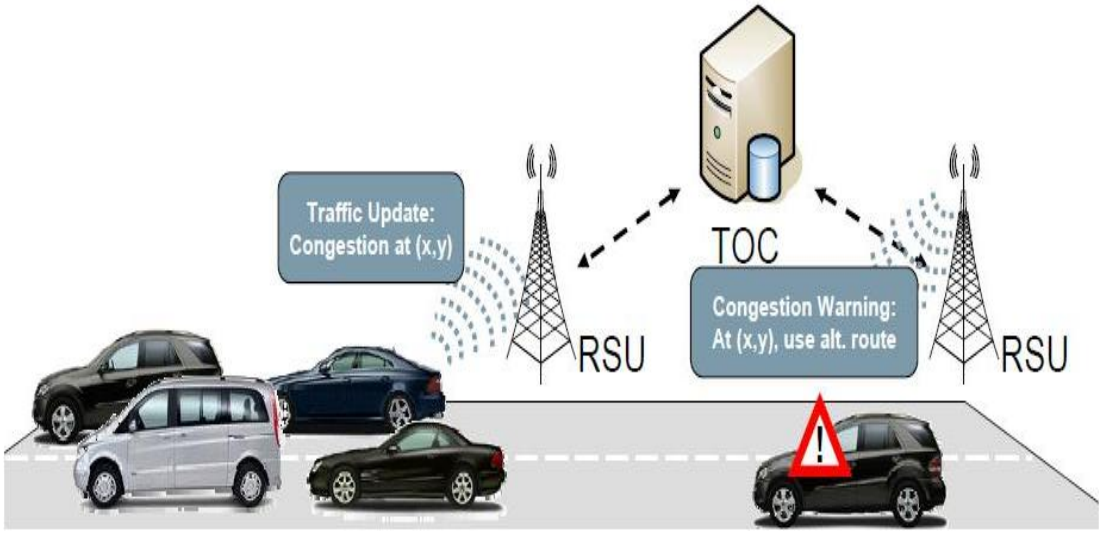

Figure 1.2: Traffic Management [4]

- Enhanced Driver Comfort: VANET offers general message transmission between vehicles.

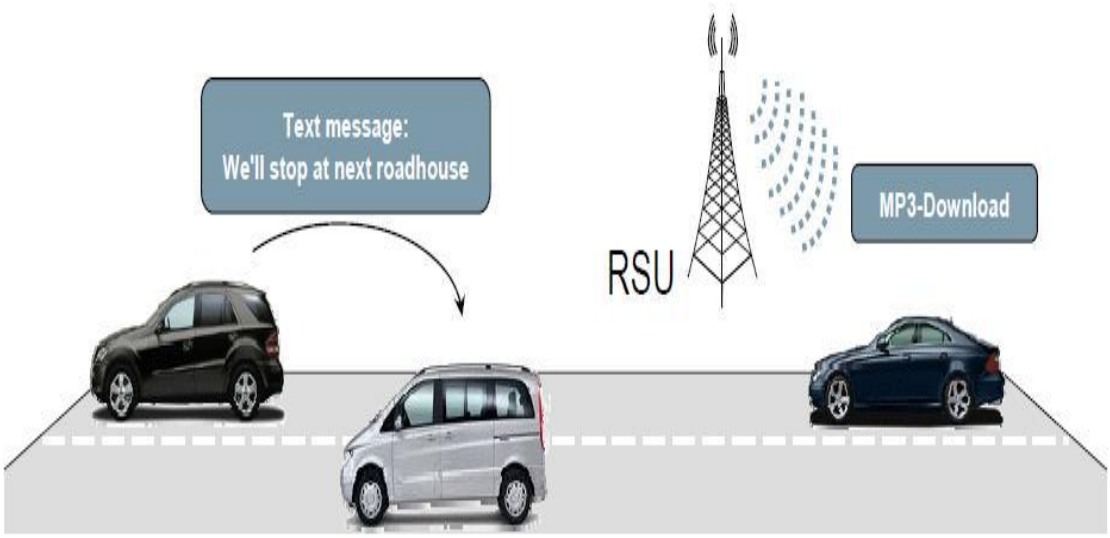

Figure 1.3: Enhanced Driver Comfort [4]

- Maintenance: VANET offers vehicles to send message to the workshop in case the vechicles face any problems. Also software of vehicles can be updated from car manufacturers server.

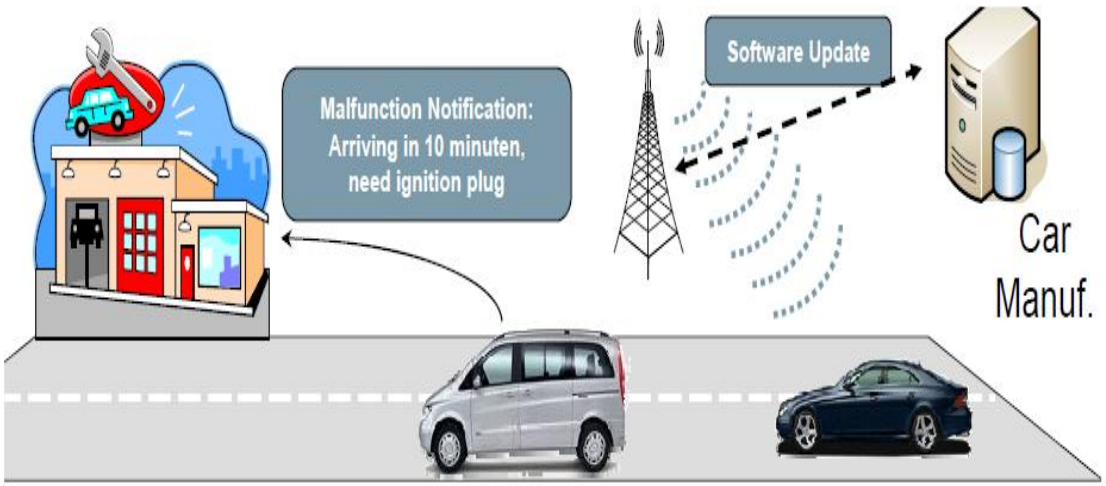

Figure 1.4: Maintenance [4]

1.3 Current Implementations

1.3.1 CARLINK

The aim of this project is to develop an intelligent wireless traffic service platform between cars supported by wireless transceivers beside the road [13]. The primary applications

are real-time local weather data, urban transport traffic management, and urban information broadcasting. Cars have integrated wireless transceivers to communicate with base stations located beside the road. In addition, cars may also communicate between each other as members of an ad-hoc network. Base stations provide real-time information (e.g. local weather) to the cars driving past. At the same time, cars gather real-time data (weather, traffic density) and deliver this information back to base stations. Base stations provide all gathered information to the central unit, which updates its databases and provides information on current traffic conditions, weather etc. back to cars driving past the base stations.

1.3.2 CAR 2 CAR

The mission and the objectives of the CAR 2 CAR Communication Consortium [14] are

- the development and release of an open European standard for cooperative Intelligent Transport Systems and associated validation process with focus on Inter-Vehicle Communication Systems.
- to be a key contributor to the development of a European standard and associated validation process for Vehicle-2-Roadside Infrastructure Communication being interoperable with the specified inter-vehicle communication standard.
- to provide its specifications and contributions to the standardisation organisations including in particular ETSI TC ITS in order to achieve common European standards for ITS.
- to push the harmonisation of Car-2-Car Communication Standards worldwide.
- to promote the allocation of a royalty free European wide exclusive frequency band for Car-2-Car Applications.
- to develop realistic deployment strategies and business models to speed-up the market penetration.
- to demonstrate the Car-2-Car System as proof of technical and commercial feasibility.

1.3.3 WILL WARN

The three-year WILLWARN subproject is developing, integrating and validating a safety application that warns the driver whenever a safety-related critical situation occuring beyond the driver's field of view [15]. This includes the development of on-board hazard detection, in-car warning management, and decentralised warning distribution by vehicle-to-vehicle communication on a road network. Positioning, relevance checks, message transport, and on-board message evaluation will enable a low-cost and reliable solution for wireless local danger warnings.

The key issues of WILLWARN include:

- Improved safety through vehicle-to-vehicle and vehicle-to-infrastructure communication
- High benefit for the user even at low equipment rates using cars as relays for transporting messages in a road network
- Design of a basic system at low cost

1.3.4 Network on Wheels

Your electronic assistant has already informed you about the impending situation and to slow down the car long before the danger comes into sight. The information has been transmitted to you by oncoming cars and the location of the traffic jam was forwarded from car to car through the traffic in front of you. Hence, an alternative route can be immediately calculated in order to safe you and others from delays and stress [16].

1.3.5 Cooperative Vehicle Infrastructure System(CVIS)

Intelligent Co-operative Systems are the next big challenge in automotive electronics and ITS. Intelligent Co-operative Systems that are based on vehicle-to-vehicle and vehicle (V2V) and Vehicle to Infrastructure (V2I) communications hold the promise of great improvements both in the efficiency of the transport systems and in the safety of all road users. Indeed intelligent Co-operative Systems increase the "time horizon", the quality and reliability of information available to the drivers about their immediate environment, the other vehicles and road users, enabling improved driving conditions leading to enhanced safety and mobility efficiency. Similarly, Co-operative Systems offer increased

information about the vehicles, their location and the road conditions to the road operators and infrastructure, allowing optimized and safer use of the available road network, and better response to incidents and hazards [17].

Activities have been started in USA, Japan and Europe. Europe is well placed in the research on autonomous and stand-alone systems that have been researched in EU-funded programmes for over a decade, and have benefited from the support of political initiatives such as eSafety.

1.3.6 SafeSpot

SAFESPOT creates dynamic cooperative networks where the vehicles and the road infrastructure communicate to share information gathered on board and at the roadside to enhance the drivers' perception of the vehicle surroundings. Road accidents prevention via a SAFETY MARGIN ASSISTANT to detect potentially dangerous situations in advance and extend the drivers' awareness of the surroundings in space and time [18].

SAFESPOT is an integrated research project co-funded by the European Commission Information Society Technologies among the initiatives of the 6^{th} Framework Program .

1.4 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, Background, describes general overview of 802.11p and WAVE characteristics. Also, the difference between 802.11 and 802.11 p is described.

Chapter 3, Literature Survey and Important Observation, describes general overview of VANET, Chain Collision Avoidance (CCA), Simple Highway Mobility Model (SHWM) and using multiple channels at same time using CMMP method. Also, summary of the papers have been explained.

Chapter 4, Problem, describes the general problem, Challenges for solving problem and Assumptions taken while solving problem.

Chapter 5, Simulation System, includes study of various open source VANET simulation software. eWorld generates SUMO files after converting .OSM (Open Street Map) file. MOVE rapidly generates realistic mobility models for VANET simulations which is built on top of SUMO. NS-2 is a discrete event simulator useful for testing of networking

environment. Also steps for generating scenarios have been described.

Chapter 6, Proposed Solution, includes the proposed solution for solving the problem in order to achieve the objective of the project work. Detailed flow chart have been described for formation of cluster, role of cluster header and role of cluster member. It will also cover the comparison analysis of the results with respect to No. of Vehicles and different speed in Ahmedabad city scenario of VANET.

Finally, in chapter 7 concluding remarks and future work is presented.

Chapter 2

Background

2.1 802.11p

The protocol IEEE 802.11p is an amendment of the IEEE 802.11-2007 protocol for wireless networks which focuses on the improvement of the performance of the CSMA/CA networks in highly mobile ad-hoc networks.

This protocol is still in draft status pending for approval by the IEEE organization. It is the proposed standard of the DSRC —Direct Short Range Communications— in the vehicular environment by the IEEE organization, and considering the global success of the IEEE 802.11 standard, the derived IEEE 802.11p version has the potential of becoming a single global standard for the communications in Intelligent Transportation Systems (ITS).

The design requirements of the standard are the following:

- Longer ranges of operation: target of aprox. 1000m.
- High relative speed between nodes: up to 500 km per hour. Doppler effect has a high impact in the communication.
- Extreme multi-path environment.
- Operation of multiple potentially overlapping ad-hoc networks.
- High quality of service (QoS).
- Support for vehicular-oriented applications such as safety message broadcasting.

The implementation of IEEE 802.11p is easily understandable considering the other amendments of the IEEE 802.11 protocol [10]. At this point the aspects 802.11p has in common with other of the IEEE 802.11 family protocols will be explained, which provide a very comprehensive way to understand the protocol:

- The physical layer of the IEEE 802.11p protocol is based on the one of IEEE 802.11a: similar orthogonal frequency-division multiplexing (OFDM) based modulation is used and the frequency allocation at the 5GHz band.
- Medium Access Control (MAC) is that of IEEE 802.11: Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).
- For Quality of Service (QoS) the implementation has been adapted from the protocol IEEE 802.11e: Four Access Categories (ACs). Each AC has its own transmission queue with built-in priorities.

IEEE 802.11p does not only implement features from other IEEE 802.11 protocols, as the design requirements are unique in the protocol family. The main qualities which differentiate this protocol, required to meet the essential requirements of high speed and short communication intervals using exclusively the available resources, are:

- Dedicated frequency band (5.850 to 5.925 GHz).
- The channel bandwidth differs: 10MHz per channel instead of 20MHz. This implies the data rate is also halved, alongside the symbol and guard periods being doubled.
- Short Interframe Space (SIFS) times increased in order to prevent long-range issues.
- Polling-based Collision-Free Phase (CFP), present in other versions of IEEE 802.11, is not part of the 11p amendment as it would not be effective in a extremely dynamic environment. Therefore, no guarantee of timely delivered real-time data packets can be offered.
- Introduction of the WAVE operation mode: allows fast communication setup when certain conditions are met.

2.1.1 Physical layer (PHY)

The Federal Communication Commission (FCC) of the U.S. approved 75 MHz bandwidth at 5.850-5.925 GHz frequency band. The available band is divided into seven 10 MHz bandwidth channels (CHs 172 to 184), of which CH 178 is designated as the Control Channel (CCH). CCH usage is limited to broadcast of safety-related data and transmission of control and management messages. The remaining channels are Service Channels (SCHs) available for non-safety related data transmission, including general Internet service in case it is provided [10].

In other geographical areas the channel allocation may slightly differ to provide compatibility with already implemented services. Even though the setting for the PHY configuration are not globally unique, the band used is always alike, making the different allocations a challenge in terms of device distribution and global compatibility, but not critical in terms of research.

2.1.2 Medium Access Control (MAC)

The IEEE 802.11p proposal for MAC layer is a limited subset of the IEEE 802.11p, where only CSMA/CA random access scheme is available for the contention based phase (CBP), with no collision-free phase (CFP), as previously introduced [10].

To prevent privacy issues for the users, whom would be easily traceable when using the standard IEEE 802.11 MAC permanent addressing approach, a OBU in a VANET frequently changes its MAC address: at startup time the system randomly generates a new MAC address, and may change it even during operation, as long as there are no ongoing communication sessions.

2.1.3 QoS

CBP in IEEE 802.11p uses the IEEE 802.11e enhanced distributed channel access mechanism (EDCA) where four ACs provide four QoS levels. To ensure the medium access priority there are different arbitrary inter-frame spacing (AIFS) and contention windows (CWs) defined for each QoS level.

Within QoS classes the collisions are not prevented by the EDCA. After a packet collision has occurred, a back-off time is randomly chosen from the CW interval. The CW size is also inherent to the priority level, giving high priority packets the higher

probability of channel access after a collision. The initial size of the CW is given by the factor CWmin. Each time a transmission attempt fails, the CW size is doubled until reaching the size given by the parameter CWmax.

It has to be reminded that even though this priority system increases the probability of certain packets to access the wireless channel, there are no guarantees that this will happen before a predefined deadline, due to the absence of CFP.

2.2 WAVE mode

IEEE 802.11 MAC operations described above imply a considerable latency. This makes the standard handover approach unsuitable for safety communications, which require the fastest possible data exchange and therefore cannot afford the delay introduced by scanning the medium and handshake to establish the communication [11].

In order to solve this the "WAVE mode" was developed: all the radio channels are set up in the same channel and have the same configuration for that subset of parameters —such as the basic service set identifier (BSSID)- required to enable instant communication. This allows a station (STA) in WAVE mode to transmit and receive data frames with the predefined BSSID value regardless of whether the node belongs or not to a basic service set (BSS), called in this mode a WAVE BSS (WBSS). This procedure allows two nodes to instantly start a communication by only setting up the predefined channel and the wildcard BSSID.

A WAVE station uses a on demand beacon to advertise a WBSS, which contains all the needed information for receiver stations to notice the services offered in the WBSS as well as the information needed to configure itself to become a member of the WBSS. Accordingly, a STA can decide whether to join and complete the process of becoming part of a WBSS by only receiving a single WAVE advertisement beacon with no further interaction with the advertiser.

This approach provides a low overhead WBSS setup by removing all association and authentication phases. However, it requires complementary mechanisms at upper layers, the already introduced P1609.x protocol family among others, to manage the WBSS as well as to provide security, features required to prevent deceptive information, potential attacks within the WBSS such as denial of service, etc.

The lack of an association phase when a vehicle enters the WBSS prevents the MAC

address of the OBU to be passed to the advertiser, further enhancing privacy preventing the RSU to identify a WBSS member. On the other hand, the absence of an association and de-association process prevents any RSU to know whether a OBU has left its transmission range.

It has to be noted that for non-safety communications that require long periods of connectivity —such as VoIP, audio or video streaming. . .— seamless handover capabilities when moving between RSUs transmission ranges are still required.

Chapter 3

Literature Survey and Important observations

3.1 Previous Work

In [1], a CCA application is studied to evaluate the conditions required to retransmit notification alerts in case of emergency deceleration in a platoon of vehicles driving in convoy. Ensuring prompt delivery of emergency brake notifications is crucial for CCA applications. When a collision occurs, cars behind the first colliding vehicle react according to one of two possible patterns: starting to brake due to a notification coming from the first vehicle1, or beginning to decelerate after noticing a reduction in speed of the car immediately ahead. To test the worst case situation, cars do not have the chance of modifying their trajectory of movement. Speed is expressed in m/s, intervehicular distance is varied in steps of two meters, ranging from 6 to 70 meters. Signal transmission power is only accounted for the case with communication capabilities, and takes two values: 22 and 28 dBm. For the drivers reaction time, two cases are studied: one in which vehicles react automatically to the reception of a notification (without any intervention of the driver), and which takes a value of 0.2 s, and the other case, in which after receiving the notification, a driver has to react to this information, thus needing an additional uniformly distributed interval of 0.5-1 s to start braking.

The dependence of collided vehicles against intervehicular distance actually relies upon the speed of vehicles, the reaction time to the event, the transmission power (only for CCA) and to a remarkable extent the percentage of CCA support among vehicles

inside the platoon. Reaction time can increase the number of accidents if the brake system relies on the drivers reaction when compared with the automatic brake system of the vehicle, mostly for middle intervehicular distances. Speed always plays a crucial role in results. Obviously, the higher the speed of vehicles, the greater the number of accidents. Transmission power is also a critical factor when we compare simulations of CCA applications for different values of this parameter.

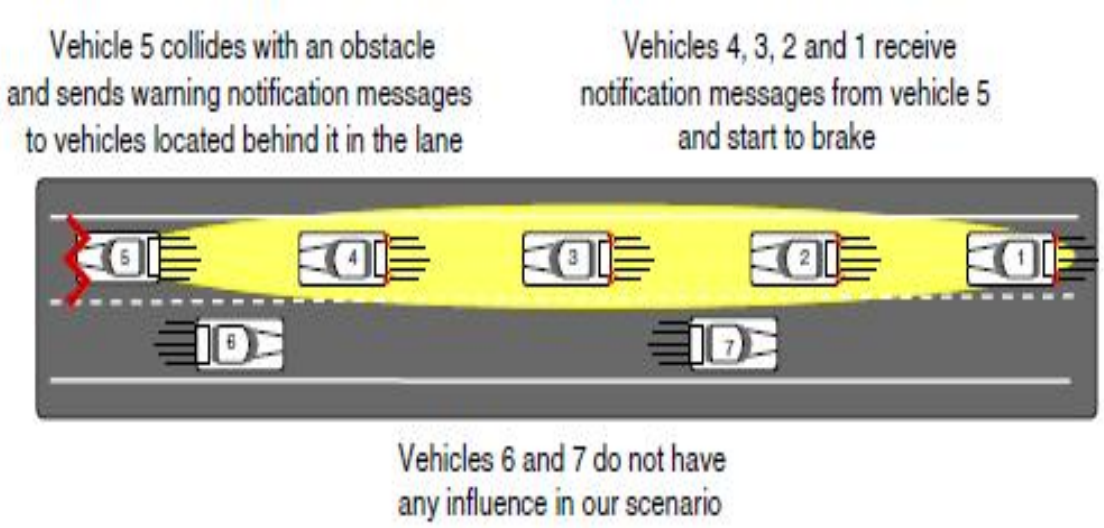

Figure 3.1: Simulation scenario with 802.11p communication capabilities

In [2] the work is focused in the simple highway mobility model. Road Side Units (RSU) is more in the city areas whereas they are less in outside city. When a vehicle enters a city area, it directly sends its request to the Road Side Unit and gets the responses. When the vehicles move outside the city area, it does not get the response from the Road Side Units, due to the limited availability of the roadside unit in that area.

The Clustering algorithm proposed in this model splits the VANET area into a number of clusters. Each cluster has a cluster head. The cluster head may be either RSU or any one of the vehicles with good database storage and access capabilities. A new Cluster head election algorithm is used to select a prominent vehicle as a cluster head. All the cluster heads are periodically updated whenever a new service enters in the network. All the cluster heads are synchronized in a specific time interval. The cluster heads are synchronized to ensure that all the cluster heads have the same value.

In the VANET, the cluster area remains the same and the size of the cluster changes only when the number of vehicles increases suddenly. If the node crosses the boundary

frequently then the cluster head election algorithm often elects a new cluster head and all the cluster heads are synchronized in a specific time interval. The cluster heads are synchronized to ensure that they have latest service updates.

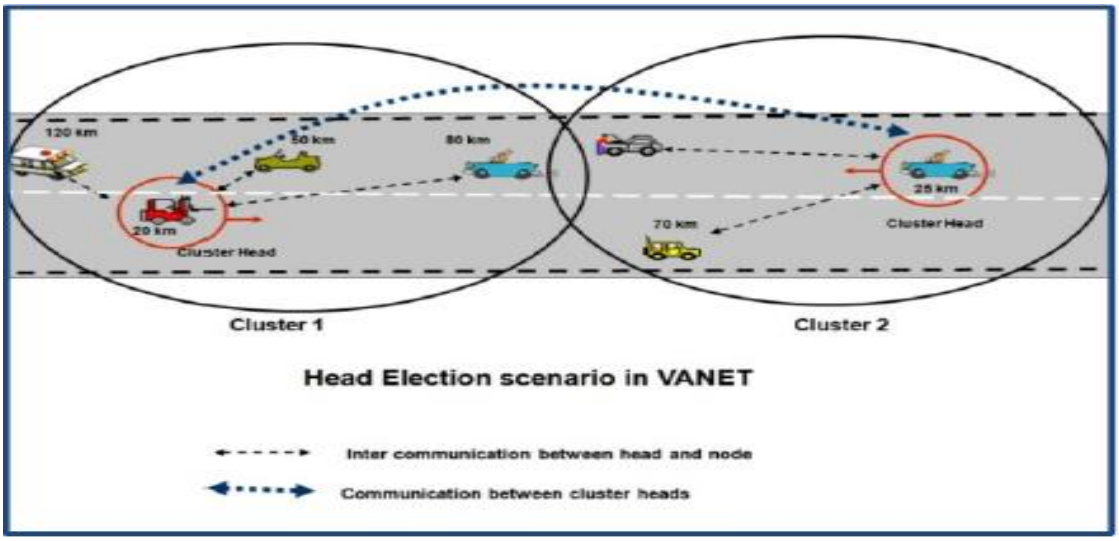

Figure 3.2: Simple Highway Mobility Model (SHWM) Scenario

In [3], a newly proposed multi-channel MAC protocol based on cluster head for contention free considering a feature of VANET. Cluster head manages the channel status table of cluster and periodically broadcasts CUL (Channel Usage List) in control channel. Each station transmits RCA (Request Channel Assignment) to cluster head to get an allocation of data channel in control channel.

Moreover, compares with the single-channel, multichannel has some strong points, such as follows. Firstly, when the single-channel is used, the biggest data size which can be handled will be restrained by the channel wide. But if the multichannel is used, the biggest data size can be enlarged immediately. Secondly, much lower normalized propagation delay will be caused in multichannel than in the single-channel. The normalized propagation delay is a ratio of the propagation per time unit when the data packet is transmitted. As a result, the ratio probability of which the collision can be avoided becomes much higher. Thirdly, when the multichannel is used, the QoS (Quality of Service) can be achieved much easily than single-channel.

In order to solve the following problems and increase the network data handle quantity, the DCA (Dynamic Channel Assignment) was proposed. In DCA, one control channel and N data channels are assigned, each node should check whether the data channel is

available for the data transmition. In order to check the available of the data channel, each node should maintain two kinds of tables such as CUL (Current Usage List) and FCL (Free Channel List).

CUL[i] (i means the list number of the used channel) has CUL[i].host, CUL[i].ch, CUL[i].rel-time, these 3 kinds of field. CUL[i].ch means the data channel which is being used now, the neighbor nodes which using this channel are CUL[i].host. CUL[i].rel-time has the information about the time when the nodes using this channel will change to use other channel. The station which wants to transmit the data will calculate which channel can be used now through the CUL, and then it will transmit the RTS packet along with the FCL.

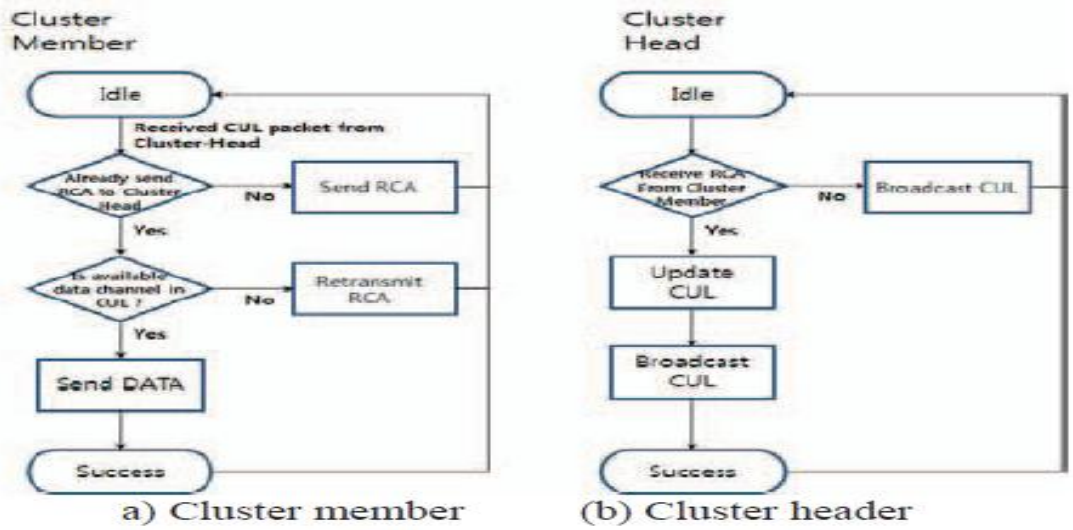

Figure 3.3: CMMP algorithm

3.2 Summary

In [1], the vehicle which is far away from the first colliding vehicle do not receive the warning message, so they cannot start braking their vehicle. In [2], there is no transition when the vehicle moves from city area into the highway area. So at that time cluster should be formed automatically. In [3], after CUL is updated, it needs to be broadcast. If all clusters needs to send data then contention will occur and CUL will be updated every time.

Chapter 4

Problem

When the vehicles move from the area where there are more RSUs to the area where there are less RSUs, or if the signal strength decreases then automatically the cluster should be formed. Similarly when the strength increases then cluster should be broken and it should shift to infrastructure mode where all vehicles should communicate with RSU instead of cluster head.

Also the cluster head should transmit all the buffered data to the RSU when it communicates with the RSU.

Mode.jpg

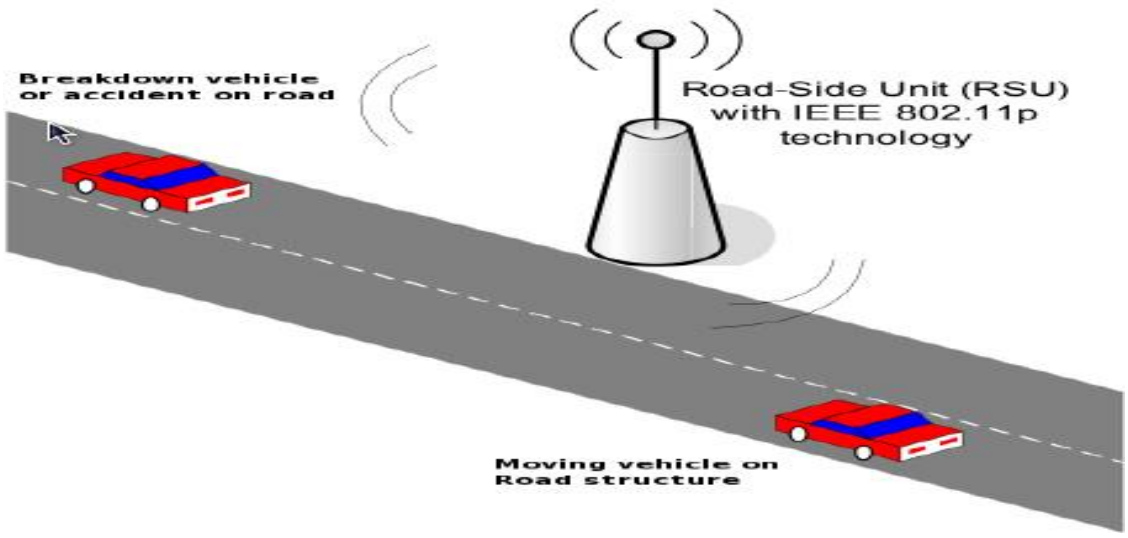

Figure 4.1: Infrastructure Mode

Mode.jpg

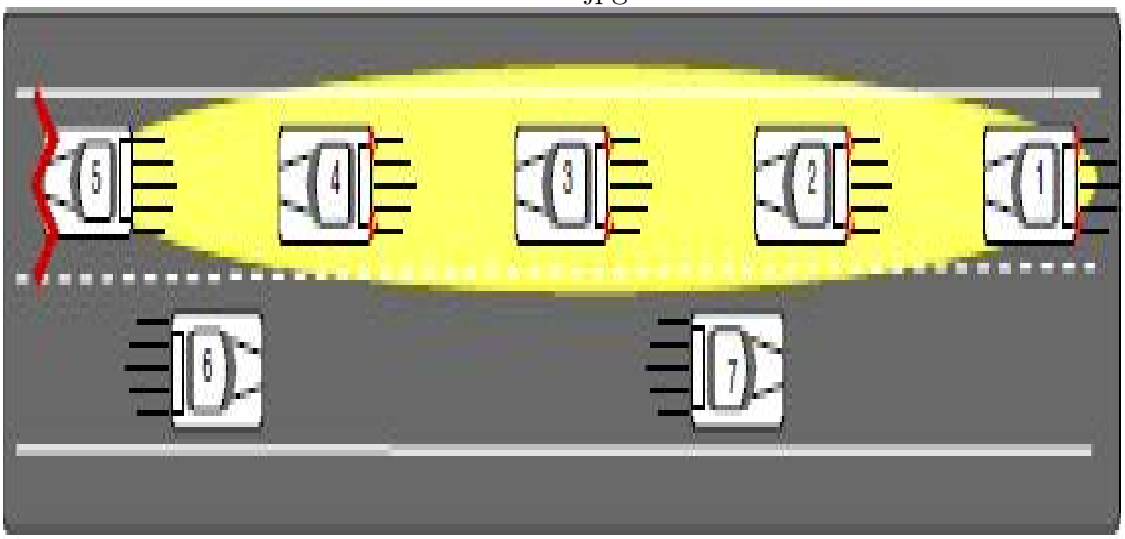

Figure 4.2: Ad-Hoc Mode

4.1 Challenges

1. When the vehicle breaks down then in that case we need to detect the failure. If it is on the side lane then in that case no emergency message needs to be sent. But if it halts on middle of the road then emergency message should be sent to the vehicles coming behind.

2. In wireless environment there are more changes of packet loss than in wired network. Also after specific time if the acknowledgement is not received then we need to retransmit the packet.

3. In case of congestion, there are no mechanism to detect congestion. If there are number of vehicles on petrol pump then it cannot be considered as congestion.

4. In VANET as vehicles move in different speeds, the topology changes very rapidly. In changing topology, it is very difficult to transmit the messages between vehicles.

4.2 Assumptions

1. All the vehicles have onboard units

2. If accident have taken place then the on-board unit will automatically detect accident and send the message to either cluster head or RSU. The OBU will detect

accident by any mechanism. In case of accident the driver may not have time to press the manual button so message will be sent automatically.

3. Also there will be emergency/manual button which driver can press if the vehicle have broken down in middle of the road. In case the fuel gets empty on middle of the road then driver can press manual button which will send emergency message to vehicles coming behind.

Chapter 5

Simulation System

There are three main techniques to analyze the behavior of a system: Analytical Modeling, Computer Simulations and Real Time Physical Measurements. Analytical Modeling may be impossible for complex systems such as the one of this research and Real Time Physical Measurements would require a very long time to be performed and a considerable investment in equipment and resources. Computer Simulation is the only reasonable approach to the quantitative analysis of both traffic and computer networks for this research.

Even though the simulators share a main set of features, each one has its own distinctive characteristics, which will be the parameters which will determine the best option for our simulation requirements.

The goal for any simulator is to accurately model and predict the behavior of a real world environment. Developers are provided with information on feasibility and reflectivity important for the implementation of the system without investing significant time and money. Deploying and testing VANETs involves high cost and time. Simulations of VANETs often involve large and heterogeneous scenarios. Mobility behavior of node in VANET significantly affects simulation results. In a vehicular network, nodes (vehicles) can only move along streets, prompting the need for a road model. In VANETs nodes do not move independently of each other but they move according to well established vehicular traffic models.

VANET mobility generators are used to generate traces of the vehicle's motion that can be usually saved and subsequently imported into a network simulator in order to study the performances of the protocol/application. It is important to generate realistic

movement traces in order to thoroughly evaluate VANET protocols because in general performances depend on the vehicles movement traces. The inputs of the mobility generator include the road model, scenario parameters like maximum vehicular speed, vehicle arrivals and departure rate etc.. The output of the trace contains the location of each vehicle at every time instant for the entire simulation time and their mobility profiles. Examples are SUMO, MOVE, FreeSim and VanetMobiSim.

Network simulators allow researchers to study how the network would behave under different circumstances. Users can then customize the simulator to fulfill their analysis needs. Network simulators are relatively fast and inexpensive compared to cost and time involved in setting up an entire experiment containing multiple networked computers, routers and data links. They allow researchers to test scenarios that might be difficult or costly to emulate with real hardware, especially in VANETs. Network simulators perform detailed packet level simulation of source, destinations, data traffic transmission, reception, route, links, and channels. Network simulators are particularly useful to test new networking protocols or to propose modifications on it. Examples are NS-2, GloMoSim and JiST/SWANS. Most existing network simulators are developed for MANETs and hence require VANET extensions before they can be used to simulate vehicular networks.

Simulation is therefore, the most common approach to developing or testing new protocol for a VANET. Choosing a right simulation tools has been a key step to get accurate prediction of real world environment. This chapter will cover the MOVE, SUMO and NS-2 with its architecture.

5.1 Mobility Simulators

Mobility simulators classification ranges from sub-microscopic to macroscopic depending on the level of detail of the simulation. This is reflected on the smallest entity considered by the simulator. Macroscopic simulators consider the whole traffic flow as the basic entity.

On the other hand, microscopic simulation considers the vehicle the smallest simulation unit. There are simulators which are in-between macroscopic and microscopic, referred as mesoscopic. The latter consider individual vehicles moving between queues, which are the main simulated entity.

There are also sub-microscopic simulators which consider not only each vehicle, but

also the components of them, as the engine or the gear-box, and their parameters. The different granularities are represented in Fig. 5.1.

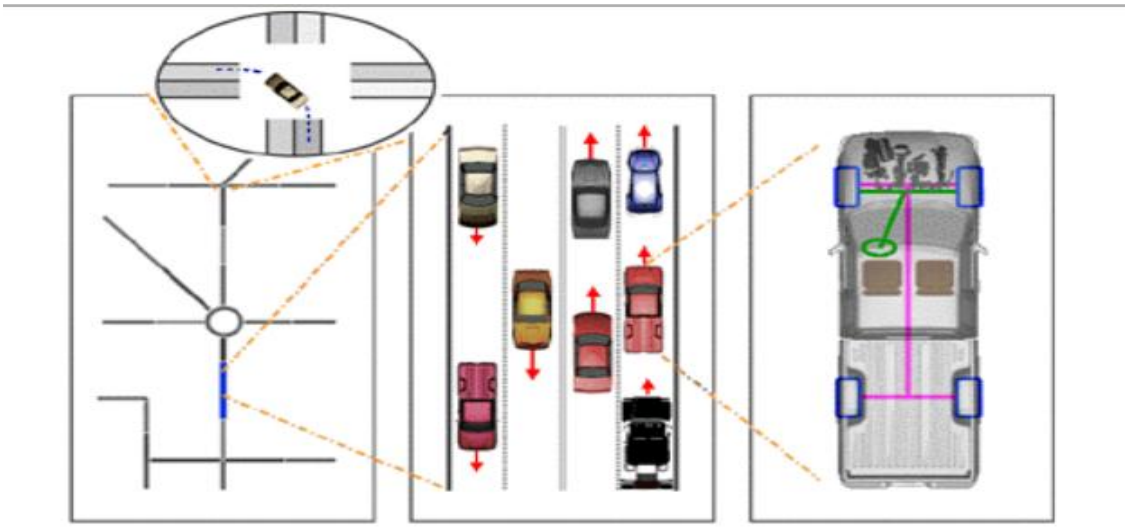

Figure 5.1: Mobility simulator categories. From left to right: macroscopic, microscopic, submicroscopic (within the circle: mesoscopic)

For VANET simulations, where every individual vehicle will be considered a node and the simulation of the vehicle components and their status are not relevant, the most adequate approach to mobility simulation is microscopic. It provides enough resolution of the system as to provide realistic traces, but without the overload of simulating submicroscopic details which would not provide relevant information for this research.

5.1.1 eWorld

eWorld is a framework to enrich mapping data with additional information such as environment events like traffic jams or icy areas and provide this enhanced data to other applications, e.g. traffic simulators (SUMO). eWorld allows to import mapping data in the OpenStreetMap (OSM) format either from files or directly through a graphical interface from their website. Imported data can be explored via panning and zooming. eWorld provides with different information via several panels displaying details of hovered or selected items. Its main capability is the ability to enrich imported data with environment events. Data manipulated this way, can be exported to different simulators, a database or simply be saved in eWorlds own file format.

Step 1: Importing .OSM file

For importing .osm file, click on Import -> Import from .osm file. Select the .osm file

from its location. Click on Import button. Figure 5.2 shows the eWorld environment after importing OSM file. This figure shows the S.G. Highway intersecting with Drive-In Thaltej road of Ahmedabad city.

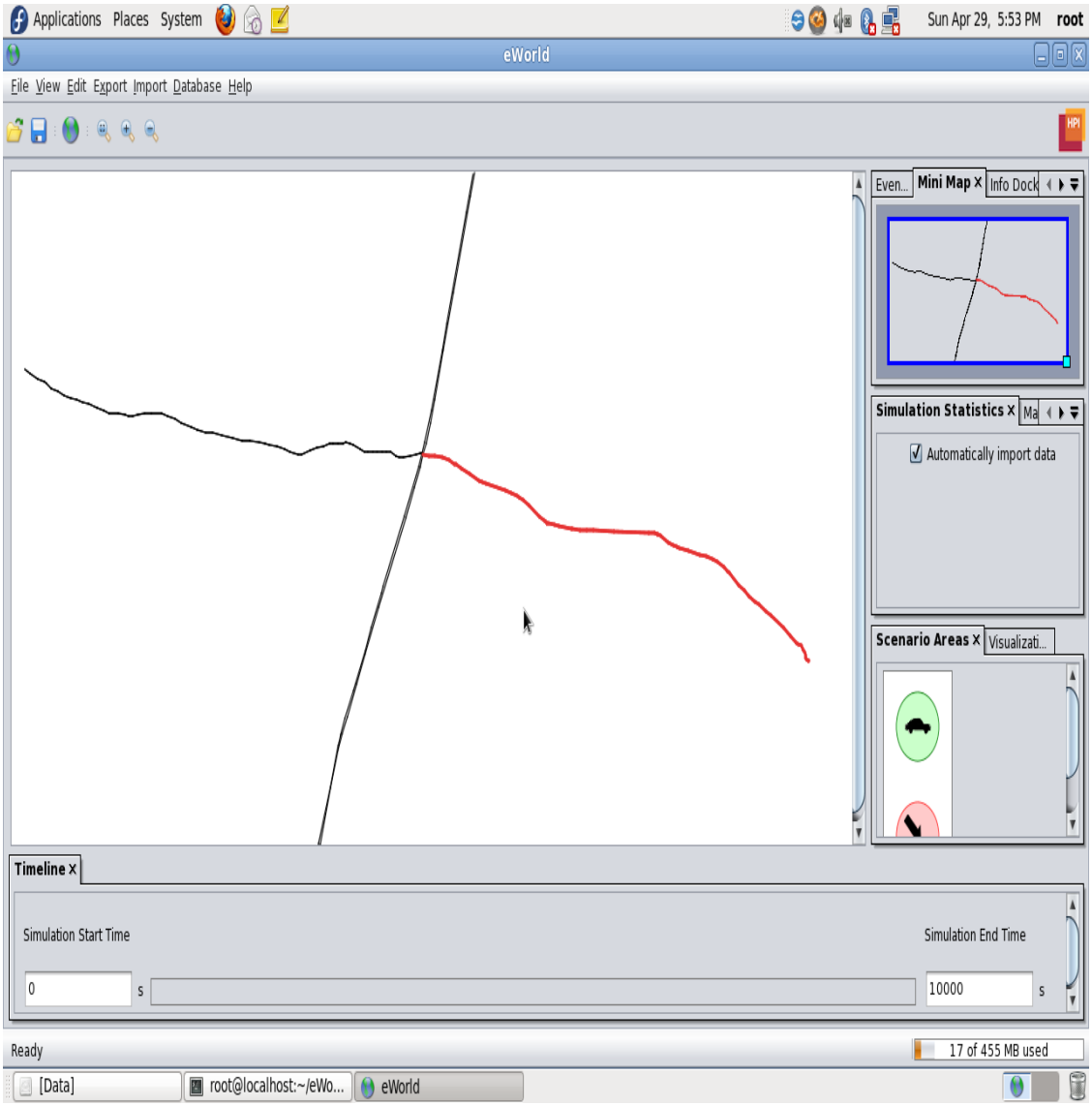

Figure 5.2: eWorld 0.9.3 Environment

Step 2: Exporting to SUMO

For exporting to SUMO, click on Export -> Export to SUMO in main menu. It will prompt for the Location where files need to be created and also the name of files. It will also ask for location of Netconvert needed to convert .osm into .net file. Once you click on Expo .osm will be converted into .net file with .edg and .nod files. Figure 5.3 shows how to export to SUMO.

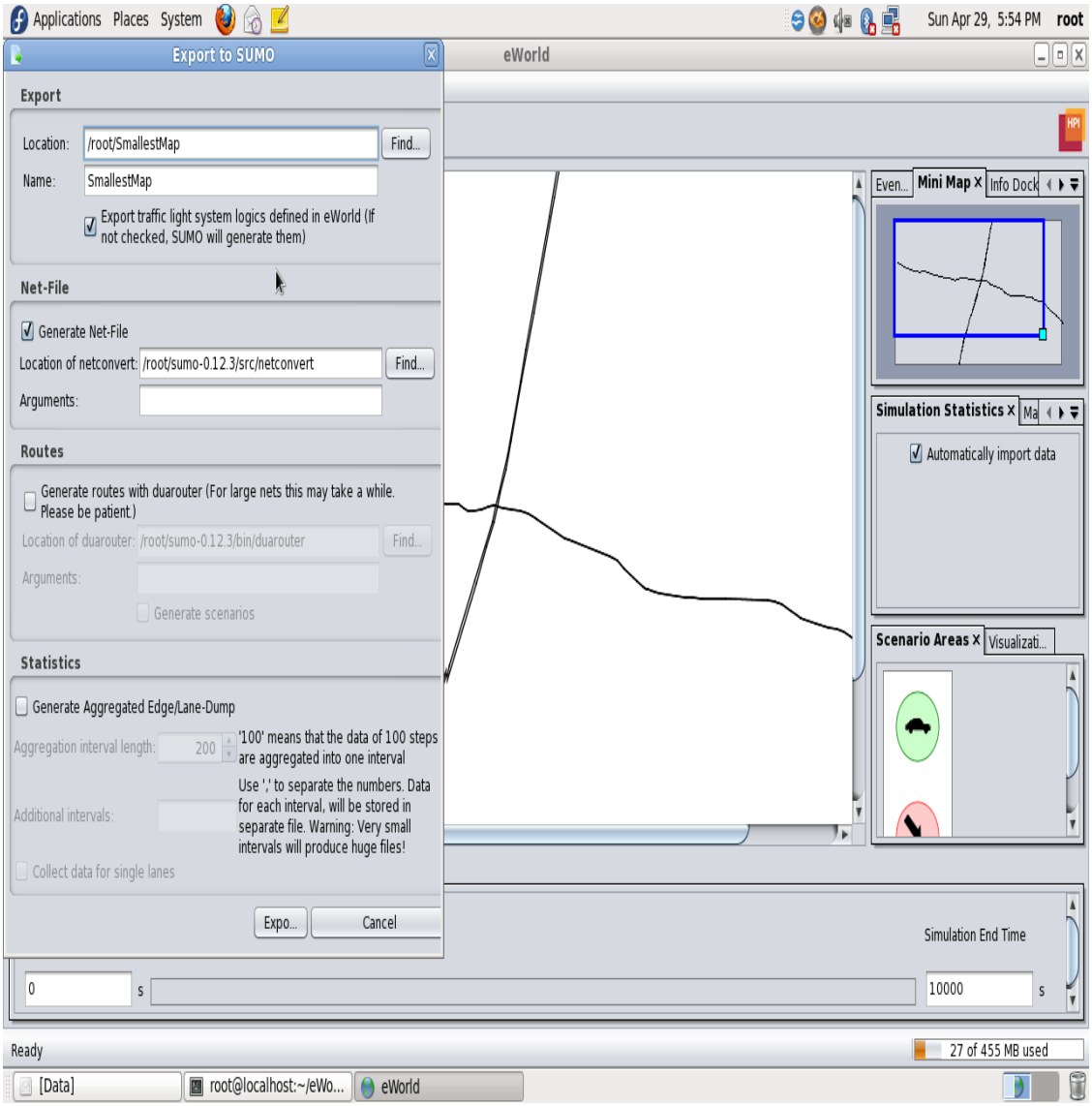

Figure 5.3: eWorld Exporting to SUMO

5.1.2 Simulation of Urban MObility

The development of SUMO started in the year 2000. The major reason for the development of an open source, microscopic road traffic simulation was to support the traffic research community with a tool into which own algorithms can be implemented and evaluated with, without the need to regard all the things needed to obtain a complete traffic simulation, such as implementing and/or setting up methods for dealing with road networks, demand, and traffic controls. By supplying such a tool, the DLR wanted to i) make the implemented algorithms more comparable, as a common architecture and model base is used, and ii) gain additional help from other contributors [12].

"Simulation of Urban MObility", or "SUMO" for short, is an open source, microscopic, multi-modal traffic simulation. It allows to simulate how a given traffic demand which consists of single vehicles moves through a given road network. The simulation allows to address a large set of traffic management topics. It is purely microscopic: each vehicle is modelled explicitly, has an own route, and moves individually through the network.

Simulation of Urban MObility is a microscopic open source traffic simulator based on command-line which incorporates realistic traffic simulation algorithms, with the possibility to have different types of vehicles, different networks -from generated grid, spider-web or random artificial roads to imported VISUM, Tiger, OSM, etc. models, and has a high speed performance. This features match the design criteria the considered when the software was developed: "the software shall be fast and it shall be portable".

SUMO as the complete software package incorporates not only the mobility simulator SUMO, but also a bundle of applications to enhance the generation of networks as well as the import/export capabilities of the software. Another included application is GUISIM, which adds user graphic interface to the SUMO simulator. It has installation packages available for Windows and Linux operating systems, but as it only uses standard C++ and portable libraries it is easy to compile and execute in any other operating system.

5.1.3 MObility model generator for VEhicular networks(MOVE)

MOVE is built on top of an open source micro-traffic simulator SUMO [21]. Simulation of Urban MObility (SUMO) is a microscopic, space continuous and time discrete traffic simulator written in C++ capable to provide accurate and realistic mobility patterns. The project started as an open source project in 2001 with the goal to support the traffic research community with a common platform to test and compare models of vehicle behaviour, traffic light optimization, routing etc.

MOVE [21] is an extension to SUMO that adds a GUI for describing maps, defining vehicle movement and allows the user to import real world map databases such as TIGER [20] and Google Earth. The output of MOVE is a mobility trace file that contains information of realistic vehicle movements which can be immediately used by popular simulation tools such as NS-2 or Qualnet. In addition, by providing a set of graphical user interfaces that automate the simulation script generation, MOVE allows the user to quickly generate realistic simulation scenarios without the hassle of writing simulation

scripts as well as learning about the internal details of the simulator.

Figure 5.4 shows MOVE modules and defines flow how we can create mobility and then simulate it with netwok simulators. Users input information of Map Editor and Vehicle Movement Editor is then fed into SUMO to generate a mobility trace which can be immediately used by a simulation tool such as ns-2 or qualnet to simulate realistic vehicle movements. Users can also visualize the generated mobility trace with SUMO by clicking on the "Visualization" button, as shown in Figure 5.5 . MOVE consists of two main components: Map Editor and Vehicle Movement Editor, as shown in Figure 5.5.

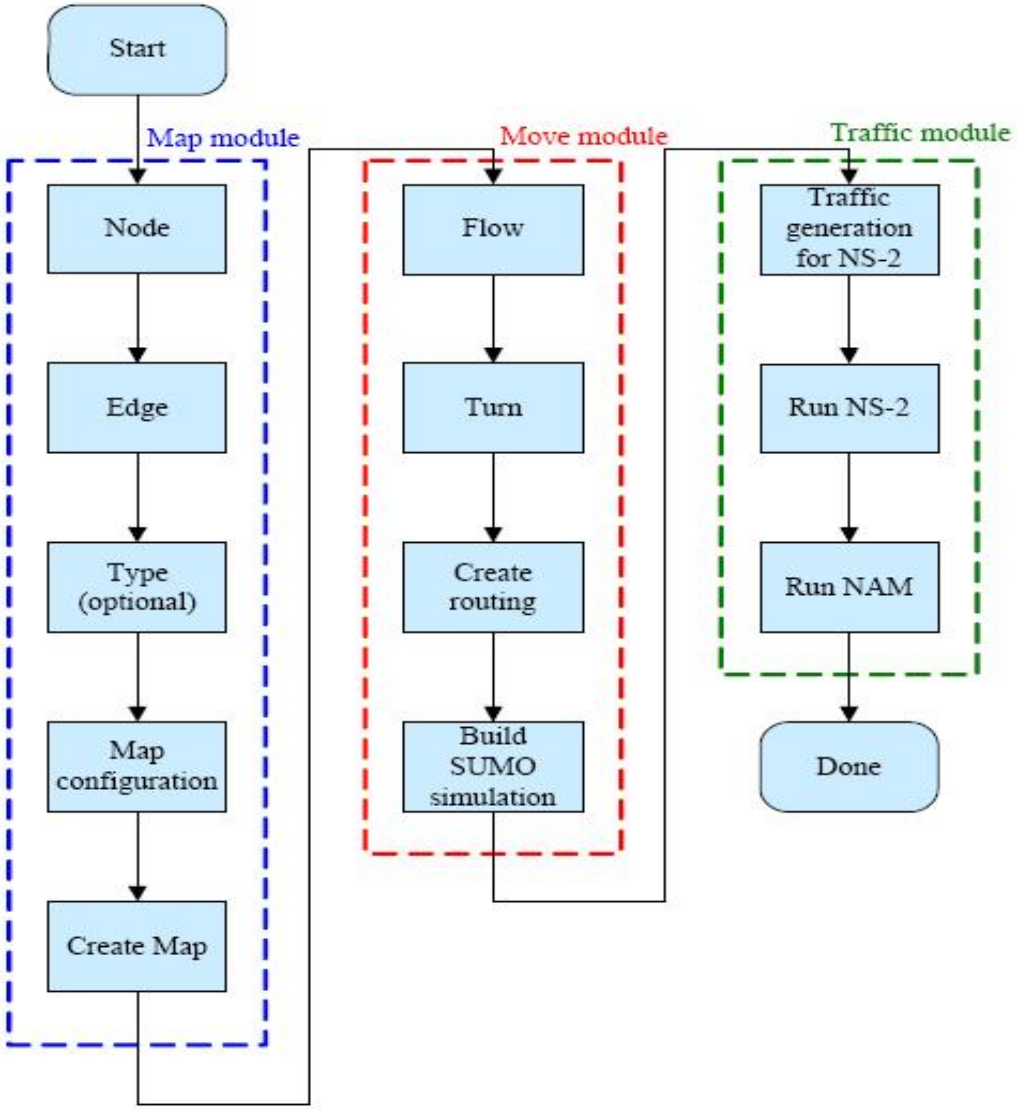

Figure 5.4: Modules of MOVE [21]

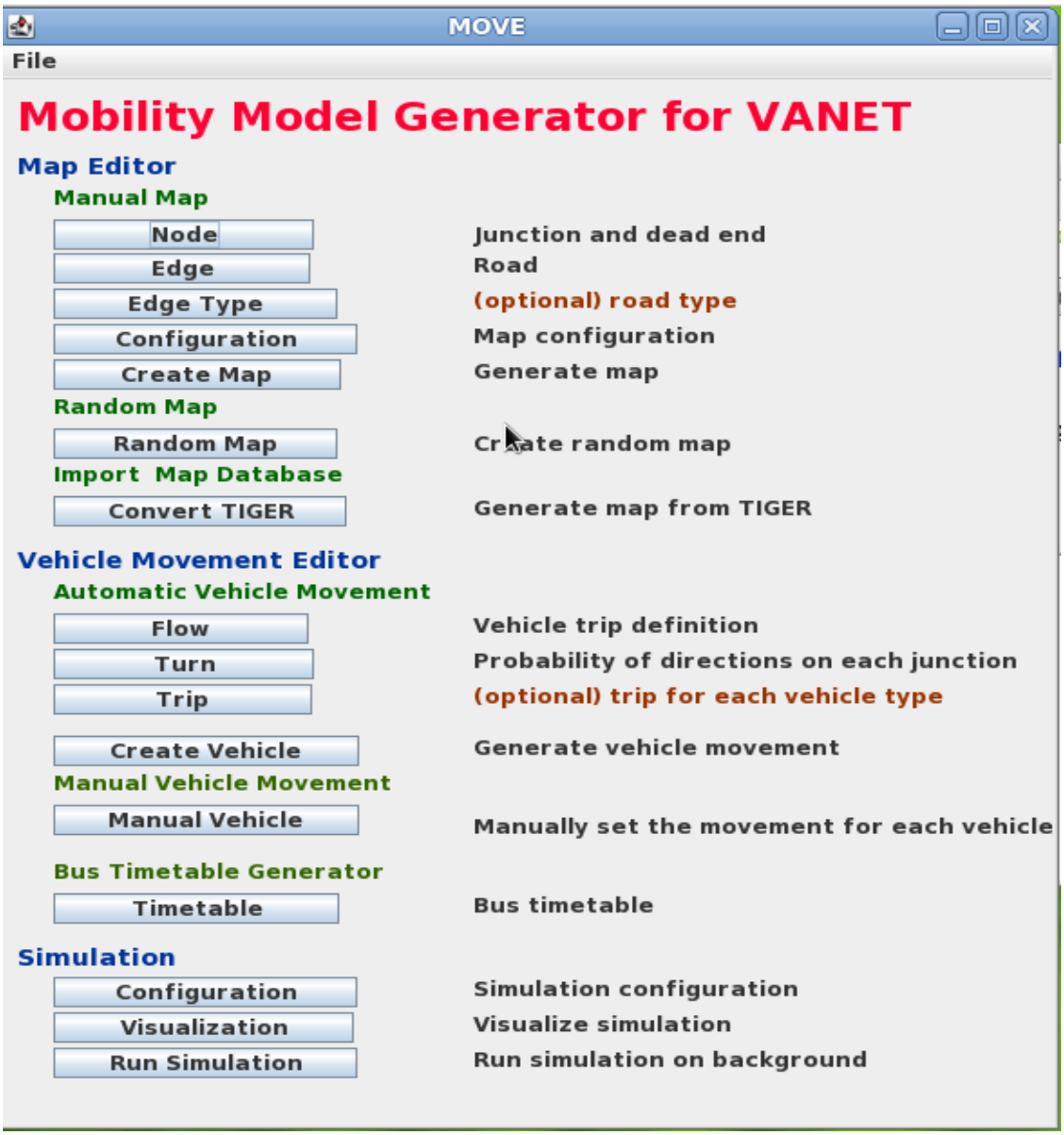

Figure 5.5: Mobility Editor of MOVE

5.2 Network Simulators

5.2.1 NS-2

ns-2 in its different versions is one of the most popular simulation environments for research. It has a hybrid approach to programming simulations with both C++ and an object-oriented version of Tcl scripting called OTcl. This duality can lead to confusion when not familiar with the system, but it proves to be very convenient once the user becomes acquainted with it. The modules are developed using C++, in order to provide

higher simulation speeds by the use of compiled code.

C++ modules are configured and executed via OTcl scripts, which provide the description of the simulation environment and the configuration parameters for each module involved. This OTcl scripts are not compiled but interpreted by the ns software. This makes the set-up of simulations very easy and convenient to batch, as there is no compilation needed to run the scripts, and these contain all the required configuration parameters for the C++ modules.

This duality becomes critical when it comes to develop or modify modules. The modules have two parts: one programmed using C++ and other OTcl. This is required to provide the usability features previously mentioned.

There is an All-in-One package available for most of the releases. These versions include the network simulator, network animator –NAM– and xGraph in the latest version available at the moment of the creation of the package. The installation is not quite straightforward if you are not using one of the systems supported out-of-the-box for that version, but is easy to find community-developed scripts to compile and install the software properly.The installation of extra modules may require additions and modifications in the configuration files in order to work, being usually simple and well documented.

There is an extensive documentation for the network simulator [22] and its modules, in addition to *.tcl example files provided in the distribution in order to both validate the installation of the simulator and learn how to script for the different areas of application of the simulator.

The disadvantage of ns-2 is mainly the limited scalability in terms of number of nodes being simulated, which is not a fixed limit, but it depends on the simulation parameters. This fact is related with the lack of memory management of ns-2: it may require multiple times the amount of memory than some of its alternatives for similar simulations [23]. This is in part a consequence of the use of interpreted software (OTcl), which in 1989 when the ns project was born was a very convenient method to improve the simulation work-flow. However, at present, when the compilation process is not time-consuming, it is considered an unnecessary legacy burden when conducting large simulations.

Another important disadvantage has already been introduced when stated that ns-2 is in its different versions the most used software: not all modules are updated and valid for all the versions. There have been different points in the development where a number

of modules stopped performing properly, so there is a considerable number of research executed with older versions as those are able to execute the modules required by the developers. Outdated versions lack general improvements and patches on different parts of the software [24] which may influence the simulation results and their validity.

5.3 Configuring MOVE

This is the main page of MOVE. First we need to generate topology and vehicles movement using mobility model.

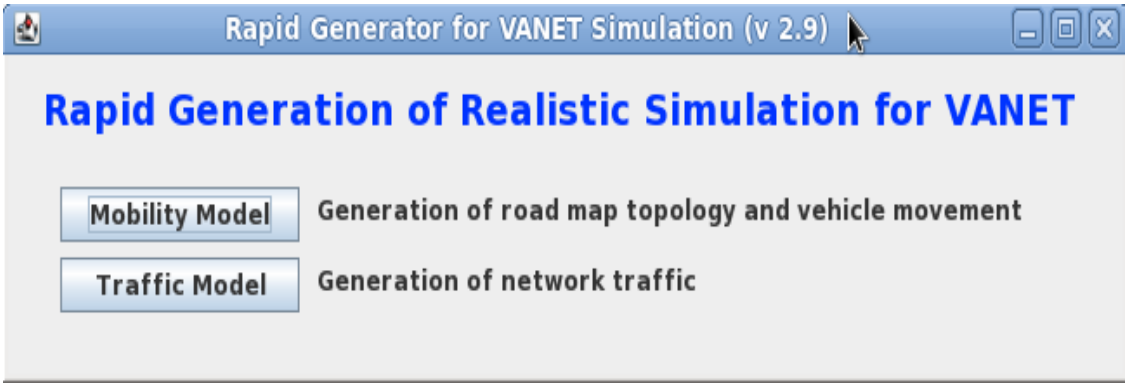

Figure 5.6: Main screen of MOVE

5.3.1 Mobility Generation

The Map Editor is used to create the road topology. It provides three different ways to create the road map - the map can be manually created by the user, generated automatically, or imported from existing real world maps such as publicly available TIGER database from U.S. Census Bureau. Manual generation of the map requires inputs of two types of information, nodes and edges. A "node" is one particular point on the map which can be either a junction or the dead end of the roads. The junction nodes can be either normal road junctions or traffic lights. The edge is the road that connects two points (nodes) on a map. The attributes associated with an edge include speed limit, number of lanes, the road priority and the road length.

The road map can also be generated automatically without any user input. Three types of random maps are currently available as grid, spider and random networks. There are some parameters associated with different types of random maps such as number of grids and the number of spider arms and circles. Also we can generate a realistic map by

importing real world maps from publicly available database from internet.

Step 1: Using Flow definition (Example1.flow.xml))

Already <name>.nod.xml, <name>.edg.xml and <name>.net.xml files are generated

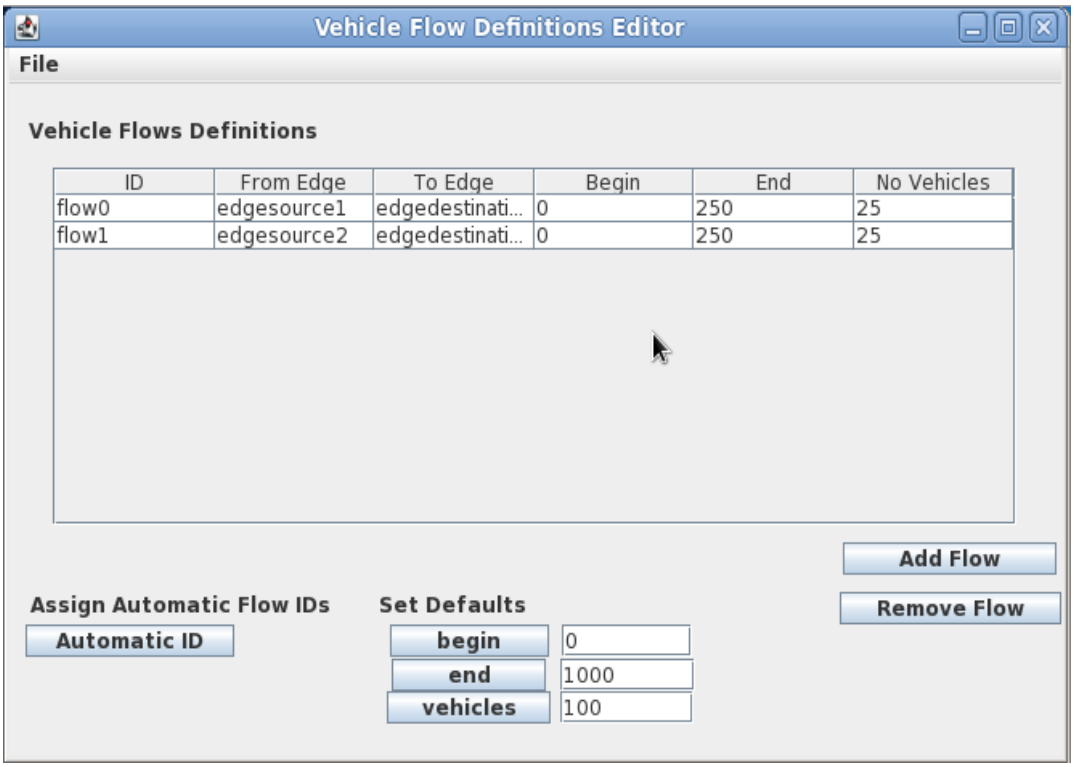

Figure 5.7: Generating Flow definition

from eWorld. So now flow should be defined for vehicles. Simply select "Flow" from MOVE main menu. This editor will specify the groups of vehicle movements flow on simulation. Using these groups of flows, movements of the vehicles will be generated between the two specified edges. For example if you have: edge1 (connected to node 1 and 2) edge2 (connected to node 2 and 3) edge3 (connected to node 3 and 4) If you want a car to move from node 1 to 2 to 3 (1-2-3), you must assign the flow to be from edge1 to edge3 (NOT edge1 to edge2). If you assign it from edge1 to edge2, it will only move halfway (from 1- 2). The movement pattern is all ways from the beginning node of a source edge to the beginning node of a destination edge. Save the file as <name>.flow.xml (Example1.flow.xml).

Step 2: Automatic vehicle movements (Example1.rou.xml)

Select your previously created map file (e.g. <name>.net.xml). Specify your output

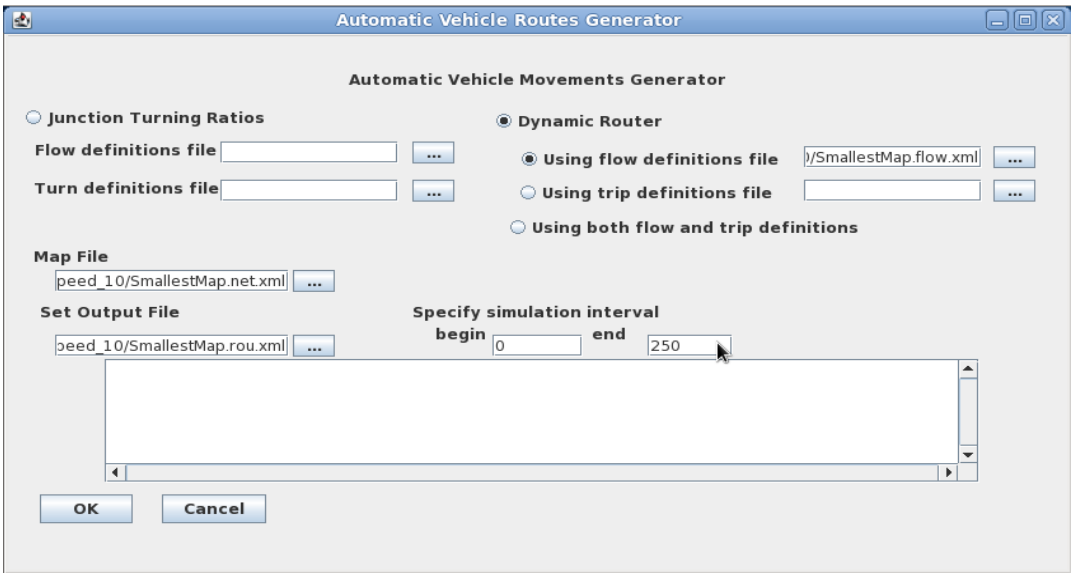

Figure 5.8: Generating vehicle movements

file location and name it as <name>.rou.xml (Example1.rou.xml). Also select previously created flow.xml in Flow Definitions File. Set the beginning and end time of simulation. Finally click OK. The rou.xml file will be automatically generated.

Step 3: Simulation setup (Example1.sumo.tr and Example1.sumo.cfg)

After the map and movement files are generated, you need to specify the configurations of

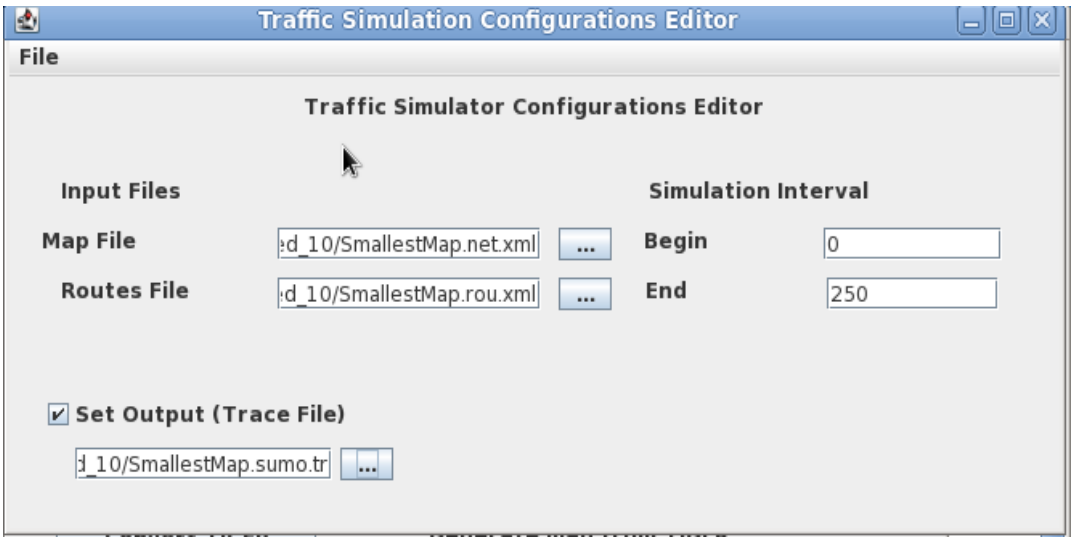

Figure 5.9: Generating configuration file for SUMO

the simulation. Select "Configuration" at the bottom on MOVE main menu. Specify the

<name>.net.xml (i.e Example.net.xml) and <name>.rou.xml (i.e. Example1.rou.xml) location and specify the beginning and end time of simulation. For creating the trace file, we need to check the checkbox and specify trace output name (e.g. <name>.sumo.tr). Then save the file as <name>.sumo.cfg (i.e. Example1.sumo.cfg).

Step 4: Visualize Simulation

Now, you can select "Visualization" see the actual movements on vehicle. So the SUMO window will open. This is the map or net file which has been created using nodes and edges. Now set the delay to 50/100 ms depending on the speed with which vehicles should travel on the map. In this topology the vehicles are generated from top left corner of the topology. They are seen as yellow triangles on the edges/road of topology. Also on junction there are traffic signals seen as red or green. We can zoom in or zoom out to view the topology.

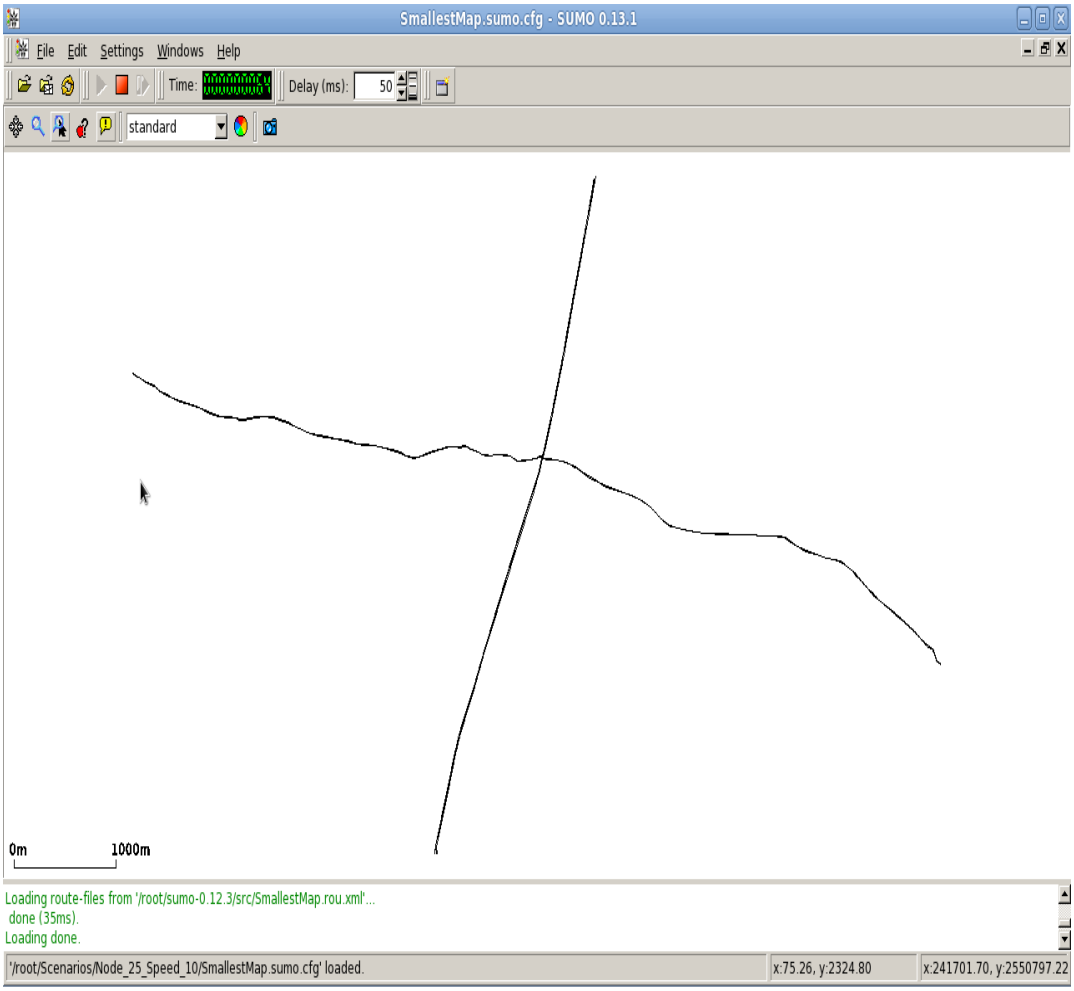

Figure 5.10: Visualization of Topology created

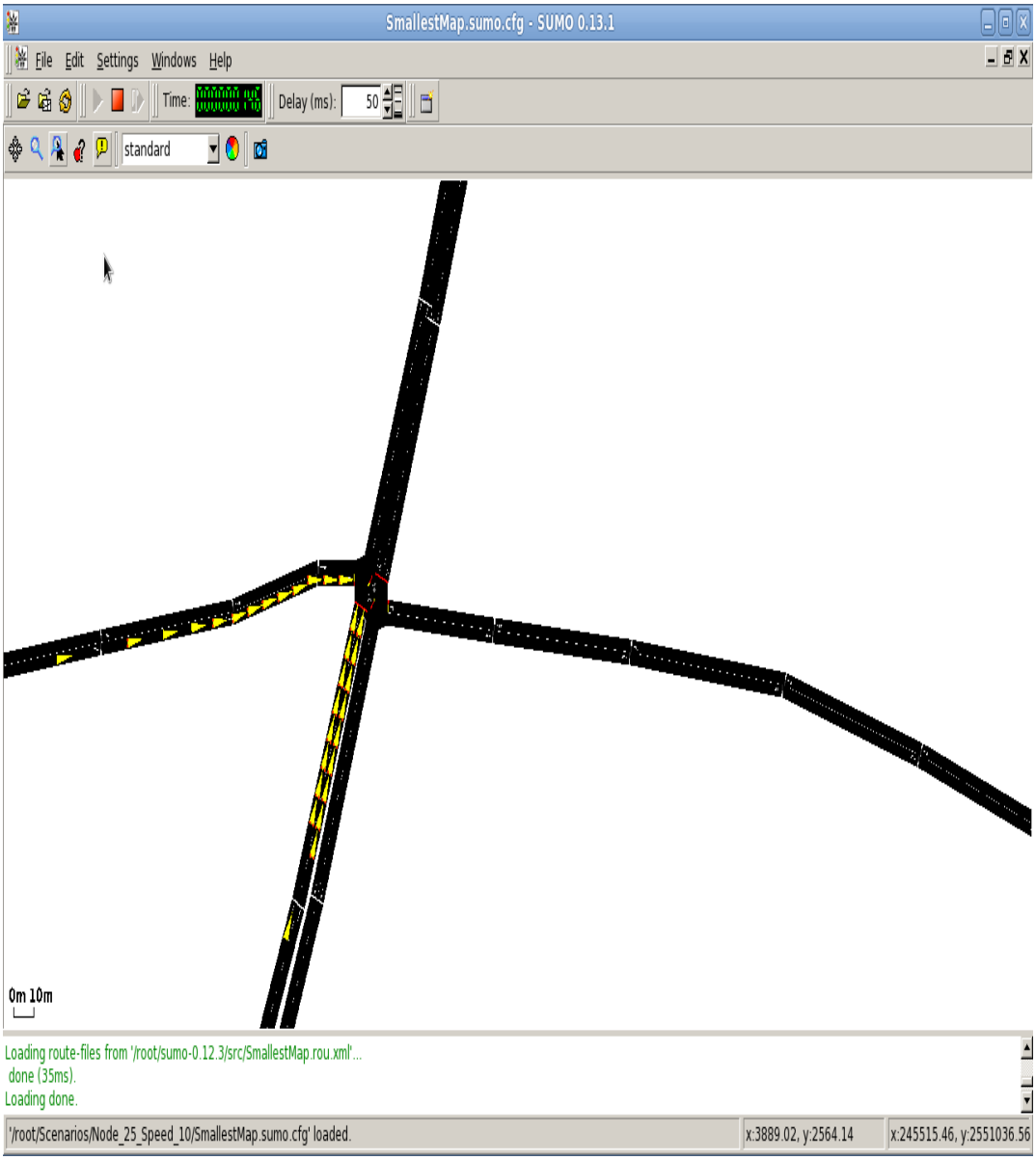

Figure 5.11: Visualization of Topology created with vehicles moving at time 146 sec showing Traffic Junction

5.3.2 Traffic Model Generation

First, Select "Traffic Model" from MOVE main menu entering traffic model. The traffic model generator consists of two main sections: for NS-2 and Qualnet. Right now, MOVE only supports static mobility available. We will try to deploy dynamic mobility in future.

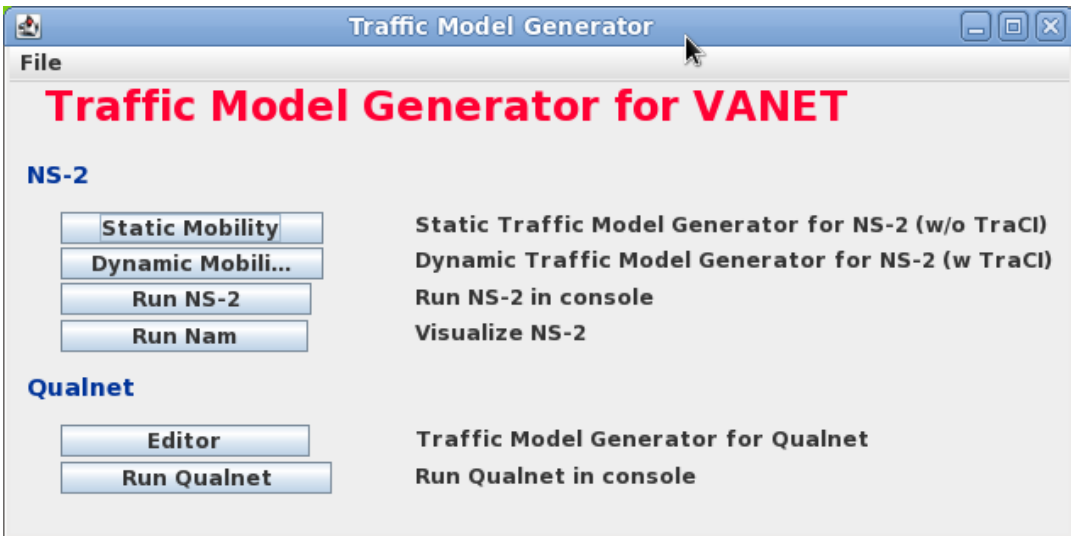

Figure 5.12: Traffic Mobility Model of VANET

Step 1: Traffic Model Generator for NS-2 (Example1.tcl)

This editor will generate the traffic simulation file (a tcl file) for NS-2 simulation tool. First import MOVE Trace (i.e. Example1.sumo.tr) and .net.xml (i.e. Example1.net.xml) file for script generator. After loading is done, specify the options of the TCL simulation. For the example choose NAM trace file and specify the file location. Then we add the connections between each mobile node specified in the table on the left side and assign the transport protocol (e.g. tcp/udp).

In "mobile nodes starting positions" table, you can see different time, node ID, and initial position. You can refer this information to add connection in right-side. Note, the node ID depend on the name of flow in SUMO. For example, if a flow calls test0, the generated node ID should be called test0_0, test0_1, , test0_x, where x is index in the flow. Please check the node ID. If you input a non-exist node in the table, the TCL file will have not correct node ID. When you are done, select File->Save or Save As and put it as <name>.tcl (Example1.tcl).

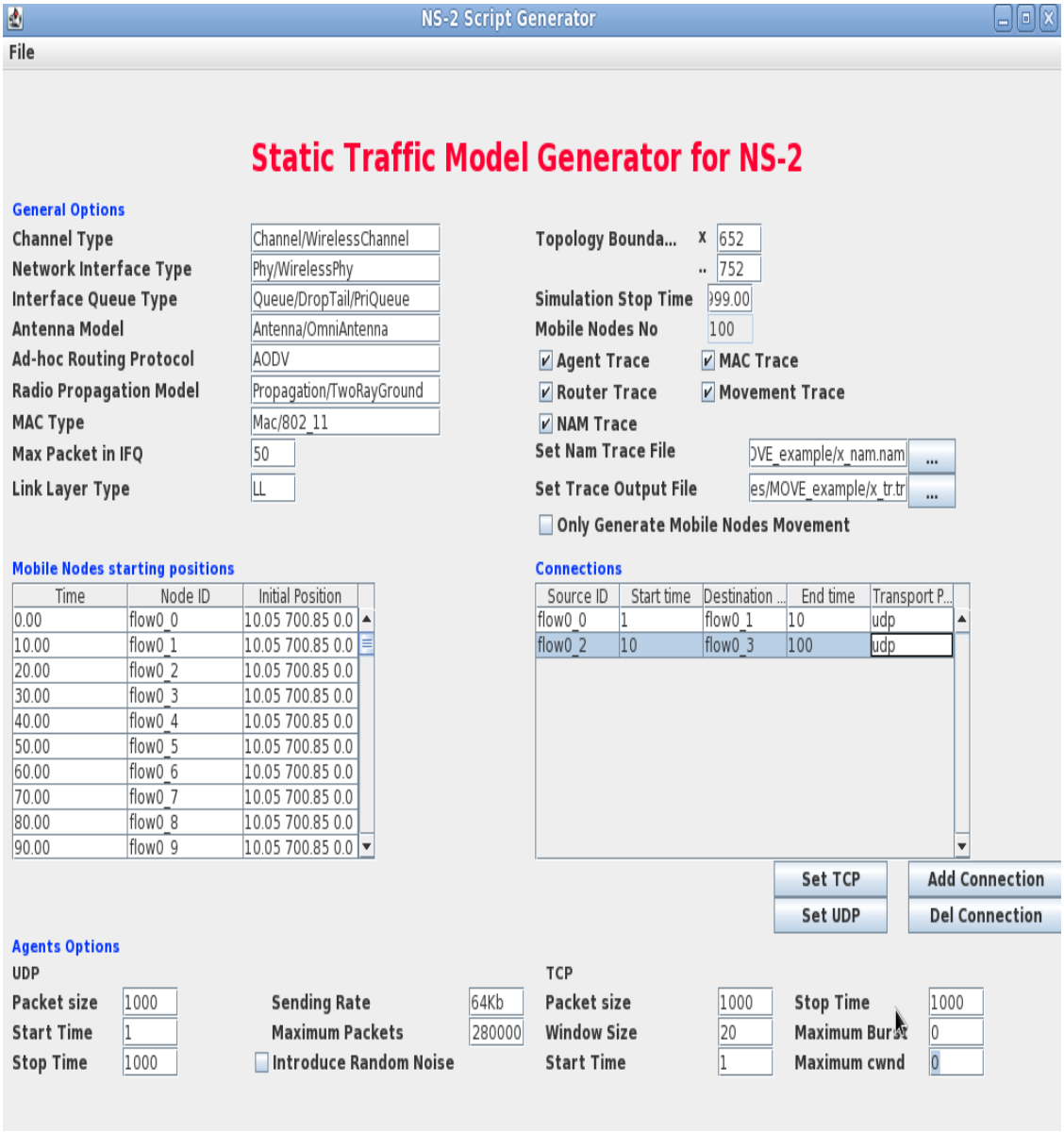

Figure 5.13: Static Traffic Model Generator for NS-2

Step 2: Run NS-2 in background console (Examaple1.nam and Example1.tr)

After, select "Run NS-2" runs NS-2 in console. Finally, run the simulation with <name>.tcl (Example1.tcl) After that, you can either run the NS-2 script in your own shell or using the program's NS-2 script runner (by choosing the Run NS-2 button from the traffic model main menu).

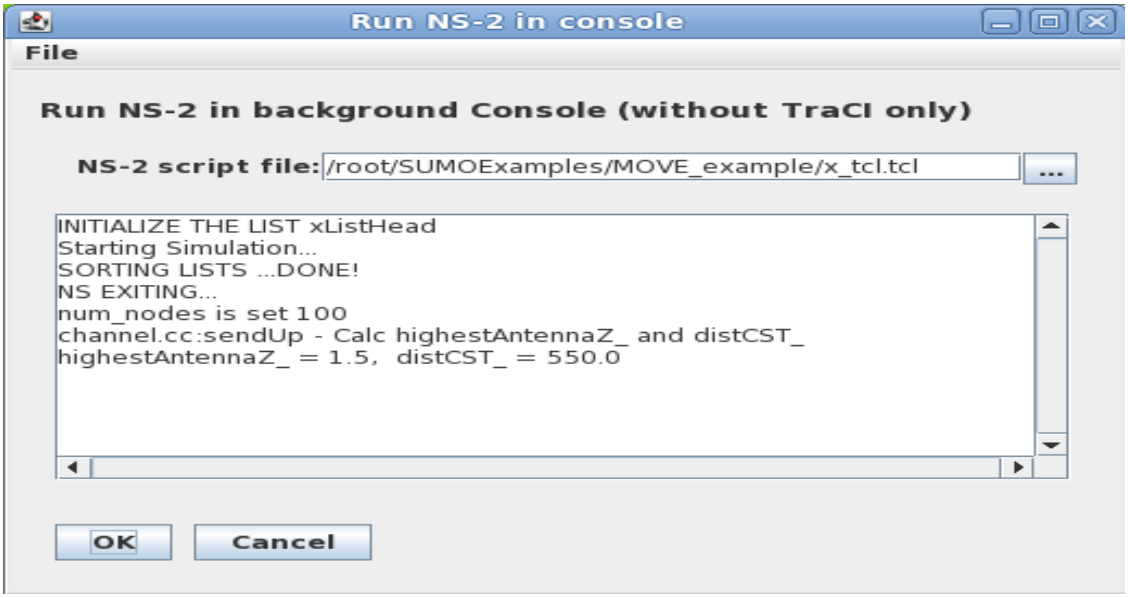

Figure 5.14: Generating NAM file from TCL file

Step 3: Visualize NS-2

Finally, you can call the NAM trace runner from the main menu.

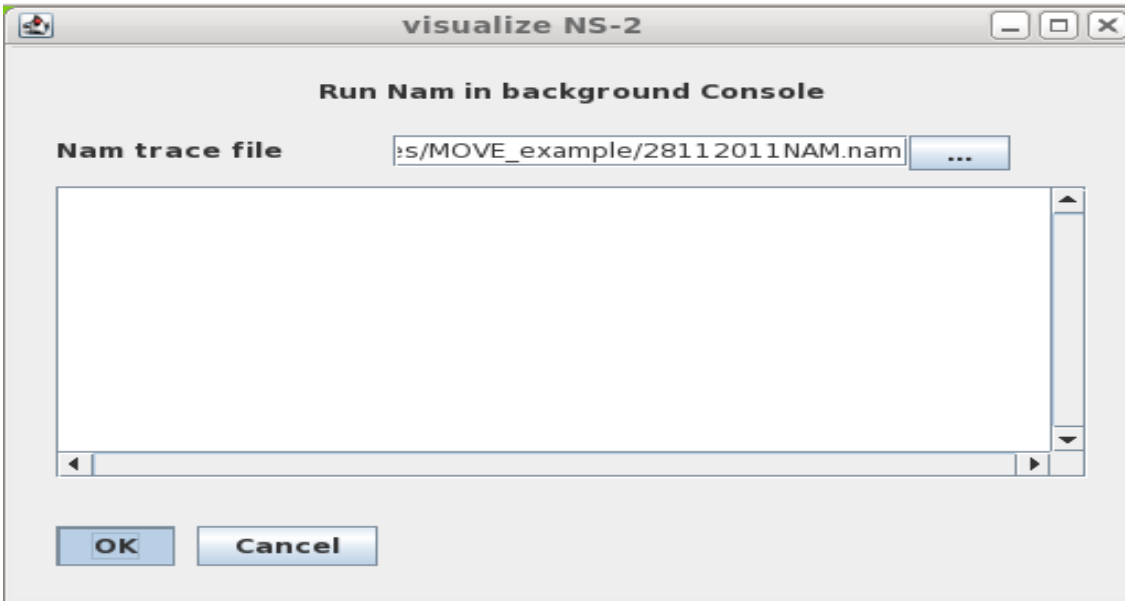

Figure 5.15: Generating NAM file from TCL file

(a) Example of Vehicle to Vehicle communication

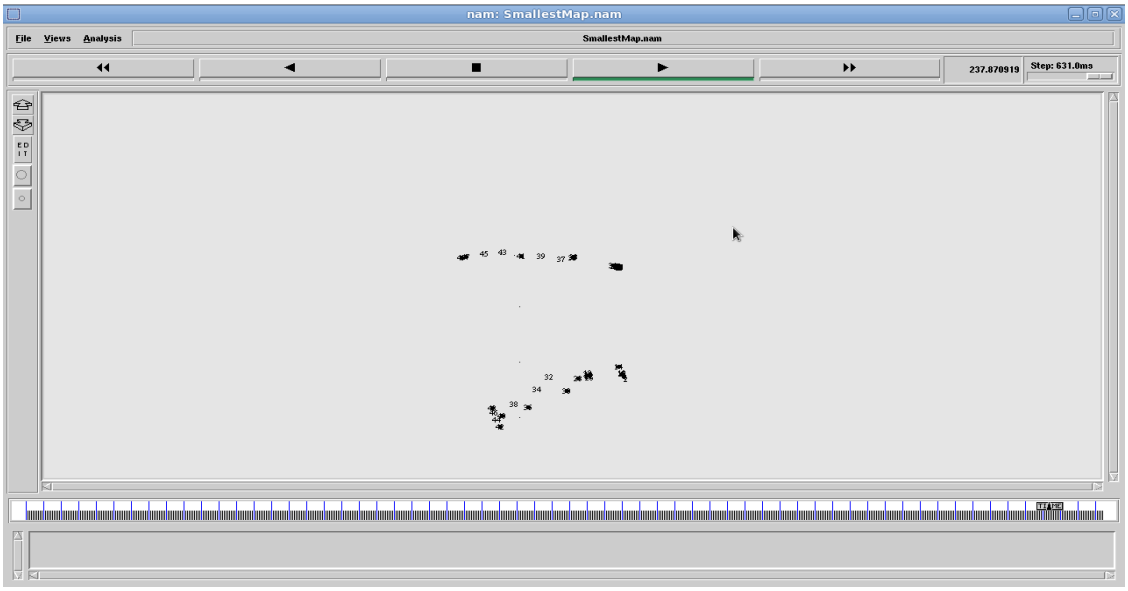

Figure 5.16: Vehicle to Vehicle communication

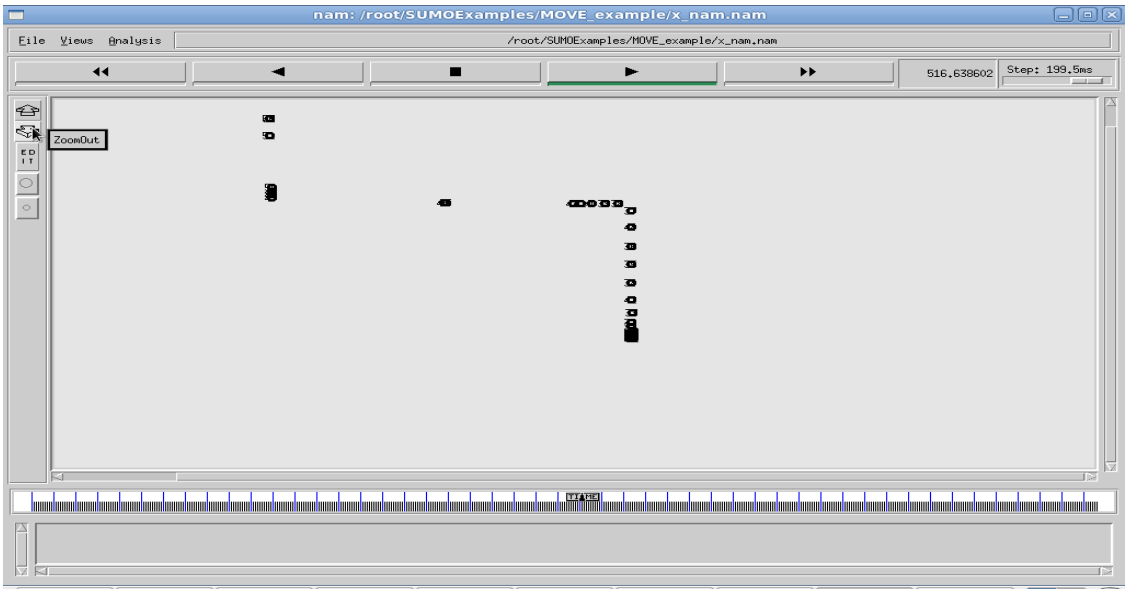

Figure 5.17: Vehicle to Vehicle communication

(a) Example of Vehicle to Infrastructure communication

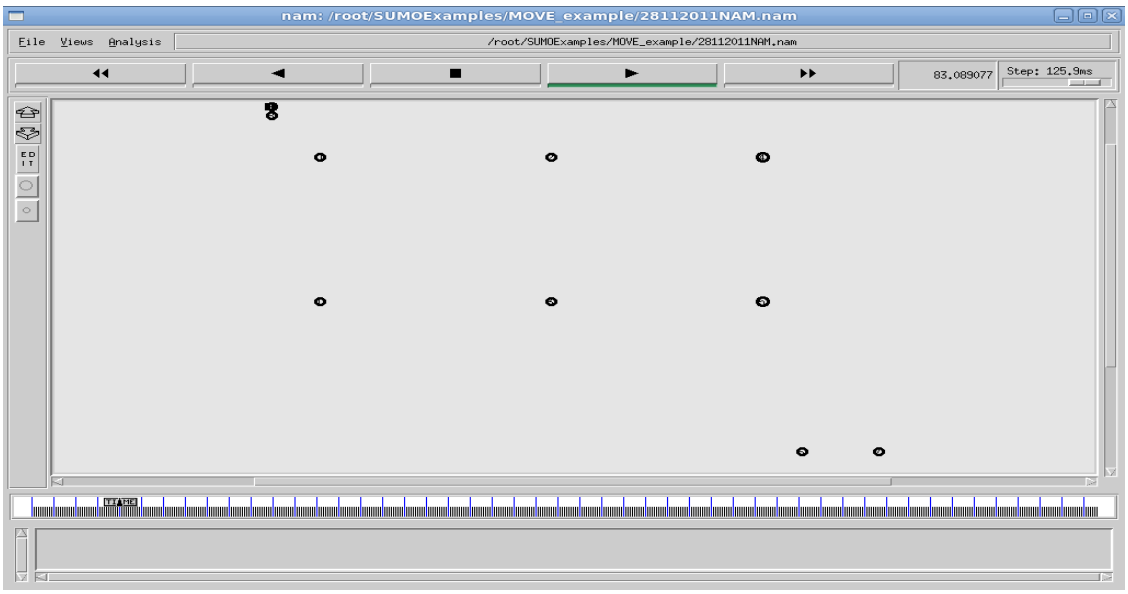

Figure 5.18: Vehicle to Infrastructure communication

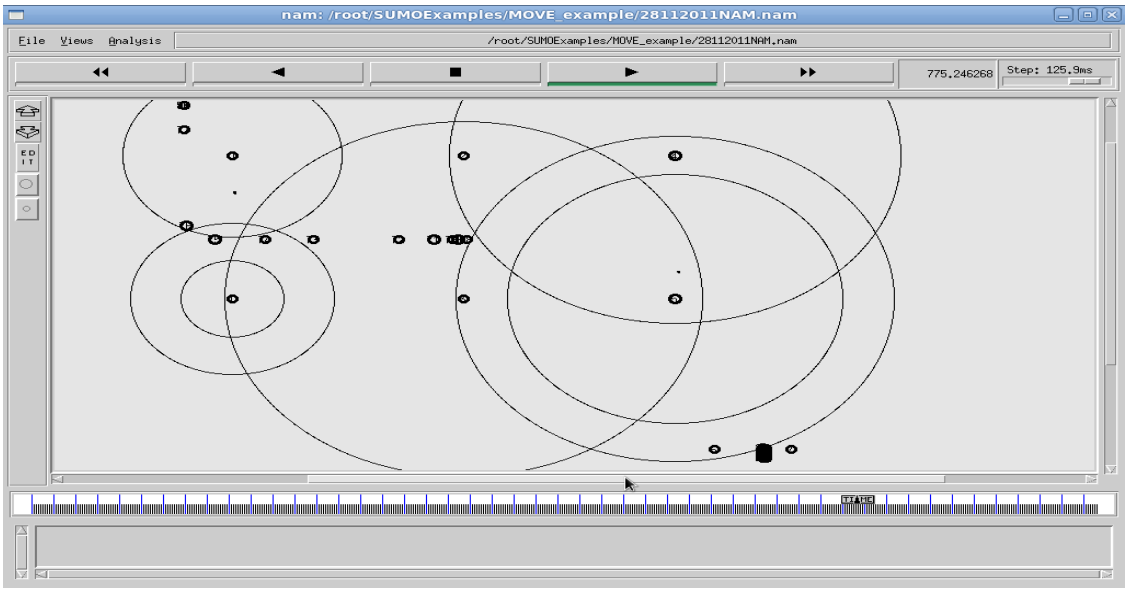

Figure 5.19: Vehicle to Infrastructure communication

Chapter 6

Proposed Solution

6.1 Proposal

Based on the assumptions mentioned in 4.1, proposal for the application under consideration is:

- First of all the vehicle will try to establish the connection with Road Side Unit (RSU). If the vehicle successfully establish the connection with the RSU then in that case infrastructure mode will be enabled. In infrastructure mode, the vehicle will directly communicate with RSU's.
- By default infrastructure mode will be enabled. If the onboard unit present in the vehicle does not receive the acknowledgement from the RSU, then it can be assumed that there is no RSU in the range.
- In case the OBU on vehicle does not receive the acknowledgement from RSU, the Ad-hoc mode will be enabled.
- In Ad-hoc mode, the vehicles would form cluster. Each cluster would have cluster head. All vehicles in cluster would communicate with cluster head. All cluster head would communicate with each other.
- Also cluster head would have to buffer data i.e. store the communication messages received from the other members of cluster. There is a threshold value for how much time the cluster head would store the data in case RSU is not detected.

- Once RSU is detected and the currently it is in Ad-hoc mode, then the cluster head would transmit all the buffered data to the RSU. Also OBU will change its mode from Ad-hoc to Infrastructure.
- While deleting the buffered data, the OBU will not delete any priority data from the available data i.e. any accident related information or any break-down related information.

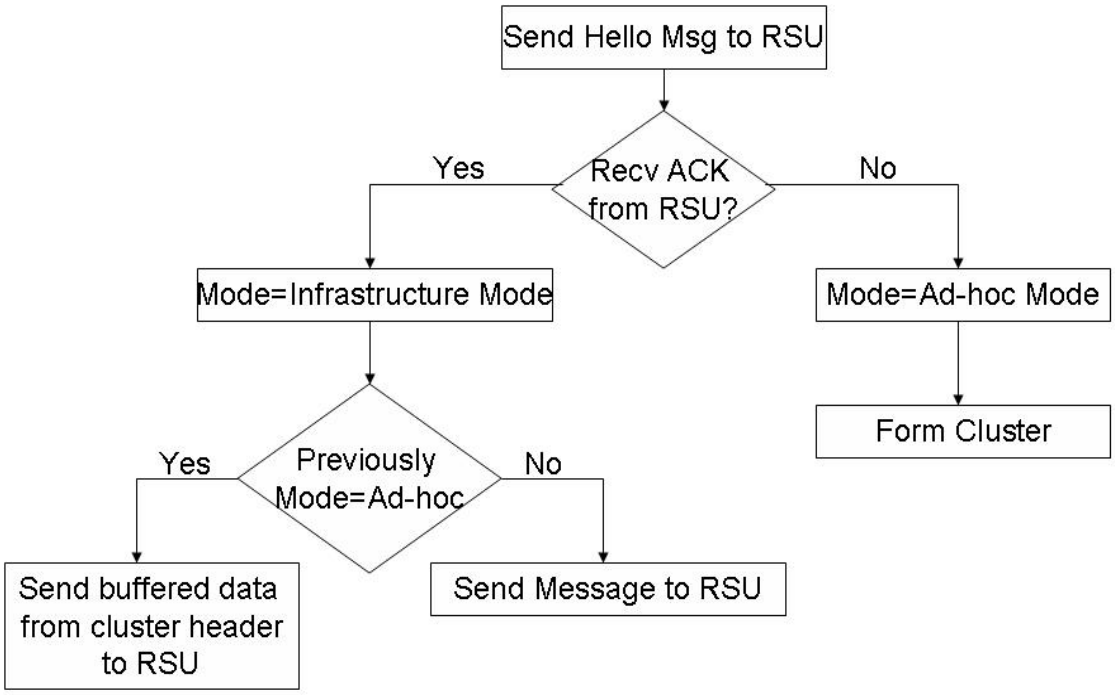

Figure 6.1: Proposed Solution

6.1.1 Cluster Election

- Initially all the nodes will broadcast Hello message.
- The node which will receive Hello message will reply with Acknowledgement message (ACKs).
- Each node will receive ACK messages from their neighboring nodes. Each node will count the neighboring nodes with help of ACKs received.
- Now every node will broadcast their total number of neighbors.
- As a result, the node which will have maximum number of neighbors and minimum ID will be elected as cluster header.

- Now, every node will broadcast the cluster head ID decided with help of number of neighboring nodes and minimum ID.
- Once cluster head is elected, every cluster member will send message to cluster head only.

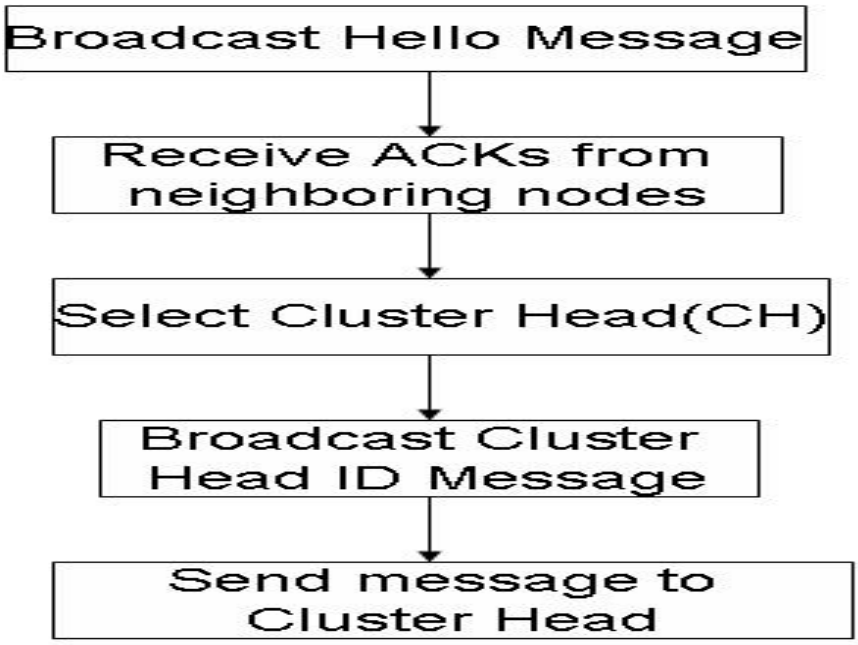

Figure 6.2: Flow Chart for Cluster Election

6.1.2 Role of Cluster Header

- Cluster Header will receive messages from cluster members.
- Once cluster head receives message from cluster member, it needs to send Acknowledgement (ACK) to cluster member.
- It will send this message to other cluster headers.
- Also it will buffer the messages received from cluster members.
- If the memory buffer exceeds, then in that case it will delete non priority messages. It will not delete safety related messages from memory buffer.
- It will also receive messages from other cluster heads and it will forward the messages to respective cluster member depending upon type of message. If it is safety related message, it will forward to all the cluster members.

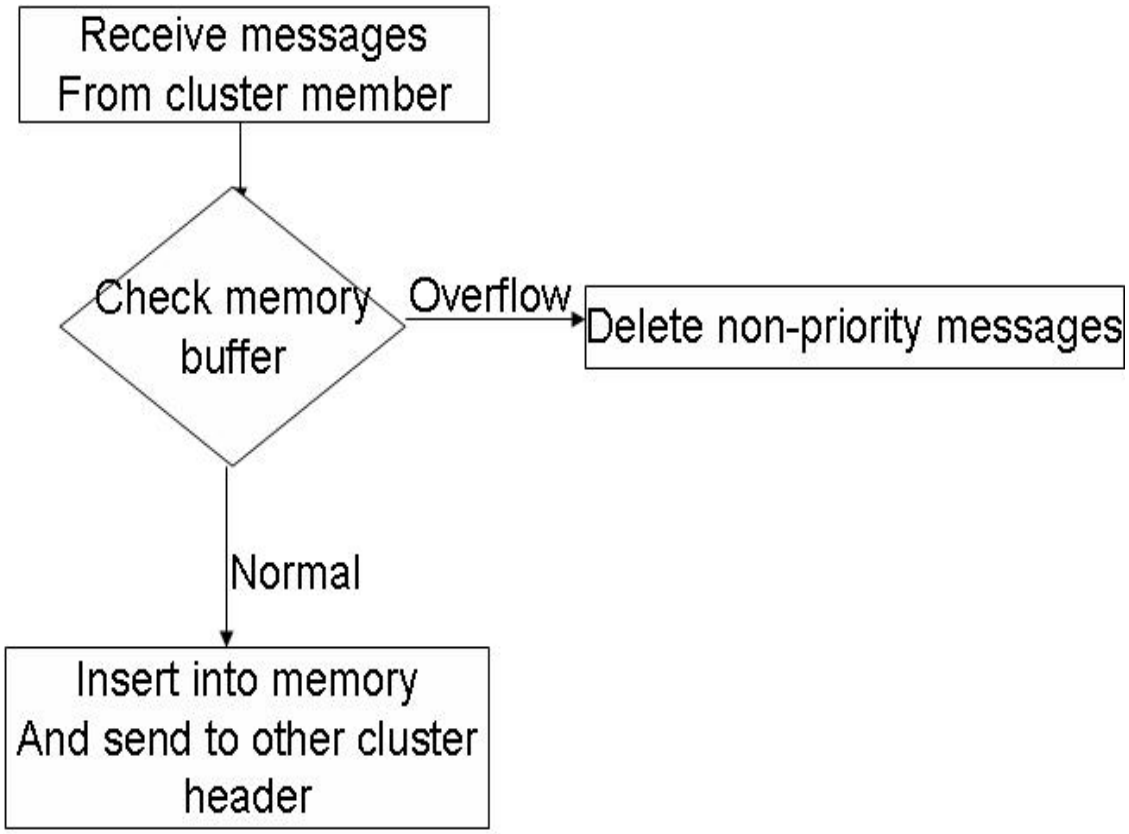

Figure 6.3: Role of Cluster Header

6.1.3 Role of Cluster Members

- Once cluster head is elected, cluster member will send message to respective cluster head.
- Cluster head should send ACK to cluster member from where it received message.
- If no ACK is received from cluster head (CH), then it can be assumed that cluster has been broken. So again cluster formation procedure should be repeated.
- If ACK is received then it can send further messages to cluster head.

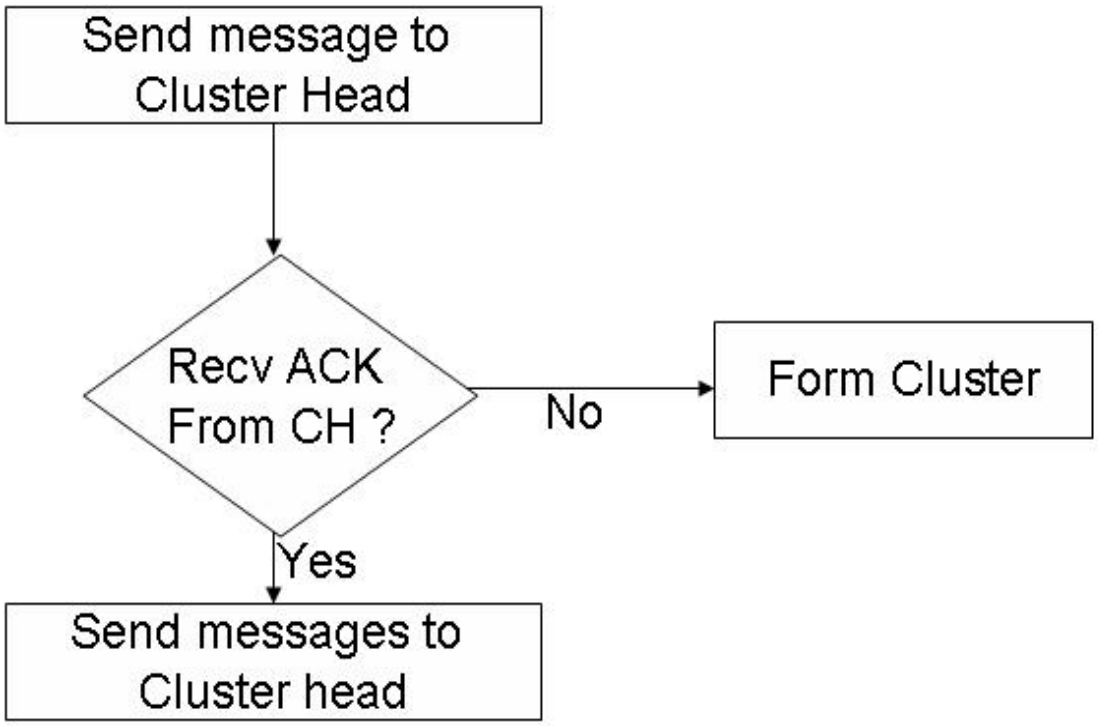

Figure 6.4: Role of Cluster Member

6.2 Simulation Parameter

For all scenarios mobility of vehicle is generated with eWorld and MOVE which work on SUMO, which is open source software.

We had integrated proposed solution in AODV protocol in NS-2.34. This chapter includes the simulation parameters for test cases which are carried out on the NS-2 for varying No. of Nodes and Speed of vehicles. Table 6.1 shows simulation parameters used in the city as well as highway scene for varying No. of nodes and speed of vehicles.

6.3 Results and Analysis

The performance of the proposed solution is compared with basic AODV protocol in terms of Average End-to-End delay and Packet Delivery Ratio. All experiment results presented in this section are average of five simulation runs for all the two different parameters for performance metrics. The performance metrics measurements are with respect to No. of vehicles and speed of vehicles.

The following metrics were chosen for evaluating the performance of protocols:

- **Average End-to-End delay (E2E delay)**: It is the calculation of typical time

Parameters	Simulated Values
Antenna Model	Omnidirectinal antenna
Radio Propogation Model	TwoRayGround
Physical Layer	WirelessPhy/OFDM
MAC Type	IEEE 802.11p
Interface Queue Type	priority Queue (50 Packets)
Routing Protocols	AODV
Simulation Time	250 second
Dimension	7973 X 4282
No. of vehicles	25, 50, 75, 100
Speed	10 m/s, 15 m/s, 25 m/s
No. of UDP connections	2, 3, 4, 5

Table 6.1: Simulation Parameters

taken by packet (in average packets) to cover its journey from the source end to the destination end.

- **Packet Delivery Ratio (PDR)**: This metric gives the ratio of the total data packets successfully received at the destination and total number of data packets generated at source.

6.3.1 Average End to End Delay

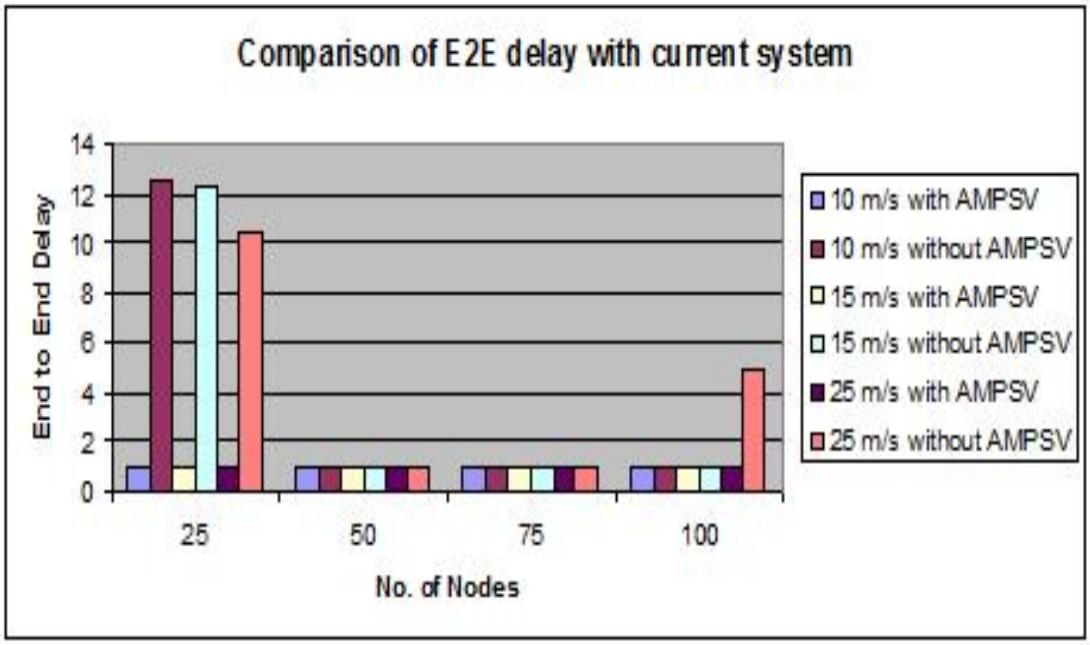

Figure 6.5: Comparison of End-to-End delay with current system

By analyzing results of Ahmedabad city with varying No. of nodes and speed as

shown in Figure 6.5 performance improvement in Avg. End-to-End delay can be seen. But as the number of vehicle increases, average end to end delay increases in the present system, while in the proposed solution it remains constant.

6.3.2 Packet Delivery Ratio

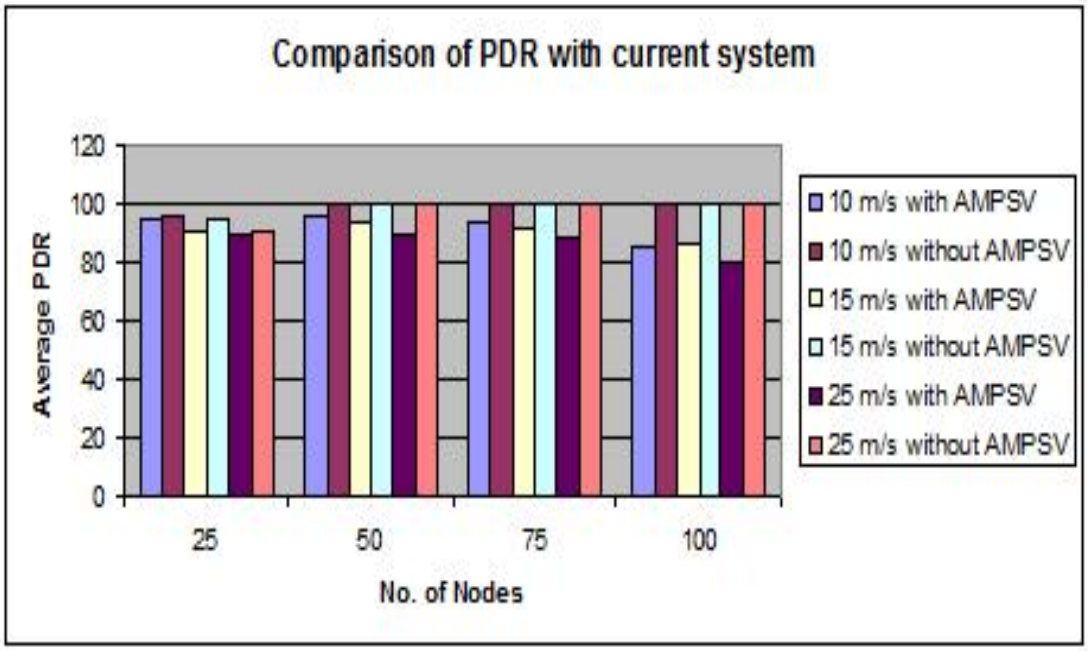

Figure 6.6: Comparison of PDR with current system

By analyzing results of Ahmedabad city with varying No. of nodes and speed as shown in Figure 6.6, we can find that the Packet Delivery Ratio is less when compared with the current implementation. The PDR decreases because at the time of cluster election, the messages are discarded at MAC layer. When cluster election algorithm works, HELLO messages are broadcast and so at that time some of messages are discarded. As a result PDR is less for new proposed solution.

6.4 Summary

Results indicate that the proposed solution AMPSV which has been implemented in AODV class has reduced Average End-to-End Delay. But it has also degraded Packet Delivery Ratio to some extent. The main reason behind degradation of PDR is the cluster election algorithm which is repeated after periodic time of 2 seconds. During cluster election, broadcast messages are generated resulting in discarding of some of the unicast messages.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Inherent VANET characteristics makes data transmission quite challenging for different type of application and scenarios.

The simulation test scenarios for Ahmedabad city are deployed by NS-2 along with number of vehicles and speed. Simulation results show that with help of clustering we can achieve better end-to-end delay when compared with the existing system. But the time taken to form cluster, clustering algorithm and the periodicity of running cluster election algorithm are major factors which affect the results.

Because of cluster election algorithm, the packet delivery ratio decreases when compared with the current system. When cluster election algorithm is executing, because of broadcast message the packets are discarded at MAC layer itself.

7.2 Future Work

The AMPSV algorithm performs better than the current clustering scenarios as end-to-end delay decreases and also number of hops also decreases between source and destination. But PDR decreases because of cluster election algorithm. So, effective clustering algorithm needs to be selected so that number of broadcast messages can be decreased.

Bibliography

[1] *Juan-Bautista Tomas-Gabarron, Esteban Egea-Lopez, Joan Garcia-Haro, Rocio Murcia-Hernandez, "Performance evaluation of a CCA application for VANETs using IEEE 802.11p"*, 978-1-4244-6826-3/10 2010 IEEE.

[2] *B. Ramakrishnan, Dr. R. S. Rajesh, R. S. Shaji, "Performance Analysis of 802.11 and 802.11p in Cluster Based Simple Highway Model"*, B. Ramakrishnan et al. / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 1 (5) , 2010, 420-426

[3] *TaeOh Kim, SungDae Jung, SangSun Lee, "CMMP : Clustering-Based Multi-Channel MAC Protocol In VANET"*,978-0-7695-3925-6/09 2009 IEEE, 2009 Second International Conference on Computer and Electrical Engineering.

[4] *Frank Kargl,"Vehicular Communications and VANETs"*, CCC Ulm, Ulm University.

[5] *Pablo Urmeneta, "Simulation and Improvement of the Handover process in IEEE 802.11p based VANETs (Vehicle Ad-hoc NETworks)"*, College of Electronics and Information Engineering, Tongji University, Shanghai, November 2010.

[6] *B. Ramakrishnan, Dr. R. S. Rajesh, R. S. Shaji, "CBVANET: A Cluster Based Vehicular Adhoc Network Model for Simple Highway Communication"*, International Journal of Advanced Networking and Applications, Volume: 02, Issue: 04, Pages: 755-761 (2011)

[7] *B. Ramakrishnan, Dr. R. S. Rajesh, R. S. Shaji, "An Intelligent Routing Protocol for Vehicle safety communication in Highway Environments"*, Journal of Computing, Volume 2, Issue 11, November 2010, ISSN 2151-9617

[8] *B. Ramakrishnan, Dr. R. S. Rajesh, R. S. Shaji, "Analysis of Routing Protocols for Highway Model without Using Roadside Unit and Cluster"*, International Journal of Scientific & Engineering Research, Volume 2, Issue 1, January-2011 ISSN 2229-5518

[9] *D.Rajini Girinath, S.Selvan, "A Novel Cluster based Routing Algorithm for Hybrid Mobility Model in VANET"*, International Journal of Computer Applications (0975 - 8887), Volume 1 No. 15, 2010

[10] *Daniel Jiang, Luca Delgrossi, "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments"*, Vehicular Technology Conference, IEEE 2008, ISSN 1550-2252

[11] *Sebastian Grafling, Petri Mahonen, Janne Riihijarvi, "Performance Evaluation of IEEE 1609 WAVE and IEEE 802.11p for Vehicular Communications"*, Ubiquitous and Future Networks (ICUFN), 2010

[12] http://sumo.sourceforge.net/

[13] http://carlink.lcc.uma.es/

[14] http://www.car-2-car.org/

[15] http://www.prevent-ip.org/en/prevent_subprojects/safe_speed_and_safe_following/willwarn/willwa

[16] http://www.network-on-wheels.de/

[17] http://www.cvisproject.org/

[18] http://www.safespot-eu.org/

[19] Krajzewicz D and Rossel C, S̈imulation of Urban MObility (SUMO)". German Aerospace Centre, 2007. http://sumo.sourceforge.net/index.shtml.

[20] TIGER Topologically Integrated GEographic Encoding and Referencing. http://www.census.gov/geo/www/tiger/.

[21] MOVE (MObility model generator for VEhicular networks):R̈apid Generation of Realistic Simulation for VANET", 2007. http://lens1.csie.ncku.edu.tw/MOVE/index.htm.

[22] The VINT Project, ns Manual (formerly known as ns Notes and Documentation), May 2010.

[23] ïns-2: Tips and statistical data for running large simulations in ns,"http://www.isi.edu/nsnam/ns/ns-largesim.html

[24] ns-2 Changelog, http://www.isi.edu/nsnam/ns/CHANGES.html.

Appendix A

List of Publications

Following paper have been communicated to following conferences and we are waiting for response:

- Nikunj Doshi, Prof. Vijay Ukani, ***Adaptive Message Passing Scheme in VANET***, at "The First International Workshop on Vehicular Communications, Sensing, and Computing (VCSC) 2012" in conjunction with IEEE SECON 2012
- Nikunj Doshi, Prof. Vijay Ukani, ***Adaptive Message Passing Scheme in VANET***, at "The 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2012)", Shanghai, China

Appendix B

Installation steps for MOVE

To install MOVE following additional softwares need to be installed:

- Linux
- Java SDK 1.6 - http://java.sun.com
- SUMO version: 0.13.1 - http://sumo.sourceforge.net
  - Xerces (XML-parser) - http://xerces.apache.org/xerces-c/index.html
  - FOX-Toolkit (GUI Toolkit) - http://www.fox-toolkit.org/
  - PROJ (Cartographic Projections Library): http://www.remotesensing.org/proj/
  - GDAL (Geospatial Data Abstraction Library): http://www.remotesensing.org/gdal/
- NS2 version: 2.34 (all-in-one) or later - http://www.isi.edu/nsnam/ns/

B.1 Installation steps for SUMO

To be able to run SUMO on Linux, just follow these four steps:

- Install all of the required tools and libraries
- Get the source code
- Build the SUMO binaries
- Install the SUMO binaries to another path (optional)

Prerequisites to install SUMO:

- yum install libtiff-devel
- yum install libpng-devel
- yum install libjpeg-devel
- yum install libXft-devel
- yum install zlib-devel
- yum install bzip2-devel
- yum install mesa-libGLU-devel
- yum install mesa-libGL-devel
- yum install glibc-devel

Prerequisites to install gdal and proj4:

- yum install gcc-c++
- yum install gcc
- yum install libpng
- yum install libtiff

B.1.1 Installation steps for PROJ

```
tar -zvxf proj-4.7.0.tar.gz
./configure
make
make install
```

B.1.2 Installation steps for GDAL

```
tar -zvxf gdal-1.7.3.tar.gz
cd gdal-1.7.3
./configure --with-static-proj4=/usr/local/lib --with-threads --with libtiff=internal
make
make install
```

B.1.3 Installation steps for FOX-Toolkit

```
tar -zvxf fox-1.6.43.tar.gz
cd fox-1.6.43
./configure --with-opengl=yes --prefix=$HOME
make
make install
```

B.1.4 Installation steps for XERCESC

```
tar -zvxf xerces-c-3.0.1.tar.gz
cd xerces-c-3.0.1
export XERCESCROOT=$HOME/xerces-c-3.0.1
./configure --prefix=$HOME
make
make install
```

B.1.5 Installation steps for SUMO

```
tar -zvxf sumo-src-0.13.1.tar.gz
cd sumo-0.13.1/
./configure --with-fox=$HOME --with-proj-gdal=$HOME --with-xerces=$HOME --prefix=$HOM
make
make install
```

Installation of SUMO binaries

The installation can be done with the help of RPM files that are available for Fedora 14.

1. Download all the RPM files: Here is the following link for downloading all the rpm files related to SUMO:

   `http://download.opensuse.org/repositories/home:/behrisch/Fedora_14/i386/`

2. Install all the RPM files downloaded in following sequences using command rpm -ivh packagename :

   (a) `libfox1_6-1.6.43-22.1.i386.rpm`

(b) `fox16-1.6.43-22.1.i386.rpm`

(c) `fox16-devel-1.6.43-22.1.i386.rpm`

(d) `fox16-devel-static-1.6.43-22.1.i386.rpm`

(e) `fox16-example-apps-1.6.43-22.1.i386.rpm`

(f) `libt4k_common0-0.1.1-10.1.i386.rpm`

(g) `t4k_common-0.1.1-10.1.i386.rpm`

(h) `t4k_common-devel-0.1.1-10.1.i386.rpm`

(i) `tuxmath-2.0.3-21.3.i386.rpm`

(j) `sumo-0.13.0-7.4.i386.rpm`

(k) `sumo-svn-836.1.i386.rpm`

3. Install all the missing files using "yum install" which are needed while running rpms.

While installing SUMO using RPMs versions of all packages were implemented:

- FOX Toolkit: 1.6.43
- GDAL: 1.7.3-9
- PROJ: 4.7.0-3
- XERCES: 3.0.1-20
- SUMO: 0.13.1

As a result the source of SUMO was builded and installed successfully with help of source and binaries.

B.2 Installation steps for NS-2

1. su , password
2. `yum install libX11-devel libXext-devel libXau-devel libXmu-devel`
3. exit
4. tar xvf ns-allinone-2.34.tar.gz

5. cd ns-allinone-2.34
6. ./install
7. cd
8. cd root
9. export PATH=/home/nikunj/NS2/ns-allinone-2.34/bin:/home/nikunj/NS2/ns-allinone-2
   \item \begin{verbatim} export LD_LIBRARY_PATH=/home/nikunj/NS2/ns-allinone-2.34/o
10. export TCL_LIBRARY=/home/nikunj/NS2/ns-allinone-2.34/tcl8.4.18/library
11. cd ns-2.34; ./validate
12. Put following code in /root/.bash_profile

```
#LD_LIBRARY_PATH
OTCL_LIB=/home/nikunj/NS2/ns-allinone-2.34/otcl-1.13
NS2_LIB=/home/nikunj/NS2/ns-allinone-2.34/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB

#TCL_LIBRARY
TCL_LIB=/home/nikunj/NS2/ns-allinone-2.34/tcl8.4.18/library
export TCL_LIBRARY=$TCL_LIB

#PATH
XGRAPH=/home/nikunj/NS2/ns-allinone-2.34/bin:/home/nikunj/NS2/ns-allinone-2.34/to
NS=/home/nikunj/NS2/ns-allinone-2.34/ns-2.34/
NAM=/home/nikunj/NS2/ns-allinone-2.34/nam-1.14/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

Index