

**IEEE WAVE (802.11P/1609.X) BASED
ADAPTIVE TRAFFIC CONTROL SYSTEM FOR A FOUR WAY
JUNCTION
USING VEHICLE TO VEHICLE (V2V) COMMUNICATION**

**IEEE WAVE (802.11P/1609.X) BASED
ADAPTIVE TRAFFIC CONTROL SYSTEM FOR A FOUR WAY
JUNCTION
USING VEHICLE TO VEHICLE (V2V) COMMUNICATION**

Undergraduate graduation project Report
submitted in partial fulfillment of
the requirements for the
Degree of Bachelor of Science of Engineering
in

Department of Electronic & Telecommunication Engineering
University of Moratuwa

Supervisor

Dr. Ajith Pasqual

Project Group

R.P.R.L. Ranatunga	080408F
R.P.L. Rupananda	080429U
K. C. Sirimanna	080476J
W.A.A.D. Srimal	080488X

Approval of the Department of Electronic & Telecommunication Engineering

.....
Head, Department Of Electronic &
Telecommunication Engineering

This is to certify that we have read this project and that in our opinion it is fully adequate,
in cope and quality, as an Undergraduate Graduation Project.

Supervisor:
Name & Signature

Date:

Abstract

**IEEE WAVE (802.11P/1609.X) BASED
ADAPTIVE TRAFFIC CONTROL SYSTEM FOR A FOUR WAY
JUNCTION
USING VEHICLE TO VEHICLE (V2V) COMMUNICATION**

Supervisor: Dr. Ajith Pasqual

Keywords: WAVE, IEEE 802.11p, VANET, VEINS

Traffic coordination in intersections is a very studied and challenging topic. We develop a traffic control algorithm and design a virtual traffic light protocol that can dynamically optimize the flow of traffic in road intersections without requiring any roadside infrastructure. We use IEEE 802.11p/1609.x protocols known as Wireless Access in Vehicular Environments (WAVE) as the standards for the Vehicle to Vehicle (V2V) data dissemination. WAVE is a next generation intelligent transportation technology that aims to improve transportation environments and road traffic control is a promising application of WAVE standards.

To our dearest parents and teachers

ACKNOWLEDGEMENT

We would like to pay our gratitude towards Dr. Ajith Pasqual, the supervisor of our project for the immense help and support given from the beginning till the end of our project by advising, encouraging and helping us.

We also thank our project coordinator Dr. Rohan Munasinghe for advising and guiding us throughout the project to encourage us.

Last but not least, we also like to thank all the academic and non-academic staff members for the help and support given in making us achieve our objective and making the project a success.

TABLE OF CONTENTS

	Page
APPROVAL	iii
ABSTRACT	iv
DEDICATIONS	v
ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS	vii
1 INTRODUCTION.....	1
1.1 Intelligent Transportation Systems.....	1
1.2 Vehicle to Vehicle Communication (V2V).....	1
1.3 Traditional Traffic Handling.....	3
1.4 Existing Adaptive Traffic Light Systems.....	4
1.4.1 SCOOT System.....	4
1.4.2 SCATS System.....	6
1.5 WAVE Based Adaptive Traffic Control System.....	7
1.6 WAVE System Architecture.....	9
1.7 WAVE Physical Layer.....	11
1.8 WAVE Channel Coordination.....	12
1.9 WAVE Basic Service Set.....	14
1.10 WAVE Communication Protocols	
1.10.1 Internet Protocol Version 6 (IPv6).....	15
1.10.2 WAVE Short Message Protocol (WSMP).....	15
1.11 WAVE Management Plane.....	16

1.12 WAVE Synchronization	16
2 RESEARCH AND DEVELOPMENT.....	18
2.1 Simulating VANETs.....	18
2.2 VANET Mobility Models.....	19
2.3 EstiNet Network Simulator and Emulator	21
2.4 Vehicles in Network Simulation-VEINS.....	23
2.4.1 Traffic simulation.....	24
2.4.2 Network Simulation -OMNeT++.....	27
2.4.3 Mixim (Mixed Simulator).....	28
2.4.4 Traci	29
2.5 Simulation Setup.....	30
2.5.1 Sumo Simulation Setup	30
2.5.2 Omnet++ Simulation Setup.....	30
2.6 Algorithm.....	32
2.6.1 Lane connection at a four way junction.....	32
2.6.2 Queues arriving to a four leg intersection.....	33
2.6.3 Wave Short Message (WSM) Format.....	37
2.6.3.1 WSM Header and Data field.....	38
2.6.4 Computation of the Algorithm with WAVE Short Messages.....	39
2.6.5 Developing the algorithm with the VIENS-2.0 Framework.....	44
2.6.5.1 TraCIScenarioManager.cc.....	44
2.6.5.2 TraCIMobility.cc.....	46
2.6.5.3 BaseWaveApplLayer.cc.....	48
2.6.5.4 WaveShortMessage.cc.....	48
2.6.5.5 TraCIDemo11p.cc.....	51
2.6.5.6 Class hierarchy of the modified classes.....	52

3 RESULTS.....53

3.1 Queue Lengths.....53

3.2 Accumulated Waiting Time.....55

3.3 Fairness of the Algorithm.....56

4 CONCLUSIONS AND FUTURE WORK

4.1 Conclusion.....60

4.2 Future Work.....61

LIST OF FIGURES AND TABLES

REFERENCES

Chapter 1

INTRODUCTION

1.1 Intelligent Transportation Systems

Intelligent Transportation System (ITS) is an emerging concept and a wide research area in the current technological innovations. One of the applications of this concept consists of providing our vehicles and roads with capabilities to make the road network more secure and to make our time on the road more enjoyable.

- Traffic information
- Collision avoidance
- Possible detours
- Weather information
- Internet access
- Interactive chat
- Network games
- Following vehicles

These applications are typical examples of what we call an Intelligent Transportation System (ITS) whose goal is to improve security, efficiency and enjoyment in road transport through the use of new technologies for information and communication.

1.2 Vehicle to Vehicle Communication (V2V)

Traditional traffic management systems are based on centralized infrastructures where cameras and sensors implemented along the road collect information on density and traffic state and transmit this data to a central unit to process it and make appropriate decisions. This type of system is very costly in terms of deployment and is characterized by a long reaction time for processing and information transfer in a context where information transmission delay is vital and is extremely important in this type of system. In addition, these devices placed on roads require periodic and expensive maintenance. Consequently, for large scale deployment of this type of system, important investment is required in the communication and sensor infrastructure.

However, with the rapid development of wireless communication technologies, location and sensors, a new decentralized (or semi-centralized) architecture based on vehicle-to-vehicle communications (V2V) has created a very real interest these last few years for car manufacturers, R&D community and telecom operators. This type of architecture relies on a distributed and autonomous system and is made up of the vehicles themselves without the support of a fixed infrastructure for data routing. In this case, we are talking about a vehicular ad hoc network (VANET). An example of an urban VANET network is illustrated in Figure 1.1.

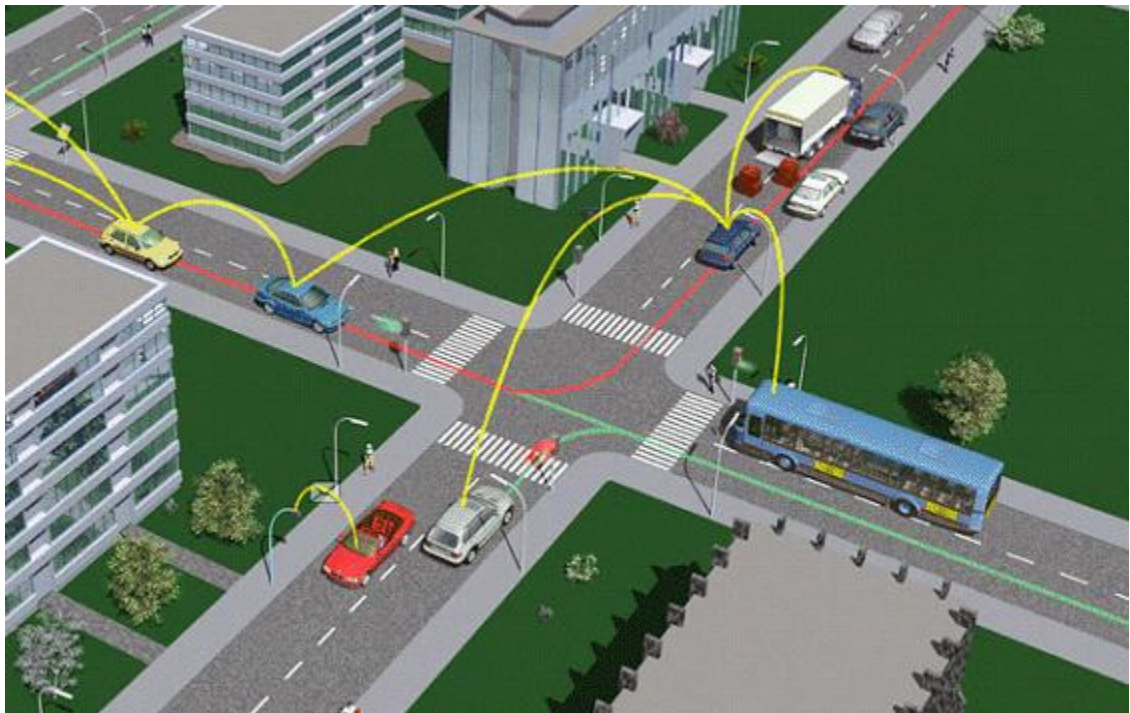


Figure 1.1: Example of VANET network

Two categories of draft standards provide outlines for vehicular networks and V2V communication. These standards constitute a category of IEEE standards for a special mode of operation of IEEE 802.11 for vehicular networks called Wireless Access in Vehicular Environments (WAVE). 802.11p is an extension to 802.11 Wireless LAN medium access layer (MAC) and physical layer (PHY) specification. IEEE 802.11p aims to provide specifications needed for MAC and PHY layers for specific needs of vehicular networks. IEEE 1609 is a family of standards which deals with issues such as management and security of the network.

- 1609.1 -Resource Manager: This standard provides a resource manager for WAVE, allowing communication between remote applications and vehicles
- 1609.2 -Security Services for Applications and Management Messages
- 1609.3 -Networking Services: This standard addresses network layer issues in WAVE
- 1609.4 -Multi-channel Operation: This standard deals with communications through multiple channels

The current state of these standards is trial-use. A vehicular communication networks which complies with the above standards supports both vehicular on-board units (OBU) and roadside units (RSU). RSU acts similar to a wireless LAN access point and can provide communications with infrastructure. Also, if required, RSU must be able to allocate channels to OBUs. There is a third type of communicating nodes called Public Safety OBU (PSOBU) which is a vehicle with capabilities of providing services normally offered by RSU. These units are mainly utilized in police cars, fire trucks, and ambulances in emergency situations. In our case, we only use OBUs for V2V communication.

1.3 Traditional Traffic Handling

Conventional way of traffic handling consists of fixed road side infrastructures. Normally traffic lights have three colors, red, yellow and green to indicate the signals to vehicles and pedestrians. The main drawback of traditional traffic light systems is the inability to adapt to the prevailing traffic patterns which is an important aspect in efficient traffic management. Basically each signal light or color is assigned a fixed time interval regardless of the traffic congestion. Even though there are adaptive fixed traffic lights, they are also not fully adaptive as one would expect. And those adaptive traffic lights can be very expensive. And also it is neither feasible nor practical to implement fixed roadside infrastructures to each and every junction, as there are thousands of junctions in a selected road network, and also due to the installation costs and maintenance costs of each fixed node.

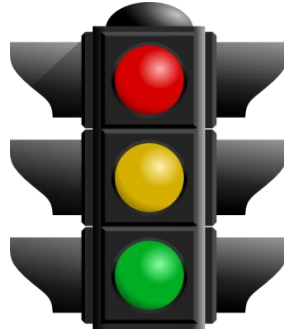


Figure 1.2: Traditional Traffic Lights

1.4 Existing Adaptive Traffic Light Systems

There are adaptive traffic lights systems in some parts of the world which have been implemented to provide better traffic management. But these systems are very costly and need a lot of maintaining capability as well. Two of the most popular adaptive traffic lights systems in the world are,

1. SCOOT
2. SCATS

1.4.1 SCOOT System [2]

Information on the physical layout of the road network and how the traffic signals control the individual traffic streams are stored in the SCOOT database. Any adaptive traffic control system relies upon good detection of the current conditions in real-time to allow a quick and effective response to any changes in the current traffic situation. SCOOT detects vehicles at the start of each approach to every controlled intersection. It models the progression of the traffic from the detector through the stop line, taking due account of the state of the signals and any consequent queues. The information from the model is used to optimize the signals to minimize the network delay.

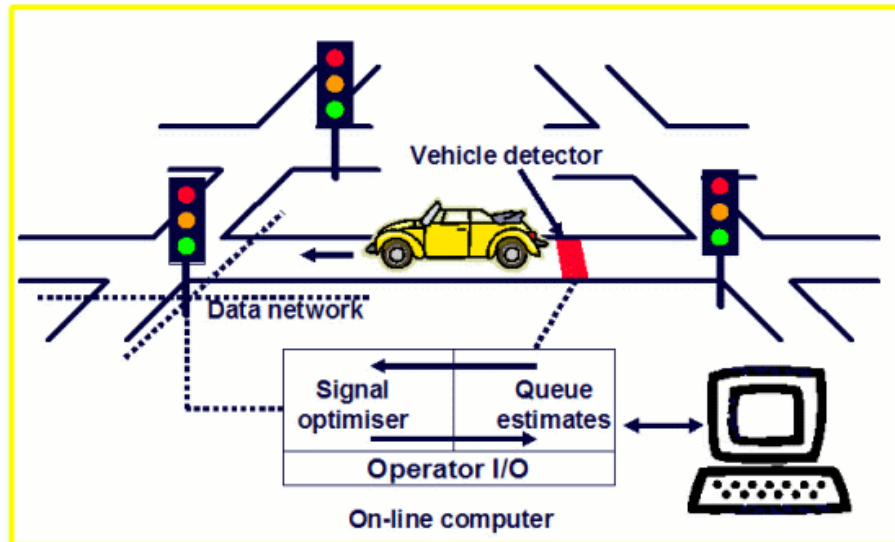


Figure 1.3: How SCOOT Works

SCOOT systems are designed to meet the user's requirements. There is a central processor hosting the SCOOT Kernel integrated with the company specific UTC software that controls communications to the on-street equipment and provides the operator interface. This processor and associated networked terminals may be installed in a control room. Alternatively for smaller systems, the processor may be installed in the authority's server room and engineers control the system through client software on their desk top computers. In larger systems there may be a control room staffed by operators, with selected engineers also having system access from their desktops. This access may extend to viewing and controlling CCTV to compare on-street conditions with the SCOOT model.



Figure 1.4: SCOOT Control Room

1.4.2 SCATS System [3]

SCATS is an area wide traffic management system that operates under the Windows environment. It controls the cycle time, green splits and offsets for traffic control intersections and mid-block pedestrian crossings. With the inclusion of vehicle detectors, it can adaptively modify these values to optimize the operation to suit the prevailing traffic. Alternatively, it can manage intersections in fixed-time mode where it can change plans by time of day, day of week. It is designed to coordinate traffic signals for networks or for arterial roads.

Intersection connections to a regional traffic control computer can be permanent or may be on-demand using dial-in or dial-out facilities. Each regional computer can manage up to 250 intersections. A SCATS system can have up to 64 regional computers. Monitoring is provided by a graphical user interface. Up to 100 users can connect to a SCATS central manager at the same time. Up to 30 users can connect to a single regional computer simultaneously. Performance monitoring, alarm condition notification and data configuration facilities are included. SCATS automatically collects alarm and event information, operational and performance data and historical data. SCATS operates automatically but operation intervention is provided for use in emergencies.

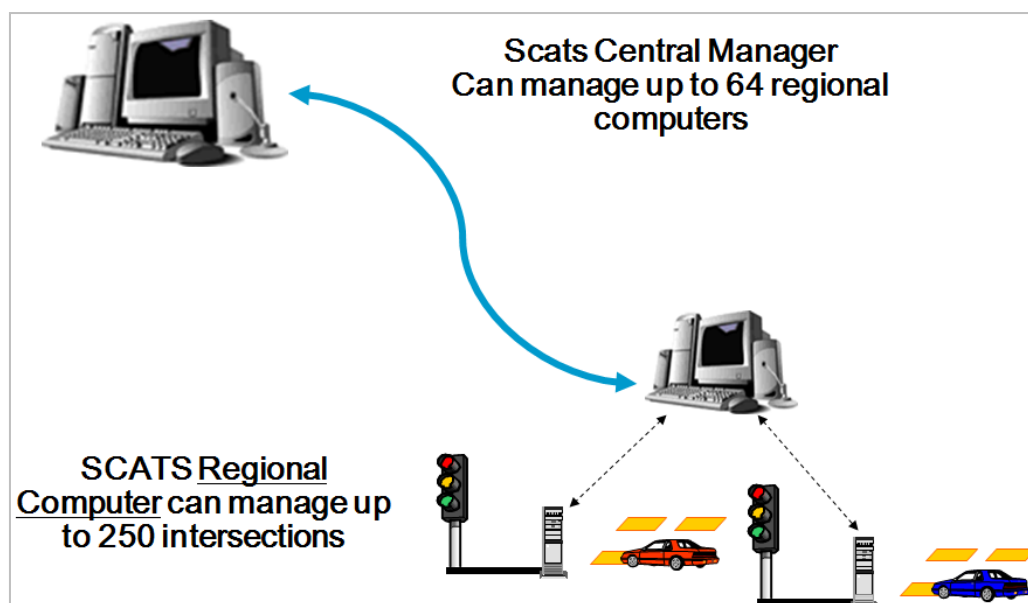


Figure 1.5: SCATS System

But the problem with these two systems and other adaptive traffic control systems is that they are very costly and has to be operated by technical experts. These systems have to be continuously monitored and hence need a lot of technical staff and cost a lot for operational maintenance. Below figure 1.6 shows a comparison of each system based on the overall cost per road intersection.

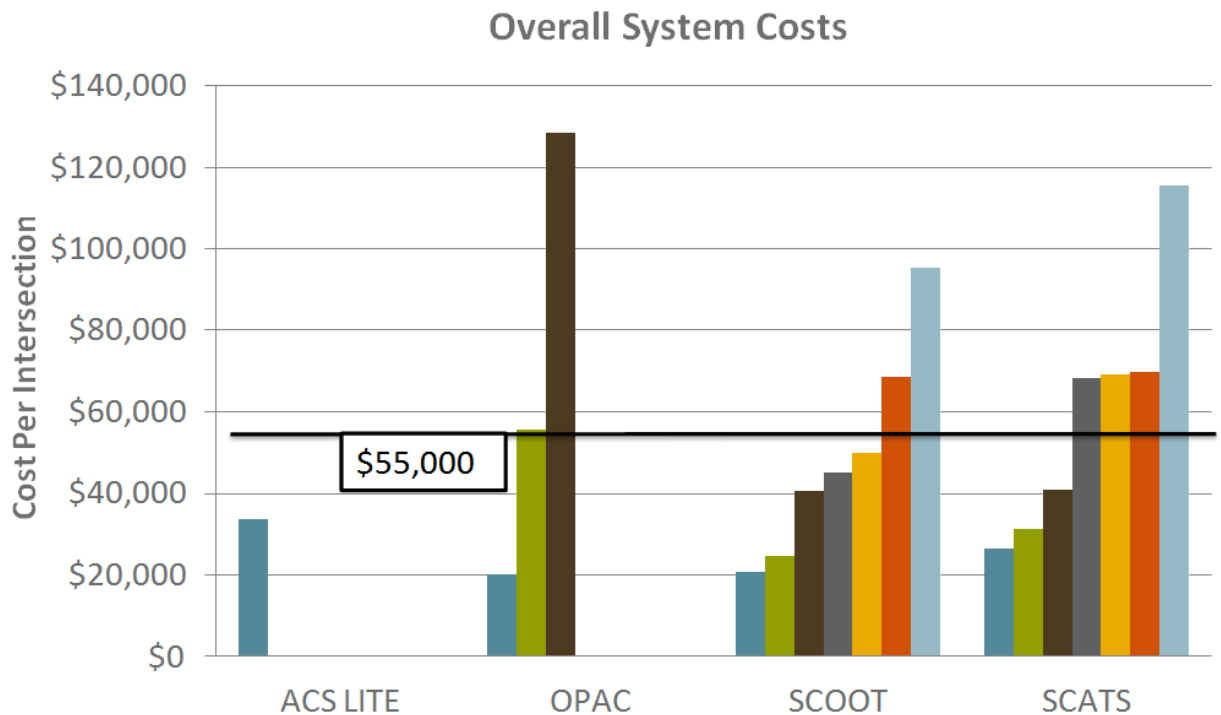


Figure 1.6: Cost Comparison of Existing Adaptive Traffic Light Systems

1.5 WAVE Based Adaptive Traffic Control System

One of the main application areas for Vehicular Ad Hoc Networks (VANET) is the mitigation of traffic congestion. This is a very important problem, as its cost has been estimated to reach a significant percentage of the GDP in some of the developed countries. In theory, the solution to this problem could follow three different approaches.

- 1) Reducing the total number of vehicles on the roads
- 2) Increasing the existing road infrastructure
- 3) Increasing traffic flow on the existing road infrastructure.

In economic terms, the most sustainable approach to reduce congestion would be based on increasing the throughput of the already existing road infrastructure, which is also the relevant approach for VANET-based research. It is clear that a wider exploration of the road network will result in higher conflicts at road intersections. Considering these road junctions as a crucial resource that is responsible for the continuity of traffic flows, we focus in our project in the role of VANET for the efficient management of these fundamental components of the road network. Currently, a small percentage of the intersections of the road network are equipped with adaptive traffic lights, providing the state-of-the-art in terms of intersection management to maximize traffic flow.

What we propose in our project is the migration of these roadside-based traffic lights to in-vehicle virtual signs supported only by the advent of vehicle-to-vehicle (V2V) communication. In our proposal, elected vehicles act as temporary road junction infrastructures and broadcast traffic light messages that are conveyed to drivers through in-vehicle displays. The improvement in traffic flow would result not only from the optimized management of individual intersections, which is enabled by the neighborhood awareness of VANET protocols, but also from the scalability of our solution that renders signalized control of intersections truly omnipresent. This omnipresence would allow maximizing the throughput of the complete road network, rather than the reduced number of road junctions that are currently managed by traffic lights. This is particularly important in the context of traffic, as a single bottleneck can rapidly propagate congestion through the road network.

If we summarize what we have implemented in our project,

- Implementing a Virtual Traffic Lights System for a four way junction.
- There will be no physical roadside traffic controlling infrastructures.
- Vehicles will communicate with each other using WAVE V2V communication.
- Vehicles will act as Virtual Traffic Lights.

- The system will temporarily choose a vehicle to make traffic handling decisions.
- The decision will be passed to the other vehicles at the junction.
- The message will be passed to the drivers through an inbuilt display on the dashboards of each vehicle.

Assumptions that we have made in our system,

- We are considering only a simple four way junction.
- Vehicles will not turn to left or right at the junction.
- There are two lanes at each side of the junction. Some vehicles are approaching the junction in one lane and the other lane is for the vehicles going in the opposite direction.
- Every vehicle has to be near perfectly in line of the queue and there is only one queue on each lane.

We have developed an algorithm for our system to work as expected and we simulated the developed system on open source simulators. We have used a traffic simulator called Sumo and a network simulator called OMNeT++, and a common interface called Veins has been used to combine those two simulators.

1.6 WAVE System Architecture

Worldwide, hundreds of projects and consortiums are competing in developing a robust set of standards for VANET environments. In USA, the Dedicated Short Range Communication (DSRC) Committee of the IEEE Transportation Technology Council is preparing the new Wireless Access in Vehicular Environments (WAVE) standard, which will be illustrated in this section. This section presents a brief overview of the IEEE WAVE system architecture as an indication of the current state of standardization process. WAVE system Architecture is a set of standards that describes the communication stack of vehicular nodes and the physical air-link between them (Fig. 2-6). Any RSU may have two interfaces, one for the wireless

WAVE stack and the other for external interfaces like Ethernet that may be used to enable connectivity to the Internet. Similarly, each OBU may have two interfaces, one for the wireless WAVE stack and the other for sensor-connections and human interaction.

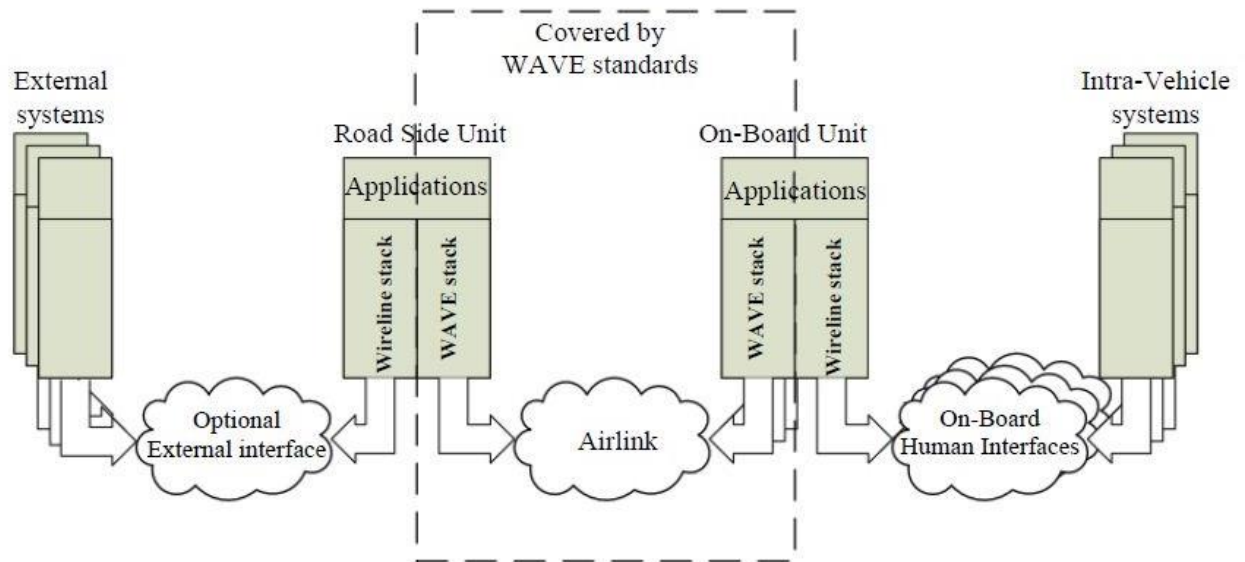


Figure 1.7: WAVE System Components

WAVE standard consists of five complementary parts as follows,

- 1) 802.11p Wireless Access in Vehicular Environments (WAVE), which is an amendment to the well-known IEEE 802.11 Wireless LAN Standard and covers the physical layer of the system.
- 2) 1609.1 Resource Manager that covers optional recommendations for the application layer.
- 3) 1609.2 Security Services for Applications and Management Messages that covers security, secure message formatting, processing, and exchange.
- 4) 1609.3 Networking Services that covers the WAVE communication stack.
- 5) 1609.4 Multi-Channel Operation that covers the arrangement of multiple channels and how they should be used.

The WAVE communication stack and the coordination between standards are shown in Fig. 2-7. Definition and operation of each layer of the stack will be demystified in the following sections.

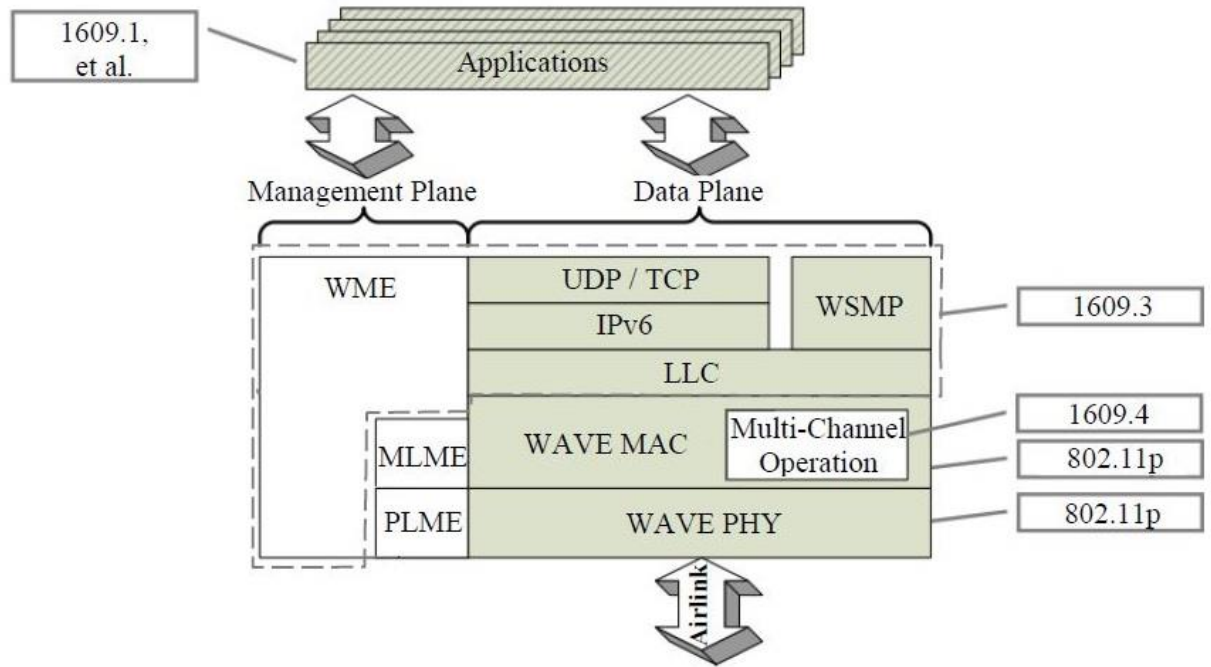


Figure 1.8: WAVE Protocol Stack

1.7 WAVE Physical Layer

A bandwidth of 75 MHz has been allocated in the 5.9 GHz band (5.850 – 5.925 GHz) for applications of the DSRC. The WAVE spectrum is composed of seven channels of 10 MHz each, as shown in Fig. 2-8, with an option of grouping two adjacent channels to have a spectrum of 20 MHz. Channel 178 is the only control channel (CCH), and other channels are service channels (SCH). Channels 175 and 181 are the 20 MHz channels. Note that channel numbering are defined according to the relation,

$$\text{Channel center frequency} = 5 \text{ GHz} + (5 \times \text{channel number}) \text{ MHz}$$

The modulation scheme used by WAVE is the Orthogonal Frequency Division Multiplexing (OFDM) using 52 orthogonal subcarriers. The OFDM is a multi-carrier modulation scheme where data is split into multiple lower rate streams. Each stream is used to modulate one of the closely spaced orthogonal subcarriers. The primary

advantage of OFDM is its ability to cope with frequency-selective fading due to multipath channels without complex equalization filters. This modulation scheme enables data rates of 3, 4.5, 6, 9, 12, 18, 24, and 27 Mbit/s in the 10 MHz channels and up to 54 Mbit/s in the 20 MHz channels. The orthogonal subcarriers should be modulated using BPSK (Binary Phase Shift Keying), QPSK (Quadrature Phase-Shift Keying), 16-QAM (Quadrature Amplitude Modulation), or 64-QAM depending on the data rate required.

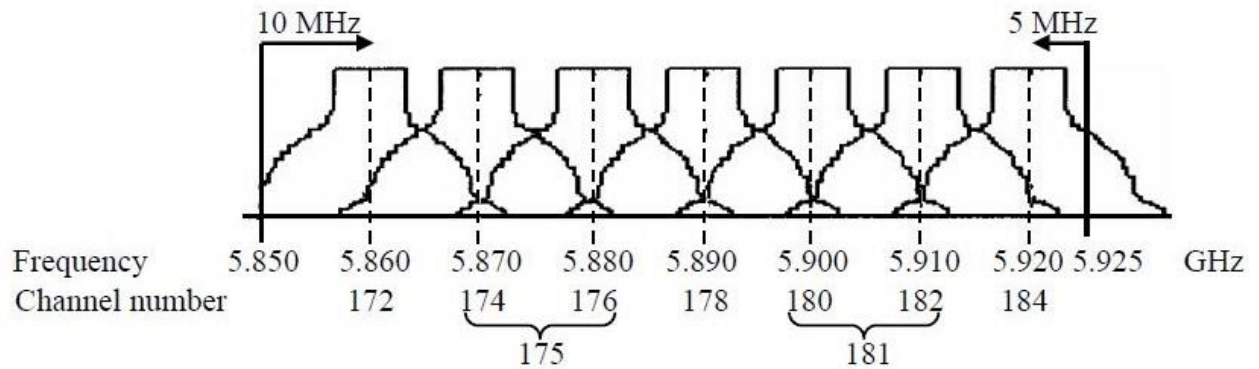


Figure 1.9: Spectrums of WAVE Channels

Before leaving the physical layer, Table 1.1 summarizes some of the physical-dependant parameters related to 802.11 MAC.

Characteristic	Value for WAVE
Time Slot	16 μ s
SIFS	32 μ s
DIFS	64 μ s

Table 1.1: WAVE Physical Characteristics

1.8 WAVE Channel Coordination

The WAVE spectrum is composed of only one control channel (CCH) and six service channels (SCH). The control channel is considered as the public room for all WAVE devices and its critical resource. Efficient organization and minimization of traffic on the CCH is a challenging problem. The CCH should only be used for service advertisement frames and broadcast messages (i.e. when the transmitter has not

negotiated with a specific receiver yet); however, no active connections between two or more devices are allowed to exchange data over the CCH (i.e. after handshaking, the transmitter and receiver must pursue talking in another channel). The channel access function used to organize contention over the CCH (and SCHs as well) is the EDCA. Table 1.2 summarizes CW and AIFSN parameters for different access categories over the CCH. [Note: $CW_{min}=15$ and $CW_{max}=1023$]

ACI	AC	CW_{min}	CW_{max}	AIFSN
0	Background	CW_{min}	CW_{max}	9
1	Best Effort	$(CW_{min} + 1)/2 - 1$	CW_{min}	6
2	Video	$(CW_{min} + 1)/4 - 1$	$(CW_{min} + 1)/2 - 1$	3
3	Voice	$(CW_{min} + 1)/4 - 1$	$(CW_{min} + 1)/2 - 1$	2

Table 1.2: EDCA Parameter Set Used in CCH

The other six SCHs are considered as private rooms for any connection to exchange long streams of data. Before initiating a connection over a SCH, a node must first join an active logical private network (namely, the WAVE Basic Service Set ‘WBSS’). Advertisement of new services should be transmitted over the CCH, however, actual data exchange of the service is done over any SCH. Table 1.3 summarizes CW and AIFSN parameters for different access categories over SCHs.

ACI	AC	CW_{min}	CW_{max}	AIFSN
0	Background	CW_{min}	CW_{max}	7
1	Best Effort	CW_{min}	CW_{max}	3
2	Video	$(CW_{min} + 1)/2 - 1$	CW_{min}	2
3	Voice	$(CW_{min} + 1)/4 - 1$	$(CW_{min} + 1)/2 - 1$	2

Table 1.3: Default EDCA parameter set used in SCH

1.9 WAVE Basic Service Set

The WAVE Basic Service Set (WBSS) is a concept that should be clear before discussing the deployed communication protocols. Due to the distributed manner of WAVE protocols, applications that want to establish a new connection with remote devices must first announce for the new service on the CCH within a WBSS advertisement frame. The WBSS advertisement frame contains the originating application, intended recipient devices (which could be a broadcast), data rate and the intended SCH to be used.

On receiving of the WBSS advertisement frame, the provider node as well as user nodes should switch to the indicated SCH to proceed with data exchange. Hence, the WBSS is a logical private network of two or more WAVE devices having same active application(s) and participating in data exchange over any of the SCHs (no WBSS is allowed on the CCH).

Any node can announce for a new WBSS while other nodes, on receiving of the advertisement frame, have the right to join it according to their currently active applications. A device can join only one WBSS at any time. A WBSS can support services for multiple applications and can be joined by many users.

There are two types of WBSS, persistent WBSS and non-persistent WBSS. A persistent WBSS is announced periodically in each CCH interval (the time interval when all WAVE nodes listen to the CCH). This type could be used to support services of indefinite lifetime (e.g. a RSU offering Internet access) so that they can be joined by nodes that newly come into range. A non-persistent WBSS is announced only once on its initiation, and could be used to support WBSS with limited lifetime.

1.10 WAVE Communication Protocols

WAVE supports two protocol stacks, the standard Internet Protocol Version 6 (IPv6) and a new specially designed WAVE Short Message Protocol (WSMP).

1.10.1 Internet Protocol Version 6 (IPv6)

WAVE networking services support data exchange using the Internet Protocol version 6 (IPv6) [25] with both TCP and UDP at the transport layer. The existence of IPv6 protocol in the wireless device within vehicles opens the Internet access with a tremendous variety of possible applications. Connection using IPv6 is permitted only on SCHs after joining a WBSS.

1.10.2 WAVE Short Message Protocol (WSMP)

The WAVE Short Message Protocol (WSMP) is a new protocol designed especially for an optimized operation in WAVE environments. If any node prefers not to join a WBSS (for example, a transmitter has a short data to broadcast) it will have to use only WSMP over the CCH. WSMP is used for direct transmission of short messages without joining WBSS. Messages of this protocol are designed to consume minimal channel capacity. Hence, it is the only protocol allowed over the CCH (and may be used on any SCH as well). The suggested frame format of a WAVE Short Message (WSM) is shown in Fig. 1.9 (lengths are in octets of bits).

1	1	1	1	1	4	2	variable
WSM Version	Security Type	Channel Number	Data Rate	Tx Power Level	Provider Service Identifier	WSM Length	WSM Data

Figure 1.9: WSM Frame Format

The WSM Version is used version of WSMP. The Security Type indicates the security processing of the WSM Data (i.e. the transmitter application can sign or encrypt the message with an indication in security field). The Channel Number is used to identify the radio channel used for the WSM. The 'Data Rate' indicates the data rate used for the WSM. The Tx Power Level indicates the transmit power used for the WSM. The Provider Service Identifier identifies the application that originated the WSM (each application will have a unique number). The WSM Length indicates the

length in octets of the following WSM Data field (limited to 1400 in its default value). The WSM Data contains the application data being transferred.

1.11 WAVE Management Plane

The WAVE management plane is considered a logical low-level database of the system and performs system configuration and maintenance functions. It consists of the WAVE management entity (WME) with a special part to serve the MAC layer namely MAC layer management entity (MLME) and another one to serve the physical layer namely Physical layer management entity (PLME). Examples of its use include:

- Prior to the first operation of the transceiver (i.e. network configuration phase) different system parameters are loaded into the device's WME. This field is known as Local Information.
- Active applications register their parameters with the WME. Therefore, MAC layer can determine whether a received WBSS advertisement is of interest to any of its applications or not. This field is known as User Service Information.
- The WME is responsible for generating the WAVE service advertisement frame on an application request. This field is known as Provider Service Information.
- On the initiation or joining of a WBSS, network parameters are registered in the WME.

1.12 WAVE Synchronization

During data exchange within a WBSS over a SCH, critical events (e.g. public safety related messages) and new service advertisements with higher priorities may take place over the CCH. Thereupon, the WAVE system requires that all participating devices should monitor the CCH during a small common time interval (CCH interval) on a regular basis. WAVE depends on GPS devices to acquire synchronization with reference to the Coordinated Universal Time (UTC). Each UTC second is divided into ten sync intervals, which in turn divided into a CCH interval followed by a SCH interval separated by a guard interval, as shown in Fig. 1.10.

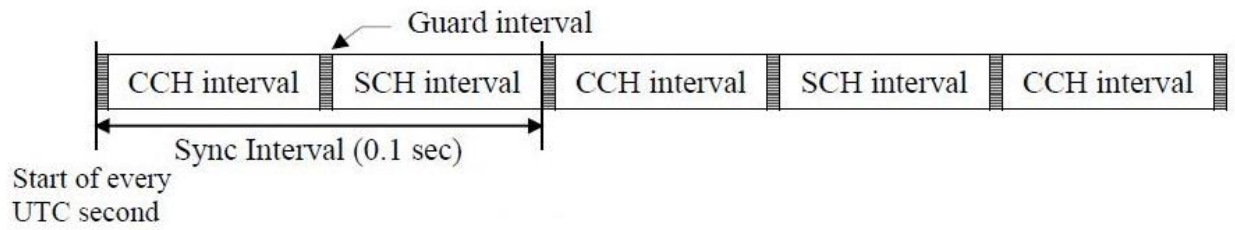


Figure 1.10: WAVE Synchronization

Devices without access to a precise timing signal (e.g. GPS) may acquire synchronization from other WAVE devices upon receiving of WAVE advertisement frames, as the time will be included within the frame. This concludes the introduction of the field of study.

Chapter 2

RESEARCH AND DEVELOPMENT

2.1 Simulating VANETs

Wireless vehicular ad hoc network (VANET) research plays an important role within the field of Intelligent Transportation Systems (ITS). To evaluate VANET protocols and services, the first step is to perform an outdoor experiment. Many wireless technologies such as GPRS, IEEE 802.11p and IEEE 802.16 have been proposed for reliable traffic information. Before the technology hits the ground and can meet the expectations, a series of experiments should be performed to test it. These experiments could be expensive and highly complex to inherit all kinds of situation. Implementing it in public road networks may arise following difficulties.

- Testing on a public road network is not safe.
- It may generate additional traffic jams.
- Hardware modules are very expensive.
- Frequency spectrums are not available yet.
- Vehicles and roadside devices should redesign to support vehicular communication.
- Difficult to implement very large vehicular network (Ex. a city)

Using the simulation approach in the protocol design and evaluation phases for such a network is economic, flexible, and safe. Two major trends have been observed in the research community when it comes to simulating VANETS. The two are explained below.

1) Stand-alone Network Simulator

A network simulator is used to simulate the networking aspect of the VANET and a mobility framework within the network simulator, coupled with a mobility model is used to model the mobility of the vehicles in the VANET.

2) Network Simulator Coupled with Traffic Simulator

In this method of simulating VANETS, a network simulator is coupled with a dedicated traffic simulator. A network simulator that can be used to study such a network must be able to

- ✓ Simulate communication and network protocols (i.e. network simulation),
- ✓ Simulate road and vehicle traffic (i.e., traffic simulation),
- ✓ Tightly integrate both network and traffic simulations so that there is a feedback loop between them during simulation.

Here the network simulators are only responsible for simulating the networking protocols used by the VANET. The mobility of each node in the simulation is controlled by the dedicated Traffic Simulator and the updated position is sent to the Network Simulator periodically.

2.2 VANET Mobility Models

The Mobility Model governs the set of rules that define movement pattern of nodes in ad-hoc network. Network simulators can then, by using this information, create random topologies based on nodes position and perform some tasks between the nodes.

Using VANET pose a challenge and that is how to separate a mobility model at Macroscopic and Microscopic level. Mobility Model includes some constraints like streets, lights, roads, buildings, cars, vehicular movements and inter-vehicle behavior. These constraints are divided into two parts that are dealt with separately. The node mobility includes streets, lights, roads, buildings etc. and is classified as Macroscopic, whereas the movement of vehicles and their behaviors are classified as Microscopic. We can also analyze mobility model as Traffic generator and Motion generator. Motion constraints are designed by car driver habits, cars and pedestrians and describe each vehicle movement. The Traffic generator creates random topologies from maps and defines the vehicular behavior under environment. The mobility model is described by the framework, which includes topological maps like lanes, roads, streets, obstacles in mobility and communication model, car velocities, the attraction and repulsion points, based on traffic densities relating to how the simulation time could vary, vehicular distribution on roads and intelligent driving pattern.

There are various models, which can generate mobility patterns based on certain criteria. While it is hard to present real world traffic scenarios in a single simulation model, ways can be adopted to develop a protocol suite which can support the implementation. The mobility patterns can be generated from various models. These models are described below.

I. Survey models

Survey models represent realistic human behavior in urban mesh environments. The model relies on data collected through surveys performed on human activities.

II. Event driven models / trace models

Vehicle traces are created by directly extracting generic mobility patterns from vehicle movement traces. Though this approach gives a rather realistic mobility model, it has a major drawback of limited availability of vehicular traces.

III. Software Oriented Models

Various simulators like VISIM, CORSIM and TRANSIM are able to generate the traces of urban microscopic traffic. VANETMobiSim uses TIGER database and Voronoi graphs to extract road topologies, maps, streets etc. for the network simulators. The problems with such simulators are that they can only operate at traffic level and cannot generate realistic levels of details. Moreover the inter-operability with network simulators and the generated level of details seems insufficient for network simulators

IV. Synthetic Model

The majority of the work has been carried out in the area of synthetic modeling. All models in this category use mathematical equations to develop realistic mobility models. The strength of mathematical models is validated by comparing them with real mobility models.

2.3 Stand-alone Network Simulator - EstiNet Network Simulator and Emulator (NCTUns)

NCTUns (National Chiao Tung University Network Simulator) based on Harvard simulator proposed by S.Y. Wang in 2002. NCTUns is purely written in C++ with a powerful GUI support. The user need not worry about the complex coding as NCTUns hides the complexities with a few mouse clicks. Since its first release, NCTUns has supported a distributed simulation approach to run simulation tasks concurrently.

NCTUns 1.0 was originally developed as a network simulator with unique network simulation capabilities. In the release of NCTUns 4.0, NCTUns incorporates traffic simulation (e.g., road network construction and microscopic vehicle mobility models with its existing network simulation, tightly integrates them together, and provides a fast feedback loop between them. The simulator was commercialized after the release of NCTUns 6.0 and rebranded as **EstiNet**. EstiNet has become a useful simulation platform for wireless vehicular ad hoc network research.

Due to the interest in ITS (Intelligent Transportation System) research, a lot of simulation tools are being developed to support vehicular simulation. NCTUns also showed an interest in ITS project and added support of ITS simulation in its version 4.0. Unlike Veins, NCTUns tightly couples traffic and network simulators inside a single module to provide single vehicular network environment. After the release of version 4, it added the following support for ITS simulation.

- ✓ Driver Behavior Model
- ✓ Network road construction
- ✓ RSU (Roadside unit) Simulation
- ✓ Onboard unit (OBU) device equipped with IEEE 802.11(b) Ad hoc mode
- ✓ IEEE 802.11(b) infrastructure mode
- ✓ GPRS radio
- ✓ DVB RCTS satellite radio

NCTUns can simulate multiple wireless interfaces inside one node including 802.11(p) interface. After the release of version 5, NCTUns enhanced its usability of

the ITS project by supporting a large network simulation with the possibility of automatic road assignment using SHAPE-format map file. It uses a car agent module to control vehicular movement dynamic on road with the possibility of autopilot assignments and pre-defined assignment. In autopilot assignment, all the vehicles in the topology are assigned with automatic parameters to control dynamic traffic flow during the simulation. In a pre-defined assignment, a user has to manually assign values for traffic flow. With its intelligent driving behavior the car agent can model a car to obey certain parameters like traffic light, nearby vehicle, changing the lane, taking the turn and car following model. There are four types of road assignment in NCTUns, which allow users to define their custom topology. With the powerful use of GUI tool, it is now possible to deploy the vehicles automatically.

Talking of network's perspective, NCTUns can simulate 802.11a, 802.11b, 802.11g and 802.11p technologies. It includes free space, two ray ground and free space with a shadowing path loss model. It further includes Rayleigh and Ricean as a fading model. NCTUns implements directional, bidirectional and rotating antenna types. The SNR calculation is cumulative and the signal strength is determined from the sender's and receiver's perspective point. NCTUns implements block objects to introduce the hindering object between wireless signals. The Wall object can completely block the wireless signal or can attenuate the signal with a specified value. The hindering object gives good simulation environment to observe the effects of multi hop wireless network simulation. During the simulation, each node is allowed to send either a UDP or TCP packet. However, there is a limitation in NCTUns. Most of the Network simulators allow multiple TCP/IP versions (Tahoe and New Reno) inside single simulators whereas; NCTUns allows a single instance of TCP/IP version.

Unlike VEINS, NCTUns integrates traffic and network simulators inside with a feedback to support vehicular network simulation. However, NCTUns can support a maximum of only 4096 nodes inside a single simulation.

2.4 Vehicles in Network Simulation-VEINS

We needed a robust simulation platform for VANET where in the results of the simulation are not entirely dependent and heavily influenced by the mobility model in use. This led to the development of VEINS (Vehicles in Network Simulation) an open source simulation framework which makes use of two simulators OMNeT++ for handling the network simulation and SUMO for handling microscopic road traffic simulation. The two simulators run in parallel and communicate over a TCP connection

VEINS is the outcome of the 'Veins Research Project' [5], and was developed by the team comprising of Christoph Sommer, Dr.Falko Dressler and David Eckhoff at the Computer Networks and Communication Systems Department, University of Erlangen, Germany. The main aim of this project was to realistically simulate Inter-Vehicle Communications (IVC). Also they realized the need of bidirectional coupling between the traffic and network simulation needed to realistically simulate the effects of VANET applications on the mobility of the vehicles. To this end, a simulation framework was created by coupling a dedicated network

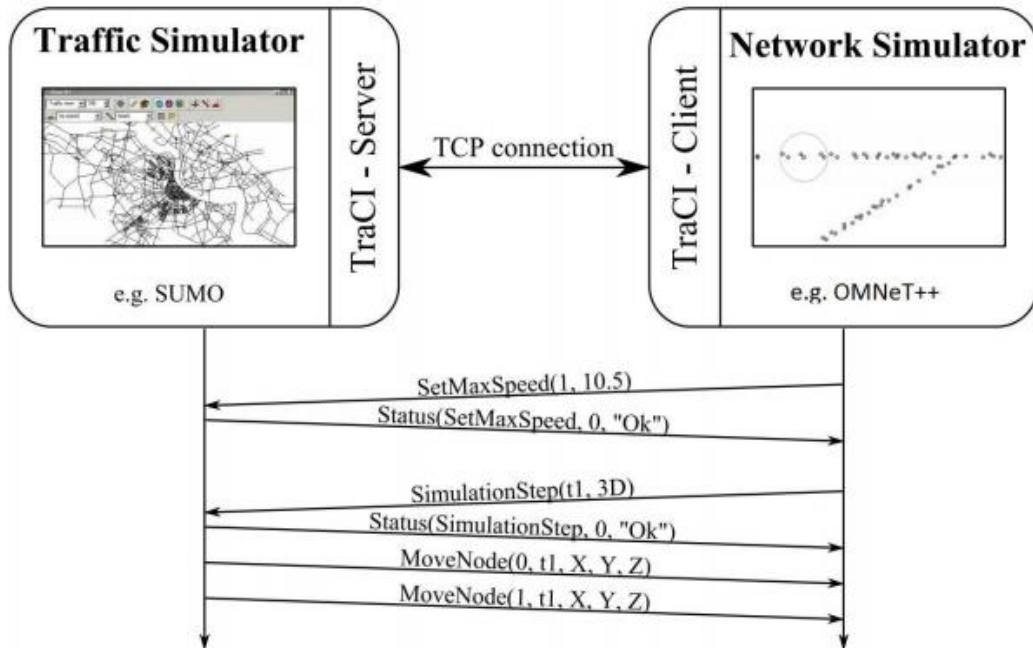


Figure 2.1: Network Simulator OMNeT++ coupled with Traffic simulator SUMO using TraCI

simulator, OMNeT++ and a dedicated traffic simulator, SUMO (Simulation of Urban Mobility). TraCI (Traffic Control Interface) served as the glue between the two simulators making bidirectional coupling a possibility. This lets the network simulation also influence the traffic simulation and vice versa.

Following are some of the important features:

- Completely open source, hence can be extended as required
- Relies on a trusted vehicular mobility model.
- Models IEEE 802.11p and IEEE 1609.4 DSRC/WAVE network layers in full detail
 - ✓ Multi-channel operation
 - ✓ QoS channel access
 - ✓ Noise and interference effects
- Simulations can be deployed on a single workstation or extended to a computer cluster.
- Obstacle model which accounts for shadowing effects caused by buildings.

2.4.1 Traffic simulation

A transportation system or a road network is made up with all the landmarks of the entire area. It consists of junctions, traffic light systems arterial routes, roundabouts, downtown grid systems, etc. Simulation of transportation systems or Traffic simulation is the mathematical modeling of transportation systems, through the application of computer software to better help plan, design and operate transportation systems.

Traffic modeling is an extremely difficult task due to its high complexity and chaotic organization. Therefore it is difficult to predict an exact traffic model for a selected road network. Researchers are using traffic simulators to predict traffic patterns for their applications.

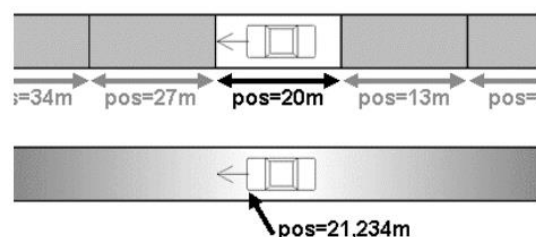


Figure 2.2: Space-discrete vs. Space-continuous simulation

Traffic is conditioned by values like the weather, the infrastructure within the region or other incidents affecting the system. It is not highly conditioned by an individual's private wish for mobility –making up around 60% of traffic – and due to this, neither departure times nor fixed and earlier known routes are available. This is a great problem for modeling traffic itself. Especially the private transit leads to an impossibility of describing traffic by the use of mathematical formulas. Both, a modern human being's wishes to leave and arrive at certain places and at certain times on the one hand, and the movement of the vehicle on the street on the other, influence traffic and one another: the street network's work load depends on the drivers' departure times and determines the speed of movement. This complexity yields in varied behavior of the whole system where system means the generation of traffic and traffic itself, and as no valid mathematical models that take into account all these influences are available, simulation is the only way to show weak points of the street network or predict its traffic.

To understand simulation, it is important to understand the concept of system state, which is a set of variables that contains enough information to describe the evolution of the system over time. System state can be either discrete or continuous. Traffic simulation models are classified according to discrete and continuous time, state, and space

Simulation methods in transportation can employ a selection of theories, including probability and statistics, differential equations and numerical methods.

2.4.1.1 Simulation Of Urban Mobility- Sumo

"Simulation of Urban MObility", or "SUMO" [6] 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 addressing a large set of traffic management topics. It is purely microscopic: each vehicle is modeled explicitly, has an own route, and moves individually through the network.

The following are the features provided by SUMO:

- Open Source
- Time-discrete vehicle movement - where the position of each vehicle in the scenario is updated every time step.
- Multiple vehicle types like cars, trucks, vans, each with unique characteristics like acceleration, reacceleration and top speed associate with them.
- Multi-lane streets with lane changing
- Vehicles follow right of way rules
- OpenGL graphical user interface which provides a graphical view of the ongoing simulation with features like locating any tracking vehicles during runtime.
- Interoperability with other applications during runtime.
- Scenarios form OpenStreetMaps can be imported. OpenStreetMaps is an online wiki of the world map, which allows users to update and maintain maps. Also, it allows exporting of map data in several for-mats. SUMO supports importing
 - ✓ Road networks
 - ✓ Speed limits
 - ✓ Lane counts
 - ✓ Traffic lights
 - ✓ Turn restrictions
- All configuration and data input like route and network definitions are accepted in the form of XML files.

Also, it provides a suite of other supporting applications which help import/generate the network and traffic for a simulation. The following applications are of interest to our work:

- **Sumo-gui:** Provides a Graphical User Interface to the simulation, enabling the user to observe the simulation in action. Allows display properties relating to the vehicles and road networks to be modified. It has a feature enabling the user to follow a particular vehicle also available. Displays information pertaining to the different components forming the scenario graphically.

- **Netconvert:** Enables the user to import road networks from different formats and generate a road network that conforms with SUMO's format. This tool provides support to import maps from .xml files.
- **Duarouter:** Computes the shortest possible routes between source and destination pairs for a vehicle given the road network and other parameters.

2.4.2 Network Simulation - OMNeT++

OMNeT++ is a C++ based object oriented discrete event network simulation framework with a generic architecture and can be used to simulate the following:

- ✓ Wired and Wireless communication networks
- ✓ Protocol modeling
- ✓ Multiprocessors and distributed hardware systems

OMNeT++ was developed at the Technical University of Budapest, Department of Telecommunications and is available for free for academic use. It provides the infrastructure and tools for writing simulations, and is not a ready to use Network Simulator by itself. A simulation model in OMNeT++ consists of a well architected arrangement of reusable 'modules' which are connected to each other and can pass messages between each other. The smallest building block in a simulation model is called the 'simple module'. Simple modules are written in C++, and can be made to behave as desired. Simple modules are grouped together into compound modules. Both simple and compound modules communicate by sending messages to each other's 'gates' over connections that span between modules or other compound modules.

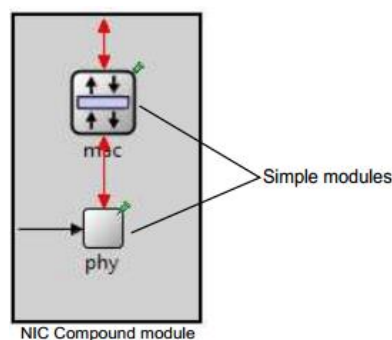


Figure 2.3: Compound module in OMNeT++

A compound module forming a Network Interface Card (NIC) in OMNeT++ is shown in Figure 2.3. Simple modules can be used to model algorithms using C++ wherever necessary. OMNeT++ provides the tools required to define the structure of the entire system by providing the following features:

- Hierarchically nested modules.
- Inter-Module communication using messages through channels.
- Flexible module parameters.
- Network Description language (NED) used to define network topology.
- Graphical and text editing of NED files. An OMNeT++ simulation model consists of the following:
 - .ned files coded using the Network Description language (NED) which describe the position and connections between various modules. Also, values for parameters relating to the simple modules can be defined here.
 - .msg files which contain the message definitions defining various message types with data fields which are later translated by OMNeT++ to C++ classes.
 - C++ source files for simple modules.
 - Simulation kernel used for managing the simulation and simulation class library, all in C++.
 - .ini file is used to specify modifiable parameters explicitly for all the modules involved at any level of the hierarchy.

2.4.3 Mixim (Mixed Simulator)

MiXiM is an OMNeT++ modeling framework for simulating fixed and mobile wireless networks. MiXiM stands for 'Mixed Simulator'. It provides detailed models of various layers of the OSI model along with accurate models for radio wave propagation and interference estimation. It also has a vast library of networking protocols that can be incorporated into any simulation. Since it follows the module hierarchy of simple modules forming more complex compound modules, a great deal of flexibility is available when it comes to developing new compound modules. Also,

user defined simple modules can be added to the framework, enabling the user to model custom compound modules.

The VEINS simulation framework comes as part of MiXiM, adding support for IEEE 802.11p, IEEE 1609.4 and IEEE 1609.3 (WSMP). Also, the obstacles model is available as a simple module.

2.4.4 Traci [7]

TraCI [7] stands for Traffic Control Interface, and this is what gives the outside world access to an already running simulation in SUMO in real time. It uses a TCP based client/server based architecture to provide access to SUMO. As part of the VEINS project, the TraCI Scenario Manager module in OMNeT++ was introduced as part of the MiXiM framework. This gives us the ability to couple OMNET and SUMO together as shown in Figure 2.1. It enables a network simulation running in OMNeT++ to send TraCI commands which control the traffic simulation in SUMO. The TraCI protocol explains in detail the messaging and message format to be used. There are different classes of TraCI commands, those of interest to this paper are listed below.

- **Vehicle Value Retrieval:** This suite of commands queries SUMO about the value of a certain variable pertaining to a particular vehicle from the last time step.
- **Route Value Retrieval:** This suite of commands queries SUMO about information pertaining to a certain vehicle route in the simulation.
- **Change Vehicle State:** This suite of commands sets the value of variables related to a particular vehicle.

2.5 Simulation Setup

The mobility of the vehicles should be modeled first using the SUMO traffic. Then the application is configured with the OMNeT++ network simulator. The following sections will explain how this was achieved.

2.5.1 Sumo Simulation Setup

2.5.1.1 Building Road Networks from XML-descriptions

There are two files used in defining sumo road networks. Junctions are described in the "MyNodes.nod.xml" file and the connections between these junctions are described in "MyEdges.edg.xml" file. Following steps were taken in this process.

- Node descriptions

The node (junction) is described by its attributes. There we defined junction id, location and junction type.

- Edge Descriptions

Here we defined the attributes of the roads which are connecting to the each junction.

2.5.1.2 Generating A Sumo Network Using Netconvert

SUMO provides a command line application called Netconvert which is used to import and generate road networks that can be used by SUMO and other tools present in the SUMO package. Also, it projects the geographic co-ordinates (Latitude and Longitude) to x-y coordinates and also applies the required offset to translate the axes to the first quadrant. The following command line arguments are passed to achieve the desired road network.

2, 5.1.3 Generating Vehicle Routes Using DUAROUTER

The resulting file from the previous step is now used to generate a Trips file using the RandomTrips python script provided as part of the SUMO package. This script generated a set of random trip definitions for routes between two points in the road network. These are then used in the creation of vehicle routes.

Previously generated files are now used to generate the vehicle routes using the Duarouter.exe application which again is a part of the SUMO package.

2.5.1.4 SUMO Configuration File

Each scenario in sumo has a .sumo.cfg file associated with it which points to the corresponding .net.xml and .rou.xml files to be used, and also the start and end time of the simulation.

2.5.1.5 SUMO-GUI Configuration

To display the right vehicle colors as set by the Public Safety Application in OMNET through TraCI, the 'View Settings' in the SUMO-GUI application are set to display the 'given/assigned vehicle color'.

2.5.2 Omnet++ Simulation Setup

The simulation in OMNeT++ can be controlled by supplying the required modifiable parameters per module that is being used in the simulation using the .ini file corresponding to that simulation.

2.6 Algorithm

We developed an algorithm for traffic controlling at a four way junction where there are 12 incoming traffic streams as depicted in the following diagram.

2.6.1 Lane connection at a four way junction

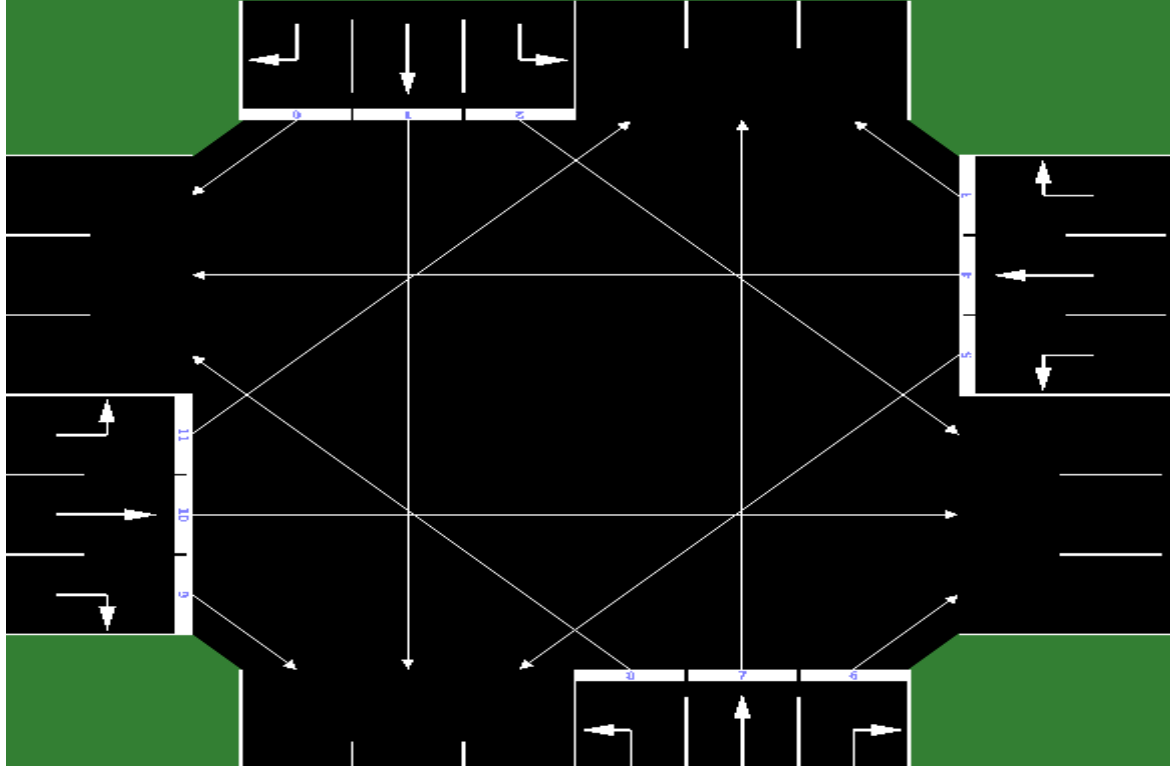


Fig.2.4: Lane connection at a four way junction.

As per to the above diagram, junction traffic condition is not effecting on the traffic streams from lane 0, 3, 6 and 9. Therefore those traffic streams can be allowed at any time without taking in to account for calculation of the adaptive traffic controlling function. One of the following combinations of traffic streams can be allowed to pass through the junction at one time and we have to decide which combination should be allowed at a certain time and how much time period should it be allowed to keep an optimal traffic flow at the junction.

(1,2) , (1,5), (1,7), (4,5), (4,8), (4,10), (7,8), (7,11), (10,11), (10,2), (2,8), (5,11).

Webster developed some equations for the optimal cycle length and the green phase time assignment. They are the basis of fixed-time control which has been widely used. Then Akcelik modified Webster's theory for the over-saturated scenario in a new

signal timing algorithm called ARRB. These methods perform well with low computational costs when traffic conditions are consistent with historical records, but cannot respond to real-time variations. With the development of variety of inexpensive sensors which can be installed at junctions, a number of adaptive traffic control systems have been deployed all over the world, such as SCOOT, SCATS, OPAC and RHODES. These systems are not fully adaptive as vehicles do not provide data in an active manner.

For the simplicity, we can analyse the signal control problem for a single intersection in a simplified mathematical model for a two phase signalized intersection model.

2.6.2 Queues arriving to a four leg intersection

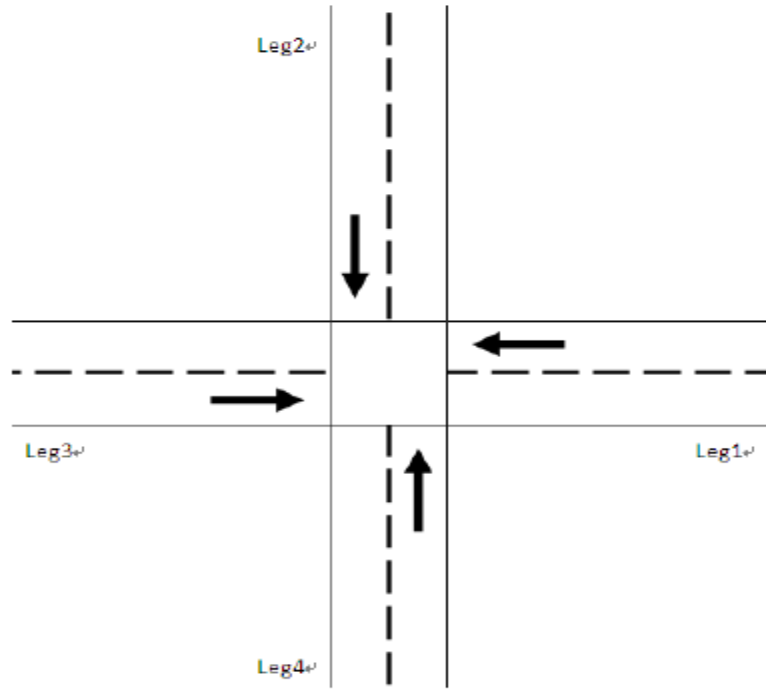


Fig.2.5: Queues arriving to a four leg intersection

In this shape, The Leg 1 and Leg 3 are in phase 1 and The Leg 2 and Leg 4 are in phase 2. The queue length is an important variable that describes the traffic state of an intersection. The queue evolves as,

$$Q_i(n + 1) = Q_i(n) + q_i(n) - d_i(n)S_i(n) \quad (2.1)$$

Where,

$i = 1, 2, 3 \dots \dots \dots M$; index of the traffic stream

$n = 0, 1, 2, \dots \dots \dots N - 1$; discretized time intervals

$Q_i(n)$; in unit of number of vehicles, is the queue length of
the i^{th} stream at the onset of the n^{th} time interval

$q_i(n)$; is the number of vehicles that arrive to the i^{th}
queue in the n^{th} time interval

$d_i(n)$; is the number of vehicles that depart from the i^{th}
queue in the n^{th} time interval

$S_i(n)$; which takes 0 (for stop) or 1 (for go), is the signal state
of the i^{th} stream in the n^{th} time interval.

In fixed-time control and fuzzy intelligent control model, the control variables were considered as follows. For phase 1,

$$(S_1, S_2, S_3, S_4) = (0, 1, 0, 1) \quad (2.2)$$

It means that traffic light is green in lanes 2 and 4 and red in lanes 1 and 3. Therefore, the vehicles can go in lanes 2 and 4 and they should stop in s 1 and 3. On the other hand, for phase 2,

$$(S_1, S_2, S_3, S_4) = (1, 0, 1, 0) \quad (2.3)$$

This means that traffic light is green in lanes 1 and 3 and red in lanes 2 and 4. Therefore, the vehicles can go in lanes 1 and 3 and they should stop in lanes 2 and 4.

Integrating the length of queue with respect to time yields the average vehicle's waiting time in the queue. Let T denotes the length of the discretized time interval. If T is short enough, the vehicles arrivals can be treated as being uniform in every time interval. Hence, by integrating (1) yields,

$$W_i(n + 1) = W_i(n) + TQ_i(n) + \frac{T}{2}q_i(n) - \frac{T}{2}d_i(n)S_i(n) \quad (2.4)$$

Where,

T ; length of the discretized time interval

$W_i(n)$; average vehicle wise waiting time of the i^{th} queue from the beginning of the period to the onset of the n^{th} time interval

Equations (2.1) and (2.4) are the state-space equations describing the dynamic evolution of the traffic state at a single intersection. The waiting time and the number of vehicles are popular performance indices for traffic signal control. The waiting time is used here as the performance index. Therefore, the optimization objective is to minimize the total waiting time considering each lane together.

$$\min \left\{ W(n + 1) = \sum_{i=0}^M W_i(n + 1) \right\} \quad (2.5)$$

As an example let's consider a case where the lane 1 and lane 3 is open during the n^{th} time interval and then the parameters at the beginning of $(n + 1)^{th}$ time interval for each lane as follows,

Lane 1:

$$W(n)= 0 \quad Q= 80 \quad q=10 \quad d=20 \quad s=1$$

Lane 3:

$$W(n)= 0 \quad Q= 100 \quad q=30 \quad d=20 \quad s=1$$

Lane 2:

$$W(n)= 800 \quad Q= 15 \quad q=5 \quad d=0 \quad s=0$$

Lane 4:

$$W(n)= 1200 \quad Q= 60 \quad q=10 \quad d=0 \quad s=0$$

Let's assume the discretized time interval be $T=20s$, then we can calculate waiting time for the time period $(n+1)$ in each lane as follows,

$$\text{For Lane 1: } W(n+1) = 20 \times 80 + 10 \times 10 - 10 \times 20 = 1500$$

$$\text{For Lane 3: } W(n+1) = 20 \times 100 + 10 \times 30 - 10 \times 20 = 2100$$

$$\text{For Lane 2: } W(n+1) = 400 + 20 \times 25 + 10 \times 5 = 950$$

$$\text{For Lane 4: } W(n+1) = 1200 + 20 \times 80 + 10 \times 10 = 2900$$

Aggregated waiting time for phase 1 (that is $S=1$ for lane 1 and lane 3),

$$1500 + 2100 = 3600$$

Aggregated waiting time for phase 2 (that is $S=1$ for lane 2 and lane 4),

$$950 + 2900 = 3850$$

That is waiting time for phase 2 is higher, therefore the traffic controlling system will decide to set phase 2 in the next time interval (that is $(n + 1)^{th}$ time interval).

Due to the use of fixed-time intervals, the traffic signal can only be changed after a fixed period of time. In certain circumstances, this may result in waste of time. Therefore the algorithm can be further improved by assigning dynamic discretized time periods depending on the traffic condition at the junction.

Suppose that lane 1 and lane 3 have more vehicles waiting to pass through and lane 2 and lane 4 have lesser number of vehicles, at a time phase 2 will be appeared (lane 2 and 4 open) at the junction due to the increment of $W(n)$ term of the formula. But much of the green light interval will be wasted if the fixed time T is more than enough for the available number of vehicles on lane 2 and lane 4. In such situations, it is more efficient to allocate shorter time intervals for lane 2 and lane 4 while giving longer intervals for lane 1 and lane 3. On the other hand the transportation efficiency of a lane can increase by allocating continuous longer time intervals for a particular lane. Therefore we decided to allocate time intervals within the range of 10s to 60s proportional to the average queue length of vehicles that belong to a same phase.

Fairness is another important fact that we should consider in this traffic controlling problem. The above algorithm allocates dynamic time intervals for each phase independently without considering the previous stages. A situation may occur in which the algorithm decides to continuously give green light to the same direction of flow (say, leg 1 and leg 3) in consecutive time intervals if the queues in that direction are relatively much longer than the queues in the other direction. As per to the above algorithm, it will increase the waiting time in the term $W(n)$, but we can introduce a policy so that the a queue should deprive the chance for green light at most particular times as an example say 3 times, by this mechanism we can guarantee a fairness for each lane.

2.6.3 Wave Short Message (WSM) Format

Wave Short Message (WSM) is the packet type we used for communication between vehicles. It supports a high data rate with a low latency communication between devices. The WAVE system supports both IP and non-IP applications. For the IP applications, it supports IPv6 traffics and supports WAVE short message services for the non-IP applications. The WAVE Short Message Protocol (WSMP) ensures proper addressing and routing service systems while enhancing high priority and time sensitive communication to vehicles.

WSMs can be sent on both control channel and service channel. It doesn't require any setup before packet transmission and packet size is limited to 1400 bytes.

WAVE accommodates two protocol stacks:

- 1) Standard Internet Protocol (IPv6).
- 2) WAVE short message protocol (WSMP).

WSMP is responsible for network layer activities of the WAVE communication stack and it generates WSMs as per to the application layer requests and pass them to the Physical layer. The WSMP allows applications to directly control physical layer characteristics (e.g. channel number and transmitter power) used in transmitting the messages. A sending application also provides the MAC address of the destination device, including the possibility of a broadcast address. WSMs are delivered to the

correct application at a destination based on Provider Service Identifier (PSID). WSMs are designed to consume minimal channel capacity.

WSM consists of two fields, the header and the data field. The header is fixed size and contains several fixed fields while the data field is dynamic and user can customize the content. In our project we introduced several parameters to the data field as depicted below and these parameters are used for the computation of the above traffic controlling algorithm.

2.6.3.1 WSM Header and Data field

Header	Data field
Wsm Version	isApproaching
Security Type	isOnQueue
Channel Number	noOfVehiclesBehindMe
Data Rate	backVehicleId
priority	frontVehicleId
psid	lane
opp_string psc_var;	allowedLane
wsm Length	decisionCount
sender Address	isOnGoingQueue
recipient Address	state
serial	
sender Pos	
time stamp	

Table.2.1: WSM Header and Data field

2.6.4 Computation of the Algorithm with WAVE Short Messages.

Every vehicle is equipped with a WAVE onboard unit which has attached GPS module. Therefore vehicles know their exact locations (in our project we assumed the 5m accuracy of GPS modules); also there is a database with GPS locations of each and every junction. In this way vehicles know when they reach to a junction.

- 1) When a vehicle reaches less than or equal to 200m to the junction, the driver will be alerted to turn the vehicle to the appropriate lane and then the WAVE module will begin broadcasting WSMs.
- 2) Then if there is a vehicle in front of it which is already a queue member, that front vehicle can measure the distance between itself and the approaching vehicle. If the distance is less than 10m, it will detect the back vehicle as the immediate follower and then the front vehicle will send a unicast WSM to the approaching vehicle in order to add the vehicle to the queue. At the same time the front vehicle will unicast a WSM to its front vehicle notifying that the queue length has been increased by one, this message will be forwarded up to the leader and it is called line forwarding.
- 3) Arrived vehicle is now a queue member and it will store the front vehicle ID obtained from the previous unicast WSM. In this manner the process will continue and the queue length increases, if more vehicles are arriving towards the junction.
- 4) If there is no other vehicle on the lane when a vehicle approaches to the junction, the vehicle will set itself as the leader as soon as it reaches 10m closer to the junction.

In this way each vehicle knows how many vehicles are on the queue behind them at any instance. This scenario is depicted in the following flow chart.

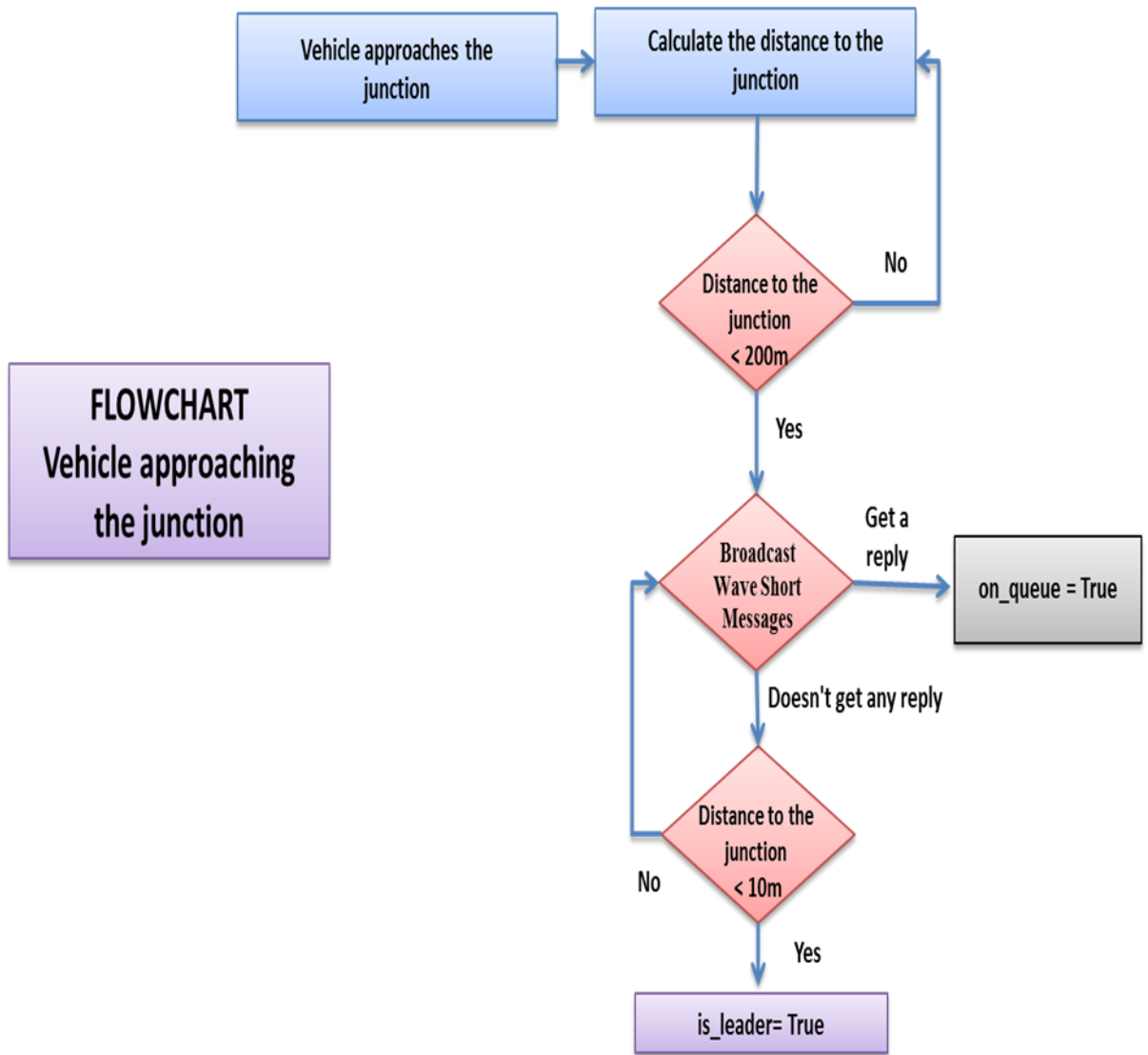


Fig.2.6: Flow chart of an approaching vehicle

Vehicles that are already on the queue will handle WSMs as depicted in the following flow chart. Last member of the queue is responsible for adding a new comer to the queue, first member of the queue is called ‘the leader’ and it is responsible for receiving necessary parameters, computation of the algorithm, collaboratively takes a unique decision with the other leaders and pass the taken decision to its queue members. All the other members except the leader are responsible for line forwarding update messages to acknowledge the leader. This scenario is depicted in the following flow chart.

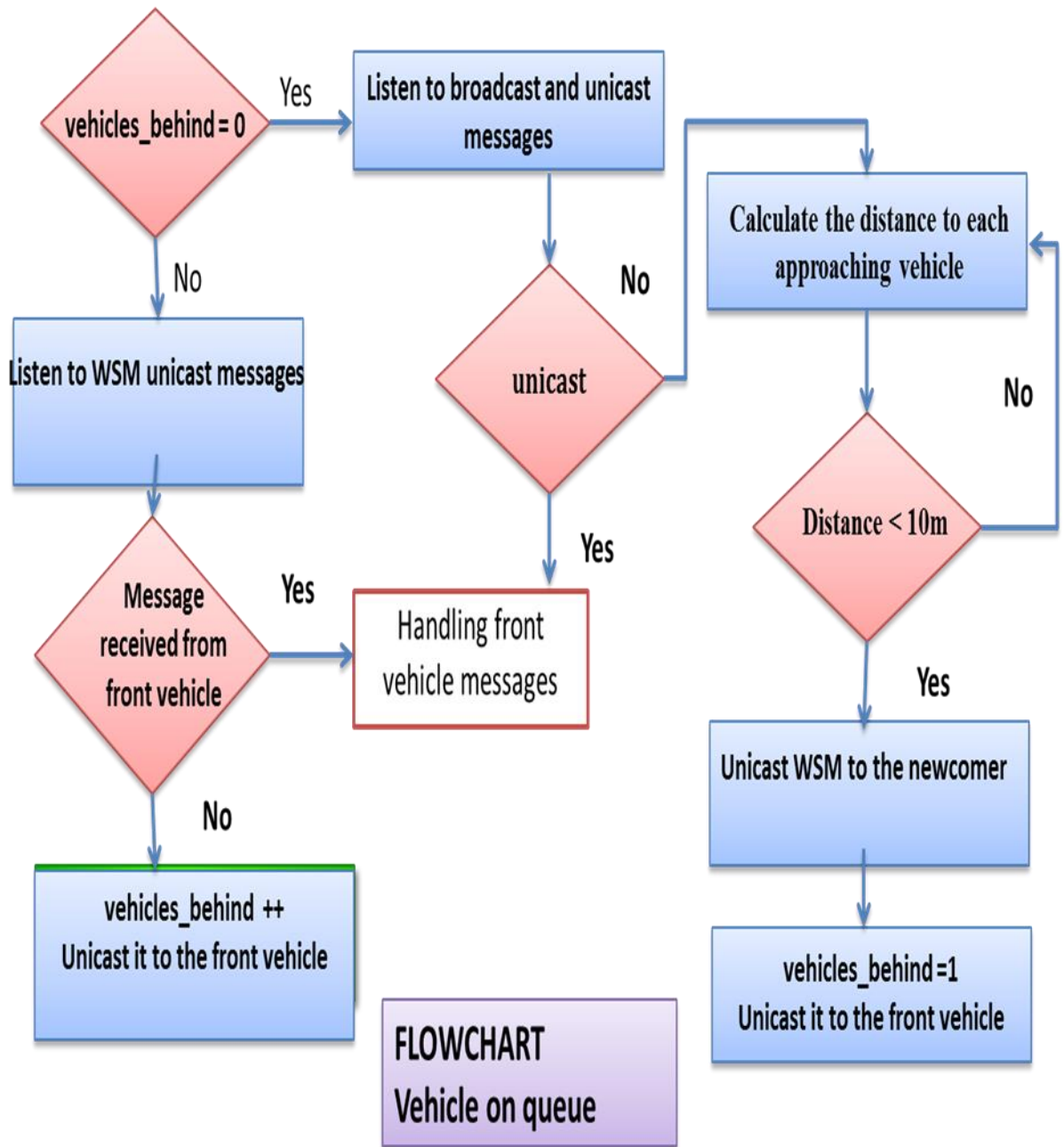


Fig.2.7: Flow chart of queue members

- After a vehicle becoming the leader, it is responsible for taking the decision. This decision consists of the lane numbers which are going to be allowed and the number of vehicles allowed to passing through.
- According to this simplified case, there will be at most 4 leaders at each leg. All the leaders will broadcast WSMs containing details about their legs. This process begins just after it became the leader and at the same time these leaders will be listening for those details from the other leaders.

- Upon reception of traffic details from each leader, a leader can compute the algorithm, but it is not required for every leader to take the same decision in a distributed manner and also it increases the chances for decision conflicts, therefore only one leader is elected to take the decision and notify the decision to the other leaders. Hence the leader with the lowest vehicle ID will compute the decision.

This procedure is illustrated in the following flowchart.

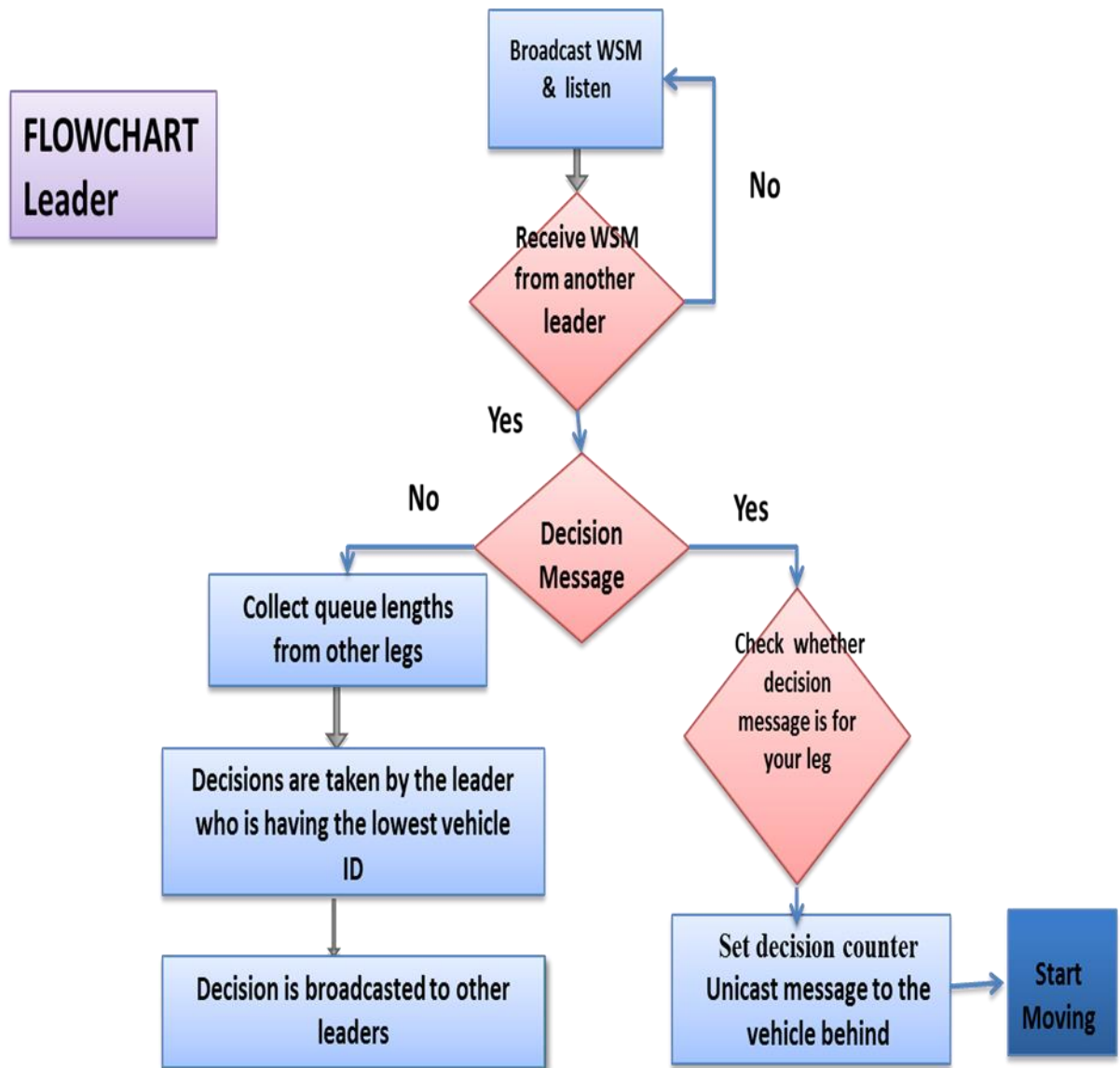


Fig.2.8: Flow chart of a leader

Then the taken decision will pass to the queue members using a line forwarding mechanism in backward direction, this will be directed by the leader of the relevant lanes. The process is illustrated in the following flowchart.

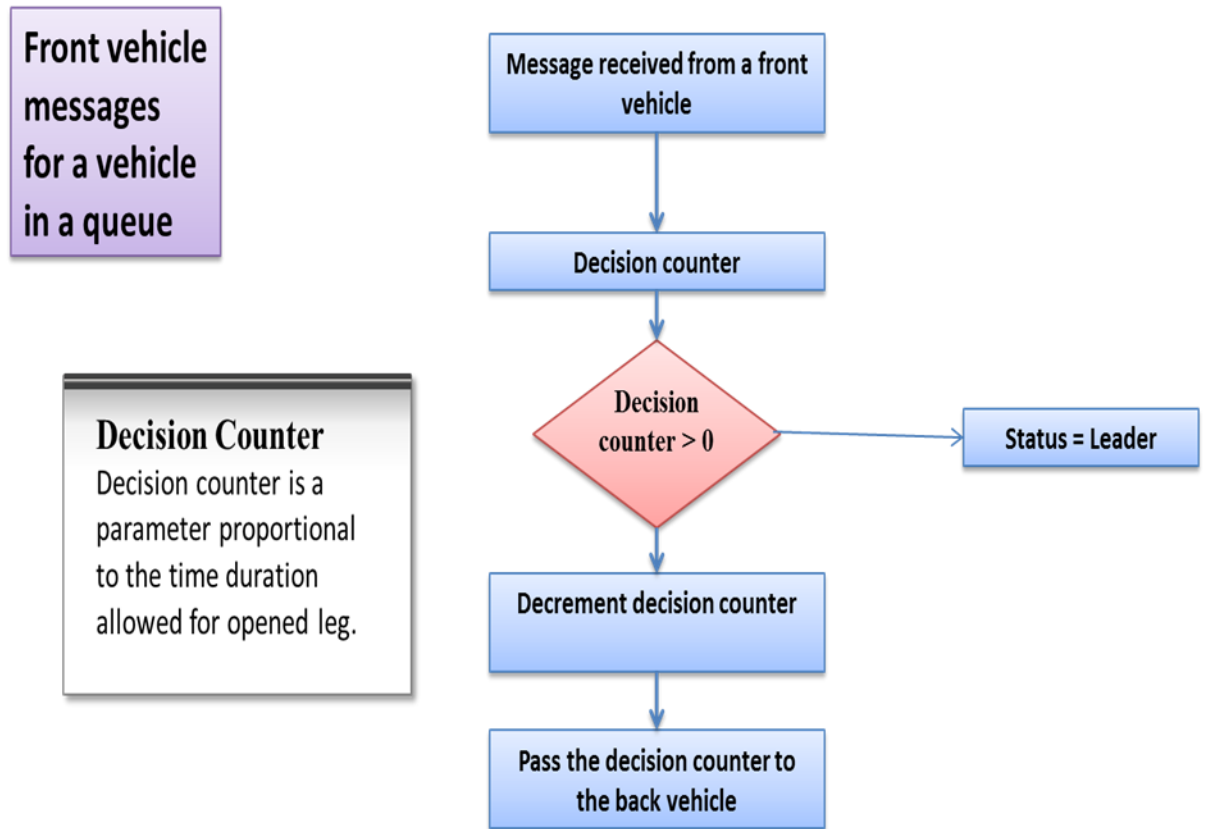


Fig.2.9: Flow chart of the decision distribution

According to our algorithm, a dynamic time interval will be allocated for the chosen lanes to passing through the junction. Onboard units of vehicles on the permitted lanes will display green light and a countdown timer to indicate drivers about the lane open time. In our project we decided to choose a number proportional to the allowed time interval where this number indicates the number of vehicles allowed to pass through.

Even though the algorithm is implemented for the simplified case where there are only two phases at the junction, we can extend the system for the multilane case as described at the beginning of this discussion. In such situation there will be at most 12 leaders and the leaders from lane number 0, 3, 6 and 9 will not be participating for the computation of this traffic controlling algorithm as those lanes do not experience any congestion due to the junction traffic condition.

2.6.5 Developing the algorithm with the VIENS-2.0 Framework

This framework is written in C++ programming language and in this project we modified majorly 4 C++ class files and corresponding header files. Namely,

- 1) TraCIDemo11p.cc
- 2) TraCIMobility.cc
- 3) BaseWaveApplLayer.cc
- 4) WaveShortMessage.cc
- 5) TraCIScenarioManager.cc

2.6.5.1 TraCIScenarioManager.cc

TraCIScenarioManager connects OMNeT++ to a TraCI server running road traffic simulations. It sets up and controls simulation experiments, moving nodes with the help of a TraCIMobility module. This module creates and moves nodes controlled by a TraCI server and all these created nodes must have a TraCIMobility sub-module.

There are some parameters and methods used in this class to handle communication with the TraCI server. Some of those important parameters are listed below,

- bool debug - whether to emit debug messages (default - true).
- simtime_t connectAt - when to connect to TraCI server (must be the initial time step of the server)
- simtime_t firstStepAt - when to start synchronizing with the TraCI server (-1: immediately after connecting)
- simtime_t updateInterval - time interval of hosts' position updates (default - 1s)
- string moduleType - module type to be used in the simulation for each managed vehicle
- string moduleName - module name to be used in the simulation for each managed vehicle
- string moduleDisplayString - module display String to be used in the simulation for each managed vehicle
- string host - TraCI server hostname (default – localhost)

- int port – TraCI server port (default - 8888 (9999 for sumo-launchd))
- bool autoShutdown - Shutdown module as soon as no more vehicles are in the simulation (default – true).
- int margin - margin to add to all received vehicle positions (default – 25).
- void* socketPtr - a socket to connect to the TraCI server.

Important functions used in the class to communicate with the TraCI server,

- void addModule – used for creating a new node in the simulation. node_id, node type, node name, display String, position coordinate, road_id, speed and angle should be provided.
- cModule* getManagedModule - returns a pointer to the managed module named moduleName, or 0 if no module can be found
- void deleteModule – removes a node from the simulation.
- uint32_t getCurrentTimeMs - get current simulation time (in ms)
- void executeOneTimestep - read and execute all commands for the next timestep
- void connect – start connection to the TraCI server.
- bool isInRegionOfInterest - returns whether a given position lies within the simulation's region of interest. Modules are destroyed and re-created as managed vehicles leave and re-enter the region of interest.
- TraCIBuffer queryTraCI - sends a single command via TraCI, checks status response, returns additional responses.
- TraCIBuffer queryTraCIOptional - sends a single command via TraCI, expects no reply, returns true if successful.
- string makeTraCICommand - returns byte-buffer containing a TraCI command with optional parameters.
- void sendTraCIMessage - sends a message via TraCI (after adding the header).
- string receiveTraCIMessage - receives a message via TraCI (and strips the header).
- Coord traci2omnet – convert TraCI coordinates to OMNeT++ coordinates.
- TraCICoord omnet2traci - convert OMNeT++ coordinates to TraCI coordinates.

- double traci2omnetAngle - convert TraCI angle to OMNeT++ angle (in rad).
- double omnet2traciAngle - convert OMNeT++ angle (in rad) to TraCI angle.
- void commandSetSpeedMode – set the speed mode.
- void commandSetVehicleColor – set the colour of a vehicle.
- void commandSetSpeed – set the speed of a node.
- string commandGetEdgeId – get the current road id of a node.
- string commandGetCurrentEdgeOnRoute – current edge lane id of a node.
- string commandGetLaneId - current lane id of a node.
- double commandGetLanePosition – returns lane position when it is given the lane id.
- double commandGetLaneLength – returns the length of the lane.
- double commandGetLaneMaxSpeed – returns maximum speed as defined in the route file.
- double commandGetLaneMeanSpeed – returns the mean speed.

Here the command ‘void commandSetVehicleColor’ did not come with the APIs and hence we created that function in order to identify state change of vehicles during the simulation.

2.6.5.2 *TraCIMobility.cc*

TraCIMobility is a mobility module for hosts controlled by TraCIScenarioManager. It receives position and state updates from an external module and updates the parent module accordingly.

Each node has a TraCIMobility object and it uses this object to send mobility commands to the TraCI interface and to receive location updates from the interface. TraCIMobility module is an extended class of BaseMobility.cc, therefore it has useful parameters inherited from the superclass.

Some of important parameters we used are listed below,

- Move move – Stores the current position and move pattern of the host.
- simtime_t updateInterval – Time interval (in seconds) to update the hosts position.
- cMessage* moveMsg - Self message to trigger movement.

- double currntX – current X coordinate of the node.
- double currntY - current Y coordinate of the node.
- double prvPosX - previous X coordinate of the node.
- double prvPosY - previous X coordinate of the node.
- bool isApproaching – whether the vehicle is moving towards the junction or moving outwards.
- double distance_to_junction – distance to the junction
- int leg - leg number, the vehicle is currently located.
- Int lane – lane number.

Some important methods that we used and modified are listed below,

- void initialize - Assigns a pointer to ConnectionManager and gets a pointer to its host. Creates a random position for a host if the position is not given as a parameter in "omnetpp.ini".
- Coord getCurrentPosition - Returns the current position at the current simulation time.
- Coord getCurrentSpeed - Returns the current speed at the current simulation time.
- void handleSelfMsg - Called upon arrival of a self messages, The only self message possible is to indicate a new movement. If the host is stationary this function is never called and every time a self message arrives makeMove is called to handle the movement. Afterward updatePosition updates the position with the display.
- void makeMove - This function is called every time a MOVE_HOST self message arrives.
- void updatePosition - This function tells that the position has changed, and it also moves the host's icon to the new position on the screen.
- void changePosition – this function is periodically called in simulation time intervals to update the position of the node. We modified this function identify approaching vehicles, assign their lanes and legs.

2.6.5.3 *BaseWaveApplLayer.cc*

This is the base layer of the WAVE application layer. The parent class of this class is *BaseApplLayer* where other applications are extended. We implemented the algorithm in an extended class of this class. The major task of this class is preparing WAVE short messages, sending WAVE short messages and listening to them.

2.6.5.4 *WaveShortMessage.cc*

Each object from this class represents a WAVE short message and this class is defined apropos to the WAVE short message protocol.

Some important parameters of this class are listed below,

- `int wsmVersion_var`
- `int securityType_var`
- `int channelNumber_var`
- `int dataRate_var`
- `int priority_var`
- `int psid_var`
- `opp_string psc_var`
- `int wsmLength_var`
- `bool isApproaching_var`
- `bool isOnQueue_var`
- `int noOfVehiclesBehindMe_var`
- `int backVehicleId_var`
- `int frontVehicleId_var`
- `int leg_var`
- `int allowedLeg_var`
- `int decisionCount_var`
- `bool isOnGoingQueue_var`
- `int state_var`

- opp_string wsmData_var
- int senderAddress_var
- int recipientAddress_var
- int serial_var
- Coord senderPos_var
- simtime_t timestamp_var

Some important methods including field getter/setter methods are listed below,

- WaveShortMessage *dup – copy a WSM.
- void parsimPack – place WSM in the buffer.
- void parsimUnpack – take WSM from the buffer.
- int getWsmVersion
- void setWsmVersion
- int getSecurityType
- void setSecurityType
- int getChannelNumber
- void setChannelNumber
- int getDataRate
- void setDataRate
- int getPriority
- void setPriority
- int getPsid
- void setPsid
- const char * getPsc
- void setPsc
- int getWsmLength
- void setWsmLength
- const char * getWsmData
- void setWsmData
- int getSenderAddress
- void setSenderAddress

- int getRecipientAddress
- void setRecipientAddress
- Coord& getSenderPos
- Coord& getSenderPos
- void setSenderPos
- simtime_t getTimestamp
- void setTimestamp
- void setIsApproaching
- bool getIsApproaching
- void setIsOnQueue
- bool getIsOnQueue
- void setBackVehicleId
- int getBackVehicleId
- void setFrontVehicleId
- int getFrontVehicleId
- void setNoOfVehiclesBehindMe
- int getNoOfVehiclesBehindMe
- void setLeg
- int getLeg
- void setLane
- int getLane
- void setAllowedLeg
- int getAllowedLeg
- void setDecisionCount
- int getDecisionCount
- void setIsOnGoingQueue
- bool getIsOnGoingQueue
- void setState
- int getState

2.6.5.5 *TraCIDemo11p.cc*

The algorithm is composed in this class and each object from this class is associated with a single vehicle in the simulation environment.

There are several important parameters we used to keep information about each vehicle as follows,

- TraCIMobility* traci – the reference of TraCIMobility object associates with the vehicle, this object is used to issue mobility commands.
- bool sentMessage – to check whether a WSM is sent or not.
- cMessage* startMyMsg – to schedule simulation.
- bool WSMenable – configuration parameter.
- int noOfVehiclesBehindMe – to measure the queue length
- int backVehicleId – to send messages to the back vehicle
- int frontVehicleId – to send messages to the front vehicle
- bool isOnQueue – a queue member or not
- int myLeg – leg number the vehicle belongs to
- int myState – state of the vehicle
- int legLength[4] – if the vehicle is a leader , this will store the queue lengths of other legs.
- int leaderId[4] – if the vehicle is a leader, this will keep vehicle ids of the other leaders.
- int allowedLeg – allowed legs by the algorithm
- int decisionCount – number of allowed vehicles to pass through
- bool isOnGoingQueue – keep whether a vehicle is passing through the junction or not.

We developed a set of functions in this class to implement the traffic controlling algorithm. Those important functions are listed below,

- void sendMessage – to send WAVE short messages to other vehicles.
- void readWSMBroadcasts – to process broad cast messages accordingly.
- void LeaderProcess – to take the decision.
- void FrontVehicleProcess – non leader front vehicles will run this function.
- double calDistance – for distance calculations.

2.6.5.6 Class hierarchy of the modified classes

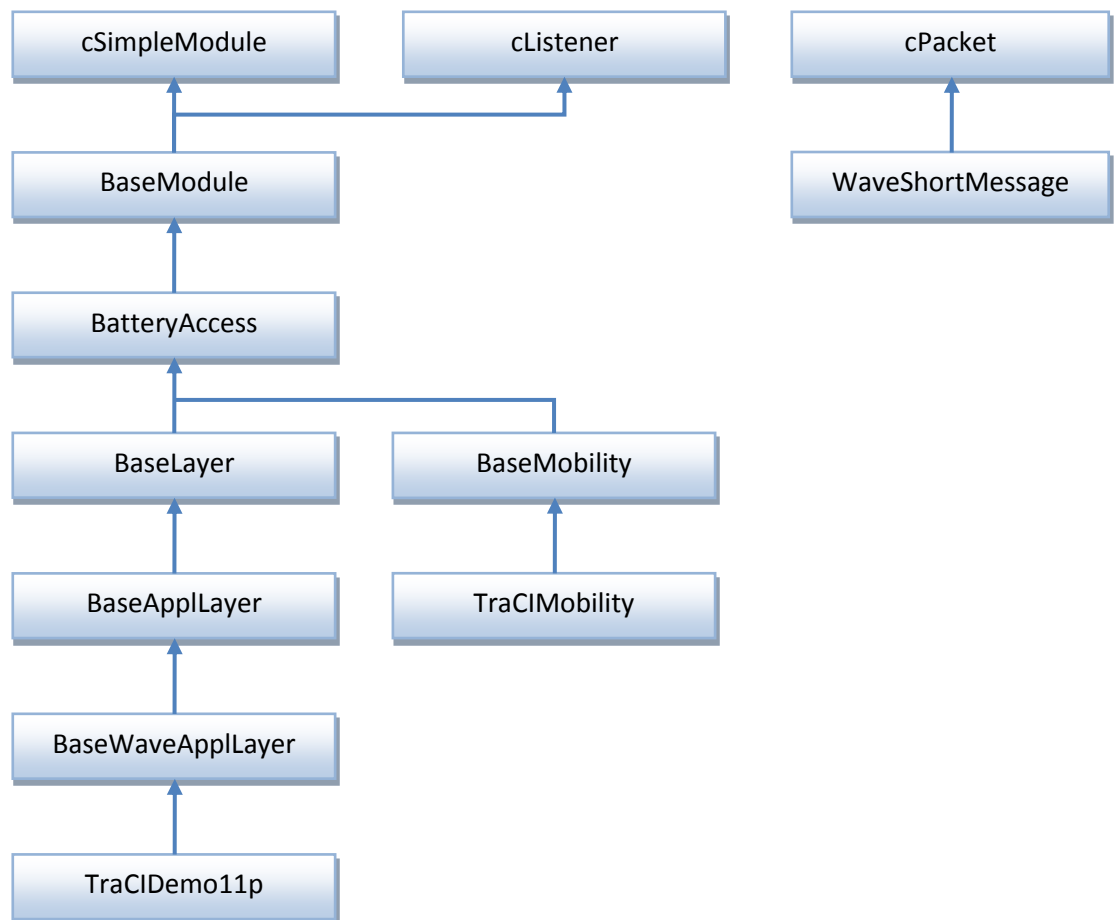


Fig.2.10: Class hierarchy of the modified classes

Chapter 3

RESULTS

3.1 Queue Lengths

The test scenario we evaluate is the simple four way junction. We will study the operations at this intersection without considering the effect of the adjacent intersections. We will focus on comparing two types of signal control strategies. Pre-timed fixed signal control as it currently is and an adaptive strategy based on communication between the vehicles. We assume all vehicles equipped with OBUs. We have run the simulation for both control methods: pre-timed and adaptive based on wireless communications. We have studied the behavior of the traffic for a period of time long enough to catch all the influences of the peak period. Here we use most commonly used traditional traffic light which has following phase time, as the pre-timed fixed signal control traffic light.

Red Light Phase	: 30 seconds
Green Light Phase	: 30 seconds
Yellow Light Phase	: 4 seconds

In fact, the differences between how queues form for the two cases can be seen in the following graphs. Figure 3.1 shows that queues lengths that form under pre-timed control vary significantly among the approaches. Thus when the shorter queues are completely discharged there will still be a large number of vehicles queued on the more demand approaches. From this moment on, some approaches will function on less than full capacity, while other will be over saturated.

Figure 3.2 presents the queue formation under adaptive traffic control mode. It can be seen how the differences among the queues on the approaches with critical flows tend to equalize. In theory, the queues on the critical paths should be equal, however, in practice several limitation appear when generating the signal plan.

3.1.1 Queue Lengths for Pre-timed Control

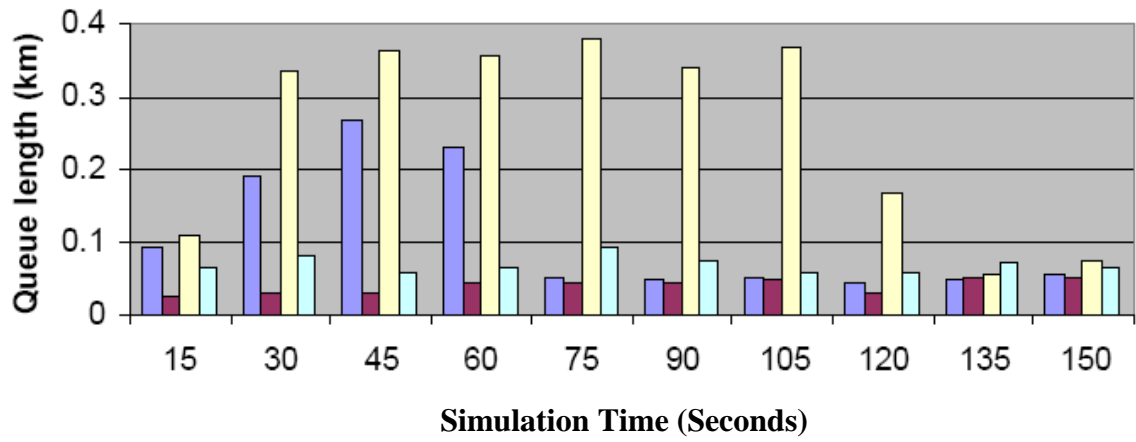


Fig.3.1: Queue Lengths for Pre-timed Control

3.1.2 Queue Lengths for Adaptive Control

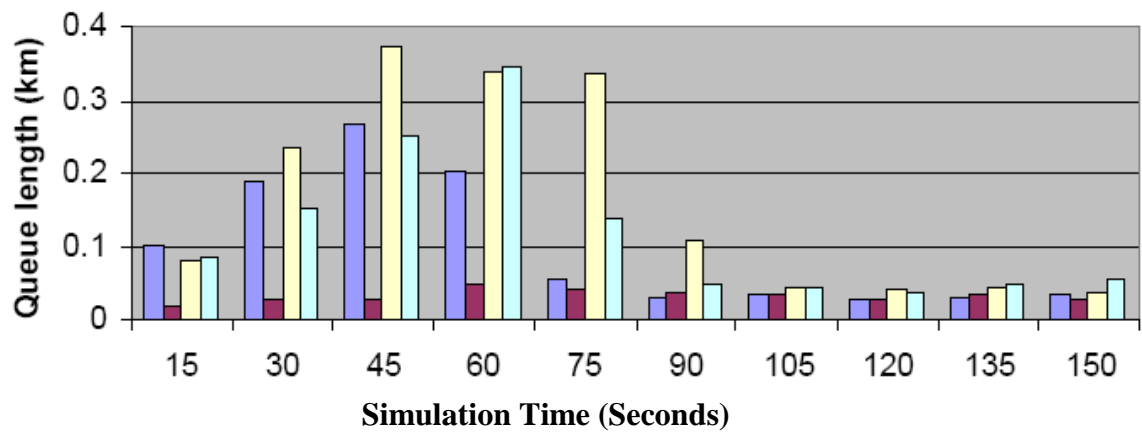


Fig.3.2: Queue Lengths for Adaptive Control



Here we used two performance metrics for performance comparison. One is the total accumulated waiting time of the vehicles to cross the junction. The unit of this performance metric is second and the lower the value, the higher the performance. The other metric concerns the fairness of an algorithm. To see whether an algorithm may sacrifice some vehicles (by letting them wait too long in front of traffic signals) to reduce the accumulated waiting time of vehicles, we also see the statistics distribution of the time saved for each vehicle and the time increased for victim vehicles. If algorithm A can result in fewer victims than algorithm B, we view that algorithm A is fairer than algorithm B.

3.2 Accumulated Waiting Time

Fig 3.3 shows the comparison of the three algorithms under traffic pattern 1, 2, and 3, when there are only 73 vehicles moving on the roads (which represent a low traffic density environment). “Traditional” means that we do not use any intelligent algorithm to control the traffic signals. They simply switch between the red and green lights every 30 seconds. “Adaptive - Original” means that we use the original algorithm proposed in to control the traffic signals. We let the algorithm assess the traffic condition to adjust the traffic signals every 20 seconds. “Adaptive- Improved” means that we use the improved algorithm, which uses dynamic time intervals and an aging mechanism, to assess the traffic condition and adjust the traffic signals. A clear finding is that the original and the improved algorithms, both of which adjust traffic signals based on real-time waiting queue lengths, greatly outperform the traditional one, which does not take the real-time traffic conditions into account. We can see that improved algorithm can reduce the accumulated waiting time of vehicles.

3.2.1 Accumulated waiting time for 73 vehicles

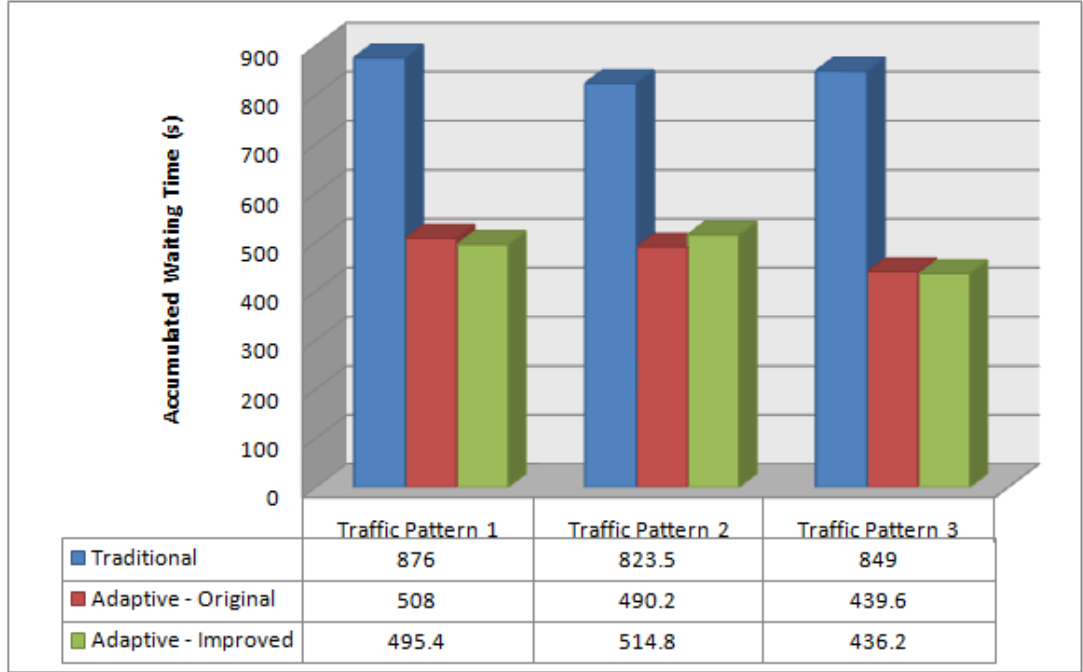


Fig 3.3: Accumulated waiting time for 73 vehicles

3.3 Fairness Of The Algorithm

In Fig. 3.4, Fig. 3.5, and Fig. 3.6, we compare the fairness of the original and the improved algorithms, under traffic pattern 1, 2, and 3, respectively, when the number of vehicles are 73. For each vehicle, we compute the difference of its waiting time between when the fixed-time(traditional) algorithm is used and when the original or the improved algorithm is used. A positive value means a waiting time decrease while a negative value means a waiting time increase. When the original or the improved algorithm is used, we make a statistics distribution of the time changes of all vehicles and classify them into various ranges. The data on the “Original” row show the distribution of the time changes of all vehicles when the original algorithm is used, as compared with the fixed-time algorithm. On the other hand, the “Improved” row show the distribution when the improved algorithm is used as compared with the fixed time algorithm.

3.3.1 Time saved for each vehicle for traffic pattern 1

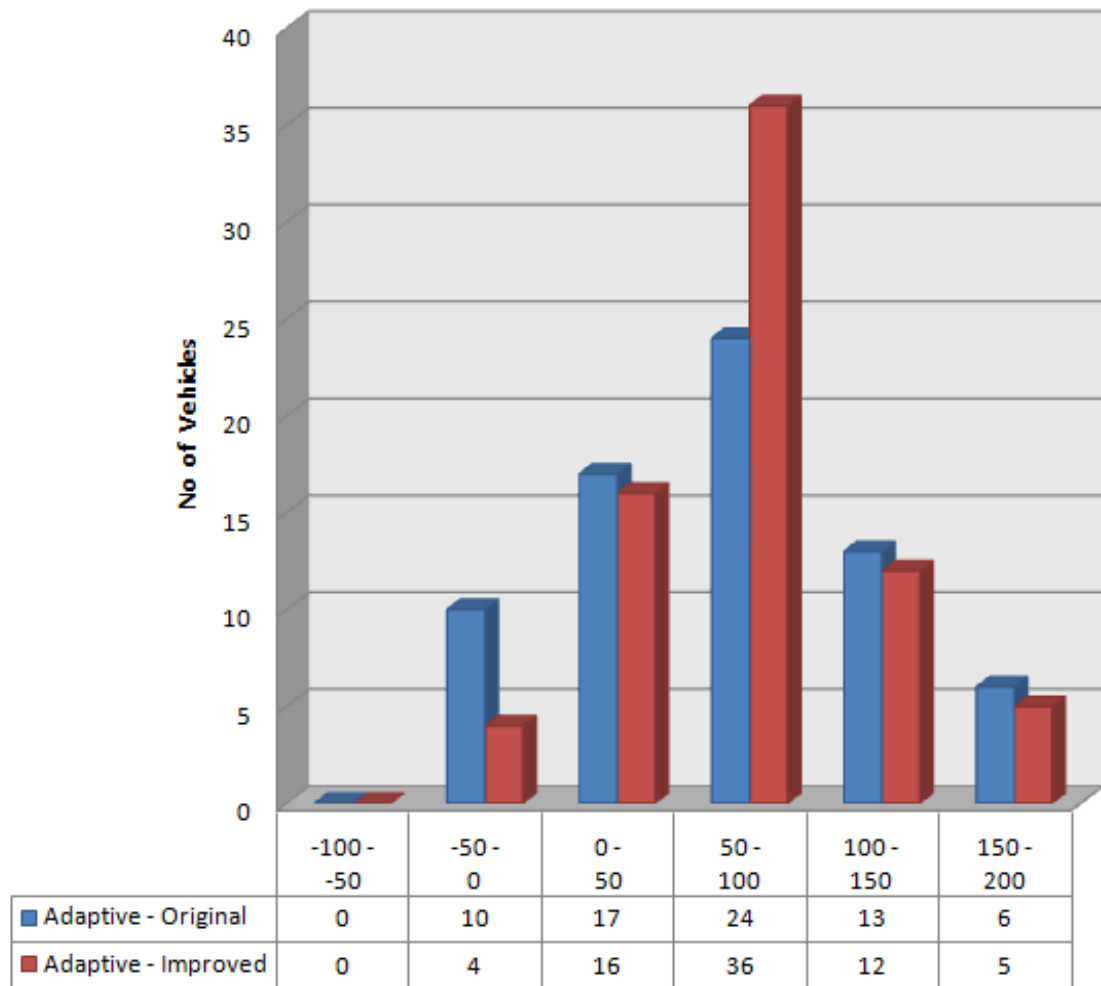


Fig.3.4: Time saved for each vehicle for traffic pattern 1 for 73 vehicles

3.3.2 Time saved for each vehicle for traffic pattern 2

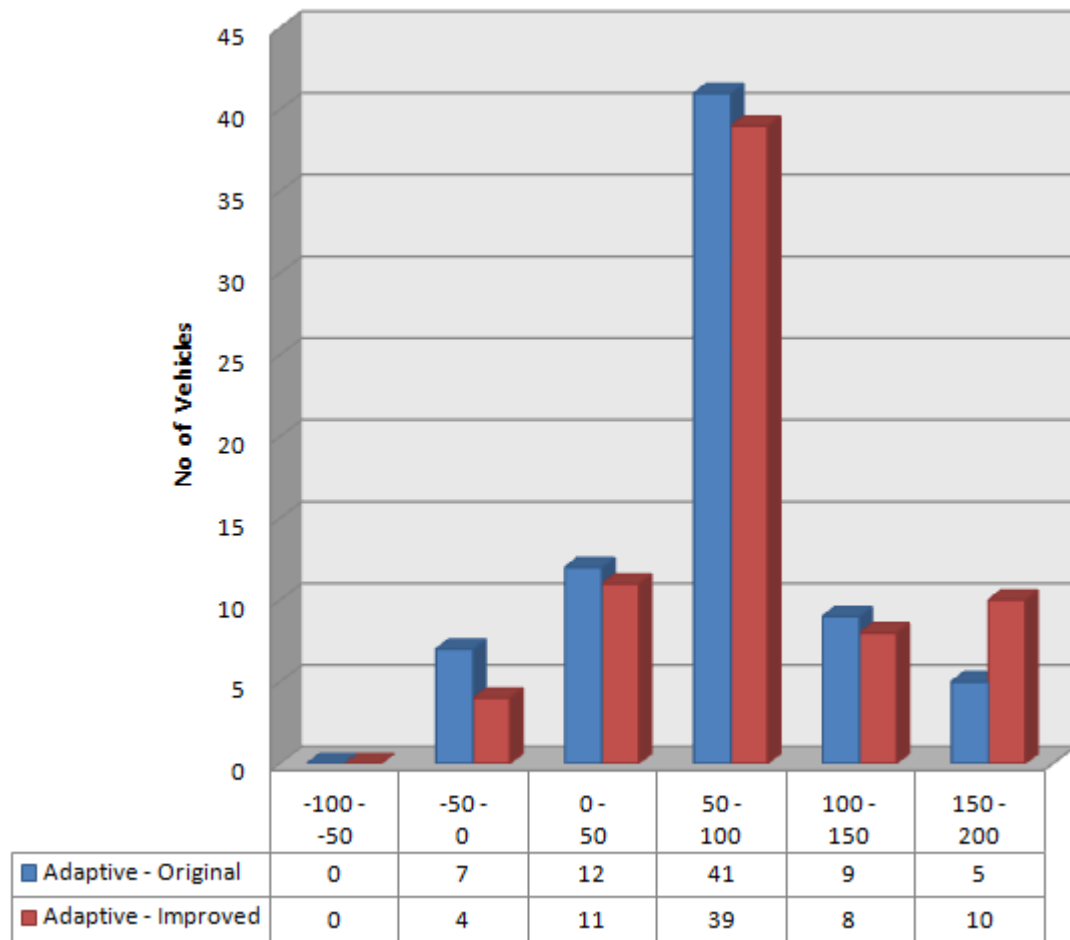


Fig.3.5: Time saved for each vehicle for traffic pattern 2 for 73 vehicles

3.3.3 Time saved for each vehicle for traffic pattern 3

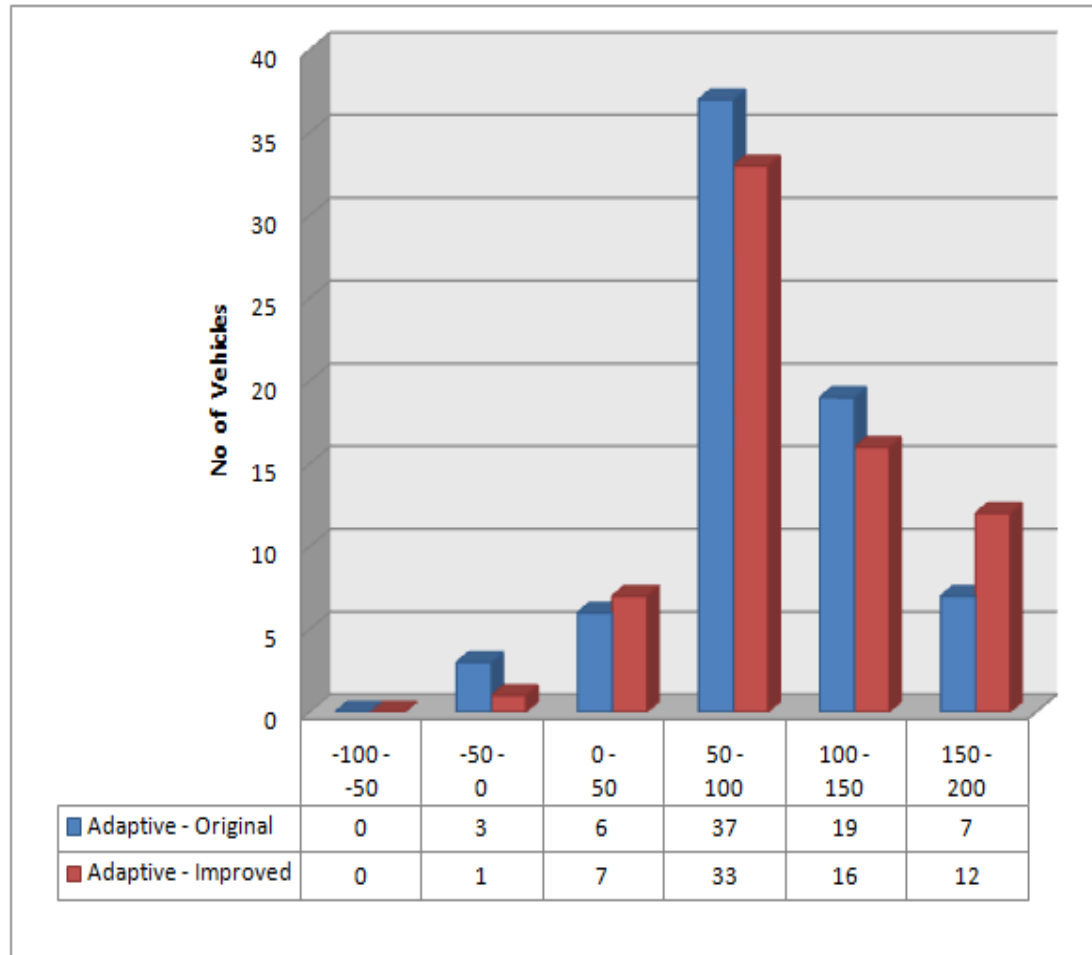


Fig.3.6: Time saved for each vehicle for traffic pattern 3 for 73 vehicles

From these figures, one first sees that most vehicles save their waiting time due to the uses of these intelligent algorithms. However, one also sees that a small number of vehicles increase their waiting time, which we call victims. The improved algorithm can reduce the number of victims than the original algorithm.

Chapter 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

In this paper, we propose a new self-organizing traffic control paradigm whereby the existing physical traffic lights are replaced by in-vehicle traffic lights. We envision that through V2V communications such in-vehicle traffic lights not only can resolve the conflicts at intersections in an autonomous and efficient manner but can also render traffic management truly ubiquitous. The simulations conducted for simple four way junction, provide compelling evidence on the viability and significant benefits of the proposed scheme in terms of the increase in flow rates (more than 60% increase at high densities). This new self-organizing traffic paradigm thus holds the potential to revolutionize traffic control, especially in urban areas.

The control method relies on data collected from vehicles. The system was fully implemented in simulation and managed to out-perform the pre-timed solution for the scenarios we have tested. The goal of an intelligent traffic signal control algorithm is to shorten the time needed for a vehicle to reach its destination. We chose such an algorithm from the literature and then used the ‘Veins’ which is an open source Inter-Vehicular Communication (IVC) simulation framework composed of an event-based network simulator(OMNeT++) and a road traffic microsimulation model(SUMO) to study its effectiveness under various conditions. We found that the chosen algorithm generally can shorten the time for a vehicle to reach its destination but may achieve the goal at the expense of some vehicles waiting too long due to unfair traffic signal control. Therefore, we improved the original algorithm and used Veins to evaluate the performance of the original and the improved algorithms.

4.2 Future Work

The adaptive traffic light control application implemented in this thesis, is a good starting point for implementing a complete Intelligent Transport System consisting of many more applications with varied features all sharing the same network. Also, many other wired and wireless networks that support remote VANET applications can be added.

In direct relation to this thesis, in order to enhance the reach and functionality of the existing application features in certain cases, it will be beneficial to implement stationary road side unit which might run a different flavor of the application, helping retransmit the WSMs, hence increasing the range, and also if connected to a backbone network, can further enhance the reach of the information.

As part of the future work, we plan to analyze the problem of coordination of traffic flows through adjacent intersections in city environments and how to solve this issue in efficient manner involving vehicular communication.

Intelligent dissemination of WSMs is another major research area concerning VANETs. The Application's range can be greatly increased, and also unwanted traffic on the network can be reduced by intelligently routing WSMs instead of using broadcast.

Also emergency vehicle handling and allowing pedestrians to cross the junction in such a intelligent road network, are very useful and important applications.

In the end, it would be of most importance to have a realistic WAVE simulator which simulates traffic patterns and also all the applications and their effects on traffic flow. This will be needed once we have real world VANETs in order to develop effective applications.

LIST OF FIGURES AND TABLES

Figure 1.1: Example of VANET network

Figure 1.2: Traditional Traffic Lights

Figure 1.3: How SCOOT Works

Figure 1.4: SCOOT Control Room

Figure 1.5: SCATS System

Figure 1.6: Cost Comparison of Existing Adaptive Traffic Light Systems

Figure 1.7: WAVE System Components

Figure 1.8: WAVE Protocol Stack

Figure 1.9: Spectrum of WAVE Channels

Figure 1.10: WSM Frame Format

Figure 1.11: WAVE Synchronization

Figure 2.1: Network Simulator OMNeT++ coupled with Traffic simulator SUMO using TraCI

Figure 2.2: Space-discrete vs. Space-continuous simulation

Figure 2.3: Compound module in OMNeT++

Figure 2.4: Lane connection at a four way junction.

Figure 2.5: Queues arriving to a four leg intersection

Figure 2.6: Flow chart of an approaching vehicle

Figure 2.7: Flow chart of queue members

Figure 2.8: Flow chart of a leader

Figure 2.9: Flow chart of the decision distribution

Figure 2.10: Class hierarchy of the modified classes

Table 1.1: WAVE Physical Characteristics

Table 1.2: EDCA Parameter Set Used in CCH

Table 1.3: Default EDCA parameter set used in SCH

Table.2.1: WSM Header and Data field

REFERENCES

- [1] W. Fisher, “Development of DSRC/WAVE Standards”, IEEE 802.11-07/2045r0, June 2007.
- [2] SCOOT Advice Leaflets , Peek Traffic Limited, Siemens Traffic Controls and TRL Limited . www.scoot-utc.com/adviceLeaflets.php (March 1, 2013)
- [3] Sydney coordinated adaptive traffic system, www.scats.com.au (March 1, 2013)
- [4] C. Jones, B. Morrison, S. Siromaskul, “Adaptive Traffic Control Systems – South Carolina Case Studies” February 2012.
- [5] C. Sommer, R. German, and F. Dressler. Bidirectionally coupled net-work and road traffic simulation for improved ivc analysis. Mobile
- [6] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. Sumo - sim-ulation of urban mobility: An overview. In SIMUL 2011, The Third International Conference on Advances in System Simulation, pages 63–68, Barcelona, Spain, October 2011. Computing, IEEE Transactions on, 10(1):3 –15, jan. 2011.
- [7] Stefan Fischer, Jean-pierre Hubaux, Axel Wegener, Michal Pi, Maxim Raya, and Horst Hellbr. Traci: an interface for coupling road traffic and network simulators. Proceedings of the 11th communications and networking simulation symposium, pages 155–163, 2008.
- [8]SUMO User Document
http://sumo.sourceforge.net/doc/current/docs/userdoc/SUMO_User_Documentation.html (March 3, 2013)
- [9] Aamir Hassan,VANET Simulation ,Master’s Thesis in Electrical Engineering . School of Information Science, Computer and Electrical Engineering Halmstad University
- [10] IEEE. Draft amendment to wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Wireless access in vehicular environments. Technical report, IEEE P802.11ptm/D2.01, 2007.