Performance Analysis of WiFi Direct for Vehicular Communication

Balasundram Arunn

(158024L)

Thesis submitted in partial fulfillment of the requirements for the degree
Master of Science

Department of Electronic and Telecommunication Engineering

University of Moratuwa
Sri Lanka

February 2017

Declaration

I declare that this is my own work, and this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or institute of higher learning, and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part, in print, electronic, or any other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The candidate, whose signature appears above, carried out research for the MSc dissertation under our supervision.

Signature: Date:

Signature: Date:

Abstract

Vehicular communication is the key enabler of intelligent transport services (ITS). Vehicular ad-hoc networks can be considered to be the integral component of such communication. The state of art dedicated short range communication (DSRC), which is a technology defined for vehicular communication, requires dedicated hardware. This hinders the penetration of ITS, especially in developing countries. In this thesis, we focus on analyzing the feasibility of using Wi-Fi Direct (WD), which is readily available on many smartphones, as an alternative communication technology for VANETs.

We simulate VANETs using DSRC and WD with the help of network simulator NS3 and traffic simulator SUMO. We validate our model first using existing results, and perform simulations to evaluate the performance of both single and multi-hop communications. Metrics such as throughput, end-to-end delay, packet receiving/loss ratios for both WD and DSRC are considered.

As expected, DSRC demonstrates a better performance with regards to most of the measured parameters. However, we observe that the performance of WD is not drastically inferior. Delays is the most crucial performance measure in a VANET. Experiments with different WD modifications show that the delays in WD based VANETs can be reduced by modifying the WD protocol. As a whole, our results indicate the potential of WD as an alternative communication technology for VANETs. Several performance gaps are identified and suggestions are provided in order to enhance WD and bridge those gaps.

Index terms— Wi-Fi Direct, Dedicated short range communication, Vehicular ad-hoc networks.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors Dr. Tharaka Samarasinghe and Prof.Dileeka Dias for making time in their busy schedule to support me, throughout my master study. I would like to thank Dr.Tharaka Samarasinghe for his continuous support in my study and research, sharing his expertise, for his patience and motivation. I would like to thank Prof. Dileeka Dias for her guidance from the completion of my undergraduate education and providing support in various ways throughout my post graduate studies.

Beside my supervisors, I would like to thank Dr. Asanga Udugama for being my progress committee chair and also help me in research by giving valuable suggestions and guidance. I would like to thank Dr. Chandika Wavegedra for being my progress committee member, and for also guiding me with insightful comments. I would also like to thank Dr. Chamitha De Alwis and Dr. Ruwan Udayanga Weerasuriya for being in the panel of examiners and help in improving the thesis.

I would like to thank my university, The University of Moratuwa, for providing financial support and other facilities to conduct my research. I would like to also thank Dialog mobile communication research laboratory at the University of Moratuwa for accommodating and supporting me throughout the study. My sincere thanks also goes to the staff of Dialog mobile communication research laboratory for their support during my stay in the lab.

Last but not least, I would like to express my deepest gratitude to my loving parents and siblings for their support and care through my life.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
DSRC	Dedicated Short Range Communication
VANET	Vehicular ad-hoc networks
OBU	On Board Unit
RSU	Road Site Unit
WPS	Wi-Fi protected setup
DHCP	Dynamic Host Configuration Protocol
P2P	Peer To peer
CTS	Clear to send
RTS	Request to send
ACK	Acknowledgement
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
IP	Internet protocol

Chapter 1

Introduction

Vehicular communication has gained more attention in the recent times because of the increasing number of vehicles and the introduction of automated vehicles. Vehicular communication is the key enabler of intelligent transport servicess(ITS) and its various applications. ITS applications can be categorized into three main types [1]. Firstly, we have active road safety applications that aim to decrease the probability of accidents and save lives. Examples of active road safety applications include collision warnings, overtaking warnings, lane change alerts, emergency vehicle warnings and breaking warnings. Traffic management applications which deal with improving traffic flow, traffic coordination, and traffic assistant [1] can be considered to be the second type. To this end, speed management and co-operative navigation are two main categories of traffic management. Infotainment applications are considered to be the third, and they focus on sharing additional information such as point of interest details with the drivers.

In order to build ITS, reliable communication among vehicles and between vehicles and roadside infrastructure is needed. Additionally, some of the applications require low latent and highly accurate data transfer. Vehicular ad-hoc networks (VANETs) are widely used for the communication in the vehicular environment. Next, we will introduce VANETs.

1.1 VANETS

An ad-hoc network is a network that does not rely on existing infrastructure. In some ad-hoc networks, nodes have mobility, and such networks are called mobile ad-hoc networks (MANETs). VANET is a subcategory of MANET. In VANETs, two types of devices are used for communication, namely on board unit (OBU)

and roadside unit (RSU). Fig. 1.1 explains a VANET with two RSU and number of vehicles.

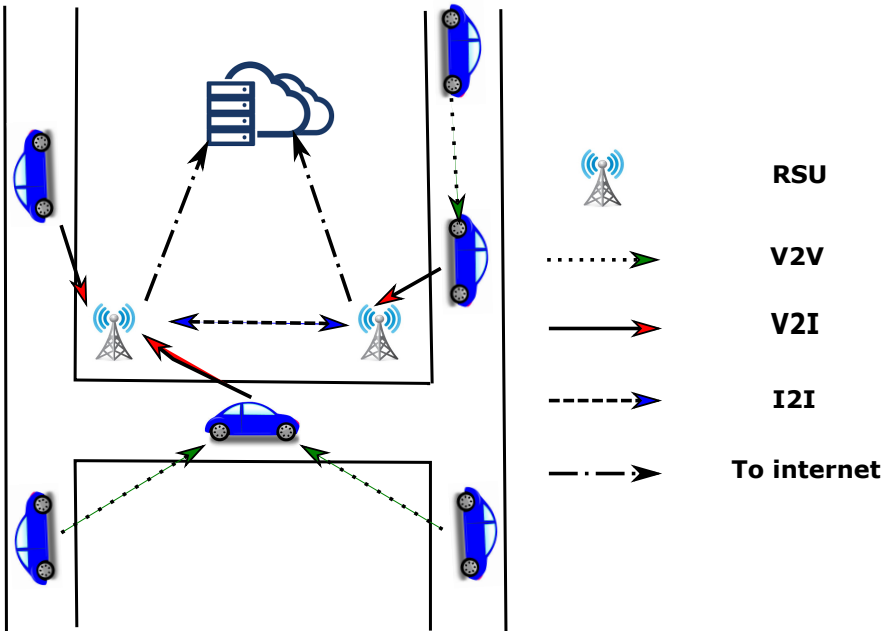

Figure 1.1: VANET architecture.

VANET has to overcome particular challenges prevailing in vehicular environments, such as rapid topology change, fading due to high speed, and multi-path propagation due to reflections from vehicles. To facilitate a robust communication in the vehicular environment, dedicated short range communication (DSRC) was introduced based on the IEEE standard IEEE802.11p.

1.2 Problem statement

Currently, DSRC is the widely used communication technology for VANETs. The majority of the OBUs and RSUs in the market, are based on DSRC. However, DSRC based systems have some drawbacks, due to its requirement of specialized hardware installation in all nodes in the network. The cost of these dedicated DSRC devices are relatively high (commonly more than two times of a flagship smartphone), and the installation requires technical expertise. Additionally, the requirement of a dedicated device will make it difficult to incorporate cyclists and pedestrians into a VANET. These issues slow down the penetration of VANETs, especially in developing countries. Hence, a viable alternative for DSRC is beneficial for the expansion of ITS.

In this work, we focus on analyzing the feasibility of using Wi-Fi Direct (WD) as an alternative communication technology for VANETs. WD is a technology defined by the Wi-Fi Alliance to enable Wi-Fi devices to exchange data without the presence of an access point. WD is available in most of the smartphones, and if not, it can also be implemented and controlled entirely by software. Today, it is reasonable to assume that the majority of vehicles in a transportation network will consist of at least one WD enabled phone. Therefore, if it is used for VANETs, the availability can be utilized to increase the penetration speeds of ITS substantially. Also, WD will alleviate the practicality issues related to incorporating stakeholders such as pedestrians and cyclists to the VANET. It should be noted that WD is implemented over existing Wi-Fi standards, and hence, future high-performance Wi-Fi standards can be easily incorporated. We believe that alternative communication technologies such as WD for VANETs will be extremely useful for developing countries, where ITS seems to be a far-fetched reality.

1.3 Approach of the research and contribution

We have analyzed the feasibility of using WD as an alternative communication technology for VANETs, using a simulation study. A suitable system model is built as the first step to analyze the performances of both WD and DSRC. Wireless channel modeling is essential for an accurate VANET simulation. For this purpose, Two-ray ground model is used as the path loss model [2], and the Nakagami model is used to model the fading [3]. Also, the constant speed model (delay=distance/propagation speed) is used to model the propagation delay. We consider two topological and network models. The first one is a small-scale model, where an RSU is placed at a junction, and two vehicles move away from the junction at a constant speed while sending data to the RSU. This model is used for studying the performance of single-hop communication. The second model, where vehicles move in clusters to and from the junction, is a large scale model used for studying the performance of VANETs with multi-hop communication. Multi-hop communication requires routing, for that purpose we use two routing protocols, AODV (Ad-hoc On-demand Distance Vector) [4] and OLSR (Optimized Link State Routing) [5].

Then, we simulate several performance measures of interest such as,

throughput, delay, and packet receiving/loss ratios. We use network simulator NS3 [6] to simulate the network part of the VANET, and traffic simulator Simulation Of Urban Mobility (SUMO) [7] to incorporate the mobility. Firstly, the topology and network parameters are configured using respective NS3 modules. Secondly, channel properties are configured according to the channel model explained above. Then, mobility traces are generated and exported using SUMO, and finally, the data required to calculate the performance measures are captured using NS3.

NS3 modules are first verified by comparing the generated results with theoretical and experimental results from the literature. Then, simulations are done for the small-scale model, while considering both communication technologies. More specifically, the throughput and the packet loss ratio are simulated. Following the small-scale simulations, the large-scale simulations are done to obtain insights into the performance of multi-hop communication. In order to generalize the results, the large-scale simulations are repeated for 200 independent runs and average results for the end-to-end delay and the packet receiving percentage are calculated. End-to-end delays of several WD modifications are also measured to get an idea about how WD protocol can be tuned for better performance in terms of delay.

As expected, the performance of DSRC is better than that of WD. However, the performance gap between the two technologies is not alarmingly high. Also, some of these performance gaps are identified, and future enhancements are suggested.

1.4 Outline of the thesis

In Chapter 2, some of the topics related to the study are explained for a better understanding about the domain of this study. Also, a literature review is provided at the end of this chapter. The system model used for this study, contains three main parts namely channel model, topological model, and network model, which is explained in Chapter 3. The simulation is done using the network simulator NS3 and traffic simulator SUMO. This simulation setup is explained in Chapter 4. Simulations are done in two phases and the performance measures are also collected and analyzed separately. An investigation of these results is provided in Chapter 5. Finally, Chapter 6 concludes the thesis with the identification of

future extensions.

Chapter 2

Background

This Chapter explains several topics which will provide a background to the research. Also, some of the related works are discussed in the later part of this Chapter. We will start by introducing DSRC, the widely used communication technology in VANETs.

2.1 Dedicated short range communications

DSRC is a standard designed to facilitate an efficient communication between vehicles. A 75 MHz of spectrum, in the 5.9 GHz frequency band, has been allocated for DSRC applications. In this 75 MHz spectrum, 5 MHz is reserved as the guard band, and seven other 10-MHz channels are also defined. These channels are configured into one control channel (CCH) and six service channels (SCHs). The CCH is dedicated for priority messages and control messages while SCHs are used to transfer other types of messages [8].

For PHY and MAC layers DSRC uses IEEE 802.11p standards, a modified version of the familiar IEEE 802.11 (Wi-Fi) standard. In the middle of the stack DSRC uses a set of standards which were defined by the IEEE 1609 Working Group. These standards are defined as 1609.4 for Channel Switching, 1609.3 for Network Services (including the WAVE Short Message Protocol WSMP), and 1609.2 for Security Services. DSRC also supports the use of well known internet protocols for the Network and Transport layers Internet Protocol (IP), User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). The choice between using WSMP or IP+UDP/TCP depends on the requirements of a given application. Single-hop messages, like in collision prevention applications, are typically using the bandwidth-efficient WSMP, while multi-hop packets are us-

Table 2.1: Comparison of PHY layer parameters between IEEE802.11a and IEEE802.11p

Parameter	IEEE802.11a	IEEE802.11p
Frequency band	2.4/5 GHz	5.9 GHz
Channel Bandwidth	20 MHz	10 MHz
Supported Data Rate (Mbps)	6, 9, 12, 18, 24, 36, 48 and 54	3,4.5, 6, 9, 12, 18, 24 and 27
FFT/IFFT Interval	3.2 μs	6.4 μs
Sub carrier Spacing	0.3125 MHz	0.15625 MHz
CP Interval	0.8 μs	1.6 μs
OFDM Symbol Interval	4 μs	8 μs

ing IP for its routing capability [9]. The PHY layer of IEEE 802.11p uses a OFDM channel with 10MHZ bandwidth. Table 2.1 compares several PHY layer parameters of both IEEE 802.11a and IEEE 802.11p.

The MAC layer of DSRC uses a new type of 802.11 communication called "outside the context of a basic service set (BSS)". In traditional 802.11, all data frames are sent between STAs that belong to the same BSS. In "outside the context of BSS", the STAs are not belong to a specific BSS. There is no MAC sub-layer setup required before STAs exchange data in this method.

2.2 Wi-Fi Direct Protocol

Wi-Fi Direct is a technology, defined by the Wi-Fi Alliance to enable data sharing without an access point (AP). WD is available in most of the smartphone and also can be implemented in other phones by software [10]. WD supports most of the Wi-Fi standards (IEEE 802.11 a/d/g/n etc). Since WD can be used for an ad-hoc network, additional to the vehicular application which we are looking for, WD can also be utilized in following applications.

- Vehicle to pedestrian communication.
- Localized chat applications.
- Communication between indoor vehicles (Eg: Folk lifts in warehouses).

WD devices form groups by taking roles as either group owner (GO) or client. Fig. 2.1 explains the structure of a WD group. Here, GO_{g1} and GO_{g2} are the group owners of group1 and group 2, respectively, and C_{gi} indicates the clients of group i. Packets are always transferred through the group owner in the original

WD implementation. Only the GO is allowed to cross connect to the external network in a WD group. Also, the GO of one group can be a client of another group, which can be seen in Fig. 2.1, where GO_{g1} is the GO of group 1 and a client of group 2. Other legacy Wi-Fi devices (not supported by WD) can also join in a WD group by considering the GO as an AP.

Some properties in the original WD implementations cans be considered as barriers in using WD for large-scale networks. In WD group packets are always transferred through the GO and if GO leaves a WD group, it needs to be re-formed, and these cause large delay in a WD network. Also, WD requires user intervention, while connecting two devices, and this affects the formation of autonomous networks, which is required for several ITS applications.

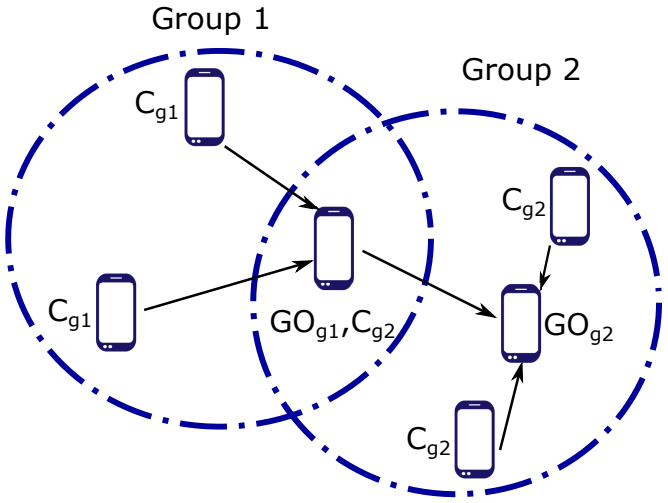

Figure 2.1: WD architecture.

Three types of group formation methods are allowed in WD [10]. Following is the complete list of steps involved in the group formation.

- **Step 1 :** Discovery of devices
- **Step 2 :** GO negotiations
- **Step 3 :** GO announces
- **Step 4 :** WPS setup phase 1
- **Step 5 :** WPS phase 2
- **Step 6 :** DHCP exchange and IP configuration

However, some of WD group formation methods require only a subset of these steps. Here, we describe the three types of WD group formation.

- **Standard formation** - This formation requires all steps listed above. Firstly, P2P devices discover themselves, followed by negotiation of roles using an intent value. One node will be elected as a GO and others will be the clients. Then, the GO announces the credentials to other devices (clients). Finally, the formation will be completed by making secured connection between clients and the GO.
- **Autonomous formation** - In this formation, only steps 1,4,5 and 6 are involved. One device will announce itself as the GO and initiate the formation. Other nodes will complete the formation by joining the autonomous GO.
- **Persistent formation** - This formation is applicable for the devices which have already formed a group, and steps 1,5 and 6 are required. If the devices, which have already formed a group, come nearby again, they could initiate the persistent formation. Devices will get the previously kept roles again in the new WD group. Since they have the information about the roles and security credentials, steps 2 and 4 are not required.

Table 2.2, contains data obtained from [10], provides data about various delays involved in WD protocol.

2.3 Delays

In this section, we discuss about dissecting delay components in a VANET. Firstly, we introduce several delay components and investigate their significance in a VANET kind of network. Then, we provide in depth details about transmission delay, which is the prominent components of the total delay. Finally, we describe the packet transfer mechanisms of both technologies and show how those mechanisms affect the total delay.

Table 2.2: Wi-Fi Direct protocol delays

Description	Delays(s)
Discovery	
Standard	4-9
Autonomous	3
Persistent	4-9
Formation	
Standard	1.5-2.5
Autonomous	1.5-2.5
Persistent	0.5-1.5
Total delay	
Standard	5-10
Autonomous	4.5-5.5
Persistent	4.5-10

2.3.1 Types of delays

Delays involved in packet transferring in a wireless sensor network, such as VANET, can be categorized into four main types.

Transmission delay - Transmission delay is the time taken for pushing the bits of the packet to the link(wireless channel in this case). Transmission delay D_t is given by

$$D_t = N/R_t, \tag{2.1}$$

where, N is the number of bits in the packet, R_t is the transmission rate.

Propagation delay - Propagation delay is the time taken for the signal to travel from sender to receiver in a wireless medium. Propagation delay D_p is given by

$$D_p = d/c, \tag{2.2}$$

where, d is the distance between sender and receiver, c is the speed of the propagating wave.

Processing delay - Processing delay is the time taken for the router to pro-

cess the packet headers. This delay is significant only when complex encryption algorithms are used or when header of the packets is modified in the application. In other instances, these delays are negligible [11] compared to the transmission delay.

Queuing delay - Queuing delay is the amount of time a packet waits for previous packets to complete transmission. This depends on the load of the network. If the load is not high, the queuing delay will be less.

2.3.2 Comparison of delays

In this sub-section, we compare the delays described above to determine the significance of the different delay components. Comparison is done via calculations, using example values listed bellow.

Packet size =600 bytes
Data rate = 1Mbps
Distance between sender and receiver =150 m
Propagation speed (Speed of light) = $3X10^8$m/s

Using the above values, the transmission delay (D_t) and the propagation delay (D_p) can be calculated.

D_t=$600X8/10^6$s=$4.8ms$
D_p=$150/3X10^8$s=$0.5\mu s$

From the above results, we can observe that the propagation delay is small compared to the transmission delay. Processing delay and queuing delays are not significant in a simple application, like the one we use. Hence, we can conclude that the transmission delay is the prominent component of the total delay. In addition to these network delays, some of the protocol delays in WD (as explained in Section 2.2) also contribute to the total delay in a WD based VANET.

2.3.3 Transmission delay in single packet transfer

This sub-section discuss the delay components that contribute to the transmission delay of a single packet transfer between two nodes. In IEEE802.11, medium access is performed using carrier sense multiple access with collision avoidance (CSMA/CA) with distributed coordination function (DCF). Some inter-frame space (IFS) intervals are used between the transmission of frames. Short IFS (SIFS) is the first type of interval, after which an ACK or data frame will be sent. DCF-IFS (DIFS) is the other IFS interval used with the DCF protocol. Two types of medium sharing methods, namely basic access and RTS/CTS access are used [12, 13].

In the basic mechanism, the sender node needs to wait for a DIFS interval before it sends the packet, and after SIFS interval, ACK will be sent back. After that, the node will wait for another DIFS interval to decide whether the packet is received successfully or not. Time interval DATA is used to indicate the time taken to transfer the data packet. Therefore, the total delay in the basic mechanism (D_{T_B}) will be as follows:

D_{TB}=DIFS+DATA+SIFS+ACK+DIFS.

In RTS/CTS mechanism, the sender node first sends the RTS(RTS) frame after waiting for a DIFS interval. Then, the receiver node will send the CTS (clear to send) frame after an SIFS interval. After that, the sender node will send the DATA FRAME after an SIFS interval. Finally, the receiver sends the ACK after waiting for another SIFS interval. Hence, the total delay on RTS/CTS mechanism D_{T_RC} will be as follows:

$D_{T_{RC}}$= RTS+SIFS+CTS+SIFS+DATA+SIFS+ACK+DIFS.

However, in DSRC, the packet will be broadcasted after a DIFS interval. Hence, single transmission delay D_{DSRC} will be given by

D_{DSRC}=DIFS+DATA.

2.3.4 Comparison of transmission delays in different type of packet transferring mechanisms

In the original WD implementation, packets are transferred by forming the group, and messages are always passed through the GO. Let's consider a situation where every node in the network needs to send a basic safety message(BSM) to every others. Assume N nodes in a network. In this network, transmission delay can be divided into three parts. Here the time taken for single packet transfer is noted as D

Part 1 : The time taken by every node to send their BSM to the GO

$$T_{clienttransmison}=(N-1)\times D$$

Part 2 : Time taken for GO to re-transmit the BSM to other nodes

$$T_{retransmison}=(N-2)\times(N-1)\times D$$

Part 3 : Time taken for GO to transfer its own BSM to other nodes

$$T_{GOtransmison}=(N-1)\times D$$

DSRC uses a broadcast mechanism to transfer BSM. If one node need to send a packet to other node, it will just broadcast it. All the nodes by the reach of the sender, will receive the broadcasted BSM. Consider a network with N number of nodes.
Time taken to transfer packets to all the nodes, T_{DSRC} is given as

$$T_{DSRC}=(N)\times D_{DSRC}$$

here, D_{DSRC} is the transmission delay of a single packet transfer in DSRC.

2.4 Simulators

A VANET simulation requires a network setup as well as a traffic configuration. In this study, network simulator NS3 is used along with traffic simulator SUMO, for simulating these to components. The network simulator NS3 and its suitability in VANET simulations, is described in the next sub-section.

2.4.1 NS3

NS3 is a widely used network simulator. The NS3 project has a solid simulation core that is well documented and easy to use and debug. Several helper classes are available in NS3 that can be used for VANET simulations [7]. NS3 follows event-based simulation, where events are initiated through calling functions, scheduled to execute at a prescribed simulation time. This is done by callback functions [14]. NS3 allows both IP and non-IP based networks, and supports a real-time scheduler for interacting with real systems.

Every simulation in NS3 is an independent program. For wireless simulations, some common steps are followed. Firstly, nodes will be initialized. Then, wireless channels and its properties will be configured using respective methods. After that, the MAC and PHY models will be installed in each node. Finally, application will be configured to send/receive data and to monitor the performance measures. Flow diagram provided in Fig. 2.2 explains the architecture of a NS3 simulation [2]. It shows, how the net devices (WD/DSRC devices) are configured with the protocol and application, and installed into nodes. Those nodes are then connected through a realistic channel.

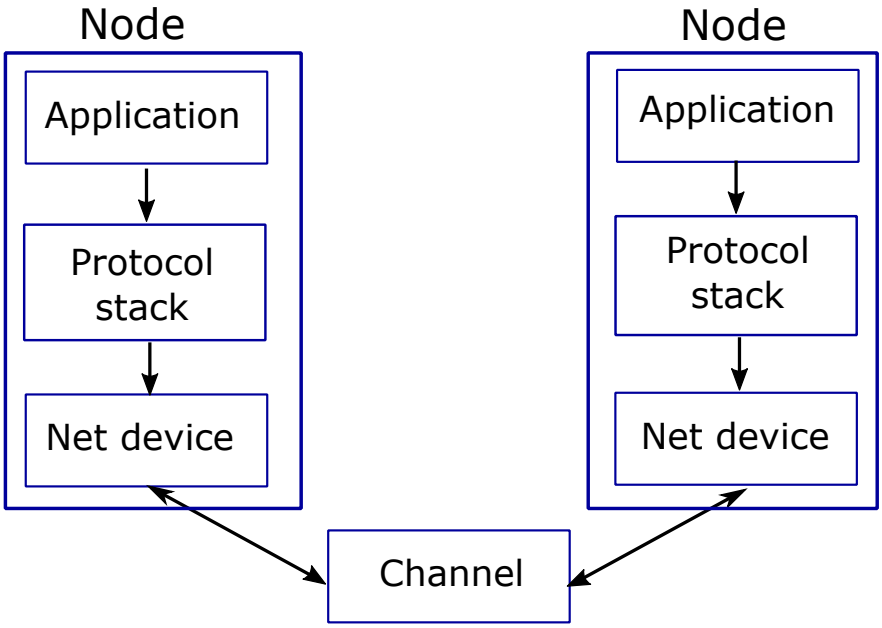

Figure 2.2: Architecture of NS3 simulation.

For VANET simulations, several NS3 modules are needed. Table 2.3 shows some of the required modules and available solutions in NS3.

Table 2.3: Availability of related NS3 modules

Requirement	Available NS3 modules
Node creation	Node container
Device configuration	Wi-Fi 802.11,DSRC (IEEE 802.11p)
Propagation	Two-ray ground path loss model, Nakagami fading, Constant speed delay model
Packet transfer	UDP packet sink, UDP client server, Onoff application
Mobility	NS2 mobility helper, Constant velocity mobility model
Data collection	Flow monitor
Visualization	NetAnim animator, PyViz visualizer
Tracing	Pcap packet tracing, ASCII tracing
Routing	AODV, OLSR, Static
Addressing	IPV4 address helper

NS3 also includes some inbuilt mobility models, that can support basic mobility patterns. However, a complex mobility model requires a dedicated traffic simulator. In this study, we use SUMO traffic simulator, to model large-scale networks. An introduction to SUMO is provided in the next sub-section.

2.4.2 SUMO

SUMO is a widely used open source traffic simulator, which is used for microscopic road network simulations [14]. Following features are available in SUMO:

- Traffic simulation with different road and vehicle configuration.
- Exportable files to use with a network simulator.
- High interoperability (use XML files only).
- High portability.

Scenarios are created by using separate XML files. Three files are needed for a SUMO simulation. Files with nod.xml and edg.xml extensions contain the road network information such as nodes and links, respectively. As shown in Fig. 2.3, using netconvert tool, nod.xml and edg.xml files are converted into net.xml file, which will contain overall detail about the road network. Files with rou.xml extension is created from flow.xml file, using a tool called duarouter, and contains the traffic demand and route information [7]. Ultimately, the net.xml and the rou.xml files will be compiled into sumo.cfg file, which can be used to visualized the road network in SUMO. Using some additional tools, SUMO files can be converted into NS3 supported mobility trace files. Fig. 2.4 explains, how SUMO

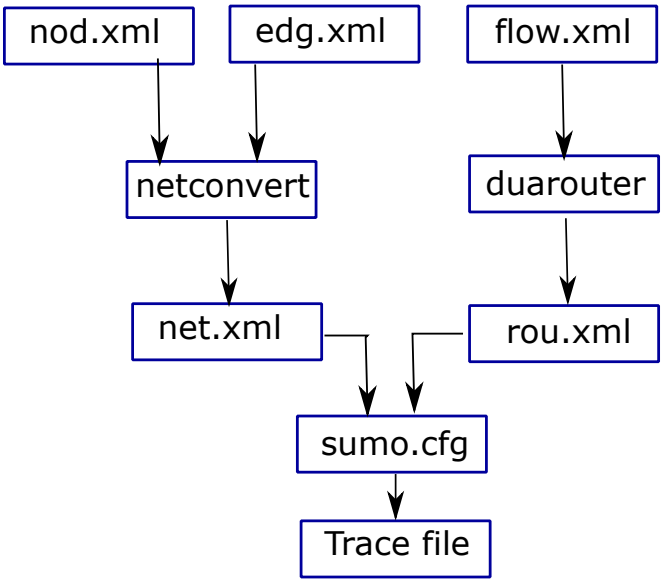

Figure 2.3: Creation of a mobility model using SUMO.

trace files(sumo.cfg) are converted into NS3 supported files using a tool called "traceExporter.py".

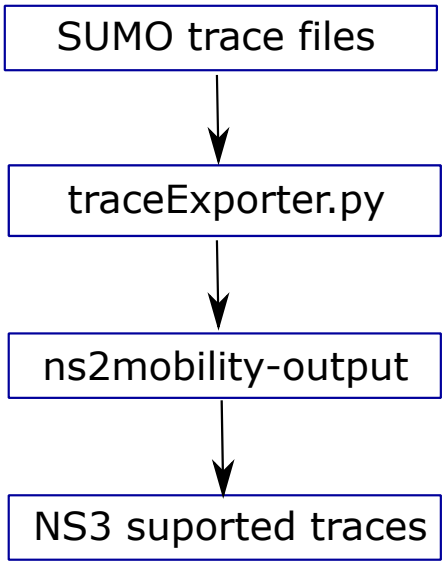

Figure 2.4: Generating NS3 supported trace files from SUMO traces.

2.5 Related works

WD is a relatively new technology and not designed with the idea of making large-scale networks. Hence, we find only a small amount of works related to VANETs based on WD. The work in [15] introduces WD as an alternative to DSRC. A the-

oretical comparison between DSRC and WD focusing on the transmission delay is provided as the main contribution. They have identified the re-transmission through GO as the main component of transmission delay, and hence, proposed a GO broadcast based method to reduce the transmission delay. Delays related to WD group formation are not included in this work. Additionally, pros and cons of using WD instead of DSRC are analyzed. The study in [16], also proposes WD as an alternative communication strategy for VANETs. They have considered the protocol delay components of WD, which is not considered in [15], and have provided an analysis on WD group formation delays. Also, they have addressed the issue of group failure because of the GO leaving the group. Algorithms are provided for selecting a backup group owner and exchanging the role between GO and backup group owner. They have demonstrated an analytical model to show the performance of their group formation algorithm.

In [17], another comparison is presented between DSRC and WD, concentrating on the MAC layer, and analyzing the average throughput and collision probabilities. They have provided a mathematical analysis by implementing a non-uniformly distributed back-off timer based on the binomial distribution for the contention window. Their goal is to reduce the collision probability and increase the throughput in a WD network.

There are few other papers which are not based on WD utilized VANETs, but directly related to our work. Out of them, [10] presents a performance evaluation of WD for device to device communication with the help of an opensource WD implementation. Firstly, they have provided a technical overview of WD by explaining the group formation and architecture. Secondly, three types of group formations are analyzed with the delays associated with every step. Finally, they have presented a study on security and power management.

Furthermore, in [18], a VANET based on smartphones is presented. In this work, the network is created by toggling Wi-Fi between hot-spot/client functionality. This solution can be implemented without rooting the smartphone and enable auto connection of devices without user intervention. They have provided an analysis on optimum time for being in each function (hot-spot/client). Experimental results are also provided with improved performance in vehicular data collection.

The work in [19] describes a smartphone based vehicular communication

system named SPIVC. With the help of a centralized server, they have used Wi-Fi and a cellular network for building SPIVC, and have done some field test for GPS accuracy and communication delays of their system. By using the results from the conducted experiments, they have determined the optimal configurations for SPIVC.

The work in [20], is a comparison of performance in a VANET, between Wi-Fi and DSRC. First, they have introduced the current standards and requirement for vehicular communication. Then, they have done experiments using off-the-self DSRC and Wi-Fi radios, for observing the effects of message size, message frequency, weather, mobility, and speed in the communication. Results from these different conditions are provided based on real world experiments.

In [21], a comparison study is done between Wi-Fi standards IEEE802.11a and IEEE802.11p. Results of some real world experiments for comparing the performance of VANETs based on IEEE802.11a and IEEE802.11p, are provided in this work. Contact time with the roadside infrastructure and throughput with different data rates, are measured as the main performance measures in this study.

Finally, the work described in [22] analyzes on incorporating the RF front end of DSRC (IEEE 802.11p) into smartphone hardware. They have modified existing Wi-Fi radios in smartphones, with the support of IEEE802.11a, to support IEEE 802.11p. They also address the power and area management issues of combining smartphones and the DSRC technology.

Chapter 3

System model

The primary goal of this study is to compare the performances of both WD and DSRC, in a VANET scenario. We present a system model for this comparison study by considering three main aspects. Firstly, we design a reliable channel model between two nodes in a VANET, which is explained in Section 3.1. Secondly, the topology of the system and the mobility of the nodes are configured, and these are explained in Section 3.2. Finally, our network model is explained in Section 3.3.

3.1 Channel model

The channel between two nodes needs to be modeled carefully so that the reliability can be improved. Two main types of attenuation can be seen in the wireless channel in a vehicular environment. The first type is the path loss, and it depends on the distance between the sender and the receiver. The path loss modeling is described in the next sub-section. In addition to the path loss, there is fading in the vehicular environment due to multi-path propagation. Therefore, a probabilistic model needs to be incorporated to model the fading, which is described in Sub-section 3.1.2.

3.1.1 Path loss model

Path loss models are used to calculate the attenuation in signal power while traveling in the propagation medium. Several models are there to evaluate the path loss in a wireless medium. Free space path loss is modeled for line of sight

communication as

$$P_r \propto 1/d, \tag{3.1}$$

where P_r is the received signal power and d is the distance between the sender and the receiver. The Fris propagation model [2] extends the free space model by integrating the antenna gain. This also assumes a clear line of sight path, and ignores scattering by atmospheric particles. In Fris model, the received power P_r is given as

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L}, \tag{3.2}$$

where P_t is the transmitted power, G_t and G_r are the transmit and receive antenna gains, λ is the wavelength of the signal and L is the system loss [2].

Two-ray ground model incorporates the heights of the antennas into the model [23]. Also, the wavelength is not a factor in the Two-ray ground model equation, where received power P_r is given by

$$P_r = \frac{P_t G_t G_r {h_t}^2 {h_r}^2}{d^4 L}, \tag{3.3}$$

where L is the system loss which is a scalar, G_t and G_r are the scalar gains at the transmit antenna and the receive antenna respectively, and h_t and h_r are the antenna heights at the transmitter and receiver in meters respectively. The Two-ray ground model assumes a line of sight path as well as a reflected path to calculate the received power. The implementation of the Two-ray ground model in NS3 is provided in [2]. The Two-ray ground model won't give proper measurements for short distances due to the oscillations caused by the constructive and destructive combination of the two rays [24]. Hence, the work in [2] introduced a crossover distance, which estimates the point where the signal will reflect off the ground. Until the crossover distance, Fris model is used instead of Two-ray ground model. The crossover distance d_{cross} is calculated as

$$d_{cross} = \frac{4\pi h_t h_r}{\lambda}, \tag{3.4}$$

Fig 3.1 explains various parameters of the channel model used in our study.

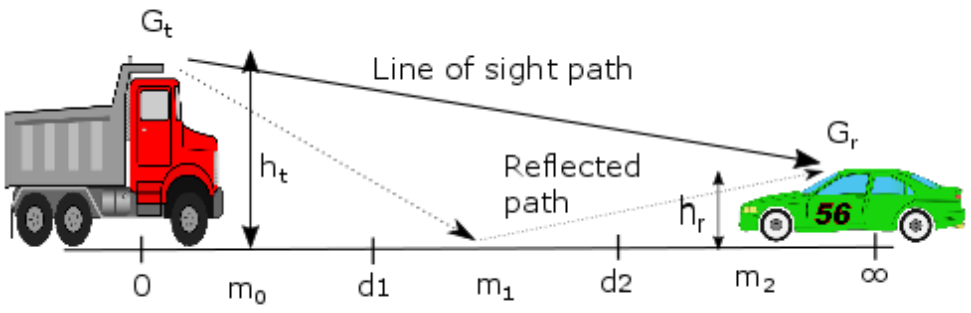

Figure 3.1: Parameters of channel model.

3.1.2 Fading model

A deterministic propagation model alone is not enough to provide a realistic channel. Randomness is mainly caused due to high mobility and multitude reflection [25]. Rayleigh and Rice distributions are also used to model the envelop of the fading channel. However, several studies indicate that Nagakami model is more suitable for vehicular environment [25–30].
According to the Nakagami fading model, the probability density function of the received signal amplitude is given by

$$f(x:m,\omega) = 2\frac{m^m}{\Gamma(m)\omega^m}x^{2m-1}\exp^{-\frac{m}{\omega}x^2}, \tag{3.5}$$

where m is the fading depth parameter, ω is the average power and $\Gamma(\cdot)$ is the Gamma function. In our setup, we consider three distant fields separated at distances d_1 and d_2, and hence, we use three fading depth parameters m_0, m_1 and m_2 (please refer to Fig. 3.1) for those three distance fields.

3.1.3 Limitation of channel models

Our channel model is configured with the proper selection of a path loss model and fading model. However, there are still some limitations in the model. Some of the recent studies suggest that obstacle modeling will give more accurate results [31–33] in a VANET environment. They have suggested a model, which will give radio-interfering conditions that are caused by the obstacle within VANET topology.

Also, in most of the scenarios, vehicles are moving at a high speed, and the velocity of the vehicles also will affect the signal propagation. It should be noted that the considered propagation models are not taking velocity as a parameter. In these cases, range is the main factor considered for calculating the received power, and velocity will be a determining factor only when readings are taken with time.

Despite these limitations, we use the model explained in Section 3.1 because of two reasons. Firstly, we feel our selection of propagation models is suitable for this study because our main concern is comparing the performance of DSRC and WD. Hence, using a similar propagation model with adequate accuracy will serve the purpose. Secondly, even though some studies implements an obstacle model for NS3, those implementations are not included in official NS3 release used in this study. Also, any propagation models that considering velocity as a factor are not implemented in NS3. Due to time constraints, we decided not to concentrate on implementing new modules in NS3, and focused on the comparison study.

3.2 Topological Model

Topology and mobility also need to be configured properly in a VANET simulation. Road networks need to be modeled, together with vehicles, and their mobility pattern. In our simulation study, two topological models are used. The first model is used for small-scale simulation, and it contains two vehicles and one RSU. The second model is a large-scale model with clusters of vehicles. Both models are designed for a junction scenario. Next, we will describe our small-scale model.

3.2.1 Small-scale model

In the first phase of the simulations, we intended to do a comparison of performance between DSRC and WD with single-hop communication. The topology model for the Phase 1 simulations, is shown in the Fig. 3.2. We can notice that, in the small-scale model, two nodes (vehicles) are moving away from the junction with velocities v_1, v_2 and sending packets to the RSU at the junction. Constant velocity model is used here to model the mobility of the vehicles.

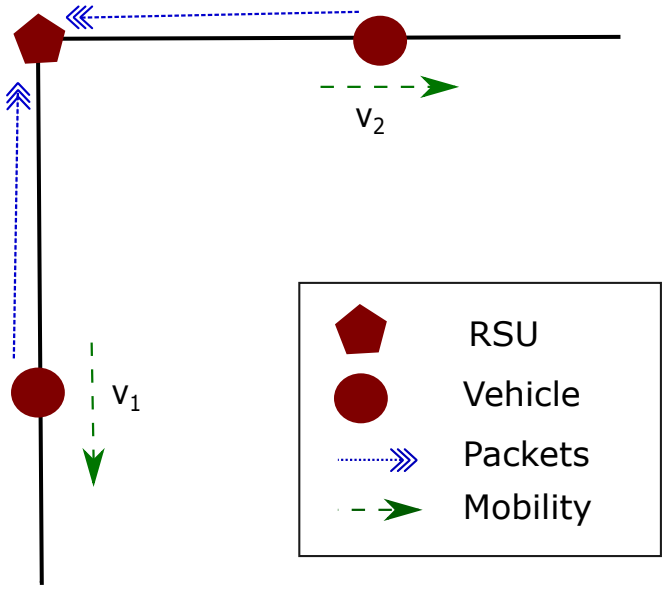

Figure 3.2: Phase 1: Small-scale model.

3.2.2 Large-scale model

The large-scale model is designed to compare the performance in multi-hop communication. In order to group the vehicles, and easily differentiate their relative positions, vehicles are divided into clusters. Vehicles are divided into C clusters, and let γ_i, where $i \in 1, \ldots, C$, be the number of vehicles in cluster i. All vehicles in cluster i move at the same velocity v_i. A junction similar to Phase 1 is considered. In this phase, some of the clusters are moving towards the junction, while some clusters are moving away from the junction. This means, some clusters are moving close to each other, and some are moving away from others. An example for Phase 2 is provided in Fig. 3.3, where $C = 4$, $\gamma_1 = \gamma_3 = 13$ and $\gamma_2 = \gamma_4 = 12$.

3.3 Network

As the final step of the system modeling, network part need to be designed. In Phase 1, the packet transfer is from the moving nodes (vehicles) to the static node (RSU) in the junction. UDP packets of size s_{p1} each are sent at a rate of r_{p1} directly to the RSU. These packets are received by the RSU, and the performance measures are calculated.

In the large scale model, a vehicle can either communicate with a vehicle inside its cluster or a different one. Such a communication process (data transfer)

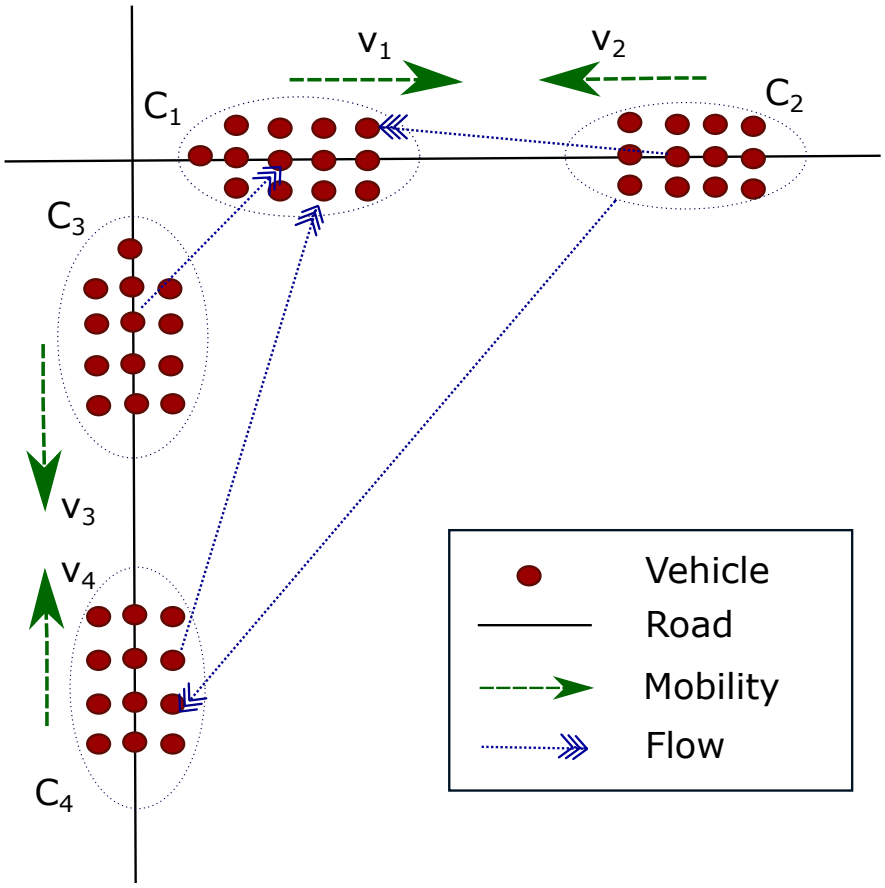

Figure 3.3: Phase 2 : large-scale model.

is called a “flow" in this thesis. Let F be the number of flows in the network. In each flow, λ_{p2} packets of size s_{p2} each are transmitted at a rate r_{p2}. Unlike in Phase 1, packets may be sent through multiple hops, in Phase 2. Intermediate vehicles are functioning as re-transmitters for this purpose. The performance measures are calculated for these flows in the simulations. Since the packet transfers in Phase 2 involve multi-hop, it requires a routing protocol, which is explained in the next sub-section.

3.3.1 Routing protocols

A routing protocol is needed for the efficient transfer of packets in a VANET, where the transmission requires multiple hops, and nodes are highly mobile. There are two main categories of routing protocols used in VANET studies. The first category is an active protocol that dynamically finds the routing path every time. The second category is proactive routing scheme, which periodically calculates the routing path, and maintains the routing table. In this study, we

select two widely used routing protocols AODV [4] and OLSR [5] from each category and compare their performance in VANETs. These two routing protocols are studied in various researches for their use in VANETs, and also some works compare their performance in VANETs [34–37].

3.3.1.1 Ad-hoc On-Demand Distance Vector (AODV)

The AODV protocol initiates a route discovery process whenever a packet needs to sent from a source node to the destination. The source node starts the process by broadcasting a route request (RREQ) packet. After that, the neighbor nodes also forward the packet to the next node until the route is found to the destination. If the maximum number of hops are reached, nodes will stop forwarding the packet. Ultimately, when an intermediate node is aware of the route to the destination, it will send a Route Reply (RREP) packet back to the source node. By this means, the source node will get to know the route to the destination, and will send the packet [37].

As explained above, the route is calculated actively in the AODV protocol, and it has its pros and cons. Using an active routing method will be beneficial for ad-hoc networks with mobility. Since the global topology of the network will change rapidly, a dynamic protocol will incorporate those changes and perform well. However, being dynamics, and calculating route every time, increases the latency in routing, and eventually, affects the throughput.

3.3.1.2 Optimized Link State Routing Protocol (OLSR)

OLSR is a proactive protocol, in which each node periodically constructs its set of neighbors with hop distance up to two [37]. An algorithm named multi-point relay (MPR) will find the active relay to cover all the two-hop neighborhoods. This algorithm also minimizes the number of active relays since, only the selected nodes (by MPR from sender) forward the packets. OLSR send link state information periodically to construct and maintain the routing tables. After the saturation, each node will have the active path to each other node in the network. Hence OLSR is a proactive protocol, it will construct the routing table periodically. This may be suitable for a network which has slow changes in topology, such as urban traffic scenarios.

Chapter 4

Simulation Setup

Our simulation setup is aimed at designing a VANET, based on the system model described in the previous chapter to compare the performances of both WD and DSRC. Several components are essential in a simulator to design a realistic VANET. A list of simulation modules is provided in Table 2.3. NS3 has a DSRC implementation with the support of IEEE802.11p protocol. However, WD has not been implemented in NS3 at the time of the project. In our simulations, we use the existing Wi-Fi implementation in NS3 to develop a model that gives similar performance to a WD based network. Since our primary focus is on the transmission delays of both networks, we have not considered some of the protocol delays in WD in the initial stage. We will first discuss several important steps of the simulation, which are summarized in Fig. 4.1 .

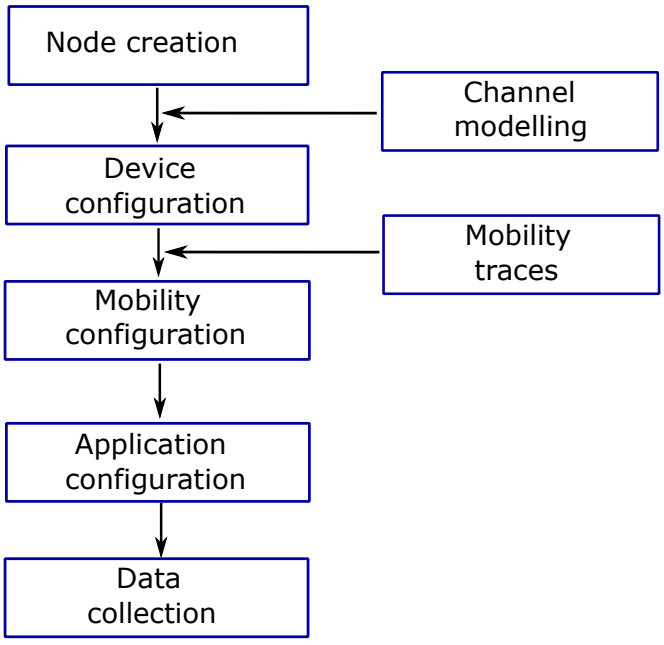

Figure 4.1: Steps of the simulation.

4.1 Node creation

The first step of the simulation is creating the nodes using the node container class, which is done as follows:

```
NodeContainer c;
```

If different types of nodes are required in a simulation, several node containers need to be defined, but in this simulation, we use all the nodes with same properties(similar device). After creating the device containers, nodes need to be created for each container, using

```
c.Create (numNodes);
```

The required number of nodes depends on the topology of the simulation, and it needs to be aligned with the mobility model. Until this point of simulation, nodes have no properties. Hence, these steps are common for both WD and DSRC simulations.

4.2 Channel modeling

Modeling the channel between two nodes is the next step in the simulation. As explained in Chapter 3, Two-ray ground model and Nakagami model are used to model the channel. Also, constant speed delay model is used to configure the propagation delay. NS3 has the implementation of these two propagation loss models and the delay model. We configure the parameters using respective helper classes. As the first step, channel is defined by using the YANS Wi-Fi PHY module as follows:

```
YansWifiChannelHelper wifiChannel;
```

After defining the channel object, delay and loss models are added to the channel using respective functions. Constant speed delay model is added to the channel using SetPropagationDelay function, as follows:

```
wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
```

Two-ray Ground path loss model is introduced and configured with its parameter as shown bellow.

```
wifiChannel.AddPropagationLoss ("ns3::TwoRayGroundPropagationLossModel",
        "Frequency",DoubleValue(freq),
        "HeightAboveZ",DoubleValue(height),
        "SystemLoss",DoubleValue(sysloss));
```

Frequency is set according to the protocol (5GHz for WD and 5.9GHz for DSRC) for the "Frequency" parameter. The height of the antenna is set as "HeightAboveZ" parameter in this function. System loss is set using the parameter "SystemLoss". Nagakami loss model is added in addition to the path loss model for giving the fading effect to the channel as follows:

```
wifiChannel.AddPropagationLoss("ns3::NakagamiPropagationLossModel",
    "Distance1",DoubleValue(d1),
    "Distance2",DoubleValue(d2),
    "m0",DoubleValue(m0),
    "m1",DoubleValue(m1),
    "m2",DoubleValue(m2));
```

Here, "Distance1","Distance2" are the distances, which divides the distance fields and "m0","m1","m2" are the fading depth parameters of the distance fields as explained in Fig. 3.1. Channel is configured in a similar manner for both technologies in order to provide the same environment for both technologies.

4.3 Device configuration

In this step, we configure the nodes with different net devices and set up their PHY, MAC layers using the helper classes. Configurations in this step are different for both technologies. We will start by describing the DSRC configuration. Initially, a IEEE802.11p device is defined using the Wifi80211pHelper module and PHY layer object is defined by using YansWifiPhyHelper module as follows:

```
Wifi80211pHelper dsrc = Wifi80211pHelper::Default();
YansWifiPhyHelper dsrcPhy = YansWifiPhyHelper::Default ();
```

After defining the PHY object, the physical layer configuration is added to the object using the set of following functions.

```
dsrcPhy.Set("TxPowerEnd",DoubleValue(power));
dsrcPhy.Set("TxPowerStart",DoubleValue(power));
dsrcPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
dsrcPhy.SetChannel (wifiChannel.Create ());
```

Here, the TxPowerEnd, TxPowerStart parameters are the maximum and minimum available powers, respectively. We set the same value for both in-order to make a constant transmission power. The last line of the above snippet shows how the channel properties are assigned to the PHY object. After configuring the PHY layer, the MAC layer object is defined using NqosWaveMacHelper module of the NS3 as follows:

```
NqosWaveMacHelper dsrcMac = NqosWaveMacHelper::Default ();
```

The "dsrc" object is configured with the device parameters as shown bellow.

```
dsrc.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode",StringValue (phyMode));
```

Using the SetRemoteStationManager, the node is configured with the Wi-Fi manager, the data mode, and control mode. Here, the ""DataMode" is the data rate of the device used.

Finally, DSRC net devices are created using NetDeviceContainer object by installing the PHY and MAC objects to the "dsrc" object.

```
NetDeviceContainer devices = dsrc.Install (dsrcPhy, dsrcMac, c);
```

The configuration of WD devices also follows the same procedure, except it uses different PHY, MAC and device modules. Following part of the code explains how these modules are defined for WD. Also, sample code for WD simulation is provided in Appendix A.1.

```
WifiHelper wifi;
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
```

4.4 Mobility

In this section, we are going to explain about configuring mobility to the nodes in the simulation. Some basic mobility models are implemented in NS3 such as

- ConstantPosition
- ConstantVelocity

- ConstantAcceleration
- RandomDirection2D
- RandomWaypoint
- RandomWalk2D

However, mobility in VANET simulation requires some complex and customized requirements, which cannot be done with the available basic mobility models in NS3. Hence, we use external means of introducing mobility. In Chapter 3, the steps are described on how a NS3 supported mobility trace file can be obtained using traffic simulator SUMO. Following code is a example of NS3 supported mobility trace file(.tcl).

```
$node_(0) set X_ 0
$node_(0) set Y_ 0
$node_(0) set Z_ 0
$node_(1) set X_ 5
$node_(1) set Y_ 0
$node_(1) set Z_ 0
$node_(2) set X_ 0
$node_(2) set Y_ 5
$node_(2) set Z_ 0
$ns_ at 1 "$node_(1) setdest 5 0 5"
$ns_ at 1 "$node_(2) setdest 0 5 5"
$ns_ at 2 "$node_(1) setdest 10 0 5"
$ns_ at 2 "$node_(2) setdest 0 10 5"
$ns_ at 3 "$node_(1) setdest 15 0 5"
$ns_ at 3 "$node_(2) setdest 0 15 5"
```

The initial positions of the nodes are indicated at the beginning. Then, periodically, the positions and velocities of each node are defined. These traces can be used in the simulation with the help of Ns2MobilityHelper module in the following way.

```
Ns2MobilityHelper ns2 = Ns2MobilityHelper ("scratch/ranTW.tcl");
ns2.Install ();
```

Above steps install the mobility model to each node in the simulation.

4.5 Application configuration

In this section, we discuss the application modules which need to be installed to transfer packets and collect data. In the Phase 1 simulations (small-scale model), two moving nodes are sending packets to the static node at the junction, at a constant data rate. An NS3 application called "OnOff" is used for this purpose as follows:

```
OnOffHelper onoff ("ns3::PacketSocketFactory", Address (socket));
onoff.SetConstantRate (DataRate (dataRate));
onoff.SetAttribute ("PacketSize", UintegerValue (packetSize));

ApplicationContainer apps = onoff.Install (c.Get (Node1));
apps.Start (Seconds (startTime));
apps.Stop (Seconds (endTime));
```

As shown above, the OnOffHelper object is configured with parameters such as data rate and packet size. This will send packets from the sender node to the receiver node at a constant data rate. After configuring the application, it is installed on all the nodes. The application is configured to start and stop at the required time, using start and stop functions.

In Phase 2 simulations, we use an application similar to Phase 1. However, instead of using "OnOff" application, we design our sender application. Sample code for this application is provided in Appendix A.2. This application will send a packet soon after the previous packet sent. Following lines show how the receiver node (hereafter it will be referred as sink node) is configured, and sink app is initiated on it.

```
Address sinkAddress1 (InetSocketAddress (ifcont.GetAddress (to1),
            sinkPort));
PacketSinkHelper packetSinkHelper1 ("ns3::UdpSocketFactory",
            InetSocketAddress(Ipv4Address::GetAny (), sinkPort));
ApplicationContainer sinkApps1 = packetSinkHelper1.Install (c.Get (to1));
sinkApps1.Start (Seconds (startTime));
sinkApps1.Stop (Seconds (endTime));
```

Firstly, the sink address is obtained from the sink node. Secondly, the sink helper object is initialized using the sink port. Then, the app is installed in the receiver node. Finally, the sink app is started and stopped according to the required times. After sink node configuration, the sender node is configured as follows:

```
Ptr<Socket> ns3UdpSocket1 = Socket::CreateSocket (c.Get (from1),
            UdpSocketFactory::GetTypeId ());

Ptr<MyApp> app1 = CreateObject<MyApp> ();
app1->Setup (ns3UdpSocket1, sinkAddress1, packetSize, numPackets,
            DataRate(datarate));
c.Get (from1)->AddApplication (app1);
app1->SetStartTime (Seconds (startTime));
app1->SetStopTime (Seconds (endTime));
```

The socket for sending the packet is created with the address of the sender node in the first two steps of the above snippet. An object for the sending application is created in the remaining steps above. This object is configured with the addresses, the size of the packet, number of packet and data rate. After these steps, the app is started and stopped according to the required simulation time. The application will be sending required number of packets in specific size with specific data rate.

In the large-scale simulations, the nodes are moving in clusters, and data transfer is via a multi-hop network. Hence, we configure two routing protocols AODV and OLSR. Following example describes the configuration of OLSR routing protocol.

```
OlsrHelper olsr;
AodvHelper aodv;
Ipv4ListRoutingHelper list;
list.Add(olsr, 10);

InternetStackHelper internet;
internet.SetRoutingHelper(list);
internet.Install (Nodes);
```

in the first four steps of the above code, routing objects are defined and added to the routing list. In the next three lines, the list is added to the Internet stack, and the routing scheme is installed on all the nodes. It should be noted that the above example is about configuring OLSR and line 4 need to be replaced with "aodv" for configuring AODV routing. After every parameter related to nodes are configured, following lines are placed to run the simulation for a specific period.

```
Simulator::Stop (Seconds (stopTime));
Simulator::Run ();
```

Simulation is started using "waf" tool. Additional parameters can be

passed as arguments with the code. Following terminal command explains the way of running the simulation.

```
./waf --run "scratch/scriptname --parameter1=value"
```

4.6 Data collection configuration

In the above sections, we have discussed how to configure the topology and packet transfer in the VANET network. In this section data collection process is explained. In Phase 1 of the simulation, data collection is straightforward since all the packets are sent towards the RSU at the junction. These statistics are calculated using functions with the help of callbacks in NS3.

However, for the large-scale simulation, data collection is more complicated. Hence, a data collection framework called "Flow monitor" is used to organize the data. Flow Monitor module is designed to measure the performance metrics in an NS3 simulation. It uses probes in network nodes to track packets through the nodes. These probes will measure some parameters such as transmitted bytes, received bytes, end-to-end delay, transmission time, received time and jitter sum for each flow. Here, flow is categorized by packets with same source and destination. This data is exported to an XML format. From this, we can calculate metrics like packet delivery ratio, throughput, and end-to-end delay etc. The following steps show how the flow monitor is installed in the simulation.

```
FlowMonitorHelper flowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll();
```

After installing the flow monitor, the classifier is defined to identify the flows using the source and destination IP.

```
Ptr<Ipv4FlowClassifier> classifier =
  DynamicCast<Ipv4FlowClassifier(flowmon.GetClassifier ());
std::map<FlowId, FlowMonitor::FlowStats> stats =
                                        monitor->GetFlowStats ();
```

Using the classifier, we can calculate performance measures, by utilizing the exported data. Following part of the code explains how the required parameters can be calculated from the flow statistics.

```
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter =
```

```
      stats.begin (); iter != stats.end (); ++iter)
  {
    Ipv4FlowClassifier::FiveTuple t =
                  classifier->FindFlow (iter->first);
    if ((t.sourceAddress == Ipv4Address(ip1)
        && t.destinationAddress == Ipv4Address(ip2)))
      {
        NS_LOG_UNCOND(iter->first << "  "
        << t.sourceAddress << "  "
        << t.destinationAddress << "  "
        << iter->second.txPackets <<"  "
        << iter->second.rxPackets << "  "
        <<iter->second.delaySum.GetSeconds() );
      }
  }
```

Using the for loop, the code is iterating through flows, and calculating the parameters of interest. In the given example, the number of received packets and delay are calculated.

4.7 Additional modules

Above sections covered the important steps in the simulation. Additional to those steps, there are some modules that can also considered important. In this section, some of these additional modules are discussed.

Firstly, we discuss on the SeedManager module, which is used to run an independent instance of the simulation. Some part of the simulation has randomized components. Hence, running one instance of the code won't give us fair results. The SeedManager is used to run independent simulations of the same code. The following part of the code is used to configure the SeedManager.

```
ns3::SeedManager::SetSeed(Seed);
ns3::SeedManager::SetRun(Run);
```

Here, the "Seed" and "Run" parameters can be changed to get the difference instances. The simulation is repeatedly run with different "Run" values.

Secondly, we discuss some of the modules that can be used to visualize the simulation results. NetAnim module is used to configure the visualization

using AnimationInterface.

```
AnimationInterface anim ("traceAnimation.xml");
anim.EnablePacketMetadata ();
```

This will give the XML output, which can be opened with NetAnim tool for the visualization. In addition to the visualization, some of the metrics are plotted with the help of the "Gnu plot" module.

Finally, we introduce some of the tracing related modules. NS3 can generate "pcap" files, which contain the packet traces. This files can be opened using tracing tools such as Wireshark and TCP dump to analyze the packet transfers.

```
WifiPhy.EnablePcapAll ("tracefile");
```

An ASCII file can also be generated in the format of .tr as the output, and can be extracted from it, as follows:

```
AsciiTraceHelper ascii;
WifiPhy.EnableAsciiAll (ascii.CreateFileStream ("tracefile.tr"));
```

4.8 Other tools

In this section, we discuss some additional tools, which supplements the NS3 modules, especially related to data processing. Tools such as Linux shell scripts, "Sed" text editor and Python programming language are used for this purpose. Separate instances of the code are need to be run with different seeds for getting the independent set of results. This is done by using a shell script in Linux.

```
#!/bin/bash
echo "200 independent runs"
cd /directory
./waf --run "scratch/simulation_script --Run=1"
./waf --run "scratch/simulation_script --Run=2"
./waf --run "scratch/simulation_script --Run=3"
./waf --run "scratch/simulation_script --Run=4"
.
.
```

All of the outputs from the simulations are exported into text files, which

are then processed using "Sed" editor to filter the required lines. Finally, Python scripts are used to process these metrics, and calculate the statistical results.

Chapter 5

Results and Discussion

In this chapter, the results of simulations that compare the performance between WD and DSRC, are discussed. More specifically, we focus on the throughput, delay and packet loss ratio in the simulations related to the small-scale model. Then, we focus on the average end-to-end delay and the average packet receiving percentage in the simulations related to the large-scale models. Parameters are set as follows, and the values are consistent with the values used in the literature [30]. The Two-ray ground propagation model parameters are set as $h_r = h_t = 1.5$m (an estimate of the vehicle height), $L = 1$ and $G_r = G_t = 1$. Transmit power (P_t) is considered to be 40mW (16dBm) for WD and 2W (33dBm) for DSRC, based on the recommended power setting in respective technology standards. In the Nakagami model, the distance fields are set by selecting $d_1 = 50$m and $d_2 = 150$m, and the fading depth parameters are set as $m_0 = 3$, $m_1 = 1.5$, $m_2 = 1$. Prior to the primary simulations, the related modules are verified using some experimental and theoretical results.

5.1 Verification

The NS3 simulation modules are validated, before starting the performance comparison, in order to make the results more trustworthy. Some of the existing results are compared with the simulation results for this validation. The small scale model is used for the validation, and results from one hop communication are compared. However, we couldn't find many of the metrics related to WD for VANET in literature. Therefore, we start our validation with DSRC modules.

To this end, the behavior of the throughput with the distance, between two nodes is compared with existing experimental results in [21]. For this simu-

lation, λ_{p1} is set at 1000, velocities of both vehicles (v_1, v_2) are set at 54 km/h and s_{p1} is set at 1000 Bytes. Values are selected for r_{p1} in accordance with [21]. Fig. 5.1 illustrates the comparison of throughput. It can be observed that the simulation results show adequate conformity with existing experimental results. More specifically, we can observe that the patterns at which the graphs change are similar, but there are some differences in values, which can be expected when experimental results are compared with simulation results. We can see that the selection of the parameters of the fading model looks reasonable as well. However, we may be able to get a closer match by fine tuning the fading parameters.

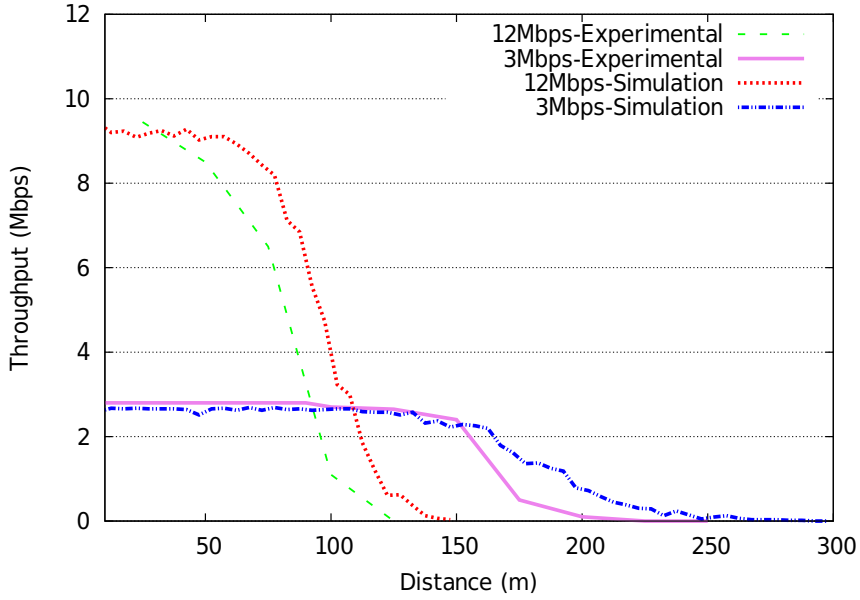

Figure 5.1: A comparison between the theoretical, experimental and simulation based throughput results.

We also intended, to validate the WD related modules, but unfortunately, a suitable data set for WD is not found in the literature. However, in our comparison of DSRC and WD, mostly, the same simulation parameters (channel properties, mobility, and applications) are used. Under our assumptions, the only key differences are with regards to transmitting power, bandwidth, and frequency (IEEE 802.11a uses 20 MHz bandwidth at 5.15 GHz while IEEE802.11p uses 10MHz at 5.9 GHz). While studying the codes and implementation documents of DSRC in NS3, we have found that the DSRC (IEEE802.11p) modules, which we considered in the simulation, are extended from the Wi-Fi (IEEE802.11) modules, by accounting for the aforementioned differences. Also, the MAC model used in the DSRC implementation is similar to the ad-hoc Wi-Fi MAC, which is used in our WD model. Because of these reasons, we can expect the WD modules

to function similarly to the DSRC modules. Therefore, although not perfect, we believe that the DSRC validation presented in Fig. 5.1 is adequate to consider the results from the WD modules to be trustworthy as well. Propagation and mobility modules have also been tested and validated by comparing with theoretical results.

5.2 Small-scale simulations

After the validation, performance measures of interest are simulated for both DSRC and WD, considering the direct packet transfer (single-hop communication), using the small scale model. Most of the parameters, that were set for the validation process are kept at the same values for these simulations as well. Firstly, we compare the packet loss ratio of both technologies. Fig. 5.2 illustrates the behavior of the packet loss ratio with the distance between the transmitter and the receiver, for both communication technologies. We can observe that the packet loss ratio increases more rapidly for WD compared to DSRC. The lower transmit power in WD can be considered as the reason for this behavior. Since the transmit power is lower for WD, the range at which it can communicate reliably is less compared to DSRC. As the two technologies operate at two different frequencies, the fading can be considered to be another reason for this behavior. However, since the frequencies are not drastically different (both in the 5GHz range), this can be considered to be a minor reason. Close observation of Fig. 5.2 also shows that distances of 50m and 150m are points of interest as the behavior of the curves change at these points. This is because of the fading model. Note that the fading depth parameters change at these two points.

Comparison of the behavior of throughput with the distance between the transmitter and the receiver is provided in Fig. 5.4. It can be seen that the throughput is constant up to a certain range, and then it starts to decline. When the distance is small, the received signal strength is above the minimum threshold for successful reception of packets. In Fig. 5.3, we can observe this, where the declining red line indicates the receive signal power and the horizontal green line indicates the threshold power for successful reception. This explains the reason for the constant throughput. As the received signal strength deteriorates with distance, the throughput starts to decrease when the distance is above a certain value. Also, we can observe that the distance at which the throughput starts to

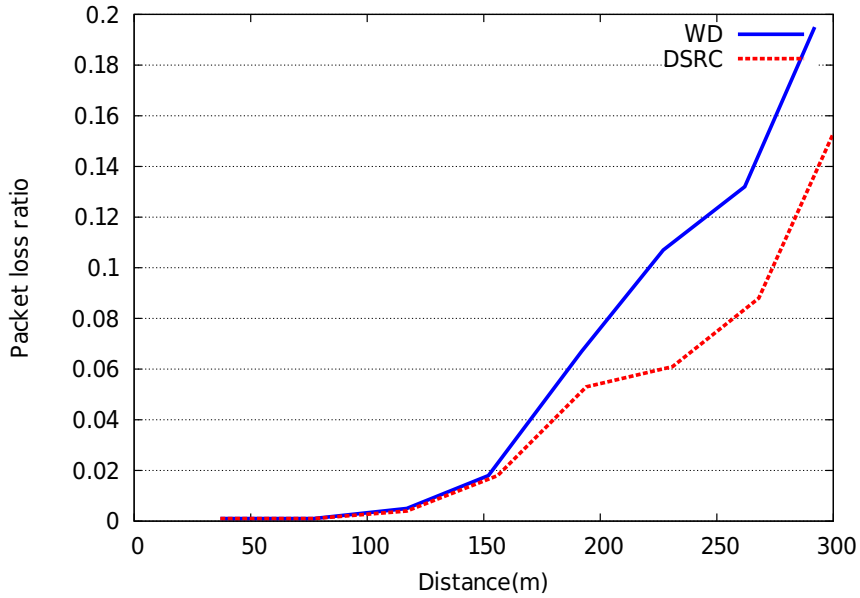

Figure 5.2: Comparison of the packet loss ratio between WD and DSRC.

deteriorate is higher for DSRC. The main reason for this is the higher transmit power used in DSRC.

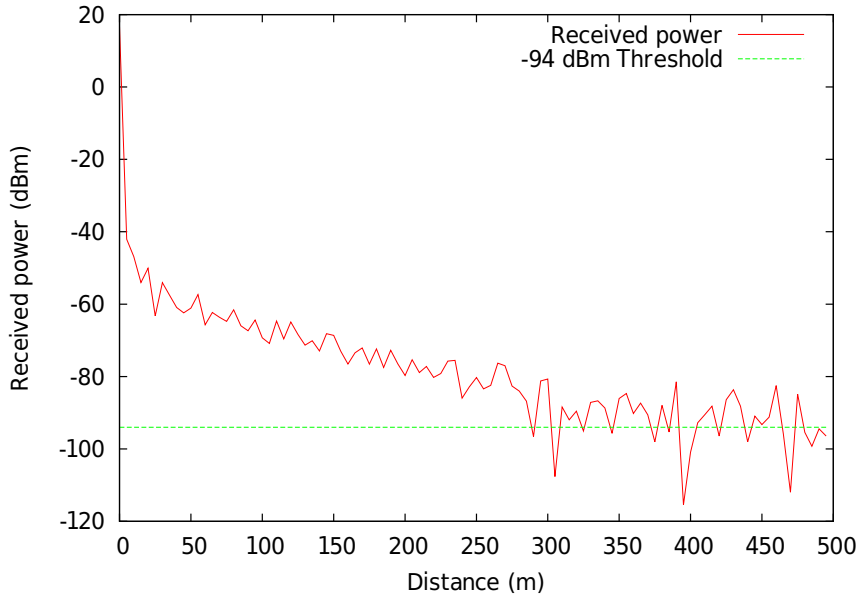

Figure 5.3: Change of received signal power with distance and threshold power for successful reception.

Another parameter of interest is the velocity at which the vehicles move. Changing the velocity will not change the behavior of our performance measures, with regards to the distance, as both the path loss model and the fading model are independent of velocity. However, the rate at which the path loss values change,

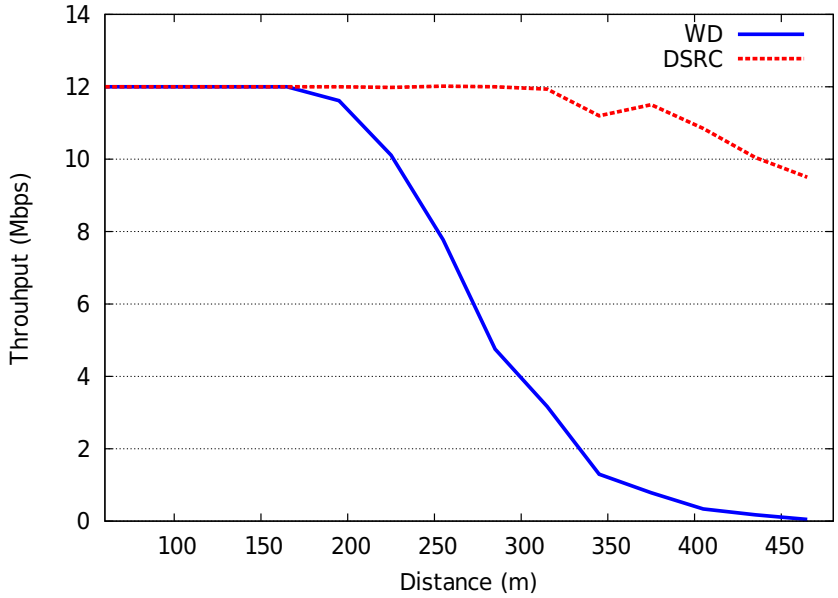

Figure 5.4: Comparison of the throughput between WD and DSRC.

depends on the velocity of the vehicles. Therefore, for different velocities, we observe how the throughout changes with time, and the results are presented in Fig. 5.5.

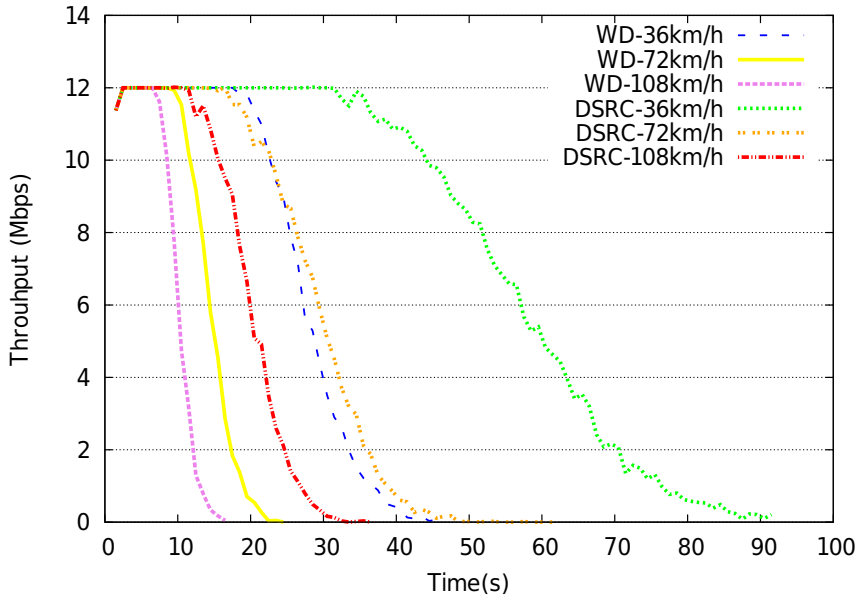

Figure 5.5: The behavior of throughput with time at different velocities.

Similar to the observations from Fig. 5.5, the throughput is constant for a certain interval, and then it starts to deteriorate. At the same velocity, the time interval with constant throughput, is longer for DSRC compared to WD.

Furthermore, for the same technology, the time interval, where the throughput is constant, reduces when the velocity is increased. This observation is consistent with the observations in Fig. 5.4 as well. Due to increasing the velocity, the vehicle will go beyond its favorable communication range much more quickly, so the throughput will start deteriorating sooner. This also means that the communication process should be handed over to another intermediate node after this time.

5.3 Large-scale simulations

The Phase 2 simulations are focused on the multi-hop communication scenario, and the large-scale topological model is used for this. Parameters in the topological and network models are set as $C = 4$, $F = 3$, $\lambda_{p2} = 1000$, $s_{p2} = 600$Bytes, $\gamma_1 = \gamma_3 = 13$, $\gamma_2 = \gamma_4 = 12$, $v_1 = v_2 = v_3 = v_4 = 54$km/h and $r_{p2} = 1$Mbps. Channel model parameters are set same as in Phase 1.

For the first set of simulations, we use a modified WD model and results are discussed next. We also investigate the delay using the current WD implementation, and these simulations are explained in Sub-section 5.3.2.

5.3.1 Conceptual Wi-Fi Direct model

The first set of simulations, is done with a conceptual WD protocol which is configured based on some assumptions. Firstly, we assume an already formed WD group, hence the protocol delays such as group formation delays are ignored in these simulations. Secondly, we assume that all the nodes can communicate with another node irrespective of the roles. Finally, we assume that broadcast is allowed in all the nodes, and making an individual connection is avoided. Some of these assumptions are based on the modification works in literature [15,16,38]. End-to-end delay and the packet receiving percentage are calculated for every flow, using the flow monitor. In order to get more generalized results, simulations are run 200 independent instances. The average end-to-end delay and the average packet receiving percentage are calculated using these results from 200 runs.

Statistical results are compared between the two communication technologies, and also between the two routing protocols. In both Fig. 5.6 and Fig. 5.7 the average end-to-end delay of WD and DSRC are illustrated for both rout-

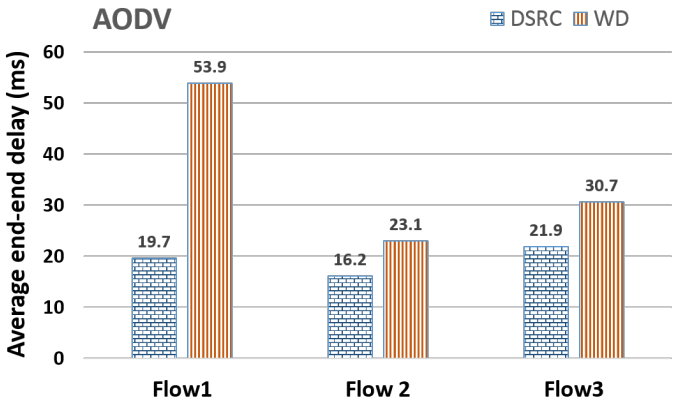

Figure 5.6: Comparison of average end-to-end delay with AODV routing.

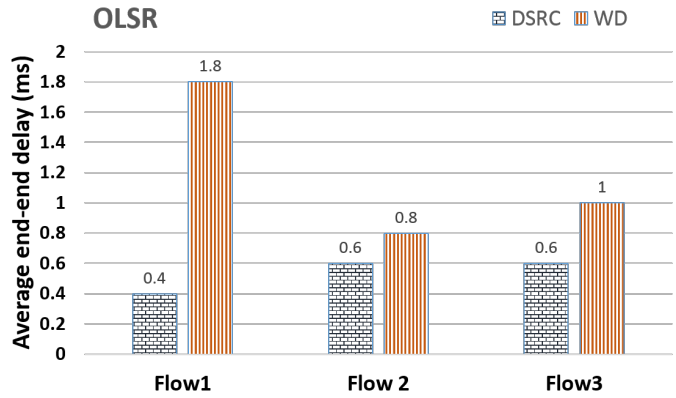

Figure 5.7: Comparison of average end-to-end delay with OLSR routing.

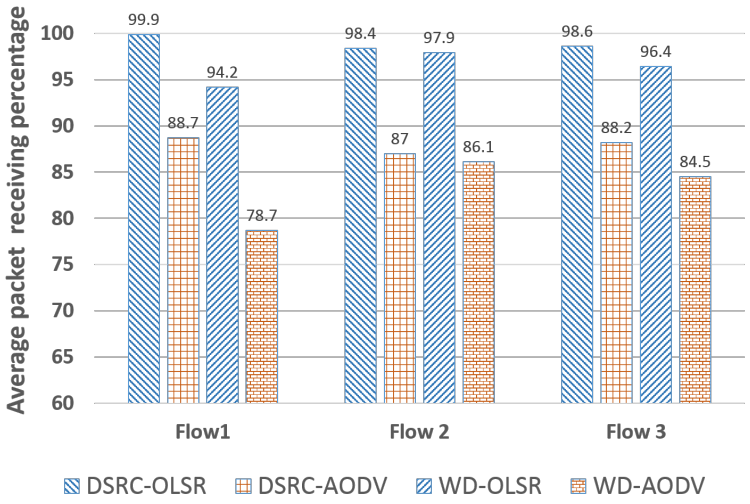

Figure 5.8: Comparison of the average packet receiving percentage.

Table 5.1: Average results of all flows.

	DSRC-AODV	WD-AODV	DSRC-OLSR	WD-OLSR
Average End-to-End Delay (ms)	28.00	58.37	0.56	0.96
Average packet receiving percentage	87.2	83.3	98.8	97.3

ing protocols in three flows. Also, Table 5.1 provides a more generalized results, where the average results of these parameters are calculated among twenty flows. We can observe that the average delay for WD is higher than DSRC in every flows in Fig. 5.6 and Fig. 5.7. The average results in Table 5.1 also indicate the same. These differences in delays can be explained as follows. When comparing the delays, the key component is the transmission delay, which is the time it takes to push the packet to the wireless link. In DSRC, the effective throughput is higher, and the packet loss is lower, as per the results generated for Phase 1. These factors make DSRC's effective transmission delay less compared to WD. Also, in terms of average delay, we can notice that OLSR performs much better compared to AODV from the simulations. Reasons are rather obvious as AODV is dynamic, and OLSR maintains a pre-calculated routing table. Fig. 5.8 illustrates the average packet receiving percentage of WD and DSRC for both OLSR and AODV and Table 5.1 shows the average results of receiving percentage for both protocols considering twenty flows. We can observe that the average packet receiving percentage is high for DSRC and OSLR, and the reasons can be explained using similar arguments to the ones used earlier.

5.3.2 Models with original Wi-Fi Direct implementation

In Sub-section 5.3.1, we have analyzed a conceptual WD model with several modifications to WD protocol. However, we also want to get an idea about a VANET with the current WD implementation. Next, we will discuss the results of the simulations on delay, using original WD implementation.

Original Wi-Fi Direct implementation - In this part, we discuss the simulations done on WD delay, with the original WD implementation. These simulations use the original WD model explained in Sub-section 2.3.4, where all the packet transfers are through the GO and GO will make individual connections with each client to send data. The comparison of average end-to-end delay, between this WD model and DSRC, is provided in Fig. 5.9. We could notice that the delay in

original implementation is higher than the conceptual implementation explained in Sub-section 5.3.1.

Original Wi-Fi Direct implementation with Group Owner broadcast

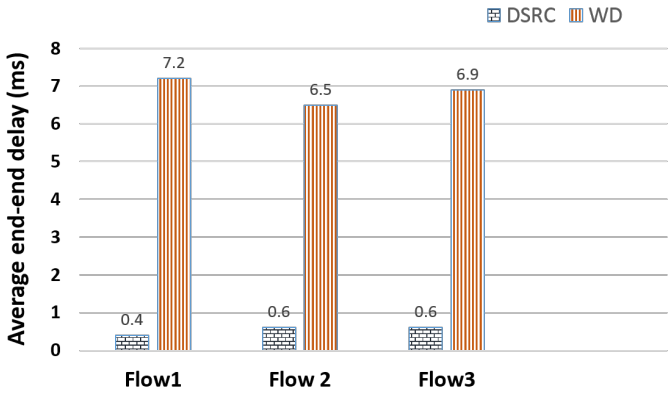

Figure 5.9: Comparison of average end-to-end delay between original WD implementation and DSRC.

mechanism - For this part of simulations, we use a mechanism which is suggested in [15],to improve the delay. According to their method, GO broadcasts the packet to clients, instead of making separate connections with every client. Also, they have assumed that the WD group is already formed, and the protocol delays are ignored. These can be implemented with some application level changes in the original WD implementation (hence without modifying the WD protocol). Fig. 5.10 compares the delays in these three types of WD based VANET implementations, from which we can observe that the delay of the broadcast mechanism is in between the original and conceptual models.

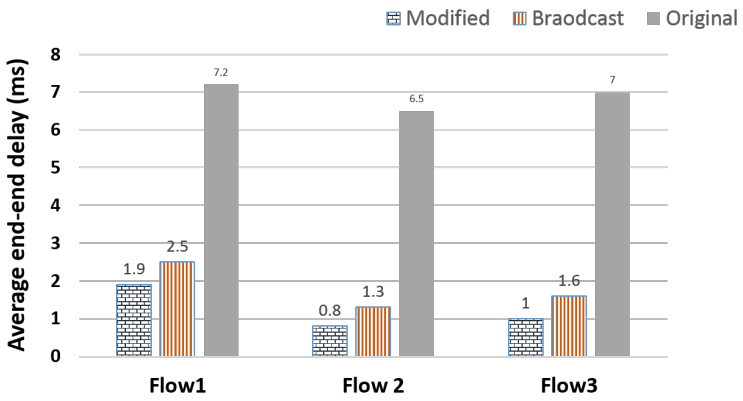

Figure 5.10: Comparison of average end-to-end delay with different WD implementations (Modified- implementation in 5.3.1, Broadcast and Original-implementations in 5.3.2)

5.4 Discussion

In this Section, we are going to analyze the results of the comparison study of performance between WD and DSRC. From the Section 5.2, we can notice that DSRC's performance in single hop communication is better, compare to WD. If we consider the packet loss ratio, the values are similar until certain range and the increase is high for WD above that range. The throughput also follows similar pattern, that is, close values for both technologies until a certain range, and increasing after that range. Results of the large-scale simulations, with conceptual WD implementation, as shown in Sub-section 5.3.1, also indicate better performance with DSRC. We can clearly observe this in the Average end-to-end delay and Average packet receiving ratio comparisons.

Better performance of DSRC, in VANET scenario, is expected. However, it is important to notice that the performance of WD is not drastically inferior. Also, we should note that the superiority of DSRC is mainly due to its higher transmit power compared to WD. The delay is the most critical parameter, and it is expected that the delay should be bellow 100ms for a safety critical application [15]. Results (Fig. 5.6, 5.7) also indicate that the average delay is below this level for both technologies in all the flows. The average packet receiving ratio is not very low for WD either, when compared to DSRC, as shown in Fig. 5.8. Even in Phase 1, the packet loss ratio (Fig. 5.2) is almost similar until 150m, and the throughput achieved with both technologies (Fig. 5.4) are very close to each other when the distance is below 200m. These results show the potential of WD as an alternative communication technology for VANETs.

From Subsection 5.3.2, we can get an idea about the delays regarding different implementations of WD (refer Fig. 5.10). We can observe that by changing the WD protocol, delay performance of WD can be enhanced.

Also, from these simulations, we can identify several performance gaps between WD and DSRC. Identifying these gaps is also very essential for future researches that focus on enhancing the WD protocol for VANETs. Some of the suggestions for enhancements are provided in Section 6.2.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Vehicular communication is essential for various ITS applications, and VANETs enable this communication. DSRC is the widely used communication technology in VANETs. However, due to the requirement of dedicated hardware device, DSRC has some barriers for a large-scale VANET implementations. In this thesis, we analyzed the feasibility of using WD as an alternative technology for DSRC, through a simulation study.

Firstly, we simulated the VANET with the help of network simulator NS3 and traffic simulator SUMO. We configured various modules related to VANET such as topology, channel, and network. Our simulation consisted of various steps namely node creation, channel modeling, device configuration, application configuration and data collection. Some additional tools were also used for data processing and visualization.

Then, NS3 modules were validated using some available results. After the validation, simulations were carried out in two phases. Phase 1 was done with a small scale model with two vehicles and an RSU in the junction. Simulations were done on single hop communication using this model. Parameters such as packet loss ratio and throughput were measured against range in this phase. The Phase 2 simulations, were done with a large scale model with clusters of vehicles. In this phase, the communication happened through multiple nodes and two routing schemes were used namely OLSR and AODV, for transfer of packets. The average end-to-end delay and average packet receiving ratio were calculated using 200 independent runs of the simulation. Also, delays of different WD implementations were analyzed in Phase 2.

Finally, we investigated the results of these simulations and they highlighted the better performance of DSRC compared to WD. However, we have pointed out that the gap in the performance is not drastically high. Hence, we sensed the feasibility of WD to be used as an alternative technology in VANET with some enhancements. Some of the suggested enhancements are provided in the next section.

6.2 Future Work

In this section, we are going to introduce some of the ideas, for the purpose of future study in this area. Some of them are related to the enhancement of the WD protocol and rest of them are application level barriers, which are needed to be overcome.

6.2.1 Reducing the delay

Reducing the delays such as the transmission delay and the group formation delay is essential for the performance of WD in VANETs. The works in [15, 16] suggest some improvement works to reduce these delays. These are needed to be incorporated and checked in the smartphones to make WD more suitable for transferring data in the context of vehicular communications. The WD protocol can be further modified in order to reduce the transmission delays by using different packet transfer patterns. It should be noted that the group formation delays are also required to be minimized for further enhancement in the performance of WD.

6.2.2 Overcome the challenges in real implementation

In addition to the delay, there are some other challenges that needed to be addressed. The first challenge is the requirement for user intervention (by entering a pin or by pressing a button) to connect two devices for the first time. This is a barrier for WD to be used in VANETs, because the driver's intervention for every connection, is not practical. Some of the existing works suggest several methods to overcome this challenge. Notably, the work in [38] suggests a method to form an autonomous WD network, without rooting the smartphone.

The second challenge is regarding to power, since smartphones need to be active all the time, if used for vehicular communications. Solutions should be found for providing power to the phone in the vehicle as well as optimizing the power.

Finally, the concerns about security and privacy also need to be addressed, to make the drivers participate willingly in a ITS system.

Appendices

Appendix A

Sample codes

A sample code for the simulation in phase 2 is provided here. This code is correspond to the modified WD experiments. Codes for other experiments explained in Section 5 are modified form this. Also, codes for DSRC simulation is slightly different form this example as explained in 4. The sending application is defined in the SenderApp.h file which is provided in A.2.

A.1 Main Application

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/olsr-helper.h"
#include "ns3/dsdv-helper.h"
#include "ns3/flow-monitor-module.h"
#include "SenderApp.h"
#include "ns3/netanim-module.h"
#include "ns3/aodv-module.h"
#include "ns3/rng-seed-manager.h"

#include <iostream>
#include <fstream>
#include <vector>
#include <string>
double startapp=15.0;
int from1=22;
int to1=48;
int from3=41;
```

```
int to3=32;
int from4=17;
int to4=44;
const char *ip1="10.1.1.23";
const char *ip2="10.1.1.49";
const char *ip5="10.1.1.42";
const char *ip6="10.1.1.33";
const char *ip7="10.1.1.18";
const char *ip8="10.1.1.45";

NS_LOG_COMPONENT_DEFINE ("Lab5_0");
using namespace ns3;
uint32_t MacTxDropCount, PhyTxDropCount, PhyRxDropCount;

void
MacTxDrop(Ptr<const Packet> p)
{
  NS_LOG_INFO("Packet Drop");
  MacTxDropCount++;
}

// for print the packet loss in certain node
void
PrintDrop(Ptr<Node> node)
{
   Simulator::Schedule(Seconds(5.0), &PrintDrop,node);
}

// for print the Tx packet loss in certain node
void
PhyTxDrop(Ptr<const Packet> p)
{
  NS_LOG_INFO("Packet Drop");
  PhyTxDropCount++;
}

// for print the  Rx packet loss in certain node
void
PhyRxDrop(Ptr<const Packet> p)
{
  NS_LOG_INFO("Packet Drop");
  PhyRxDropCount++;
}
```

```
int mymain (int argc, char *argv[],std::string a)
{

  std::string datarate=a;
  std::string phyMode ("OfdmRate18Mbps");

  uint32_t numNodes = 50;  // Number of nodes
  uint32_t packetSize = 600; // Packet size in bytes
  uint32_t numPackets = 1000; //number of packets

  std::string rtslimit = "2200";  // Limit for enable RTS/CTS
  // turn off RTS/CTS for frames below 2200 bytes
  Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold",
                      StringValue (rtslimit));
  // Fix non-unicast data rate to be the same as that of unicast
  Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",
                      StringValue (phyMode));

  NodeContainer c;
  c.Create (numNodes);


  WifiHelper wifi;

  //PHY configuration
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
  wifiPhy.SetPcapDataLinkType(YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

  //Configuring the channel
  YansWifiChannelHelper Channel;
  Channel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
  Channel.AddPropagationLoss("ns3::NakagamiPropagationLossModel",
                      "Distance1",DoubleValue(50),
                      "Distance2",DoubleValue(150),
                      "m0",DoubleValue(3),
                      "m1",DoubleValue(1.5),
                      "m2",DoubleValue(1));
  double freq=5.15e9;
 //Configuring tworay ground model
  Channel.AddPropagationLoss ("ns3::TwoRayGroundPropagationLossModel",
                      "Frequency",DoubleValue(freq),
                      "HeightAboveZ",DoubleValue(0.8),
                      "SystemLoss",DoubleValue(1));
```

```
// set the channel for PHY model
 wifiPhy.SetChannel (Channel.Create ());

 // Add a non-QoS upper mac, and disable rate control
 NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
 wifi.SetStandard (WIFI_PHY_STANDARD_80211a);
 wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                               "DataMode",StringValue (phyMode),
                               "ControlMode",StringValue (phyMode));
 // Set it to adhoc mode
 wifiMac.SetType ("ns3::AdhocWifiMac");
 NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, c);

//configuring the mobility using tcl file
 Ns2MobilityHelper ns2 = Ns2MobilityHelper ("scratch/tracefile.tcl");
 ns2.Install ();

 // Enable routing
 OlsrHelper olsr;
 AodvHelper aodv;
 DsdvHelper dsdv;

 Ipv4ListRoutingHelper list;
 list.Add (olsr, 10);

 //Install internet statck
 InternetStackHelper internet;
 internet.SetRoutingHelper (list); // has effect on the next Install ()
 internet.Install (c);

 Ipv4AddressHelper ipv4;
NS_LOG_INFO ("Assign␣IP␣Addresses.");
 ipv4.SetBase ("10.1.1.0", "255.255.255.0");
 Ipv4InterfaceContainer ifcont = ipv4.Assign (devices);

 // Create Apps
// In thois example we create 3 flows
// this can be change by changing the code here
 uint16_t sinkPort = 6; // use the same for all apps

 // UDP connection from N22 to N48
  Address sinkAddress1 (InetSocketAddress (ifcont.GetAddress (to1),
```

```
            sinkPort)); // interface of n22
PacketSinkHelper packetSinkHelper1 ("ns3::UdpSocketFactory",
            InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
ApplicationContainer sinkApps1 =
            packetSinkHelper1.Install (c.Get (to1)); //n48 as sink
sinkApps1.Start (Seconds (0.));
sinkApps1.Stop (Seconds (100.));

Ptr<Socket> ns3UdpSocket1 = Socket::CreateSocket (c.Get (from1),
            UdpSocketFactory::GetTypeId ()); //source at n22

// Create UDP application at n22
Ptr<MyApp> app1 = CreateObject<MyApp> ();
app1->Setup (ns3UdpSocket1, sinkAddress1,
            packetSize, numPackets, DataRate (datarate));
c.Get (from1)->AddApplication (app1);
app1->SetStartTime (Seconds (startapp));//31.
app1->SetStopTime (Seconds (100.));

// UDP connection from N22 to N48
Address sinkAddress1 (InetSocketAddress (ifcont.GetAddress (to1),
            sinkPort)); // interface of n22
PacketSinkHelper packetSinkHelper1 ("ns3::UdpSocketFactory",
            InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
ApplicationContainer sinkApps1 =
            packetSinkHelper1.Install (c.Get (to1)); //n48 as sink
sinkApps1.Start (Seconds (0.));
sinkApps1.Stop (Seconds (100.));

Ptr<Socket> ns3UdpSocket1 = Socket::CreateSocket (c.Get (from1),
            UdpSocketFactory::GetTypeId ()); //source at n22

// Create UDP application at n22
Ptr<MyApp> app1 = CreateObject<MyApp> ();
app1->Setup (ns3UdpSocket1, sinkAddress1,
            packetSize, numPackets, DataRate (datarate));
c.Get (from1)->AddApplication (app1);
app1->SetStartTime (Seconds (startapp));//31.
app1->SetStopTime (Seconds (100.));

// UDP connection from N41 to N32
Address sinkAddress2 (InetSocketAddress (ifcont.GetAddress (to2),
            sinkPort)); // interface of n41
PacketSinkHelper packetSinkHelper2 ("ns3::UdpSocketFactory",
```

```
                InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
  ApplicationContainer sinkApps2 =
                packetSinkHelper2.Install (c.Get (to2)); //n32 as sink
  sinkApps2.Start (Seconds (0.));
  sinkApps2.Stop (Seconds (100.));

  Ptr<Socket> ns3UdpSocket2 = Socket::CreateSocket (c.Get (from2),
                UdpSocketFactory::GetTypeId ()); //source at n41

  // Create UDP application at n41
  Ptr<MyApp> app2 = CreateObject<MyApp> ();
  app1->Setup (ns3UdpSocket2, sinkAddress2,
                packetSize, numPackets, DataRate (datarate));
  c.Get (from2)->AddApplication (app2);
  app2->SetStartTime (Seconds (startapp));
  app2->SetStopTime (Seconds (100.));

  // UDP connection from 17 to N44
  Address sinkAddress3 (InetSocketAddress (ifcont.GetAddress (to3),
                sinkPort)); // interface of n17
  PacketSinkHelper packetSinkHelper3 ("ns3::UdpSocketFactory",
                InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
  ApplicationContainer sinkApps3 =
                packetSinkHelper3.Install (c.Get (to3)); //n44 as sink
  sinkApps3.Start (Seconds (0.));
  sinkApps3.Stop (Seconds (100.));

  Ptr<Socket> ns3UdpSocket3 = Socket::CreateSocket (c.Get (from3),
                UdpSocketFactory::GetTypeId ()); //source at n17

  // Create UDP application at n17
  Ptr<MyApp> app3 = CreateObject<MyApp> ();
  app1->Setup (ns3UdpSocket3, sinkAddress3,
                packetSize, numPackets, DataRate (datarate));
  c.Get (from3)->AddApplication (app3);
  app1->SetStartTime (Seconds (startapp));
  app1->SetStopTime (Seconds (100.));

 // Install FlowMonitor on all nodes
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();


 // Trace Collisions
```

```
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*/$
␣␣␣␣␣␣␣␣ns3::WifiNetDevice/Mac/MacTxDrop", MakeCallback(&MacTxDrop));
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*/$
␣␣␣␣␣␣␣␣ns3::WifiNetDevice/Phy/PhyRxDrop", MakeCallback(&PhyRxDrop));
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*/$
␣␣␣␣␣␣␣␣ns3::WifiNetDevice/Phy/PhyTxDrop", MakeCallback(&PhyTxDrop));

  Simulator::Schedule(Seconds(5.0), &PrintDrop,c.Get(23));

  AnimationInterface anim ("Visualize.xml"); // Mandatory
  anim.EnablePacketMetadata (); // Optional



  Simulator::Stop (Seconds (100.0));
  Simulator::Run ();

  PrintDrop(c.Get(23)); // for printing drops in node 23

  // Print per flow statistics
  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier =
               DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator iter =
               stats.begin (); iter != stats.end (); ++iter)
    {

   Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);

      if ((t.sourceAddress == Ipv4Address(ip1) &&
                        t.destinationAddress == Ipv4Address(ip2))
        || (t.sourceAddress == Ipv4Address(ip3) &&
                        t.destinationAddress == Ipv4Address(ip4))
        || (t.sourceAddress == Ipv4Address(ip5) &&
                        t.destinationAddress == Ipv4Address(ip6)))
        {

NS_LOG_UNCOND(datarate <<","<<iter->first << "," << t.sourceAddress << ","
               << t.destinationAddress << ","
       << iter->second.txPackets <<","
       << iter->second.rxPackets<< ","
       <<iter->second.delaySum.GetSeconds()<<","
```

```
        <<iter->second.timesForwarded<<","
        <<iter->second.lostPackets<<","
        << iter->second.rxBytes * 8.0 /
        (iter->second.timeLastRxPacket.GetSeconds()-
                iter->second.timeFirstTxPacket.GetSeconds()) / 1024 );


        }
    }
  monitor->SerializeToXmlFile("FlowFile.flowmon", true, true);

  Simulator::Destroy ();

  return 0;
}

int main (int argc, char *argv[])
{

  CommandLine cmd;
  int Run=3;
   // to obtaing seed run value as an argument.
  cmd.AddValue ("Run", "Wifi Phy mode", Run);
  cmd.Parse (argc, argv);
  ns3::SeedManager::SetSeed(1);
  ns3::SeedManager::SetRun(Run); // Set the seed run value
  mymain(argc,argv,"1Mbps")

return 0;
}


}
```

A.2 Sender Application

```
#include "ns3/applications-module.h"
#include "ns3/core-module.h"

using namespace ns3;


class SenderApp : public Application
{
```

```
public:

  SenderApp ();
  virtual ~SenderApp();

  void Setup (Ptr<Socket> socket, Address address,
              uint32_t packetSize, uint32_t nPackets, DataRate dataRate);

private:
  virtual void StartApplication (void);
  virtual void StopApplication (void);

  void ScheduleTx (void);
  void SendPacket (void);

  Ptr<Socket>     m_socket;
  Address         m_peer;
  uint32_t        m_packetSize;
  uint32_t        m_nPackets;
  DataRate        m_dataRate;
  EventId         m_sendEvent;
  bool            m_running;
  uint32_t        m_packetsSent;
};

SenderApp::SenderApp ()
  : m_socket (0),
    m_peer (),
    m_packetSize (0),
    m_nPackets (0),
    m_dataRate (0),
    m_sendEvent (),
    m_running (false),
    m_packetsSent (0)
{
}

SenderApp::~SenderApp()
{
  m_socket = 0;
}


//Set the parameters
```

```
void
SenderApp::Setup (Ptr<Socket> socket, Address address,
                  uint32_t packetSize, uint32_t nPackets, DataRate dataRate)
{
  m_socket = socket;
  m_peer = address;
  m_packetSize = packetSize;
  m_nPackets = nPackets;
  m_dataRate = dataRate;
}

//Start sending application
void
SenderApp::StartApplication (void)
{
  m_running = true;
  m_packetsSent = 0;
  m_socket->Bind ();
  m_socket->Connect (m_peer);
  SendPacket ();
}
//Stop sending application
void
SenderApp::StopApplication (void)
{
  m_running = false;

  if (m_sendEvent.IsRunning ())
    {
      Simulator::Cancel (m_sendEvent);
    }

  if (m_socket)
    {
      m_socket->Close ();
    }
}

//function for send the packet
void
SenderApp::SendPacket (void)
{
  Ptr<Packet> packet = Create<Packet> (m_packetSize);
  m_socket->Send (packet);
```

```
  if (++m_packetsSent < m_nPackets)
    {
      ScheduleTx ();
    }
}

//Scedule the next packet using data rate
void
SenderApp::ScheduleTx (void)
{
  if (m_running)
    {
      Time tNext (Seconds (m_packetSize * 8 / static_cast<double>
        (m_dataRate.GetBitRate ())));
      m_sendEvent = Simulator::Schedule (tNext,
        &SenderApp::SendPacket, this);
    }
}
```

References

[1] G. Karagiannis, O. ltintas, E. Ekici, G.Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Communications Surveys and Tutorials*, vol. 13, pp. 584 – 616, Jul. 2011.

[2] T. D. Hewer, *High Performance Simulation and Modeling of Wireless Vehicular ad-hoc Networks.* Thesis. University College London, 2011.

[3] P. Singh and K. Lego, "Comparative study of radio propagation and mobility models in vehicular ad-hoc network," *International Journal of Computer Applications*, vol. 16, pp. 37–42, Feb. 2011.

[4] C. Perkins and E. Royer, "Adhoc on-demand distance vector routing," in *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999, pp. 90–100.

[5] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad-hoc networks," in *Proc. IEEE International Multi-Topic Conference*, 2001, pp. 62–68.

[6] P. Katkar and V. Ghorpade, "Comparative study of network simulator: NS2 and NS3," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, pp. 608–612, Mar. 2016.

[7] C. Raj, U. Upadhayaya, T. Makwana, and P. Mahida, "Simulation of VANET using NS-3 and SUMO," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, pp. 563–569, Apr. 2014.

[8] Y. J. Li, *An Overview of the DSRC/WAVE Technology.* Springer Berlin Heidelberg, 2012, vol. 74, ch. Quality, Reliability, Security and Robustness in Heterogeneous Networks, pp. 544–558.

[9] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[10] C. Daniel, A. Saavedra, and P. Serrano, "Device-to-device communication with Wi-Fi Direct: Overview and experimentation," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 96–104, Jun. 2013.

[11] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *Proc.IEEE Global Telecommunications Conference*, Dec. 2004.

[12] M. H. Manshaei and J. Hubaux, "Performance analysis of the IEEE 802.11 distributed coordination function: Bianchi model," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, Seb. 2006.

[13] A. ZÂťuquete, "Improved csma/ca protocol for ieee 802.11," in *Proc.Next Generation Internet Networks*, Apr. 2008.

[14] C. Profentzas, *Studying Routing Issues in VANETs by Using NS-3.* Thesis. Alexander Technological Educational Institute of Thessaloniki, 2012.

[15] C. Satish, *Inter-vehicular Communication for Collision Avoidance Using Wi-Fi Direct.* Thesis. Rochester Institute of Technology, 2014.

[16] P. Angadi, *Increased Persistence of Wi-Fi Direct Networks for Smart phone based Collision Avoidance.* Thesis. Rochester Institute of Technology, 2014.

[17] N. Shuhaimi, Heriansyah, T. Juhana, and A. Kurniawan, "Performance analysis for uniform and binomial distribution on contention window using DSRC and Wi-Fi Direct standard," *International Journal of Electrical and Computer Engineering*, vol. 5, no. 6, pp. 1452–1457, Dec. 2015.

[18] S. Hu, H. Liu, L. Su, H. Wang, F. Abdelzaher, P. Hui, W. Zheng, Z. Xiek, and J. Stankovick, "Towards automatic phone-to-phone communication for vehicular networking applications," in *Proc. IEEE Conference on Computer Communications*, Apr. 2014, pp. 1752–1760.

[19] W. Jin, C. Kwan, Z. Sun, H. Yang, and Q. Gan, "SPIVC: A smartphone-based inter-vehicle communication system," in *Proc.Transportation Research Board 91st Annual Meeting*, Jan. 2012.

[20] S. Lee and A. Lim, "An empirical study on ad hoc performance of DSRC and Wi-Fi vehicular communications," *International Journal of Distributed Sensor Networks*, vol. 9, no. 11, Seb. 2013.

[21] W. Lin, M. Li, K. Lan, and C. Hsu, *A Comparison of 802.11a and 802.11p for V-to-I Communication: A Measurement Study.* Springer Berlin Heidelberg, 2012, vol. 74, ch. Quality, Reliability, Security and Robustness in Heterogeneous Networks, pp. 559–570.

[22] P. Choi, J. Gao, N. Ramanathan, M. Mao, S. Xu, C. Boon, S. Fahmy, and L. Peh, "A case for leveraging 802.11p for direct phone-to-phone communications," in *Proc. IEEE International Symposium on Low Power Electronics and Design*, Aug. 2014, pp. 207–212.

[23] S. A. H. Tabatabaei, M. Fleury, N. N. Qadri, and M. Ghanbari., "Improving propagation modeling in urban environments for vehicular ad hoc networks." *IEEE Trans. Intelligent Transportation Systems.*, vol. 12, no. 3, pp. 1–12, Seb. 2011.

[24] K. Mizutani and R. Kohno, "Analysis of multipath fading due to tworay fading and vertical fluctuation of the vehicles in its inter-vehicle communications." in *Proc. IEEE 5th International Conference on Intelligent Transportation Systems*, 2002, pp. 318–323.

[25] M. Killat and H. Hartenstein, "An empirical model for probability of packet reception in vehicular ad hoc networks," *EURASIP Journal on Wireless Communications and Networking*, no. 4, Jan. 2009.

[26] J. Yin, G. Holland, T. ElBatt, F. Bai, and H. Krishnan, "DSRC channel fading analysis from empirical measurement," in *Proc.First International Conference on Communications and Networking in China*, Oct. 2006.

[27] L. Rubio, N. Cardona, S. Flores, J. Reig, and L. Juan-Llacer, "The use of semi-deterministic propagation models for the prediction of the short-term fading statistics in urban environments," in *Proc.IEEE 49th Vehicular Technology Conference, Amsterdam*, 1999, pp. 1460–1464.

[28] L. Rubio, J. Reig, and N. Cardona, "Evaluation of nakagami fading behaviour based on measurements in urban scenarios," *International Journal of Electronics and Communication*, pp. 135–138, Feb. 2007.

[29] V. Taliwal, D. Jiang, H. Mangold, C. Chen, , and R. Sengupta, "Empirical determination of channel characteristics for DSRC vehicle-to-vehicle communication," in *Proc.1st ACM International Workshop on Vehicular Ad Hoc Networks*, Oct. 2004.

[30] T. Marc, D. Jiang, and H. Hartenstein, "Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks," in *Proc. ACM International Workshop on Vehicular ad-hoc Networks*, Oct. 2004, pp. 105–110.

[31] F. J. Martinez, C. Toh, J. Cano, C. T. Calafate, and P. Manzoni, "Realistic radio propagation models (RPMs) for VANET simulations," in *Proc. IEEE Wireless Communications and Networking Conference*, May. 2009.

[32] S. E. Carpenter, "Obstacle shadowing influences in VANET safety," in *Proc.IEEE 22nd International Conference on Network Protocols*, Oct. 2014.

[33] S. E. Carpenter and M. L. Sichitiu, "An obstacle model implementation for evaluating radio shadowing with NS-3," in *Proc.Workshop on ns-3, Spain*, May. 2015, pp. 17–24.

[34] M. C. Aswathy and C. Tripti, "A cluster based enhancement to AODV for inter-vehicular communication in VANET," *International Journal of Grid Computing and Applications*, vol. 3, no. 3, pp. 41–50, Seb. 2012.

[35] S. Jibhkate, S. Khare, A. Kamble, and A. Jeyakumar, "AODV and OLSR based routing algorithm for highway and city scenarios," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 275–280, Jun. 2015.

[36] J. Toutouh, J. G-Nieto, and E. Alba, "Intelligent OLSR routing protocol optimization for VANETs," *IEEE Trans. Vehicular Technology*, vol. 61, no. 4, pp. 1884–1894, May. 2012.

[37] H. JÃľrÃťme, F. Fethi, and B. Christian, "Performance comparison of AODV and OLSR in VANETs urban environments under realistic mobility patterns," in *Proc.5th IFIP Mediterranean Ad-Hoc Networking Workshop*, Jun. 2006.

[38] P. Wong, V. Varikota, D. Nguyen, and A. Abukmail, "Automatic android-based wireless mesh networks," *Informatica*, vol. 38, no. 4, pp. 313–320, Dec. 2014.