Enhancing the Wi-Fi Direct protocol for Vehicular Ad-hoc Networks

Wageesha Nilmini Manamperi

(168070G)

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

Department of Electronic and Telecommunication Engineering

University of Moratuwa
Sri Lanka

June 2018

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 the 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 work (such as articles or books).

Wageesha — 13/06/2018

Signature: Date:

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

Signature: ***UOM Verified Signature*** Date: 13/06/2018

Signature: ***UOM Verified Signature*** Date: 13/06/2018

Abstract

We present a technique for enhancing Wi-Fi Direct (WD) for vehicular environments. Dedicated short range communication (DSRC) has been standardized for communication in Intelligent Transportation Systems (ITS). However, due to high costs at initiation, alternative communication strategies are of interest in order to facilitate the quick deployment of ITSs. WD, which is a relatively mature technology available in mobile devices, has come across as a possible alternate candidate. However, the presence of large communication delays in the WD protocol stack is a shortcoming in deploying this in highly dynamic vehicular scenarios. The objective of our work is to propose and evaluate a method to overcome some of the large transmission delays in WD. Our proposal is to use a broadcast mechanism in the downlink between the group owner (GO) and the clients of a WD group, as an alternative to the currently used peer-to-peer (P2P) method.

We study our technique by simulating a bi-directional highway scenario with multiple lanes. We set up the vehicular channel model using two well-known models: Friis propagation model and the Nakagami fading model. Performance measures such as average total delay, average energy consumption of the GO, average packet loss ratio, and average packet reception ratio are presented.

While the proposed GO Broadcast method reduces the downlink delay, it increases the probability of packet losses due to the lack of retransmissions. Our results demonstrate a gain in terms of average total delay and the average energy consumption of the GO. We use a theoretical analysis as well as a simulation study using OMNeT++. It is also shown that the degradation in performance on the downlink due to packet losses is within tolerable limits, given that the size of the group is selected properly.

Index terms— Broadcast mechanism, Group formation, Peer-to-Peer (P2P), Vehicular Ad-hoc Network (VANET), Wi-Fi Direct (WD)

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors Dr. Tharaka Samarasinghe and Prof. Dileeka Dias for their precious advice, valuable insights, and guidance throughout the research. I thank them for sharing their expertise, and specially for making time in their busy schedules to help me with my Masters.

In addition to my supervisors, I would like to thank Dr. Asanga Udugama, for being my progress committee chair and for his assistance and comments for improving and completing the research. I would also like to thank Dr. Ruwan Udayanga, for being my progress committee member and for providing valuable comments throughout my Masters.

I would like to thank my university, The University of Moratuwa, for providing me financial support throughout this year.

I wish to thank my colleagues who helped me directly and indirectly.

Last but not least, I would like to express my deepest gratitude to my beloved parents, sisters, for their endless support, love, and care.

This work was supported by the Senate Research Committee under grant SRC/LT/2015/07.

Thank you
Wageesha Nilmini Manamperi

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
ACK	Acknowledgement
AP	Access Point
BPSK	Binary Phase Shift Key
CSMA/CA	Carrier Sense Multiple Access with Collisions Avoidance
CTS	Clear to Send
D2D	Device-to-Device
DHCP	Dynamic Host Configuration Protocol
DSRC	Dedicated Short Range Communication
GO	Group Owner
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ITS	Intelligent Transportation System
IV	Intent Value
LAN	Local Area Network
MAC	Medium Access Control
MANET	Mobile Ad-hoc Network
MTU	Maximum Transmission Unit
NED	NEtwork Description
NIC	Network Interface Controller
OBU	On Board Unit
P2P	Peer To Peer
PLR	Packet Loss Ratio
PRR	Packet Reception Ratio
QoS	Quality of Services
RSSI	Received Signal Strength Indicator
RSU	Road Side Unit
RTS	Request to Send

SSRC	Stations Short Retry Count
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad-hoc Network
WD	Wi-Fi Direct
WPS	Wi-Fi Protected Setup

Chapter 1

Introduction

Intelligent transportation systems (ITSs) reshape the future of road transportation through a variety of applications to enhance road safety and travel efficiency [1]. There are two types of ITS applications: safety applications and non-safety applications. Examples of safety applications are collision detection, lane change warning, and cooperative merging [2] and require both delay and high reliability, concurrently. The non-safety applications aim to optimize the road traffic flows and to reduce the carbon dioxide emission through smart and green transportation applications. They also support infotainment applications such as parking assistance, audio-video streaming, and interactive gaming [3]. The non-safety applications do not have specific delay requirements unlike the safety applications, but the performance measures degrade with the increase in the delay and packet loss. Vehicular ad-hoc networks (VANETs) is a key enabler of ITS [4]. We introduce VANETs in Section 1.1.

1.1 VANETs

Mobile ad-hoc network (MANET) is a collection of mobile nodes which communicate directly with each other. VANET is an extension of MANETs where mobile nodes are vehicles. There is intensive research ongoing within the area of VANETs. VANET can communicate with internal and external environments using the nodes which are called on-board units (OBUs) and road-side units (RSUs). The OBU is an in-vehicle transceiver which allows exchanging data between the nodes. The RSU is a transceiver fixed on the road-side infrastructure. This communication categorizes into either vehicle-to-vehicle (V2V) communication or vehicle-to-infrastructure (V2I) communication.

V2V communication which is the basis for the setup of VANETs, plays an important role in future transportation systems. V2V communication allows vehicles to communicate directly with minimal latency as well as supports either single hop or multi-hop multicast/broadcast communication. Figure 1.1 shows a

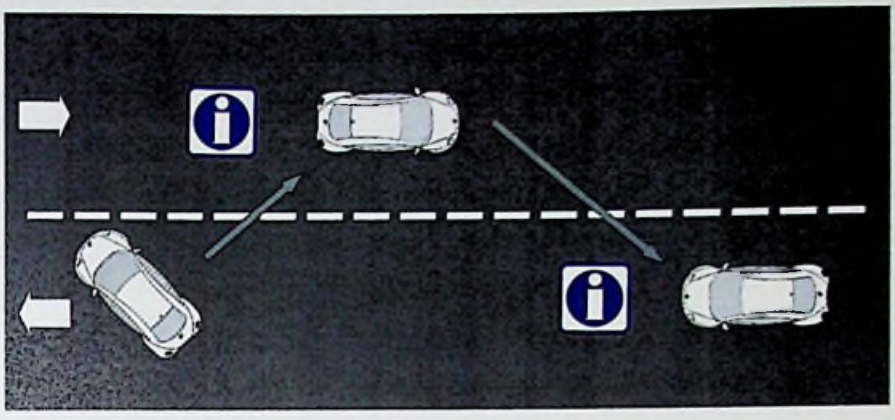


Figure 1.1: Vehicle-to-vehicle communication. Example of an event-driven messages exchange through a highway scenario.

V2V communication on a highway scenario where the nodes are vehicles whose movement is limited to the road. In Figure 1.1, a vehicle encounters a dangerous situation and sends an emergency message to a vehicle behind it. On successful reception of the message, the receiving vehicle forwards that message to the vehicle in front of it. Therefore, V2V communication provides advance warning and driving assistance to all the vehicles in the vicinity.

V2I communication allows communication between vehicles and infrastructure, for example, the ITSs infrastructure can be an RSU, a traffic light, or street signs. The V2I communication supports a single hop broadcast communication where the RSU periodically broadcasts a message to the vehicles in the vicinity. However, RSUs need to be placed every kilometer or less, to maintain an information-rich travel environment [5].

There are two types of safety-related messages exchanged in VANETs: periodic and event-driven. The periodic messages are exchanged by all the vehicles in the network which contain information about their status such as position, speed, direction, heading, etc. The event-driven messages are sent by the vehicles when a hazard event occurs which enable ITSs safety applications. These applications avoid traffic accidents and alleviate traffic congestion by sending periodic messages which monitor the locations of nearby vehicle aided by event-driven messages [6]. The different applications have different communication requirements.

The deployment of vehicular communication requires emerging technologies for the penetration of ITS. Dedicated short range communication (DSRC), which is based on the IEEE 802.11p standard, is the well known communication method

in VANETs [7,8] and allocates 75 MHz bandwidth on 5.9 GHz spectrum and has a communication range from 100 m to 1000 m. DSRC allows exchange messages through the network by dividing dedicated bandwidth into seven channels to support both safety and non-safety applications simultaneously. The data rates provide by IEEE 802.11p range from 3 to 27 Mbps [7].

Wi-Fi based solutions suggest as an alternative communication technology for VANETs. Wi-Fi direct (WD), which is based on the IEEE 802.11 infrastructure mode, and already enabled in most smart phones, has been looked upon as a possible candidate [9,10], to quick deployment of ITS services.

1.2 Why Wi-Fi direct ?

DSRC is the centerpiece of VANETs. However, it requires DSRC radio which is installed in all vehicular nodes of the network which can be either embedded to the vehicle or an additional equipment that can be connected to the vehicle externally. The embedded equipment cost of $350 and the additional equipment cost of $175 [11], hence, the capital expenditure at initiation is high [9,10]. The installation of the dedicated hardware requires technical expertise, and larger scale deployment of DSRC will take a considerable time. Also, DSRC makes it difficult to incorporate ITSs with pedestrians and cyclists due to the need of the dedicated hardware.

This hinders the penetration of ITS in developing countries, where ITS seems to be a distant reality and has also created interest into alternative communication strategies that can facilitate a quick deployment of ITS. WD, which is introduced to enhance device-to-device (D2D) communication [12], and enables Wi-Fi devices to directly communicate without passing through an access point (AP). The WD communicates both WD devices and legacy Wi-Fi devices (conventional Wi-Fi devices that does not support WD) to exchange the information. WD has been already enabled in most smart phones, may be utilized to increase the penetration of ITS. WD has the added advantage of being able to connect important stakeholders such as pedestrians and cyclists to VANET more easily compared to DSRC.

WD operates on 2.4 GHz frequency spectrum and allocates 20 MHz bandwidth whereas DSRC operates on 5.9 GHz frequency spectrum and allocates 75 MHz bandwidth which divides into seven 10 MHz channels. WD supports data rates up to 250 Mbps while DSRC supports data rates up to 27 Mbps. Therefore, WD provides higher specifications in terms of channel bandwidth and data rates

compared to DSRC.

Because of these reasons, the feasibility of using WD for VANETs as an alternative to the DSRC has been studied [9,10]. Hence, we will use WD for VANETs as the underlying technology for our work.

1.3 Focus of the thesis

In this thesis, we consider that WD as a viable alternative for DSRC. The focus is mainly on improving the WD protocol to be suited to VANETs.

It is obvious that DSRC is superior to WD in terms of performance due to having a dedicated spectrum allocation and a higher transmit power. Therefore, for safety critical applications, where reliability is paramount [1], WD may not be a suitable alternative. However, WD has a potential for non-safety applications like traffic control and given the protocol is fine tuned. There are few drawbacks of WD such as large group formation time in WD, larger transmission delay due to the multiple retransmissions by the group owner (GO) and single point of failure for the group if the GO fails or moves out of range of the group. This leads to large delays in packet delivery which is the most critical one with respect to VANETs, and [9,13,14] have focused on alleviating these issues.

In particular, we only consider the larger transmission delay in WD. WD information dissemination within a group occurs only through the GO after periodic one-hop beacon communication. This process uses a peer-to-peer (P2P) approach on the uplink (between clients and the GO) as well as the downlink (between GO and clients). Since all the clients communicate through the GO, there is a larger delay in messages sent over WD than the messages sent over DSRC. Therefore, we attempt to modify the WD protocol to overcome the above addressed drawback and improve its performance in order to be used in VANETs.

In this thesis, we focus on reducing the large transmission delay through a broadcasting mechanism, which eliminates acknowledgments and retransmissions in the link from GO to clients (downlink). In [9], proposes a broadcasting mechanism for reducing the large transmission delay in WD. The authors in [9] present a theoretical analysis, and validate their claim through numerical evaluations. Our work is centered on further investigating the idea presented in [9] through a simulation based approach (OMNeT++). A simulation based approach gives us more flexibility to fine tune the model such that it gives a better representation of the actual scenario.

1.4 Contributions and the outline of the thesis

Firstly, we discuss some of the fundamentals and background information in Chapter 2. We first explain the WD protocol architecture to provide a better understanding of the domain of study. Next, we identify the drawbacks of WD which cause large delays in packet delivery. In this thesis, we consider the delay component related to the transmission delay which addresses the data communication in VANETs. We then review the most related literature.

In Chapter 3, we present the system model. Firstly, we model the topology of the vehicles. We model a bidirectional highway with multiple lanes, and set speeds of individual vehicles, and set the spacing between vehicles in the same lane according to the 2-second rule. Secondly, we model the wireless channel, which is crucial for the performance. We model the channel using two well-known models: Friis propagation model to capture path loss and Nakagami fading model to capture multipath fading. Finally, we describe the two communication models. The first one is the P2P model where the downlink communication is based on a P2P method, and this is the methodology adopted in the standard WD protocol. In the second model, the downlink communication is based on a broadcast method, and which is called the group owner broadcast (GOB) model.

In Chapter 4, we theoretically analyze the performance measures. In particular, we derive the average total delay and average energy consumption of the GO (energy is not considered in [9]) for one data transmission cycle. We extend the theoretical analysis in [9] by considering backoff and contention for the communication channel. It provides an analytical justification for the GOB model gains in terms of average delay and average energy consumption compared to the P2P model.

In Chapter 5, we describe the algorithm implementation and simulation setup on OMNeT++ [15]. Therefore, we set up the two communication models on the initial implementation of the WD protocol on OMNeT++ [16]. We set the channel properties according to the channel model present in Chapter 3. Each vehicular node will have its own mobility model configured using the *linear mobility* module, which is inbuilt in the INET framework [17]. We configure the energy consumption of the vehicular node using the *state-based energy consumed* module, which is also implemented in the INET framework [17]. We set a timeout for GO to initiate transmitting data in the downlink. We provide the timeout using the simulation considering an ideal scenario of 100% beacon reception by all clients.

In Chapter 6, we present both theoretical and simulation results of the communication model studied in Chapter 3, and also discuss to draw further insights. In this Chapter, we provide the simulation results by averaging over 100 independent instances. In particular, we simulate the performance measures such as average total delay, average energy consumption of the GO, average packet loss ratio, and average packet reception ratio. Our simulation results are compared with the theoretical analysis described in Chapter 4 and also with the results of [9]. Simulations show that the broadcasting method leads to considerable gains in terms of delay and energy, and also show that if group sizes are set properly, the degradation of performance on the downlink due to not having retransmissions is within tolerable limits. Chapter 7 concludes the thesis.

Chapter 2

Fundamentals and Background

This chapter will first provide an overview of the WD protocol in Section 2.1. Next, we will explain the drawbacks of WD protocol which cause large delays in packet delivery in Section 2.2. Then, we will discuss the network simulator in Section 2.3. Section 2.4 provides an account of related literature.

2.1 Wi-Fi direct protocol

WD introduced by the Wi-Fi alliance and is based on the IEEE 802.11 infrastructure mode [18]. This aims to enhance the direct D2D communication [12]. WD operates on 2.4 GHz frequency spectrum and supports data rates up to 250 Mbps. The communication range of WD is less than 250 m, and allows the devices to directly communicate without passing through an AP. WD supports most of the Wi-Fi standards such as IEEE 802.11 a/d/g/n etc.

WD devices are formally known as P2P devices and communicated by forming a P2P group with 1:n topology consisting of one GO and zero or more clients [18]. The GO is the device which implements the AP-like functionalities, and the client is the device which associates with the GO. Legacy Wi-Fi devices can only connect to the group as clients. The role of the P2P device as either GO or client is specified dynamically in the group formation phase. Therefore, the P2P device supports concurrent operation where the device can execute the role of the GO and the client simultaneously. However, to support the concurrent operation, the P2P device requires either multiple medium access control (MAC) entities or virtualization techniques to time share the channel [12].

The GO is elected based on the intent value (IV) of the device where IV is an integer between 0 –15. The device which has the highest IV will become the GO [18]. If two devices have the same IV, the device in which the tie-breaker bit is set to one is selected. The IV calculation will be discussed in detail in Section 2.1.2.

The GO announces the presence of the group by sending beacons on a selected

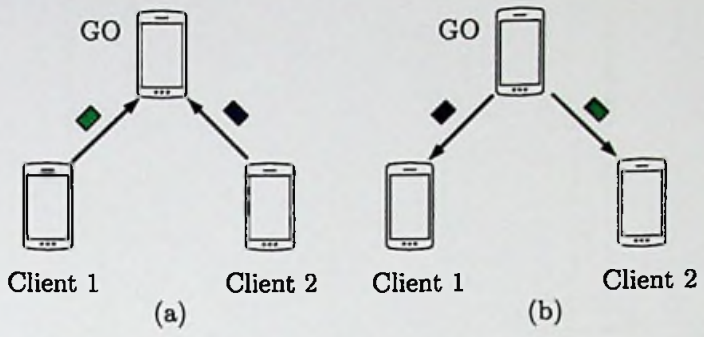


Figure 2.1: (a) Uplink: between the clients and the GO (b) Downlink: between the GO and the clients.

operating channel in the 2.4 GHz band. If the devices receive the beacons, they can join the group by sending a probe request frame to the GO. The clients can start to exchange the information after joining the group. WD information dissemination within a group occurs only through the GO after periodic one-hop beacon communication. This process uses the P2P approach on the uplink (between the clients and the GO) as well as the downlink (between the GO and the clients) as shown in Figure 2.1.

2.1.1 Group formation

WD group formation includes four phases: discovery phase, GO negotiation phase, provisioning phase and address configuration phase [18]. There are three states in the discovery phase: *scan, search* and *listen.* In the *scan* state, the devices perform a traditional Wi-Fi scan to discover any existent group or network. If the device locates a group, it can join that group by sending a probe request frame, or otherwise, the device moves to the *search* and *listen* states. In the *search* state, the device sends probe request frames on all the three social channels named as channel 1, 6 and 11 in the 2.4 GHz band. In the *listen* state, the device listens for probe requests on a selected social channel in order to respond with a probe response frame. The device switches between *search* and *listen* phases until two devices discover each other. Figure 2.2 shows an example of a device discovery phase for two P2P devices [18].

At the end of the discovery phase, the devices move to the GO negotiation phase to assign the respective roles: GO or client. The GO negotiation phase is a three-way handshake process which consists of GO negotiation request, response, and confirmation frames. The GO is elected based on the highest IV where the

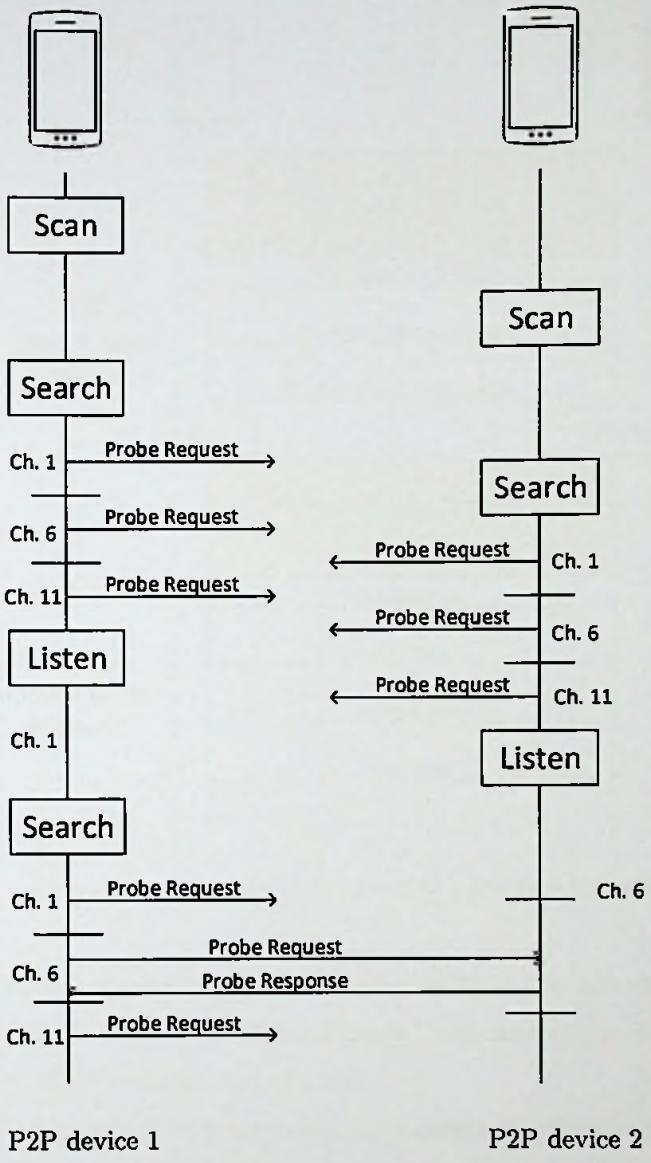


Figure 2.2: Example of a device discovery phase for two P2P devices.

devices share their IV using either GO negotiation request or response frames. The GO selects a social channel for communication within the group and starts to send the beacon frames to announce the presence of the group on the selected social channel.

In the provisioning phase, GO establishes a secure communication using Wi-Fi protected setup (WPS). There are two phases in the WPS provisioning: *phase 1* and *2*. In the provisioning *phase 1*, GO acts as a registrar which generates and shares the security keys to its clients. Then, in the provisioning *phase 2*, clients disconnect and reconnect through the provided security key.

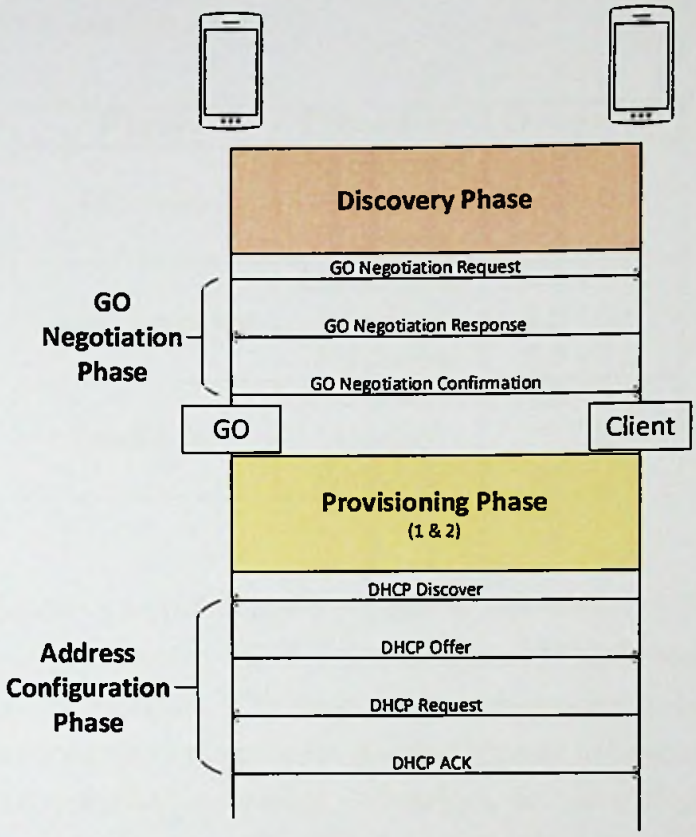


Figure 2.3: Frame exchange sequences in the standard group formation procedure [11].

In the address configuration phase, the GO assigns the internet protocol (IP) addresses to its clients using the dynamic host configuration protocol (DHCP) using the discover/offer/request/ack frames.

There are three types of group formation procedures: **standard, autonomous** and **persistent**. The **standard** procedure is the most complex group formation procedure which includes all the four group formation phases discussed above, unlike the **autonomous** and **persistent** procedures. Figure 2.3 shows the frame exchange sequences in the standard group formation procedure.

In the **autonomous** group formation procedure, a P2P device autonomously becomes the GO and initiates a group. The GO announces the presence of the group by sending the beacon on a selected social channel. Other devices which are in the discovery phase, can perform a scan and join the group as clients. Note that, the discovery phase simplified to *scan* state, and there is no *search* and *find* states. Also, there is no GO negotiation phase as the devices have already assigned their roles in the group, and directly proceed with the provisioning and address configuration phases.

Table 2.1: Delays of WD protocol for the three group formation procedures: Autonomous, Standard, and Persistent [11].

Phase	Procedure	Delays (s)
Discovery	Autonomous	3
	Standard	4 - 9
	Persistent	4 - 9
Group formation	Autonomous	1.5 - 2.5
	Standard	1.5 - 2.5
	Persistent	0.5 - 1.5
Total delay	Autonomous	4.5 - 5.5
	Standard	5 - 10
	Persistent	4.5 - 10

In the **persistent** group formation procedure, the devices will re-initiate the previously formed group using the stored network credentials and assigned roles in the initial group formation. The devices can declare a group as the persistent during the initial group formation by using a flag bit sets in beacon frames, probe responses and GO negotiation frames. Therefore, at the end of the discovery phase, the devices can discover whether they have formed a persistence group in the past with the corresponding peers. This replaces the GO negotiation phase with an invitation procedure to reform the previous group. There is no provisioning *phase 1* state since the devices re-initiate the group using the stored network credentials and directly proceed with the provisioning *phase 2* state and address configuration phase.

Table 2.1 summarizes the delay required to establish the group using the three group formation procedures [12]. In the analysis, the authors claim that the group formation delay defined as the summation of the GO negotiation phase, the provisioning phase, and the address configuration phase [12]. In particular, it shows that the autonomous group formation takes less time to form the group compared to the standard and persistent group formation procedure.

2.1.2 Intent value calculation

The IV is used to elect the GO in the negotiation phase. The definition of IV is not specified in the WD standard. Therefore, in the WD implementation on Android [19], IV sets to a fixed value whereas it sets to a random integer value between 0 –15 on OMNeT++ [16]. This takes into consideration that the GO is elected randomly in the negotiation phase. However, IV should reflect the device

capabilities in order to elect as the GO of a group.

To address this issue, an algorithm called *WD2* is introduced in [13]. In [13], the authors compute the IV based on the average received signal strength indicator (RSSI).

2.2 Drawbacks of Wi-Fi direct

As outlined in the introduction, the key challenge related to WD is to reduce the large delay associated with packet delivery. In this section, we have identified three main drawbacks of WD from the literature.

2.2.1 High group formation time

The group formation time of WD depends on both discovery phase and GO negotiation phase. In the discovery phase, the device switches between the *search* and *listen* state until it finds a peer within its range. The amount of time that a device spends on each state is randomly distributed, in the range of 100 ms and 300 ms [12]. In the *listen* state, the device selects one social channel among three in 2.4 GHz as its listen channel and listen for the probe request frames on that selected social channel. This selection is fixed until the end of the discovery phase. However, the device can select a high interference channel in the *listen* state. This causes to increase the delay in the discovery phase. Another issue in the discovery phase is that the per-channel waiting time is not explicitly defined in the 802.11 standards. The large values assign to the channel waiting time cause to increase the scanning delay. Table 2.1 shows that the total delay of the **standard** and the **persistent** group formation procedures are mainly due to the delay in the discovery phase.

In the GO negotiation phase, the GO elects based on the IV. However, in the standard WD protocol, GO is dynamically assigned using a random value without considering the device capabilities.

2.2.2 High transmission delay

WD standard specifies a P2P architecture where the clients can communicate only through the GO. This implies that no two clients can directly communicate with each other. In the uplink, where the request-to-send/clear-to-send (RTS/CTS) mechanism is used to access the channel, each client sends its data to the GO using a P2P method, and acknowledgments are used to ensure successful packet delivery. In the downlink, the GO sends the data frames back to its clients. This

implies that due to the P2P method, GO re-transmits the data frames to its clients in the downlink.

However, GO may retransmit the data frame to a client several times due to the poor channel conditions or due to the interference by other nearby devices. This retransmission will continue until the GO receives an acknowledgment frame or reaches to the maximum retry limit in the MAC layer. Therefore, when the number of nodes in the group increases, it causes to increase the retransmission time by the GO, and finally ends up with larger transmission delay for one data transmission cycle. However, this limits the number of nodes that can be directly communicated within the group.

2.2.3 Single point of failure of the group

In WD, devices are communicated by forming a group where GO facilitates the communication between its clients. If GO fails to manage the communication within the group, the entire group will tear down. This is called the single point of failure of the group. This happens either when GO fails due to power or when GO moves out of range of the group due to speed. Therefore, the group needs to be re-initiated in order to proceed further.

However, the re-initiation of the group is a lengthy process because devices either have to negotiate one of the clients as the GO or connect to available group by performing a Wi-Fi scan. This results in an additional delay which causes to increase the delay in WD packet delivery.

2.3 Simulator

In this section, we will discuss the architecture of the network simulator, OMNeT++, and the INET framework.

2.3.1 OMNeT++

OMNeT++ is an open-source, event-driven network simulator which is based on the object-oriented programming language C++. It includes an integrated development and a graphical runtime environment [20]. We can utilize OMNeT++ with INET framework to build a realistic vehicular simulation environment.

Figure 2.4 shows the modular architecture of OMNeT++ [15]. The **SIM** is the simulation kernel and class library which connects with program simulation. The **Envir** is a library that consists of general code common to all the user interfaces. The **Cmdenv, Tkenv and Qtenv** are an **Envir** based libraries

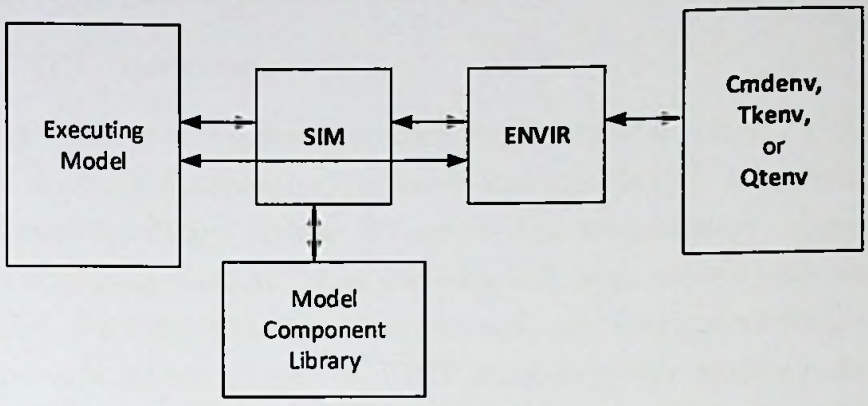


Figure 2.4: Architecture of OMNeT++.

that contain specific interface implementation. The simulation can be connected with **Cmdenv** and/or **Tkenv**. The model component library consists of simple module definition with C++ implementation, compound module types, channels, message types, networks and all other things that are connected with the model which presents in the simulation. The executing model mainly used for the simulations. This model consists of objects that instantiated from model component library instance.

A compound module is a combination of simple modules that make the system model. The module implementation is done using C++ code. For example, a module can be a host, AP, switch, and router. The modules can communicate with each other by exchanging messages. A message contains arbitrary complex data structures. The message in OMNeT++ comes with *.msg* extension and represents a frame or a packet in the real-time network simulation. The modules can send messages either directly to their destination or by a direct connection between modules through input/output gates and connections. Note that the message can be arrived from another module or the same module which is called as self-message. The gate is the input and the output interface of the modules whereas each connection is created within a single level of module hierarchy.

The user describes the structure of the network simulation model in the *.NED* file using NEtwork Description (NED) language. The NED declares the include modules, their parameters, interconnections between modules and the network topology. We can assign the parameters in either the *.NED* files or the configuration file: *omnetpp.ini*, which customize module behavior and parameterize the model topology. The *omnetpp.ini* file includes settings that control the simulation

execution and values of the model parameters.

2.3.2 INET framework

The INET framework is a modular component library in OMNeT++ which provides set of applications, agents, protocols and models [17]. It supports network protocol implementation such as IP version 4/6, transmission control protocol, and user datagram protocol. Also includes link layer models such as Point-to-Point links, Ethernet, 802.11 models and node mobility model such as wireless and mobile simulations. Moreover, INET supports power models such as energy generation, energy consumption, energy storage and routing protocols as well.

INET is an open source framework which also uses the same module concept as OMNeT++. In our simulation, we set up a wireless network using IEEE 802.11 link which assembles with modules such as *WirelessHost* and *RadioMedium*.

2.4 Related works

The concept of WD protocol was introduced in [12] and they experimentally evaluated the delay associated with group formation procedures using the open source implementation of WD in Linux. [21] presents WD based application for real-time data transmission among wireless medical devices to transfer endoscope images to a remote terminal.

A comparison between DSRC and WD protocols are given in [9,10,22]. Out of them, [10] analyzes the performance of WD and DSRC over single-hop and multi-hop communication with the use of NS3. The authors evaluate the performance measurements such as throughput, end-to-end delay, and packet receiving/ loss ratios. They show that DSRC outperforms WD in VANETs. They claim, WD as an alternative communication technology for VANETs. In [9], the authors derive the transmission delay theoretically between DSRC and WD. They show that performance of the DSRC is superior since the re-transmission through the GO increases the transmission delay in WD. [22] analyzes the non-uniformly distributed back-off timer which based on the binomial distribution for DSRC and WD, and evaluates average throughput and collision probability. They show that binomial distribution in WD improves the network throughput. They also claim WD as a potential candidate for DSRC in VANETs.

It is obvious that DSRC is superior to WD in terms of performance due to having a dedicated spectrum allocation and a higher transmit power. Therefore, for safety critical applications, where reliability is paramount [1], WD may not

be a suitable alternative.

WD has few drawbacks with large delays in packet delivery being the most critical one with respect to VANETs, and [9, 13, 14, 23, 24] have focused on alleviating these issues. To this end, [13] addresses the drawback of large group formation time in WD by proposing a new group formation algorithm that reduces the time taken for the discovery phase. In this scheme, the devices share their RSSI values and IV in the discovery phase. They experimentally evaluate the network throughput and group formation time. They show that the proposed algorithm reduces the group formation time while improves the network throughput. [23] proposes a listen channel randomization scheme to reduce the delay in the discovery phase. In this scheme, the device can switch between the three social channels in the *listen* phase to avoid the highly interfering channel when it waits for other device's probe requests. They derive the discovery delay using a Markov chain-based model and claim a significant delay reduction compared to the legacy *listen* phase of WD.

In [14, 24], the authors propose a backup group owner for reducing group failures in WD. In [24], the authors claim that backup GO will be the device which connects to the GO first. However, this is not a fair assignment in a VANET. One of the main issues associated with this method is that the backup GO can only capture the group failure, and reestablish the group after GO leaves. Therefore, in [14], the GO delegates the work to the backup GO before it leaves the group, and avoids the renegotiation which takes time. The author of [14], elects the backup GO based on the IV, and considering additional parameters such as time in P2P group and velocity of the node. This scheme modifies the group negotiation phase of the standard WD protocol.

In [9], the authors propose reducing the large transmission delay through a broadcasting mechanism instead of the P2P method, which eliminates acknowledgments and retransmissions on the downlink. This is similar to the method used in the IEEE 802.11p standard, where retransmissions are avoided [25]. The authors in [9] present a theoretical analysis in terms of total delay, and validate their claim through numerical evaluations. This takes into consideration as the best scenario where there is no contention for the communication channel. They showed that the proposed method can achieve lower delay and facilitate a large number of devices to connect with the group compared to the standard WD protocol. However, to the best of our knowledge, there is no simulation study that covers larger transmission delay experienced in WD.

Chapter 3

System Model

This chapter defines the proposed system model for enhancing the WD protocol for data communication in VANETs. Firstly, we present the topological model in the Section 3.1. Then, in Section 3.2, we model the wireless channel which suits for VANETs. Finally, the communication model is explained in Section 3.3. This system model will be used throughout the thesis for analysis and evaluation.

3.1 Topological model

The performance of the system model is influenced by the scenario topology. It is necessary to consider the parameters such as the type of the scenario, the vehicular density, and the speed of vehicles when selecting a suitable topological model. In [26–29], different vehicular environments are considered including the highways, urban, suburban and rural scenarios. In this thesis, we select a highway scenario with more realistic parameters as suggested in [30].

We consider a bidirectional highway scenario with 3 lanes in each direction, as shown in Figure 3.1. The total width of the road is 23 m and the two-way traffic lanes are separated by a median strip of 0.5 m. We assume that vehicles drive with speeds of 80 km/h, 100 km/h and 120 km/h in each lane (as shown in Figure 3.1), and vehicles do not change lanes. The spacing between vehicles in the same lane is set according to the 2-second rule.

3.2 Channel model

The vehicular channel modeling plays an important role in the performance evaluation of V2V communication system [31]. The channel is modeled using two well-known models considering both large-scale and small-scale variation in the received signal. Subsection 3.2.1 explains the path loss model which describes the large-scale variation in the received signal. The fading model which describes the small-scale variation in the received signal is explained in Subsection 3.2.2.

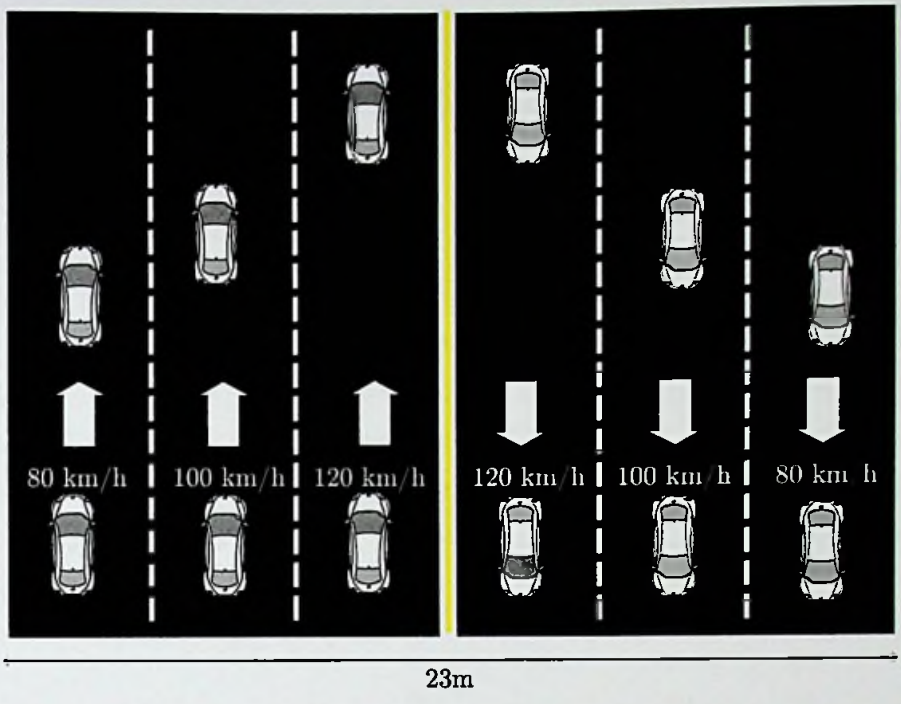


Figure 3.1: Highway scenario.

3.2.1 Path loss model

Path loss defines the variation of the received signal power with the separation distance between transmitter and receiver. According to the simplified path loss model with respect to a reference point x_0, received signal power is given by

$$P_R = P_T k \left(\frac{x_0}{x}\right)^{\gamma}$$

where P_T is the transmit power, k is a constant depending on the antenna characteristics, x is the distance between transmitter and receiver and γ is the path loss exponent. Based on the literature, it has been found that path loss exponent is a function of the carrier frequency, type of scenario, obstructions, etc [26]. The range of the path loss exponent has been derived for different scenarios such as highways, urban, suburban and rural using experimental evaluations [26, 27, 29, 32, 33]. Out of these, [26], [27], [32] show that the path loss exponent is lower than 2 in vehicular environments.

The highway scenario introduces significant signal attenuation and packet loss [34]. Therefore, we consider the signal attenuation when defining the path loss exponent. We use the Friis propagation model to capture path loss where $\gamma = 2$ [35]. Thus, the received signal power can be expressed as

$$P_R = P_T \left(\frac{G_T G_R \lambda^2}{16\pi^2 x^2} \right)$$

where G_T and G_R are the transmitter and the receiver antenna gains, respectively, λ is the wavelength of the signal.

3.2.2 Fading model

The fading model is used to capture the effect of multi-path propagation of radio waves between the transmitter and the receiver. The small-scale fading can be modelled using Rician or Rayleigh or Nakagami fading. In several studies, the Nakagami fading model has been found to be a suitable model for VANETs [36–40]. However, the Nakagami fading model can be simply transformed into either Rician or Rayleigh fading model by changing the fading depth parameter.

In the Nakagami model, the probability density function (PDF) of the amplitude of the received signal is given by

$$f(r : m, \Omega) = \frac{2m^m}{\Gamma(m)\Omega^m} r^{2m-1} \exp(-\frac{m}{\Omega} r^2),$$

where m is the fading depth parameter, Ω defines the average received power at a specific distance, and $\Gamma(.)$ is the Gamma function. The $m \in \{1, 3\}$, where $m = 1$ considers as high fading condition whereas $m = 3$ considers as low fading condition.

3.3 Communication model

We focus on the communication in a WD group. WD information dissemination within a group occurs only through the GO after periodic one-hop beacon broadcast by the GO. The key challenge related to this information dissemination is reducing the transmission delay. In our setup, the vehicles exchange information with data frames following the IEEE 802.11 standard. In the uplink, where the request-to-send/clear-to-send (RTS/CTS) mechanism is used to access the channel, each client sends its data to the GO using a P2P method, and acknowledgments are used to ensure successful packet delivery. In the downlink, the GO sends data frames back to its clients. We consider two communication models that differ from each other with respect to this downlink communication in the next subsections.

The performance measures are obtained considering one data transmission cycle. The data transmission cycle begins when the clients start to send data

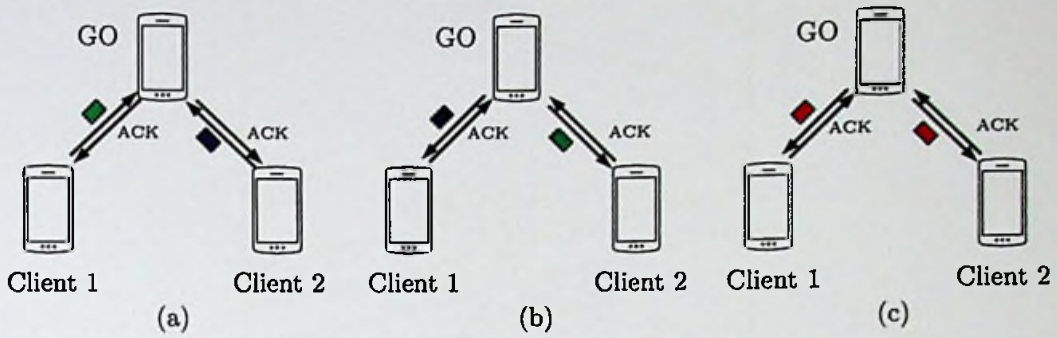


Figure 3.2: P2P model: (a) uplink where clients send its data frames to the GO (b) downlink where GO forwards the received clients data frames and (c) downlink where GO sends its own data frame to the clients.

frames to the GO on the uplink, and terminates after the GO finishes sending the data frames to its clients on the downlink.

3.3.1 Peer-to-Peer model

In the first model, the downlink communication based on a P2P method similar to the one used on the uplink. We call this the P2P model, and this methodology is currently being used in the standard WD protocol. According to this model, for a WD group of size n, $n-1$ clients send data frames to the GO on the uplink. In the downlink, GO forwards each data frame received from the clients to the remaining $n-2$ clients, and follows up by sending the data frame of the GO to the $n-1$ clients. This means, there are $(n-1)$ packet transmissions on the uplink and $(n-1)^2$ packet transmissions on the downlink.

Figure 3.2 shows the information dissemination in the P2P model for a group size of 3 which is containing two clients and the GO. Here, every data frame is acknowledged in both uplink and downlink. If not acknowledged the sender, retransmits the data frame until it receives the acknowledgment. In the downlink, the GO switches at least four times between the two clients to send the data frames.

3.3.2 Group owner broadcasting model

In the second model, the downlink communication based on a broadcast method, and we call this the group owner broadcasting (GOB) model. In this model, the GO aggregates the received data frames from the clients and the data frame of the GO into a single data frame, and broadcasts it to the $n-1$ clients. Note

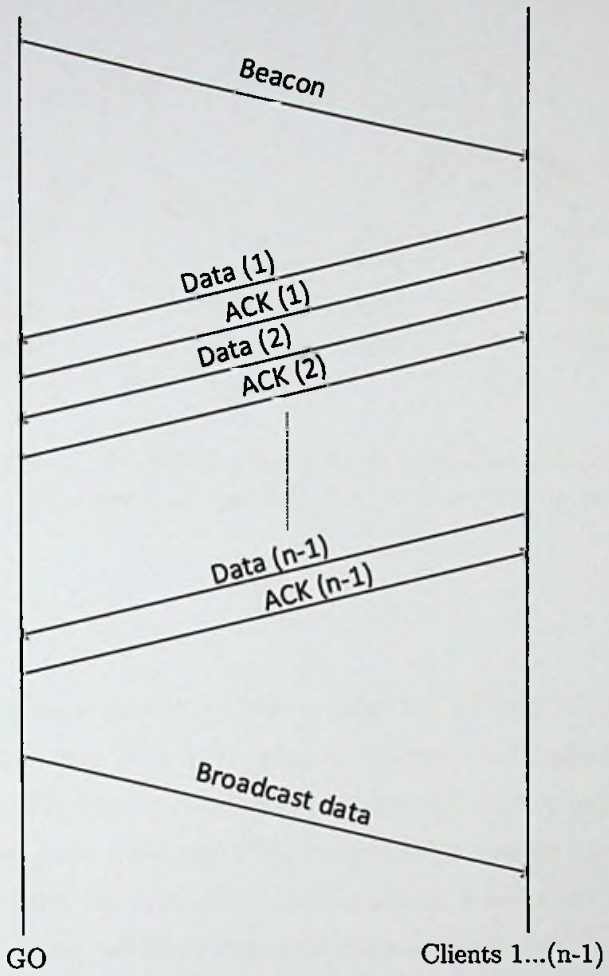


Figure 3.3: The information dissemination in the GOB model for the group size of n.

that acknowledgments are not used in this model to ensure successful reception of the broadcast packets on the downlink. Figure 3.3 shows the frame exchange sequences for information dissemination in the GOB model considering the group size of n.

Figure 3.4 shows the information dissemination in the GOB model for a group size of 3 which is containing two clients and the GO. Here, the uplink communication supports the P2P method whereas the downlink communication uses the broadcasting method. In the downlink, the GO aggregates all the data including its own data and all the received client's data into a single frame and broadcasts it to all clients.

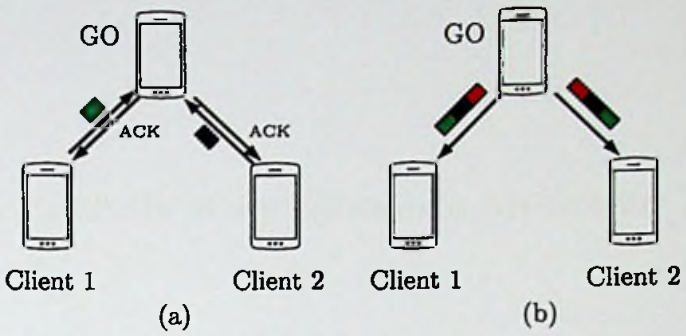


Figure 3.4: GOB model: (a) uplink where clients send its data frames to the GO (b) downlink where GO aggregates all the data into a single frame and broadcasts to the clients.

3.4 Summary

In this chapter, we have presented the system model which evaluate the performance of communication. We have used a bidirectional highway scenario as the topological model. We have established the channel model using two well-known models. Firstly, we have used the Friis propagation model to capture path loss. Secondly, we have used the Nakagami fading model to capture the effect of multipath propagation. Also, we have explained the P2P model and the GOB model, which will be analyzed in the next chapters.

Chapter 4

Theoretical Analysis of Performance Measures

In this chapter, we derive analytical expressions for the average total delay and the average energy consumption of the GO, considering both communication models. The analysis in this section is general, and in fact, it can be applied/extended to any ad-hoc network that utilizes WD for communication among groups. We note that the power budget may not be a critical constraint to a VANET. However, the derived results on energy consumption will be useful to many WD based ad-hoc networks. We will start the analysis by focusing on the average total delay in Section 4.1. In Section 4.2, we present the average energy consumption of the GO.

4.1 Average total delay

Firstly, we consider the behavior of a single station with a bidirectional discrete time Markov process $\{b(t), s(t)\}$. Where $b(t)$ be the stochastic process representing the backoff time counter and $s(t)$ be the stochastic process representing the backoff stage for a given station at time slot t [41].

Let

$$p_\tau = b_{(0,0)} \frac{1 - p_C^{z+1}}{1 - p_C} \tag{4.1}$$

be the probability that a station transmits a packet in a randomly chosen time slot, and

$$p_C = 1 - (1 - p_\tau^{n-1}) \tag{4.2}$$

be the collision probability. Where n is the number of stations, z is the station short retry count (SSRC), $b_{(0,0)}$ is the stationary distribution probability of the Markov chain at (0,0) state. The Equations 4.1 and 4.2 represent a nonlinear system which can be solved using numerical methods to obtain both p_τ and p_C.

Secondly, we consider the uplink, where each station transmits the data frames to the GO. Although the RTS/CTS mechanism is used to access the channel, there

can still be collisions among the stations due to hidden nodes and exposed nodes. Considering this fact, we obtain an expression for the average frame delay T_D, assuming that the frame drop probability is negligible since acknowledgments are used on the uplink. To this end, we can express

$$T_D = yT_L$$

where y is the average number of time slots required to successfully transmit a new frame, and T_L is the average length of a time slot which a station needs to detect the transmission of a data frame. To this end, we have

$$y = \frac{(1-2p_C)(w+1) + p_C w[1-(2p_C)^z]}{2(1-2p_C)(1-p_C)},$$

and

$$T_L = (1-p_{tr})T_E + p_{tr}p_S T_S + p_{tr}(1-p_S)T_C,$$

where w is the contention window, p_{tr} is the probability that at least one transmission occurs in the time slot of interest, T_E is the average duration of an empty slot and p_S is the conditional probability that a station achieves successful transmission given that the remaining $n-1$ stations remain silent [41]. Moreover, T_S is the average successful transmission time and T_C is the average collision time. Since all stations use the RTS/CTS access mechanism followed up by transmitting data, and an acknowledgment to confirm reception, p_{tr}, p_S, T_S and T_C can be written as

$$p_{tr} = 1-(1-p_\tau)^n,$$

$$p_s = \frac{np_t(1-p_t)^{n-1}}{1-(1-p_t)^n},$$

$$T_S = T_{RTS} + T_{CTS} + T_{DATA} + T_{ACK} + DIFS + 3(SIFS) + 4(\delta), \qquad (4.3)$$

and

$$T_c = T_{RTS} + DIFS + \delta,$$

where $DIFS$ is the DCF inter-frame space, $SIFS$ is the short inter-frame space, δ is the per packet propagation delay, T_i, $i \in \{RTS, CTS, DATA, ACK\}$, is the average time taken to transmit packet i, and given by $T_i = \text{Size}(i)/R$, where

Size(i) is the size of packet i and R is the average rate.

Finally, we formally present an expression for the average total delay of one data transmission cycle through the following theorem.

Theorem 4.1. *For a WD group of size n, the average total delay of a data transmission cycle is given by*

$$T_{P2P} = (T_S + T_D)(n-1) + T_S(n-1)^2$$

for the P2P model, and by

$$T_{GOB} = (T_S + T_D)(n-1) + DIFS + nT_{DATA} + \delta$$

for the GOB model.

Proof. The uplink is common for both models. In the uplink, $(n-1)$ clients send data frames to the GO, and due to contention and backoff, each client on the average takes $T_S + T_D$ time to successfully transmit a data frame to the GO. Therefore, the average time taken for uplink data transmission is $(T_S+T_D)(n-1)$.

When considering the downlink of the P2P model, the GO transmits the received $(n-1)$ data frames from the clients to the remaining $(n-2)$ clients, and then transmits the GO data frame to all the $(n-1)$ clients using the P2P method. This means, the GO does $(n-1)^2$ transmissions, each taking T_S on the average, and $T_S(n-1)^2$ on the average in total.

When considering the downlink of the GOB model, the GO waits for a DIFS interval, and broadcasts the aggregated data frame. Therefore, the average time taken for the downlink data transmission is $DIFS+nT_{DATA}+\delta$, which completes the proof. □

From Theorem 4.1, it is seen that the average total delay has a quadratic relationship with n in the P2P model, and a linear relationship in the GOB model. Also, we can show that the time taken for the P2P model is greater than the time taken for the GOB model, on the average, which we present through the following corollary.

Corollary 4.1. *For $n > 2$, $T_{P2P} > T_{GOB}$.*

Proof. From Equation 4.3,

$$T_S > T_{DATA} + DIFS + \delta.$$

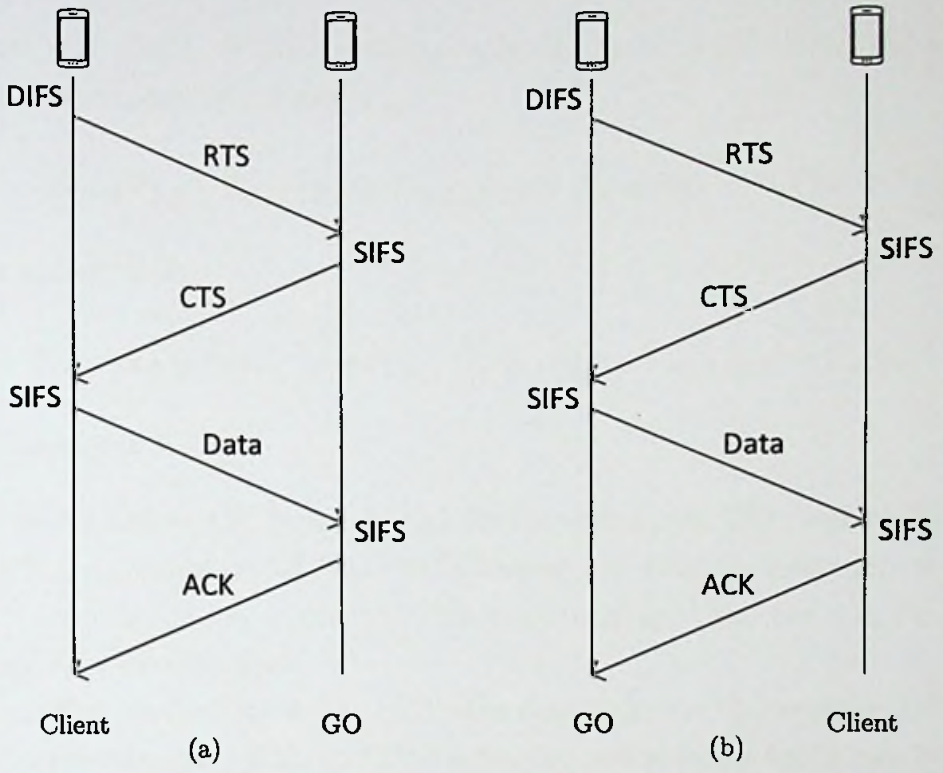


Figure 4.1: (a) Uplink and (b) downlink frames exchanged inside the group in the standard WD protocol.

Since $(n-1)^2 > n$ for $n > 2$, then

$$\begin{aligned}(n-1)^2 T_S &> n(T_{DATA} + DIFS + \delta), \\ &> nT_{DATA} + DIFS + \delta,\end{aligned}$$

which completes the proof. □

Having obtained an expression for the average total delay, we will look into the energy consumed at the GO, which is an important performance measure in an ad-hoc network, in the next section.

4.2 Average energy consumption

A station consumes energy to transmit and receive data frames, to sense the channel, and to switch from being a transmitter to a receiver or vice versa. Let, P_{TX}, P_{RX} and P_{SW} denote the power consumption at the transmitter, power consumption at the receiver and power consumption for switching, respectively. With these notations, we first obtain expressions for the average energy consumed

on the uplink and the downlink of the P2P model.

Lemma 4.1. *Let t_s be the switching interval. In the P2P model, the average energy consumption of the GO is*

$$E_{UL} = T_{RTS}P_{RX} + T_{CTS}P_{TX} + T_{DATA}P_{RX} + T_{ACK}P_{TX} + 4t_sP_{SW} + T_DP_{RX}$$

on the uplink, and

$$E_{DL} = T_{RTS}P_{TX} + T_{CTS}P_{RX} + T_{DATA}P_{TX} + T_{ACK}P_{RX} + 4t_sP_{SW}$$

on the downlink.

Proof. According to the Figure 4.1 (a), in the uplink, the GO receives RTS and DATA frames, transmits CTS and ACK frames, and switches four times between being a transmitter and a receiver. This gives us E_{UL}, where the $+T_DP_{RX}$ term accounts for the contention.

According to the Figure 4.1 (b), in the downlink, the GO receives CTS and ACK frames, transmits RTS and DATA frames, and switches four times between being a transmitter and a receiver, which gives us E_{DL}. □

Using the expressions in the above lemma, and considering both communication models, we formally present expressions for the average energy consumption of the GO through the following theorem. Proof can be obtained following the similar arguments to the ones made in the proof of Theorem 4.1.

Theorem 4.2. *For a WD group of size n, the average energy consumption of the GO for a data transmission cycle is given by*

$$E_{P2P} = (n-1)E_{UL} + (n-1)^2E_{DL}$$

for the P2P model, and by

$$E_{GOB} = (n-1)E_{UL} + nT_{DATA}P_{TX}$$

for the GOB model.

Proof. The uplink is common for both models. In the uplink, $(n-1)$ clients send data frames to the GO. From Lemma 1, GO on the average consumes E_{UL} energy to successfully receive a data frame from the client. Therefore, the average energy consumption of the GO for uplink data transmission is $E_{UL}(n-1)$.

We first consider the downlink of the P2P model. GO transmits the received $(n-1)$ data frames from the clients to the remaining $(n-2)$ clients, and then transmits the GO data frame to all the $(n-1)$ clients using the P2P method. This means, the GO does $(n-1)^2$ transmissions, each consuming E_{DL} on the average from Lemma 1, and $E_{DL}(n-1)^2$ on the average in total.

When considering the downlink of the GOB model, the GO broadcasts the aggregated data frame. Therefore, the average energy consumption of the GO for the downlink data transmission is $nT_{DATA}P_{TX}$, which completes the proof. □

It can be seen that similar to the total delay, from Theorem 4.2, the average energy consumption of the GO has a quadratic relationship with n in the P2P model, and a linear relationship in the GOB model. Also, we can show that E_{GOB} is always lower than E_{P2P} on the average. This result is formally presented through the following corollary.

Corollary 4.2. *For $n > 2$, $E_{P2P} > E_{GOB}$.*

Proof. From Lemma 4.1,

$$E_{DL} > T_{DATA}P_{TX}.$$

Since $(n-1)^2 > n$ for $n > 2$, then

$$(n-1)^2 E_{DL} > nT_{DATA}P_{TX},$$

which completes the proof. □

4.3 Summary

Having established the system model in the previous chapter, we now analyzed the performance measures theoretically only considering the communication model in Section 3.3. In this chapter, we have derived the average total delay and average energy consumption of the GO for one data transmission cycle considering both communication models.

We have obtained the average total delay considering backoff and contention for the communication channel. We have guaranteed that time taken for the GOB model is always less than the time taken for the P2P model. We have shown that energy consumption of the GO in the GOB model is always less than the P2P model.

Chapter 5

Simulation Environment and Setup

The theoretical analysis in the previous chapter was done only considering the communication model in Section 3.3. In this chapter, we focus on simulations, so that we can see the effects of the topology and the channel on performance. We will first look at the simulation environment in Section 5.1. Next, we present the simulation setup in Section 5.2.

5.1 Simulation environment

In this section, we discuss the setting up of the simulation environment using the INET framework of OMNeT++ [17]. Firstly, we assign the initial position and the mobility of the vehicular nodes in the *omnetpp.ini* configuration file according to the topological model. Then, we modify the *Ieee80211ScalarRadioMedium* module to implement the path loss and the fading models. Finally, we modify the *Ieee80211MgmtSTAWifiDirect* module [16] in the *management* layer to set up the two communication models in the Section 3.3.

We summarize the algorithm implementation in setting up the the communication models in Algorithm 1. The GO periodically sends beacons for information dissemination within the group. When clients receive the beacon, they start to send data frames to the GO. When the GO receives a data frame successfully, it checks whether the received data frame is from a client in its group, based on *clientList*. If not, the GO neglects the received data frame. If the frame is of relevance, it is decapsulated and sent to the upper layer while a copy of the received data frame is stored in the *dataQueue*. The GO keeps track on whether every client in the group has sent data frames to the GO or not. On successful reception of data frames from all clients, the *Timer* is canceled, and the GO initiates the data transmission on the downlink. The downlink data transmission will initiate when the *Timer* reaches the *GOTimeout* as well. A timeout is used to address the case of clients not receiving a beacon.

In the downlink of the P2P model, the GO forwards each frame in *dataQueue*

Algorithm 1 Data communication Algorithm for GO

```
1: n ← Size of the WD group
2: clientList ← List of clients
3: dataQueue ← Queue of client's data frames
4: GOTimeout ← Timeout for the GO
5: Top:
6: Send beacons
7: Timer ← 0
8: Start Timer
9: Uplink:
10: if successful data frame reception then
11:     if tx address of data frame in clientList then
12:         Store data frame in dataQueue
13:         if size of dataQueue = size of clientList then
14:             Cancel Timer
15:             goto Downlink.
16:         if Timer = GOTimeout then
17:             goto Downlink.
18:     else
19:         Delete the received data frame
20: Downlink:
21: if P2P model then
22:     i ← 0
23:     while i < size of clientList do
24:         Get i^th data frame from the dataQueue
25:         for each client j in clientList do
26:             if i^th frame Tx address ≠ j^th address then
27:                 Set frame Rx address to j^th address
28:                 Send data frame
29:         i ← i + 1
30:     for each client j in clientList do
31:         Set frame Rx address to j^th address
32:         Send data frame
33:     goto Top.
34: if GOB model then
35:     Create a data frame and set frame body
36:     Aggregate data frames stored in dataQueue
37:     Set frame Rx address to broadcast
38:     Send data frame
39:     goto Top.
```

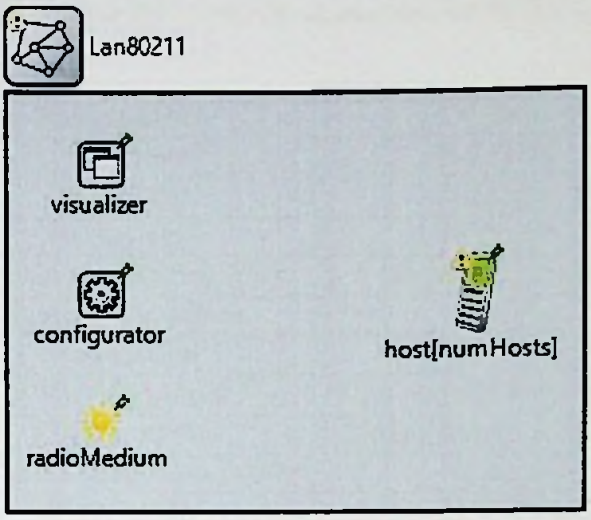


Figure 5.1: 802.11 LAN module in INET framework of OMNeT++.

to each client in *clientList*, sequentially. Then, the GO transmits the GO's data frame to each client in *clientList*, sequentially. In the downlink of the GOB model, the GO aggregates all data frames stored in *dataQueue* and its own data into one data frame, and broadcasts the frame.

5.2 Simulation setup

The simulation setup is configured according to the system model presented in Chapter 3. Firstly, we discuss the network configuration of the simulation setup in Subsection 5.2.1. Then, we present the channel model and device configuration in the Subsection 5.2.2 and Subsection 5.2.3 respectively.

5.2.1 Network configuration

A wireless 802.11 standard local area network (LAN) is used as network architecture for the simulation. Figure 5.1 shows 802.11 LAN module in INET framework of OMNeT++. It mainly consists of radio medium and wireless hosts modules. The radio medium represents the vehicular channel model which will be briefly explained in Subsection 5.2.2 and also the host represents a vehicular node in VANETs which will be explained in Subsection 5.2.3.

Table 5.1 presents the main network configuration values for simulation setup in OMNeT++. The WD protocol is based on the IEEE 802.11b standard, *i.e.*, a data rate of 6 Mbps, and a frequency of 2.4 GHz. The GO transmits beacons periodically at a rate of 5 packets per second. The transmit power is 2

Table 5.1: System parameters for WD.

Parameters	Value
Frequency	2.4 GHz
Data rate	6 Mbps
Packet size	40 Bytes
Beacon generation rate	5 packets per second
Transmitter power	2 mW
Modulation	BPSK
Path loss model	Friis propagation model
Fading model	Nakagami m $\in$ {1,3 }
Minimum reception power	-85 dBm
Noise power	-110 dBm

Table 5.2: MAC configuration values for WD.

Parameters	Value
DIFS	50 μs
SIFS	10 μs
RTS	20 bytes
CTS	14 bytes
ACK	14 bytes
Data	40 bytes

mW, and the *state-based energy consumed* module, which is implemented in the INET framework, is used to determine the energy consumption over time. In the simulated highway scenario, all vehicular nodes are configured according to the *linear mobility* model, which is also inbuilt in the INET framework. The maximum communication range is observed to be 250 m. We have set the path loss exponent as 2 [26]. The fading intensity in the Nakagami model is set to 1 or 3 to consider low and high fading environments in the simulations. The minimum reception power and background noise power of the receiver are configured to be -85 dBm and -110 dBm, respectively, at the physical (PHY) layer. More details about the vehicular node is discussed in Subsection 5.2.3.

MAC layer deploys carrier sense multiple access with collision avoidance (CSMA/CA) for every vehicular node in the group except for the broadcast frame. Table 5.2 presents the MAC configuration parameters of the WD protocol. With respect to the MAC layer, we set DIFS, SIFS as 50 μs and 10 μs, respectively.

The size of packets is set as follows: RTS = 20 bytes, CTS = 14 bytes, DATA = 40 bytes, and ACK = 14 bytes. Note that the maximum transmission unit (MTU) for IEEE 802.11 standard is 2300 bytes. This means, data intended for around 50 clients can be aggregated into the broadcast packet without any major issues.

5.2.2 Channel modeling

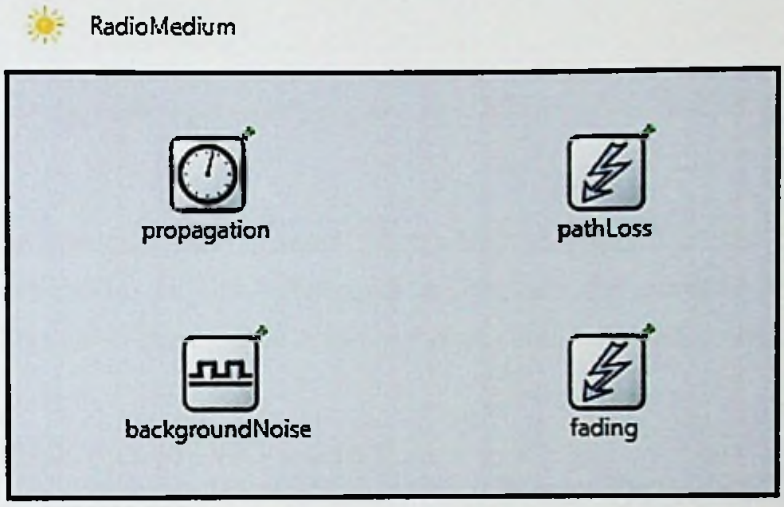


Figure 5.2: Channel model for WD in INET framwork of OMNeT++.

The channel is configured according to the channel model described in Section 3.2 where Figure 5.2 shows the vehicular channel model in the simulation setup. It basically includes path loss model, fading model, propagation model, and background noise model. Figure 5.3 shows the configuration parameters of the channel model in the INET framework.

We use Friis propagation model to capture path loss and Nakagami fading model to capture multipath fading. In addition, the constant speed propagation is used to design the propagation model which computes the propagation time as the ratio between the traveled distance over the propagation speed. The background noise model accounts the thermal noise and other fluctuations of the electromagnetic field. Both background noise power and minimum interference power are configured to -110 dBm.

5.2.3 Device configuration

A device which is a wireless host module can configure as a station, access point (AP) or ad-hoc node in INET framework of OMNeT++. We use the implemen-

```
module Ieee80211RadioMedium extends RadioMedium
{
    parameters:
        propagationType = default("ConstantSpeedPropagation");
        pathLossType = default("FreeSpacePathLoss");
        fadingType = default("NakagamiFading");
        backgroundNoise.power = default(-110dBm);
        mediumLimitCache.carrierFrequency = default(2.4GHz);
        mediumLimitCache.minReceptionPower = default(-85dBm);
        mediumLimitCache.minInterferencePower = default(-110dBm);
}
```

Figure 5.3: Configuration parameters of the channel model in the INET framework of OMNeT++.

tation of the station as the wireless host which configures to the IEEE 802.11 infrastructure mode. In this subsection, we explain the network interface card (NIC) module, mobility module and the energy consumption module of a vehicular node.

5.2.3.1 IEEE 802.11 NIC module

Figure 5.4 shows the IEEE 802.11 NIC which consists of four layers: *agent, management, MAC,* and *radio* layer. The classifier is the module that is used to calculate quality of service (QoS) parameters. The *IEEE80211AgentSTA* module is used as the *agent* layer which allows the user to control the behaviour of a station in the network. The *agent* layer controls the channel scanning, the authentication, and the association by sending commands to the management layer. The *agent* layer configuration parameters as shown in Figure 5.5, are set as follows: probe request delay = 0.1 s, minimum time for scanning the each channel = 0.15 s, maximum time for scanning the each channel = 0.3 s, timeout for the authentication and association procedure = 5 s.

The *management* layer performs the commands which are received from the *agent* layer, by exchanging beacons, probe request/ response, authentication, and association frames with *MAC* layer and reports the results back to the *agent* layer. Moreover, the *management* layer performs encapsulation when sending the data frames through the channel and decapsulation when sending the data frames to the upper (*agent*) layer. The *management* layer can configure as *IEEE80211MgmtAP*, *IEEE80211MgmtAdhoc*, *IEEE80211MgmtSTA* modules where the host acts as AP, ad-hoc node or station respectively. Here, the *management* layer is implemented using the *IEEE80211MgmtSTAWifiDirect* module

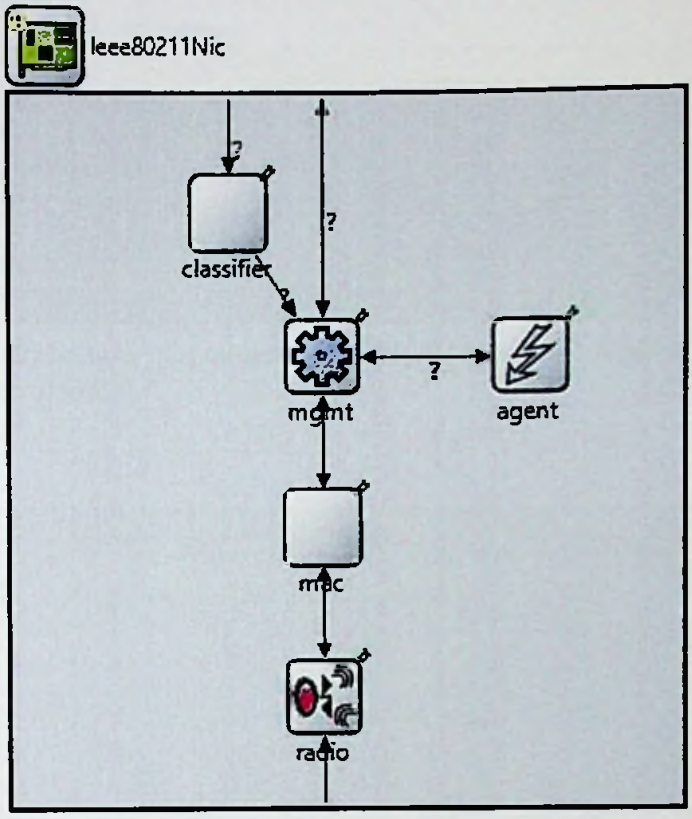


Figure 5.4: IEEE 802.11 NIC layered architecture in INET framework of OMNeT++.

by extending the *IEEE80211MgmtSTA* module [16]. This module implements the WD group formation phases such as scan, discovery, negotiation and the authentication phase. Note that, we modify the *IEEE80211MgmtSTAWifiDirect* module to set up the two communication models in the Section 3.3. The *management* layer configuration parameters as shown in Figure 5.6, are set as follows: the number of frames exchange in the authentication process = 4, whether the host supports WD or not, and whether the host configures as the GO or the client.

The IEEE 802.11 *MAC* layer performs the transmission of data/management frames received from upper layer based on the CSMA/CA protocol to access the channel. The *MAC* layer configuration parameters as shown in Figure 5.7, are set as follows: maximum queue size = 14, retry limit = 7, minimum contention window for data = 7, and minimum contention window for broadcast frame = 31.

The *radio* layer is a part of the IEEE 802.11 *PHY* layer model which exchanges the messages between *MAC* layer and radio interface. The *radio* layer is capable of transmitting and receiving the signals through the channel. It contains antenna

```
**.wlan[*].agent.activeScan = true
**.wlan[*].agent.default_ssid = ""
**.wlan[*].agent.channelsToScan = ""
**.wlan[*].agent.probeDelay = 0.1s
**.wlan[*].agent.minChannelTime = 0.15s
**.wlan[*].agent.maxChannelTime = 0.3s
**.wlan[*].agent.authenticationTimeout = 5s
**.wlan[*].agent.associationTimeout = 5s
```

Figure 5.5: Configuration parameters of the *agent* layer on IEEE 802.11 NIC in the INET framework of OMNeT++.

```
**.host[*].wlan[*].mgmt.numAuthSteps=4

**.host[0].wlan[0].mgmt.WiFiDirectUsed=true
**.host[0].wlan[0].mgmt.WiFiDirectGO=true
**.host[0].wlan[0].mgmt.strGroup="Wifi Direct Group"

**.host[*].wlan[1].mgmt.WiFiDirectUsed=true
**.host[*].wlan[1].mgmt.WiFiDirectGO=false
**.host[*].wlan[1].mgmt.strGroup="Wifi Direct Group"
```

Figure 5.6: Configuration parameters of the *management* layer on IEEE 802.11 NIC in the INET framework of OMNeT++.

```
**.mac.address = "auto"
**.mac.maxQueueSize = 14
**.mac.rtsThresholdBytes = 3000B
**.wlan[*].mac.retryLimit = 7
**.wlan[*].mac.cwMinData = 7
**.wlan[*].mac.cwMinBroadcast = 31
```

Figure 5.7: Configuration parameters of the *MAC* layer on IEEE 802.11 NIC in the INET framework of OMNeT++.

```
**.wlan[*].radio.transmitter.power = 2mW
**.wlan[*].radio.transmitter.bitrate = 6Mbps
**.wlan[*].radio.transmitter.headerBitLength = 100b
**.wlan[*].radio.transmitter.carrierFrequency = 2.4GHz
**.wlan[*].radio.transmitter.bandwidth = 20MHz
**.wlan[*].radio.receiver.sensitivity = -85dBm
**.wlan[*].radio.receiver.sinrThreshold = 4dB
```

Figure 5.8: Configuration parameters of the *radio* layer on IEEE 802.11 NIC in the INET framework of OMNeT++.

```
**.host[*].mobilityType = "LinearMobility"
**.host[*].mobility.updateInterval = 10ms
**.host[*].mobility.leaveMovementTrail = true

**.host[*].mobility.initialX = 101.875m
**.host[*].mobility.initialY = 1000m
**.host[*].mobility.initialZ = 0m

**.host[*].mobility.speed = 22.2222mps
**.host[*].mobility.angle = 270deg
**.host[*].mobility.acceleration = 0
```

Figure 5.9: Configuration parameters of the mobility module in the INET framework of OMNeT++.

model, transmitter model, receiver model and energy consumed model. The *radio* layer configuration parameters as shown in Figure 5.8, are set as follows: transmit power = 2 mW, bit rate = 6 Mbps, header bit length = 100 bits, carrier frequency = 2.4 GHz, bandwidth = 20 MHz, receiver sensitivity = -85 dBm, and signal-to-interference-plus-noise ratio (SINR) threshold = 4 dB.

5.2.3.2 Mobility module

The mobility of the nodes are configured according to the topological model explained in Section 3.1. All vehicular nodes are configured according to the *linear mobility* module, which is inbuilt in the INET framework [17] where we can configure parameters such as speed, angle and acceleration of vehicular nodes in the *omnetpp.ini* file. Figure 5.9 shows the configuration parameters of the mobility module in the *omnetpp.ini* file. The initial positions of the nodes are given by considering both node velocity and the 2-second rule spacing between the nodes.

5.2.3.3 Energy consumed module

The energy consumption of the vehicular node over time when transmitting or receiving signal configures using the *state based energy consumed* module implemented in INET framework [17] of OMNeT++. Figure 5.10 shows the configuration parameters of the energy consumed module in the *omnetpp.ini* file. We configure the power consumption at the transmitter, power consumption at the receiver and power consumption for switching, to 0.1 W, 0.01 W and 0.001 W respectively.

```
*.host*.wlan[0].radio.energyConsumerType = "↩
   StateBasedEnergyConsumer"
*.host*.wlan[0].radio.energyConsumer.offPowerConsumption = 0mW
*.host*.wlan[0].radio.energyConsumer.sleepPowerConsumption = 1mW
*.host*.wlan[0].radio.energyConsumer.switchingPowerConsumption = 1↩
   mW
*.host*.wlan[0].radio.energyConsumer.↩
   receiverReceivingPowerConsumption = 10mW
*.host*.wlan[0].radio.energyConsumer.↩
   transmitterTransmittingPowerConsumption = 100mW
```

Figure 5.10: Configuration parameters of the energy consumed module in the INET framework of OMNeT++.

5.2.3.4 Timeout for GO to initiate data transmission

One of the main concerns associated with the wireless communication is beacon loss, *i.e.*, a client of the group will not receive the beacon from the GO, and hence will not send the data frame to the GO. This causes the GO to not transmit the received data frames on the downlink until the next beacon timeout. To address this issue, a timeout for GO is introduced to initiate transmitting data on the downlink. This allows the GO to wait sufficient time until every client of its group sends data frames to the GO.

For a given group size, the GO timeout value given in the algorithm is set equal to the average delay on the uplink. The average delay on the uplink is found through prior simulations. To this end, average delay on the uplink is simulated for different values of n, and they are fitted using the goodness of fit as shown in Figure 5.11. The line of best fit is then used to set the GO timeout for any given n. An alternate theoretical approach similar to the one taken in Chapter 4 can be used to set this timeout value as well. We note that the simulation has only been set up for a basic highway scenario. Further improvements can be made using a traffic simulator like SUMO to provide additional evaluations such as the performance in urban scenarios.

5.3 Performance evaluation

In the simulation, the performance evaluation is obtained by considering the following performance measures:

1. Average total delay.

2. Average energy consumption of the GO.

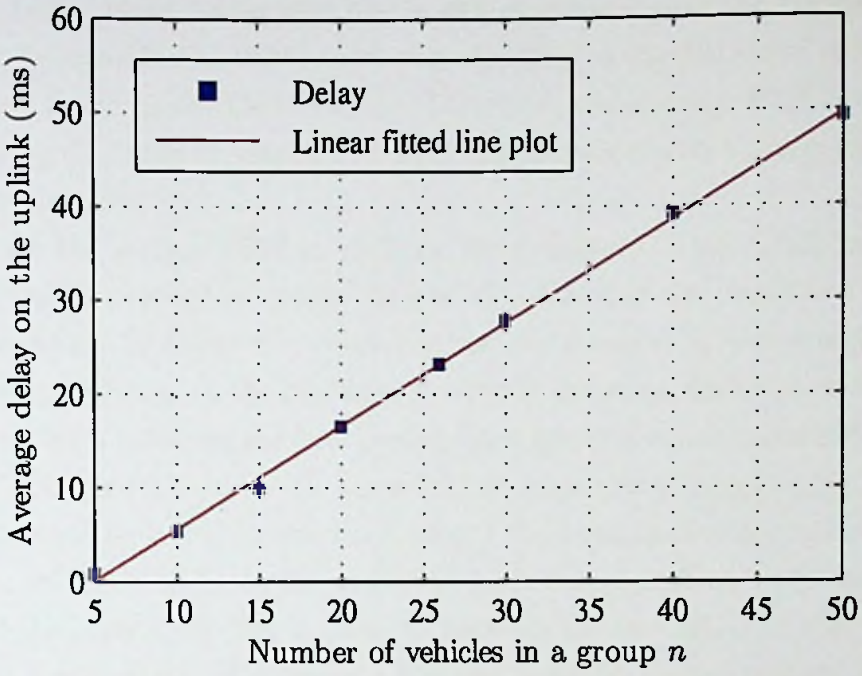


Figure 5.11: Average delay on the uplink with respect to the number of vehicles in a group.

3. Average packet loss ratio (PLR).

4. Average packet reception ratio (PRR).

The performance measures such as average total delay, average energy consumption of the GO and average PLR are evaluated and compared for the P2P and the GOB models. The average PRR is used to evaluate the reliability of the GOB model. These measures are obtained considering one data transmission cycle. The data transmission cycle begins when the clients start to send data frames to the GO on the uplink, and terminates after the GO finishes sending the data frames to its clients on the downlink.

The average total delay is the average transmission time to exchange the data frames within the group for one data transmission cycle. The average total delay for a given number of vehicles is obtained after all the vehicles have joined the group as the clients. Hence, the group formation delay is negligibly small.

The GO consumes energy to transmit and receive data frames, to sense the channel, and to switch from being a transmitter to a receiver or vice versa. The average energy consumption of the GO is obtained for one data transmission cycle, similar to the average total delay.

The average PLR is the percentage of lost data packets over one data transmission cycle. Note that packet loss is still possible due to the failure of reception the beacons in the P2P model and also due to the failure of receiving the broadcast frame in the GOB model. Therefore, the average PLR is calculated considering the losses of beacon and broadcast frames within a group for one data transmission cycle.

We use the average PRR to evaluate the reliability of the GOB model. The PRR is the percentage of clients successfully receiving the broadcast frame, on the downlink. To study the effect further, we consider a test scenario where there are two groups on the highway, moving in the same direction such that the second group is following the first group. Each group is communicating according to the GOB model, and hence, there will be inter group interference. The idea is to evaluate how this interference affects the broadcast communication. We consider the GOs to be at the center of each group, and we only consider the PRR of the clients who are located in between the two GOs, as they are more prone to interference. We simulate the test scenario for the beacon generation rate of 5 packets per second and 10 packets per second under different fading intensities. There is no synchronization between the two GOs, and hence, there will be interference in general.

Having obtained the performance measures to compare the performance of the two communication models, we will look into the results in the next chapter.

5.4 Summary

In this chapter, we have discussed the algorithm implementation and the simulation setup on OMNeT++. We have extended the WD implementation in the INET framework of OMNeT++ [16] to suit our requirement. In particular, we have modified the *Ieee80211MgmtSTAWifiDirect* module in the *management* layer to set up the two communication models and *Ieee80211ScalarRadioMedium* module to implement the channel model. We have provided the simulation setup based on the system model which is described in the Chapter 3. Also, we have presented the performance measures to evaluate both communication models.

Chapter 6

Results and Discussion

In this Chapter, we present simulation results for validation of proposed technique and the derived theoretical results, and to draw further insights. Averaging is done over 100 independent instances. In the figures, P2P-T and GOB-T represent the results obtained from the theoretical analysis, and P2P-S and GOB-S represent the simulation results, of the two communication models. Nak-m represents Nakagami fading with fading depth parameter m. Since our main objective is improving the performance of WD, a comparison with DSRC is not provided. The performance of DSRC will be definitely superior to WD due to the dedicated spectrum allocation and the higher transmit power. A comparison between the two technologies in terms of performance can be found in [10].

6.1 Average total delay

Although WD may not be best suited for safety critical applications, we focus on cases where the average delay is less than 100 ms, which is the maximum tolerable delay for a safety critical application [42].

Figure 6.1 shows how the average total delay theoretically changes with the number of vehicles in a group. We can observe that the theoretical results obtained in our work take a much larger average total delay for one data transmission cycle compared to the previous work in [9]. This is because contention for the channel is also considered in the analysis unlike [9]. We can also observe that the average total delay increases more rapidly with the number of vehicles in the P2P model compared to the GOB model. Furthermore, when comparing the average total delay for both models at $n = 15$, it can be seen that the theoretical results obtained in our work take around 59 ms and 20 ms for the P2P model and the GOB model respectively, whereas around 41 ms and 4 ms with the previous work in [9]. *Therefore, the theoretical analysis in our work shows a much larger total delay for one data transmission cycle for both communication models compared to the previous work in [9].*

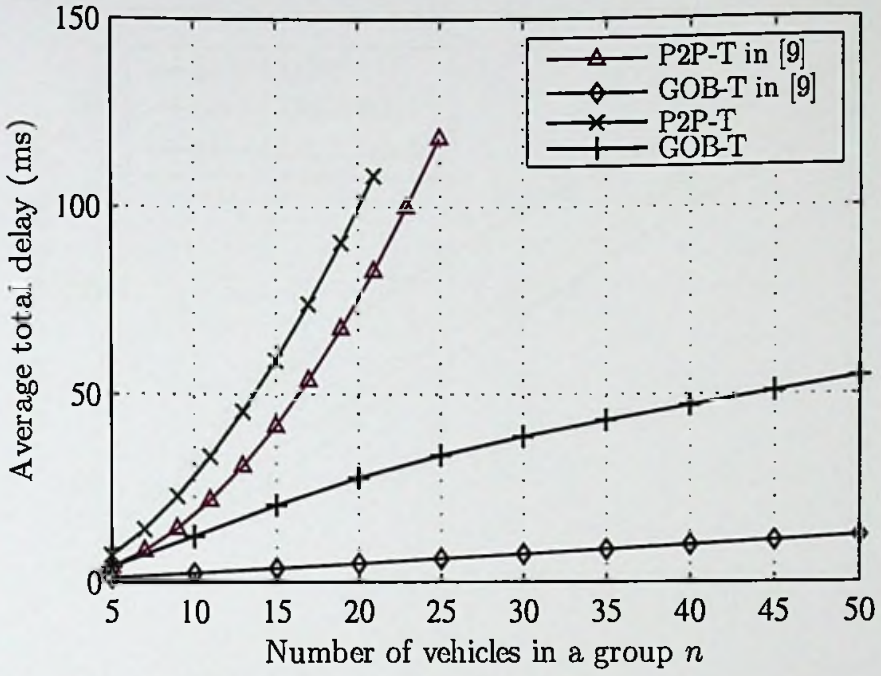


Figure 6.1: The theoretical analysis of average delay with respect to the number of vehicles in a group.

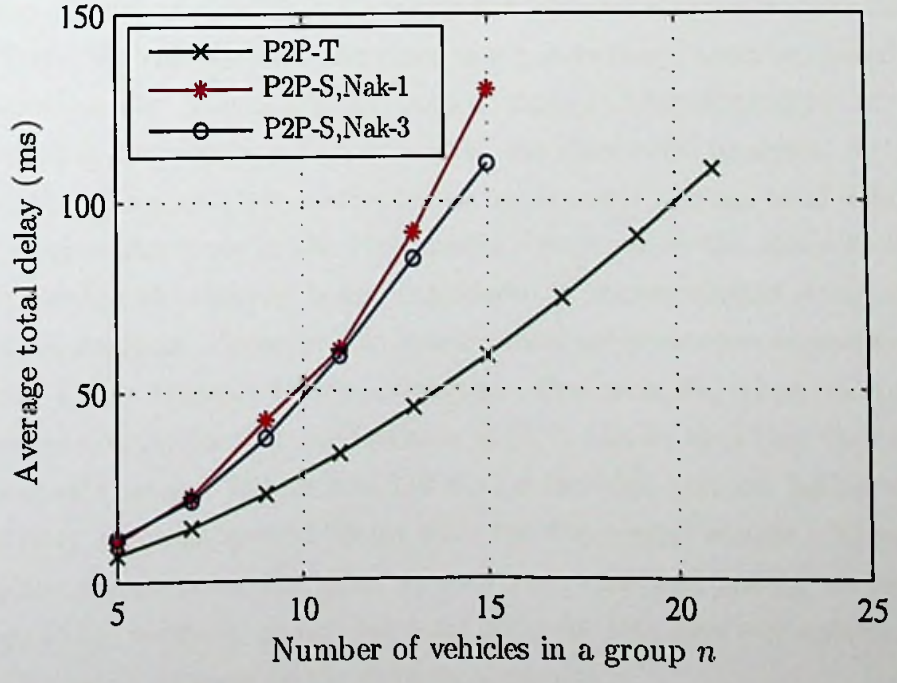


Figure 6.2: The behaviour of average delay with respect to the number of vehicles in a group for P2P model considering channel and topological models.

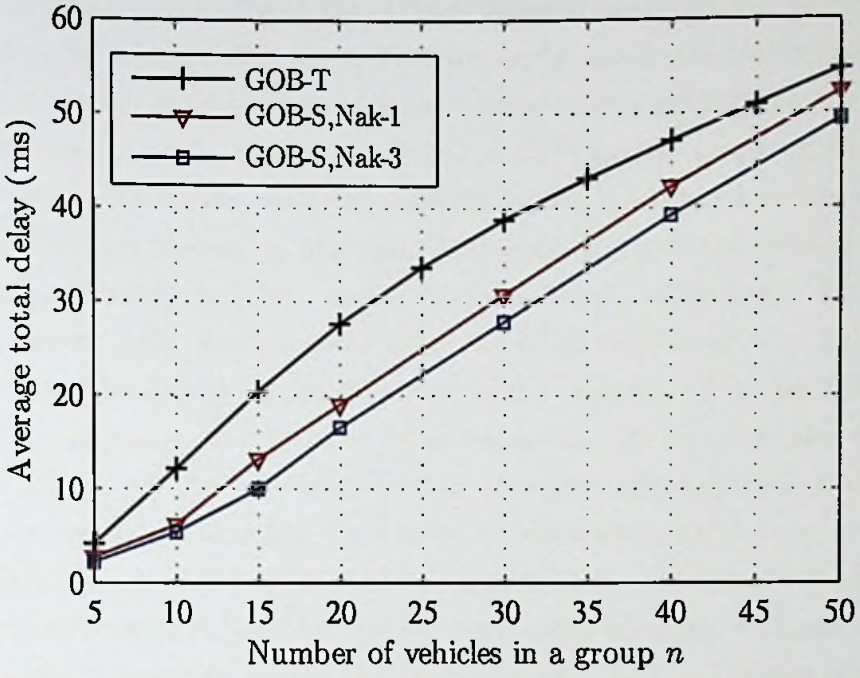


Figure 6.3: The behaviour of average delay with respect to the number of vehicles in a group for GOB model considering channel and topological models.

Figure 6.2 presents how the average total delay in the P2P model, changes with the number of vehicles in a group when channel and topological models are considered. We can observe that there is a gap between the theoretical and the simulation results. This is because both channel and topological models are also considered in the simulation study, unlike the theoretical analysis. We can also observe that the simulation results take a much larger average total delay for one data transmission cycle in the P2P model compared to the theoretical results. This is because the timeout is also considered in the simulation study unlike the theoretical analysis. Moreover, the average total delay shows comparatively small variation under different fading intensities. Furthermore, when comparing the average total delay for P2P models at $n = 15$, it can be seen that the simulation results obtain around 130 ms and 110 ms for the high and low fading intensities respectively, whereas around 59 ms with the theoretical results. *Therefore, the simulation results show the effect of both path loss and fading, as well as the topology of the network, on average total delay for one data transmission cycle in the P2P model compared to the theoretical results.*

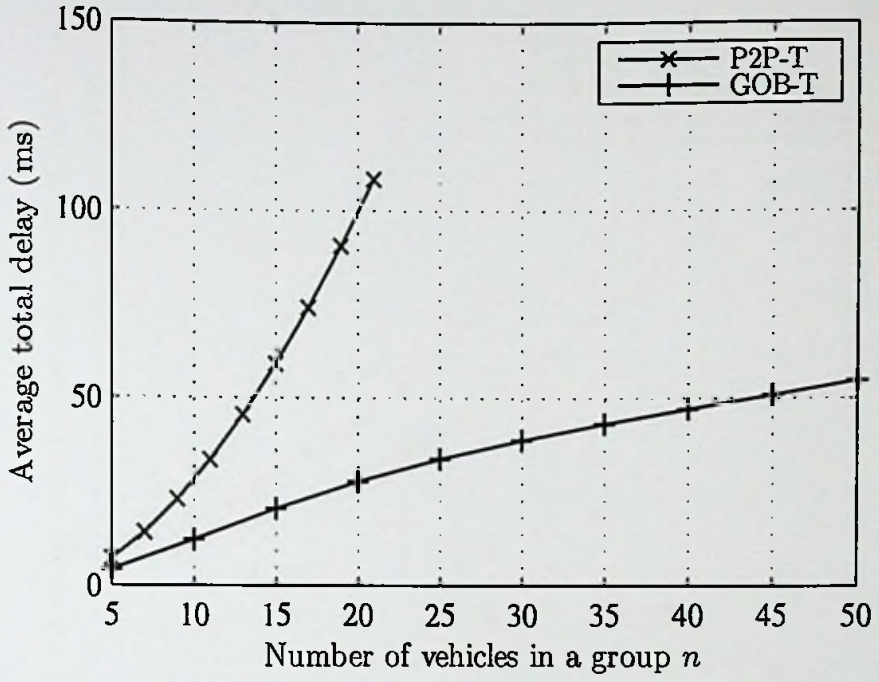
Figure 6.3 presents how the average total delay in the GOB model, changes with the number of vehicles in a group. We can observe that there is a gap be-

tween the theoretical and the simulation results which tends to decrease with the increase of the number of vehicles. This is because both channel and topological modes are also considered in the simulation study, unlike the theoretical analysis. We can also observe that the theoretical results take a much larger average total delay for one data transmission cycle in the P2P model compared to the simulation results. This is because the timeout is also considered in the simulation study, unlike the theoretical analysis. Moreover, the average total delay shows comparatively small variation under different fading intensities. Furthermore, when comparing the average total delay for GOB models at $n = 15$, it can be seen that the simulation results obtain around 13 ms and 10 ms for the high and low fading intensities respectively, whereas around 20 ms with the theoretical results. *Therefore, the simulation results show the effect of both path loss and fading, as well as the topology of the network, on average total delay for one data transmission cycle in the GOB model compared to the theoretical results.*

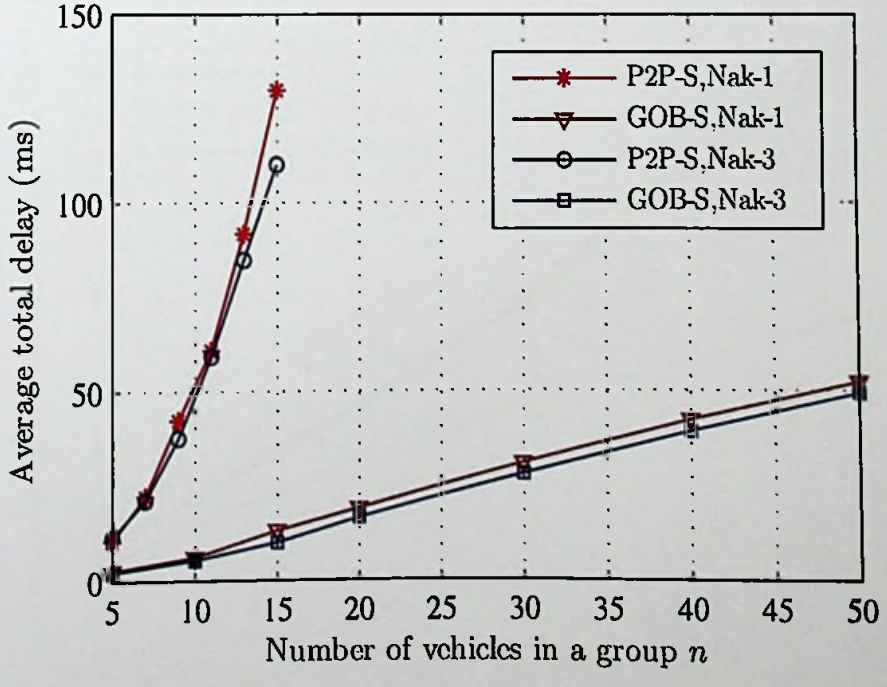
Figure 6.4 presents how the average total delay changes with the number of vehicles in a group, considering fading intensities 1 and 3. We can observe that the P2P model takes a significantly larger average total delay for one data transmission cycle compared to the GOB model. We can also observe that the average total delay increases more rapidly with the number of vehicles in the P2P model compared to the GOB model. Note that, 50 is the maximum number of vehicles that can directly communicate with the GO according to the selected radio range and the 2-second spacing between vehicles. The average total delay shows comparatively small variation under different fading intensities. Furthermore, when comparing with previous work in [9] (as shown in Figure 6.1), it can be seen that the theoretical results are more closer to the simulated scenario. The existing gap between the theoretical and the simulation results can be further reduced by fine tuning the timeout value. *Therefore, the P2P model takes a much larger average total delay for one data transmission cycle compared to the GOB model.*

6.2 Average energy consumption of the group owner

The average energy consumption of the GO with respect to the number of vehicles in a group for different fading intensities is presented in Figure 6.5. The behavior of the Figure 6.5 (a) is also similar to the one observed in Figure 6.2, and the reasons for the behavior is similar as well. In Figure 6.5 (b), we can observe that the simulation results take a much larger average energy consumption of the GO for one data transmission cycle in the GOB model compared to the theoretical

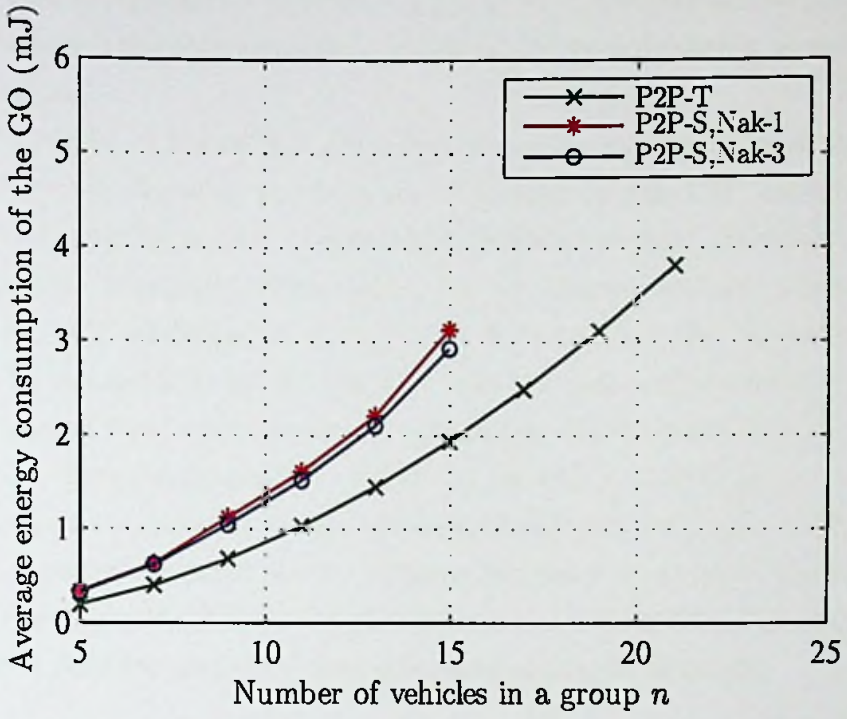


(a) Numerical evaluation of the theoretical result

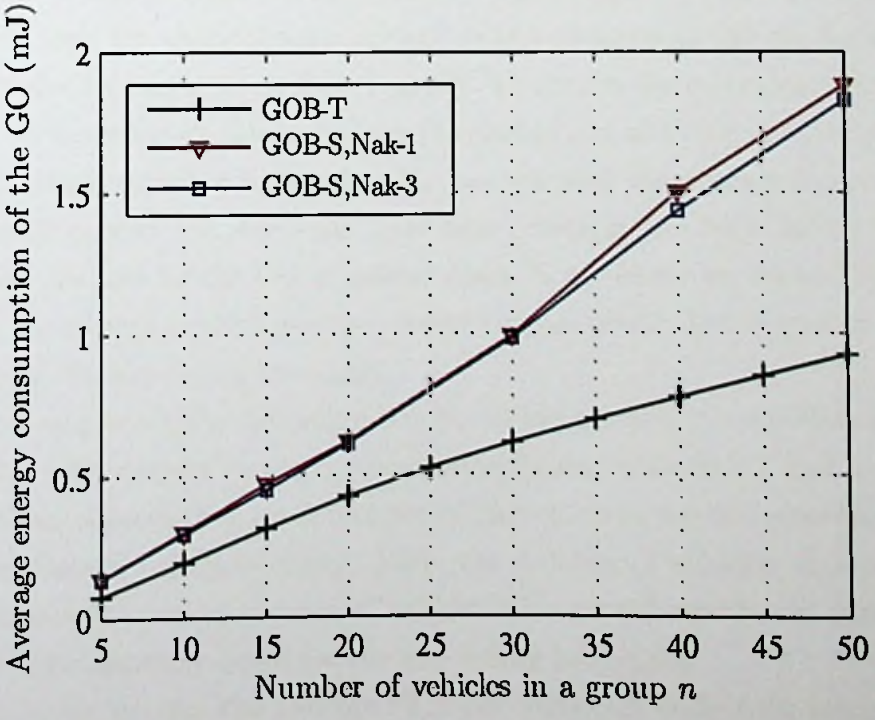


(b) Simulation study

Figure 6.4: The behaviour of average delay with respect to the number of vehicles in a group for P2P and GOB models.



(a) P2P model



(b) GOB model

Figure 6.5: The behaviour of average energy consumption of the GO with respect to the number of vehicles in a group for both communication models.

results. This is because GO switches from being a transmitter to a receiver or vice versa during the contention for the channel in the simulation as well as listen on the channel.

As illustrated in Figure 6.5, the average energy consumption of the GO increases more rapidly with the number of vehicles in the P2P model than the GOB model. Again, similar observations continue to hold for both communication models. Moreover, when comparing the average energy consumption of the GO for P2P models at $n = 15$, it can be seen that the simulation results obtain 3.13 mJ and 2.93 mJ for the high and low fading intensities respectively, whereas around 2 mJ with the theoretical results. Furthermore, when comparing the average energy consumption of the GO for GOB models at $n = 15$, it can be seen that the simulation results obtain 0.47 mJ and 0.45 mJ for the high and low fading intensities respectively, whereas around 0.3 mJ with the theoretical results. *Therefore, the P2P model consumes a much larger average energy of the GO for one data transmission cycle compared to the GOB model.*

6.3 Average packet loss ratio

The average PLR is obtained from the simulation study for the simulated highway scenario. Firstly, we consider the average received signal power of the broadcast frame through the channel with respect to the distance to the sender in Figure 6.6, considering fading intensities 1 and 3. To obtain the statistical significance, we plot 95% confidence interval where the dashed red and blue lines represent the high and the low fading intensities. We can see that the average received signal power may exceed the minimum reception power in the high fading intensity when the distance to the GO is greater than 75 m. However, up to 125 m from the GO can always achieve average received signal power that is greater than the -85dBm in the low fading intensity.

Next, we present the maximum number of lost packets per one data transmission cycle with respect to the n, considering fading intensities 1 and 3 in Figure 6.7. We can observe that lower number of packet losses are still possible with the n under the low fading intensity. Here, the number of vehicles who experience the beacon loss and the number of vehicles who experience the broadcast frame loss are approximately equal for the two fading intensities.

Finally, we present the average PLR per vehicular node with respect to the n in Figure 6.8. We can see that the GOB model achieves higher average PLR per node than the P2P model. The average PLR per node shows small variation

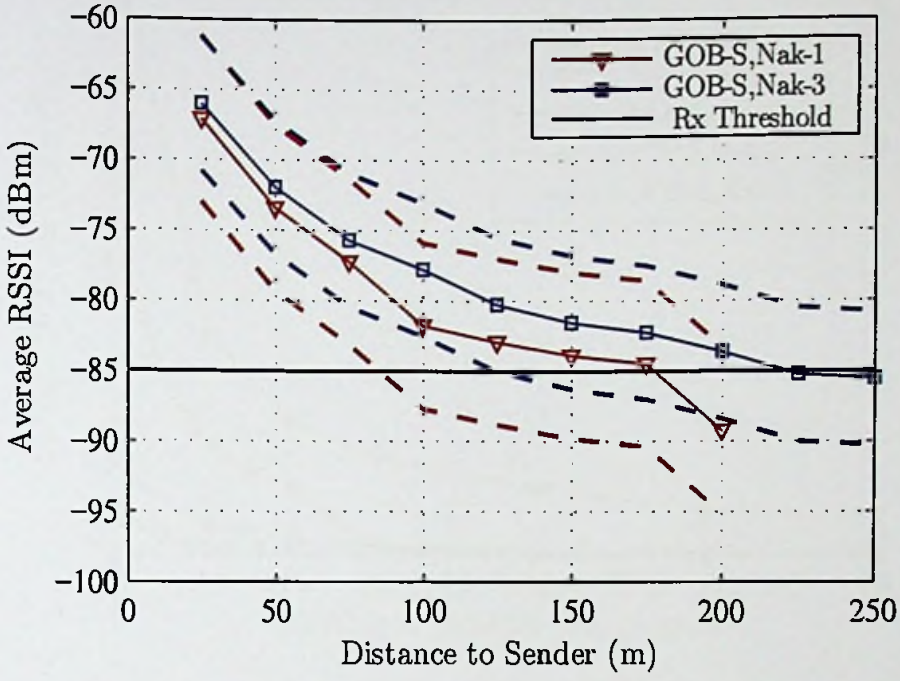


Figure 6.6: The behaviour of average received power with respect to the distance to the sender for different values of m.

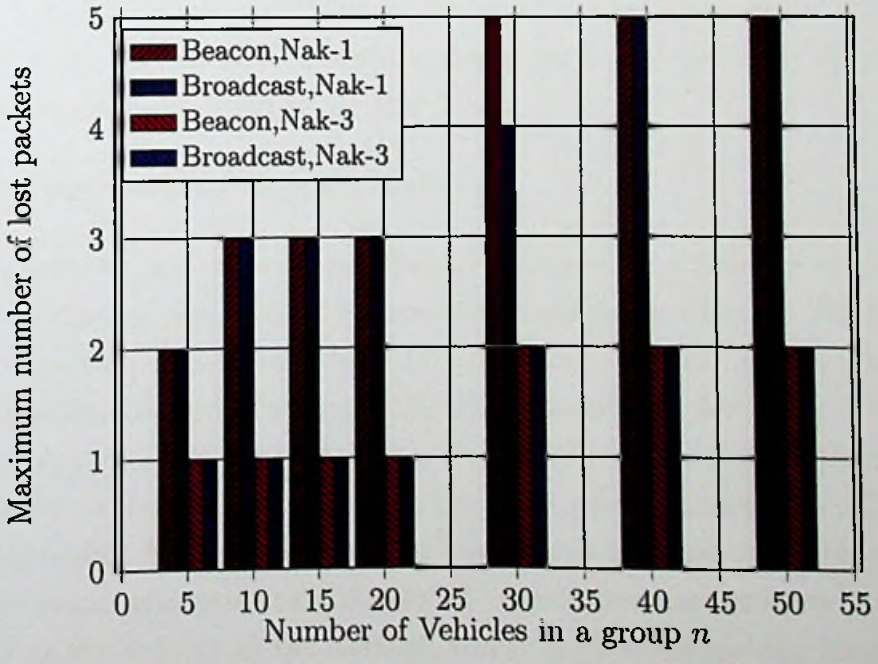


Figure 6.7: The maximum number of lost packets per one data transmission cycle with respect to the number of vehicles in a group for different values of m.

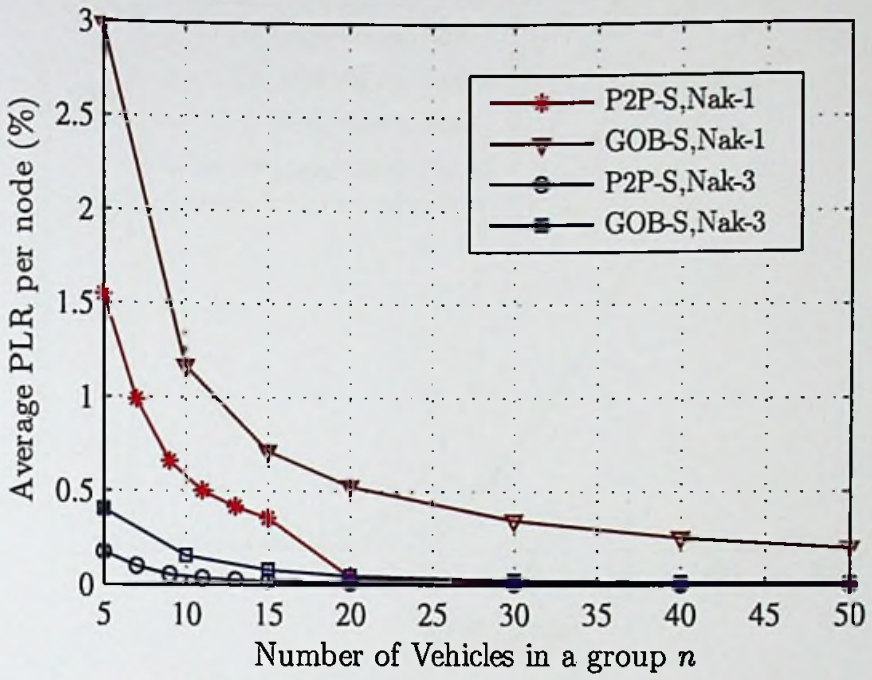


Figure 6.8: The behaviour of average PLR per vehicular node with respect to the number of vehicles in a group for different values of m.

under the different fading intensities. Also, we can have less than 0.5% PLR per node for both P2P model and GOB model in the low fading intensity. *Therefore, the GOB model demonstrates a much higher average PLR per node for one data transmission cycle compared to the P2P model.*

6.4 Average packet reception ratio

In this thesis, we use the average PRR to evaluate the reliability of the GOB model. We start by presenting the cumulative distribution function (CDF) of the signal-to-interference-plus-noise ratio (SINR) under different beacon generation rates and fading intensities in Figure 6.9. The simulation is done for 225 independent instances. We can see that approximately 50% of the clients achieve more than 35 dBm in terms of SINR, even in the worst case scenario. The CDF shifts to the right when fading conditions improve, which is rather intuitive, and also when the beacon generation rate is reduced. This is because the lower generation rate reduces the activity in the channel, which in turn reduces the interference. Note that reducing the beacon generation rate leads to higher latency as well.

Next, we present the average number of vehicles that successfully received the

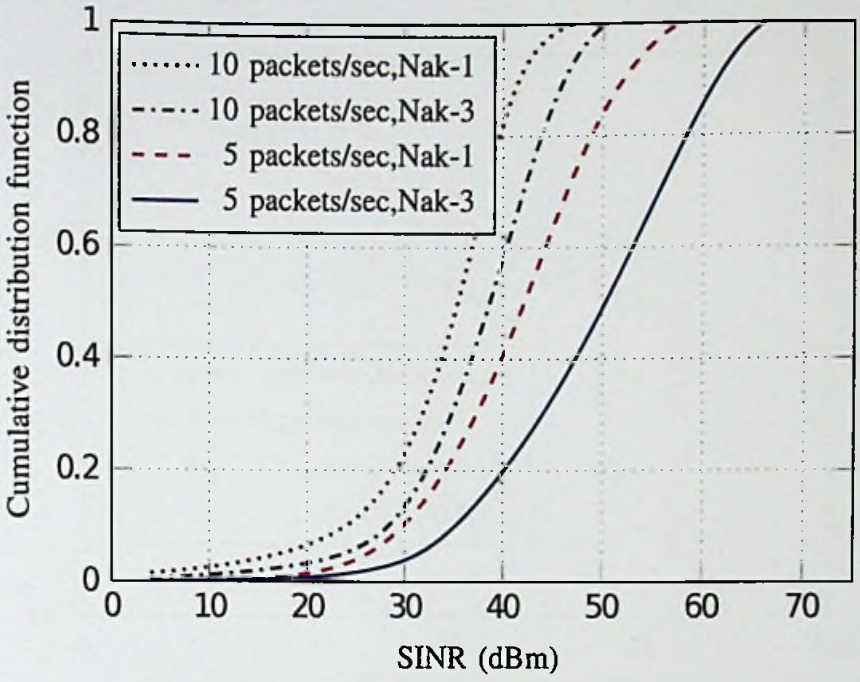


Figure 6.9: CDF of the SINR for both different beacon generation time and the different fading intensities.

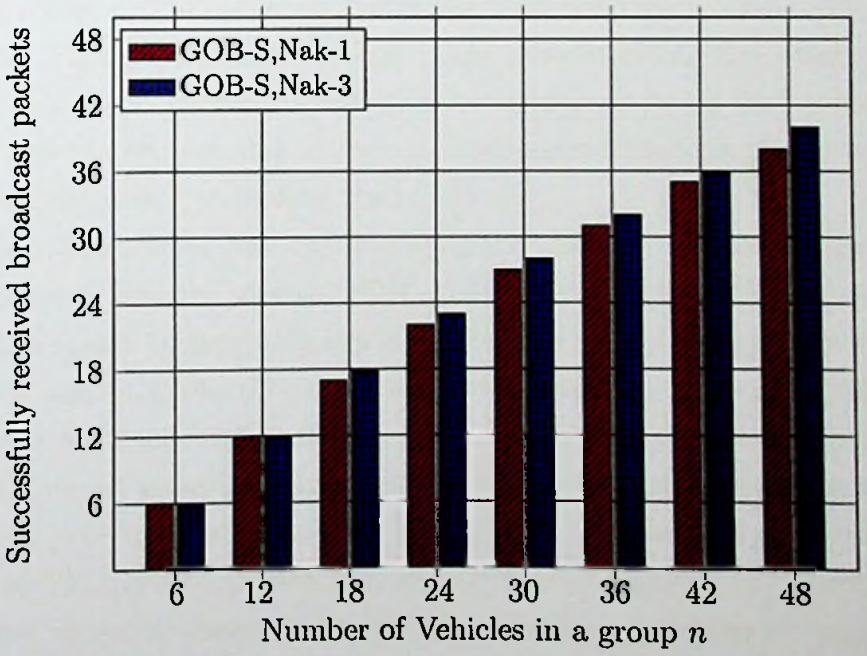


Figure 6.10: The average number of vehicles successfully received the broadcast frame with respect to the number of vehicles in a group for the beacon generation rate of 5 packets per second.

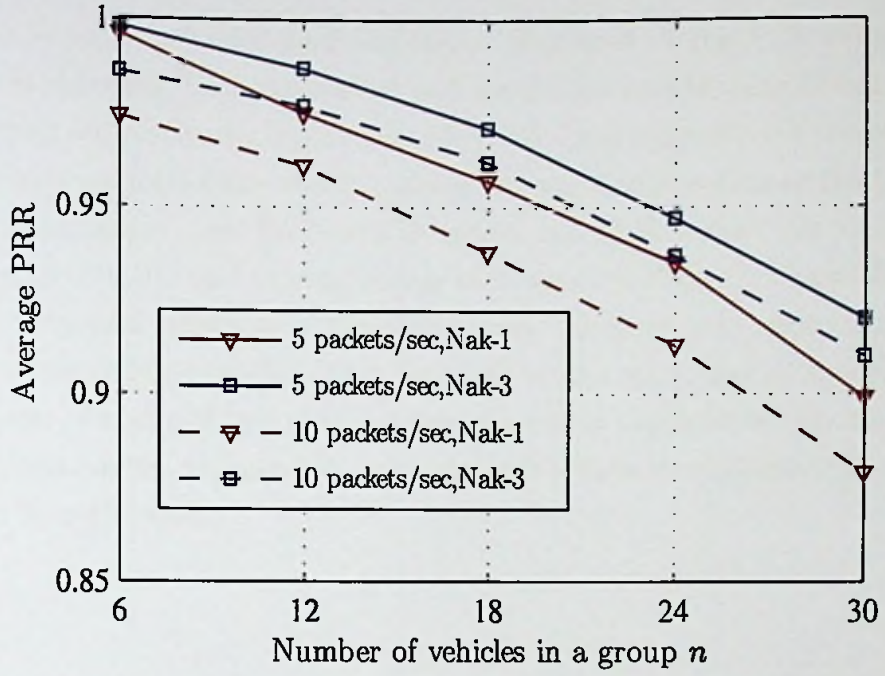


Figure 6.11: The behaviour of average PRR with respect to the number of vehicles in a group for both different beacon generation time and the different fading intensities.

broadcast frame within the group under the beacon generation rate of 5 packets per second. Figure 6.10 illustrates how many vehicles within the group received the broadcast frame successfully for this scenario considering fading intensities 1 and 3. We can observe that the group achieves considerable packet reception within the group when increasing the n.

Finally, we illustrate how the average PRR changes with n in Figure 6.11. We can observe that the average PRR gradually decreases with the number of vehicles in a group. Increasing n increases the distance of the edge users from the GO, which also, increases the path loss and decreases the received signal power. Increasing n may increase interference levels as well. However, it is interesting to note that if we set group sizes properly, we can have satisfactory performance in the network in terms of average PRR. For an example, when the group size is 12, the average PRR is greater than 95% for all cases in Figure 6.11. *Therefore, the degradation in performance on the downlink due to not having retransmissions can keep within tolerable limits by selecting group sizes properly.*

6.5 Summary

In this chapter, we have analyzed the performance of the P2P model and the GOB model using both theoretical and simulation results. The detailed insights regarding the results have been provided. We have started with the comparison of the average total delay and the average energy consumption of the GO for one data transmission cycle between P2P model and GOB model. Gains in terms of average total delay and average energy consumption of the GO have obtained for the GOB model compared to the P2P model. Also, we have shown that average packet loss ratio per node is high for GOB model compared to the P2P model. Moreover, we have shown that the degradation in performance on the downlink due to not having retransmissions can be kept within tolerable limits by selecting group sizes properly.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Wi-Fi Direct (WD) has been looked upon as a possible candidate in VANETs. WD has few drawbacks with large delays in packet delivery being the most critical one with respect to VANETs. In this thesis, we focused on improving the performance of the WD protocol so that it is better suited for communication in VANETs. In particular, it has focused on reducing large transmission delays in WD through a broadcast mechanism on the downlink between the group owner (GO) and the clients of a WD group.

First, we have modeled the topology of the vehicles. We have modeled a bidirectional highway with multiple lanes, we have set speeds of individual vehicles, and have set the spacing between vehicles in the same lane according to the 2-second rule. Secondly, we have modeled the wireless channel, which is crucial for the performance. We have modeled the channel using two well-known models: Friis propagation model to capture path loss and Nakagami fading model to capture multipath fading. We have then set up the two communication models that differ from each other with respect to the downlink communication.

Having established the system model, we have then theoretically evaluated the performance measures such as average total delay, and average energy consumption of the GO under the two communication models considering backoff and contention for the communication channel. We have showed that performance gains in terms of average delay and average energy consumption.

Then, we have performed a simulation study to fine tune the model such that it gives a better representation of the actual scenario. We have implemented the modifications in the WD protocol on OMNeT++, and have simulated the performance of communication. We have obtained the performance measures such as average total delay, average energy consumption of the GO and average packet loss ratio for both communication models.

Highlights of results:

- The theoretical analysis in our work shows a much larger total delay for one data transmission cycle for both communication models compared to the previous work in [9].
- The simulation results show the effect of both path loss and fading, as well as the topology of the network, on average total delay for one data transmission cycle in both P2P and GOB models compared to the theoretical results.
- The P2P model takes a much larger average total delay for one data transmission cycle compared to the GOB model.
- The P2P model consumes a much larger average energy of the GO for one data transmission cycle compared to the GOB model.
- The GOB model demonstrates a much higher average PLR per node for one data transmission cycle compared to the P2P model.
- The degradation in performance on the downlink due to not having retransmissions can keep within tolerable limits by selecting group sizes properly.

We have obtained gains in terms of average total delay and the average energy consumption of the GO. Although there has been no gain in terms of average packet loss ratio per node, it has given a small variation with the low fading environment.

It can be concluded that the proposed broadcast mechanism has enhanced the capability of real-time data communication in VANETs using WD. Finally, we have analyzed the average packet reception ratio using a simulation study. The simulations have also shown that the degradation in performance on the downlink due to not having retransmissions can be kept within tolerable limits properly selecting the group sizes.

7.2 Future work

Before concluding the thesis, we briefly present some ideas which will motivate future studies on this topic.

In Chapter 5, we explained about the timeout selection for GO to initiate transmitting data. We define the timeout using the simulation considering an ideal scenario of 100% beacon reception by all clients. That is, given the Figure 6.2 and Figure 6.3 with a gap between the theoretical and the simulation results. Therefore, it is necessary to further reduce the gap between the theoretical and the simulation results by fine tuning the timeout value.

The broadcast mechanism on the downlink between the GO and the clients of a WD group can be successfully implemented experimentally in the future to analyze how the multiple WD groups would further affect the results presented in this thesis.

As mentioned in the Chapter 2, this large delays in packet delivery can be further reduced by considering the WD group formation time. In the discovery phase, the device switches between the *search* and *listen* state until it moves to the GO negotiation phase. In [23], the author proposes a listen channel randomization scheme to reduce the delay of *listen* state in the discovery phase. Also, the scanning delay can be reduced by fine tuning the minimum and maximum channel waiting time. We can reduce the delay in the discovery phase by combining the above suggested methods.

The group formation time in WD can be reduced with a proper definition of the intent value (IV) based on the device capabilities in the GO negotiation phase. In [13], the authors propose a new group formation algorithm where the IV is defined by considering the RSSI value in the discovery phase. However, this not a fair consideration in a wireless network. The main idea in IV calculation is to assign the device which is more suitable to become the GO. Therefore, it is necessary to define the IV based on the RSSI, number of available neighbors, battery status, and whether the device is already a GO of another group or not. RSSI value gives the distance between two devices, and also, provides the information about the quality of the link between them. A high RSSI value between two devices will suggest that they are closed to each other and maintained a more reliable link between them. A P2P device with multiple neighbors is more suitable to become the GO since it allows its neighbors to join the group as clients. This facilitates more devices to connect to the group and allows to communicate with nearby devices. One of the main parameter associated with the wireless network is the battery status, since all the P2P devices are battery powered mobile devices. The GO is the most active member of the group which performs functions such as beaconing, forwarding and allows to cross connect the group with the cellular network. Therefore, GO consumes more power than clients. The P2P device supports concurrent operation where the device can be simultaneously operated as both GO and client. However, if a P2P device is a GO of one group, it should definitely be the client in the other group. The IV of the P2P device can be calculated by considering above parameters with the assignment of the weighted factors to each parameter.

The delay can be further reduced with a Backup group owner to avoid the

group failures in WD [14]. The combination of the above two modifications: group formation delay and single point of failure of the group, by combining with the GOB model to analyze the full case will be reducing the delay in WD packet delivery, and we believe that it is an interesting problem to be addressed in the future.

Appendices

Appendix A

Sample codes

A.1 Network configuration

Sample codes of network configuration are provided here.

IEEE 802.11 LAN module

```
package inet.examples.wireless.lan80211;

import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator↩
    ;
import inet.node.inet.WirelessHost;
import inet.node.wireless.AccessPoint;
import inet.physicallayer.ieee80211.packetlevel.↩
    Ieee80211ScalarRadioMedium;

network Lan80211
{
    parameters:
        int numHosts;
    submodules:
        host[numHosts]: WirelessHost {
            @display("i=device/cellphone");
            wlan[*].mgmtType = "Ieee80211MgmtSTAWifiDirect";
        }
        radioMedium: Ieee80211ScalarRadioMedium {
            @display("p=61,46");
        }
        configurator: IPv4NetworkConfigurator {
            config = xml("<config><interface hosts='*' address↩
                ='145.236.x.x' netmask='255.255.0.0'/></config>");
            @display("p=192,47");
        }
}
```

.INI file

The .INI file assigns the parameter values of the network. Here, we define a network with six nodes for a simulated highway scenario.

```
[General]
network = Lan80211
tkenv-plugin-path = ../../../etc/plugins
record-eventlog = true

# results record
**.scalar-recording = true
**.vector-recording = true

**.constraintAreaMinX = 100m
**.constraintAreaMinY = 500m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 125m
**.constraintAreaMaxY = 1500m
**.constraintAreaMaxZ = 0m

# Mobility module
**.host[*].mobility.initFromDisplayString = false
**.host[*].mobilityType = "LinearMobility"
**.host[*].mobility.updateInterval = 10ms
**.host[*].mobility.leaveMovementTrail = true

**.host[0].mobility.speed = 22.2222mps
**.host[1].mobility.speed = 27.7778mps
**.host[2].mobility.speed = 33.3333mps
**.host[3].mobility.speed = 33.3333mps
**.host[4].mobility.speed = 27.7778mps
**.host[5].mobility.speed = 22.2222mps

**.host[0].mobility.angle = 270deg  # one direction
**.host[1].mobility.angle = 270deg  # one direction
**.host[2].mobility.angle = 270deg  # one direction

**.host[*].mobility.acceleration = 0

**.host[3].mobility.angle = 90deg  # opposite direction
**.host[4].mobility.angle = 90deg  # opposite direction
**.host[5].mobility.angle = 90deg  # opposite direction
```

```
# Initial position of the nodes
**.host[0].mobility.initialX = 101.875m
**.host[0].mobility.initialY = 1000m
**.host[0].mobility.initialZ = 0m

**.host[1].mobility.initialX = 105.625m
**.host[1].mobility.initialY = 1000m
**.host[1].mobility.initialZ = 0m

**.host[2].mobility.initialX = 109.375m
**.host[2].mobility.initialY = 1000m
**.host[2].mobility.initialZ = 0m

**.host[3].mobility.initialX = 113.625m
**.host[3].mobility.initialY = 1000m
**.host[3].mobility.initialZ = 0m

**.host[4].mobility.initialX = 117.375m
**.host[4].mobility.initialY = 1000m
**.host[4].mobility.initialZ = 0m

**.host[5].mobility.initialX = 121.125m
**.host[5].mobility.initialY = 1000m
**.host[5].mobility.initialZ = 0m

# Energy module
*.host*.wlan[0].radio.energyConsumerType = "↩
   StateBasedEnergyConsumer"
*.host*.wlan[0].radio.energyConsumer.offPowerConsumption = 0mW
*.host*.wlan[0].radio.energyConsumer.sleepPowerConsumption = 1mW
*.host*.wlan[0].radio.energyConsumer.switchingPowerConsumption = 1↩
   mW
*.host*.wlan[0].radio.energyConsumer.↩
   receiverReceivingPowerConsumption = 10mW
*.host*.wlan[0].radio.energyConsumer.↩
   transmitterTransmittingPowerConsumption = 100mW

**.wlan*.bitrate = 6Mbps

**.host[0].wlan[0].mgmt.WiFiDirectUsed=true
**.host[0].wlan[0].mgmt.WiFiDirectGO=true
**.host[0].wlan[0].mgmt.strGroup="Wifi Direct Group"
```

```
**.host[1].wlan[0].mgmt.WiFiDirectUsed=true
**.host[1].wlan[0].mgmt.WiFiDirectGO=false
**.host[1].wlan[0].mgmt.strGroup="Wifi Direct Group"

**.host[2].wlan[0].mgmt.WiFiDirectUsed=true
**.host[2].wlan[0].mgmt.WiFiDirectGO=false
**.host[2].wlan[0].mgmt.strGroup="Wifi Direct Group"

**.host[3].wlan[0].mgmt.WiFiDirectUsed=true
**.host[3].wlan[0].mgmt.WiFiDirectGO=false
**.host[3].wlan[0].mgmt.strGroup="Wifi Direct Group"

**.host[4].wlan[0].mgmt.WiFiDirectUsed=true
**.host[4].wlan[0].mgmt.WiFiDirectGO=false
**.host[4].wlan[0].mgmt.strGroup="Wifi Direct Group"

**.host[5].wlan[0].mgmt.WiFiDirectUsed=true
**.host[5].wlan[0].mgmt.WiFiDirectGO=false
**.host[5].wlan[0].mgmt.strGroup="Wifi Direct Group"

**.host[0].wlan[0].mgmt.color="RED"
**.host[1].wlan[0].mgmt.color="RED"
**.host[2].wlan[0].mgmt.color="RED"
**.host[3].wlan[0].mgmt.color="RED"
**.host[4].wlan[0].mgmt.color="RED"
**.host[5].wlan[0].mgmt.color="RED"
```

IEEE 802.11 NIC

```
package inet.linklayer.ieee80211;

import inet.linklayer.common.IIeee8021dQoSClassifier;
import inet.linklayer.contract.IWirelessNic;
import inet.linklayer.ieee80211.mgmt.IIeee80211Mgmt;
import inet.linklayer.ieee80211.mgmt.Ieee80211AgentSTA;
import inet.physicallayer.contract.packetlevel.IRadio;

module Ieee80211Nic like IWirelessNic
{
    parameters:
        string interfaceTableModule;
        string energySourceModule = default("");
```

```
        string classifierType = default("");
        string mgmtType = default("Ieee80211MgmtSTA");
        string radioType = default("Ieee80211ScalarRadio");
        string macType = default("Ieee80211CompatibleMac");
        string opMode @enum("a","b","g","n","p") = default("b");
        double bitrate @unit("bps") = default(opMode == "b" ? 11↩
            Mbps : opMode == "p" ? 27Mbps : 54Mbps);
        bool _agentNeeded = (mgmtType == "↩
            Ieee80211MgmtSTAWifiDirect" || mgmtType=="↩
            Ieee80211MgmtSTA");
        **.opMode = opMode;
        **.bitrate = bitrate;
        @display("i=block/ifcard;bgb=336,357");
        *.interfaceTableModule = default(absPath(↩
            interfaceTableModule));
        *.energySourceModule = default(absPath(energySourceModule)↩
            );
    gates:
        input upperLayerIn;
        output upperLayerOut;
        input radioIn @labels(IRadioFrame);
        classifier: <classifierType> like IIeee8021dQoSClassifier ↩
            if classifierType != "" {
            @display("p=110,59;i=block/classifier");
        }
        agent: Ieee80211AgentSTA if _agentNeeded {
            parameters:
                @display("p=261,131");
        }
        mgmt: <mgmtType> like IIeee80211Mgmt {
            parameters:
                macModule = "^.mac";
                @display("p=157,131");
        }
        mac: <macType> like IIeee80211Mac {
            parameters:
                @display("p=157,217");
        }
        radio: <radioType> like IRadio {
            parameters:
                @display("p=157,302");
        }
    connections:
        radioIn --> { @display("m=s"); } --> radio.radioIn;
```

```
        radio.upperLayerIn <-- mac.lowerLayerOut;
        radio.upperLayerOut --> mac.lowerLayerIn;

        mac.upperLayerOut --> mgmt.macIn;
        mac.upperLayerIn <-- mgmt.macOut;

        mgmt.agentOut --> agent.mgmtIn if _agentNeeded;
        mgmt.agentIn <-- agent.mgmtOut if _agentNeeded;

        mgmt.upperLayerOut --> { @display("m=n"); } --> ↩
            upperLayerOut;
        mgmt.upperLayerIn <-- { @display("m=n"); } <-- ↩
            upperLayerIn if classifierType == "";
        mgmt.upperLayerIn <-- { @display("m=n"); } <-- classifier.↩
            out if classifierType != "";
        classifier.in <-- { @display("m=n"); } <-- upperLayerIn if↩
             classifierType != "";
}
```

Frame format of the IEEE 802.11 standard

```
cplusplus {{
#include "inet/linklayer/common/MACAddress.h"
#include "inet/linklayer/common/Ieee802Ctrl_m.h"
#include "inet/common/geometry/common/Coord.h"
}}

class noncobject MACAddress;
class noncobject Coord;
enum EtherType;
namespace inet::ieee80211;

cplusplus {{
const unsigned int LENGTH_RTS = 160;     //bits
const unsigned int LENGTH_CTS = 112;     //bits
const unsigned int LENGTH_ACK = 112;     //bits
const unsigned int LENGTH_MGMT = 28 * 8;     //bits
const unsigned int DATAFRAME_HEADER_MINLENGTH = 28 * 8;     //bits ↩
    without QoS, without Address4, without SNAP: 2 + 2 + 3*6(↩
    addresses) + 2 + 4(crc)
const unsigned int QOSCONTROL_BITS = 2 * 8;     // QoS Control ↩
    field length (bits)
```

```
const unsigned int SNAP_HEADER_BYTES = 8;
const short int MAX_NUM_FRAGMENTS = 16;
}}

enum Ieee80211FrameType
{
    // management:
    ST_ASSOCIATIONREQUEST = 0x00;
    ST_ASSOCIATIONRESPONSE = 0x01;
    ST_REASSOCIATIONREQUEST = 0x02;
    ST_REASSOCIATIONRESPONSE = 0x03;
    ST_PROBEREQUEST = 0x04;
    ST_PROBERESPONSE = 0x05;
    ST_BEACON = 0x08;
    ST_ATIM = 0x09;
    ST_DISASSOCIATION = 0x0a;
    ST_AUTHENTICATION = 0x0b;
    ST_DEAUTHENTICATION = 0x0c;
    ST_ACTION = 0x0d;
    ST_NOACKACTION = 0x0e;

    // control (CFEND/CFEND_CFACK omitted):
    ST_PSPOLL = 0x1a;
    ST_RTS = 0x1b;
    ST_CTS = 0x1c;
    ST_ACK = 0x1d;
    ST_BLOCKACK_REQ = 0x18;
    ST_BLOCKACK     = 0x19;

    // data (CFPOLL/CFACK subtypes omitted):
    ST_DATA = 0x20;
    ST_DATA_WITH_QOS = 0x28;
    //Feedback frame for multicast tramsmission
    ST_LBMS_REQUEST = 0x30;
    ST_LBMS_REPORT = 0x31;
}

// Structure to store BSM
struct Ieee80211BSMElement
{
    string DeviceID;
    string Location;
    string Speed;
}
```

```
// Data frame body format.
class Ieee80211DataFrameBody
{
    short bodyLength = 12; // assuming a 8-character SSID
    string SSID;
    Ieee80211BSMElement BSM;
}

packet Ieee80211Frame
{
    byteLength = LENGTH_ACK / 8;
    short type @enum(Ieee80211FrameType); // type and subtype
    bool toDS;
    bool fromDS;
    bool retry;
    bool moreFragments;
    simtime_t duration = -1; // "duration" in the Duration/ID ↩
       field (-1=no duration)
    short AID = -1;          // "id" (Association ID) in the ↩
       Duration/ID field (-1=no ID)
    MACAddress receiverAddress; // aka address1
    simtime_t MACArrive;
}

// Common base class for 802.11 data and management frames
packet Ieee80211DataOrMgmtFrame extends Ieee80211TwoAddressFrame
{
    byteLength = LENGTH_MGMT / 8;
    MACAddress address3;
    short fragmentNumber;
    uint16 sequenceNumber;
}

// Format of the 802.11 data frame
packet Ieee80211DataFrame extends Ieee80211DataOrMgmtFrame
{
    type = ST_DATA;          // or ST_DATA_WITH_QOS
    MACAddress address4;
    uint16 qos;
    int ackPolicy @enum(AckPolicy);
    uint8 tid;
    byteLength = 28+getBody().getBodyLength();
    Ieee80211DataFrameBody body;
```

```
}

packet Ieee80211QoSDataFrame extends Ieee80211DataFrame
{
    type = ST_DATA_WITH_QOS;
    int ackPolicy @enum(AckPolicy);
    uint8 tid;
   uint16 qos;

    byteLength = 28+getBody().getBodyLength();
    Ieee80211DataFrameBody body;
}

// 802.11 data frame with the 8-byte SNAP header (AA AA 03, 00 00 ↩
   00, <2-byte ~EtherType>)
packet Ieee80211DataFrameWithSNAP extends Ieee80211DataFrame
{
    byteLength = DATAFRAME_HEADER_MINLENGTH / 8 + ↩
       SNAP_HEADER_BYTES;
    int etherType @enum(EtherType);
}
}
```

A.2 Data transmission

Here, we provide codes of the main functions that are used for implementation of the data transmission algorithm on *IEEE80211MgmtSTAWifiDirect* module.

Initialization

```
void Ieee80211MgmtSTAWifiDirect::initialize(int stage)
{
    Ieee80211MgmtBase::initialize(stage);

    if (stage == INITSTAGE_LOCAL) {
        isAssociated = false;
        myIface = nullptr;
        numChannels = par("numChannels");
        host = getContainingNode(this);
        host->subscribe(NF_LINK_FULL_PROMISCUOUS, this);

        struct timespec ts;
```

```
        clock_gettime(CLOCK_MONOTONIC, &ts);
        srand((time_t)ts.tv_nsec);
        deviceName+="DEVICE "+std::to_string((rand()\%30000)+1);
        beaconInterval = par("beaconInterval");
        numProvisionningSteps = par("numAuthSteps");
        if((numProvisionningSteps\%2)!=0) numProvisionningSteps++;
        status=Off;
        initFormation.used=par("WiFiDirectUsed");
        initFormation.go=par("WiFiDirectGO");
        initFormation.strGroup=par("strGroup").stringValue();
        initFormation.color=par("color").stringValue();

        WATCH(status);
        WATCH(deviceName);
        WATCH(discovering);
        WATCH(isAssociated);
        WATCH_MAP(clientList);
        WATCH_MAP(dataQueue);
        WATCH(scanning);
        WATCH(assocAP);
        WATCH_LIST(apList);
    }
    else if (stage == INITSTAGE_LINK_LAYER_2) {
        IInterfaceTable *ift = findModuleFromPar<IInterfaceTable>(↩
            par("interfaceTableModule"), this);
        if (ift) {
            myIface = ift->getInterfaceByName(utils::stripnonalnum↩
                (findModuleUnderContainingNode(this)->getFullName()↩
                ).c_str());
        }
    }
}
```

Handle cMessages

In this code, we only include the actions related to the *beaconTimer* and *Timer*.

```
void Ieee80211MgmtSTAWifiDirect::handleTimer(cMessage *msg)
{
// Send beacon
if (msg == beaconTimer) {
        sendBeacon();
        scheduleAt(simTime() + beaconInterval, beaconTimer);
```

```
}else if (msg == timer) {
        EV << "GO initiate the data transmission in the downlink \n↩
           ";
        GOSendData();
}else {
         throw cRuntimeError("internal error: unrecognized timer ↩
            '\%s'", msg->getName());
    }
}
```

GO sends beacons

```
void Ieee80211MgmtSTAWifiDirect::sendBeacon()
{
    EV << "Sending beacon\n";
    Ieee80211BeaconFrame *frame = new Ieee80211BeaconFrame("Beacon↩
       ");
    Ieee80211BeaconFrameBody& body = frame->getBody();
    if(initFormation.used==true)
    {
        std::string str="DIRECT-";
        str+=initFormation.strGroup;
        body.setSSID(str.c_str());
    }else deviceName.c_str();
    body.setSupportedRates(supportedRates);
    body.setBeaconInterval(beaconInterval);
    body.getP2PIEBeaconFrame().setUsed(true);
    frame->setByteLength(28 + body.getBodyLength());
    frame->setReceiverAddress(MACAddress::BROADCAST_ADDRESS);
    frame->setFromDS(true);

    if(status == WSCProvisioning) frame->getBody().↩
       getP2PIEBeaconFrame().getCapability().↩
       setGroupCapabilityBitmap(6,true);
    sendDown(frame);
    // Clear the queue of clients data frames
    dataQueue.clear();
    // Start the Timer for GO to initiate the data transmission in↩
        the downlink
    Timer = (simtime_t) GOTimeout;
    EV << " Start Timer " << Timer  << "\n";
    timer = new cMessage("timer");
```

```
    scheduleAt(simTime() + Timer , timer);
}
```

Handle beacon frames by clients

```
void Ieee80211MgmtSTAWifiDirect::handleBeaconFrame(↩
    Ieee80211BeaconFrame *frame)
{
        if(status==P2PClient)
        {
            storeAPInfo(frame->getTransmitterAddress(), frame->↩
                getBody());
            if (isAssociated && frame->getTransmitterAddress() == ↩
                assocAP.address)
            {
                EV << "Beacon is from associated AP, restarting ↩
                    beacon timeout timer\n";
                ASSERT(assocAP.beaconTimeoutMsg != nullptr);
                cancelEvent(assocAP.beaconTimeoutMsg);
                scheduleAt(simTime() + MAX_BEACONS_MISSED * ↩
                    beaconInterval, assocAP.beaconTimeoutMsg);

                sendData();
            }
        }else
            dropManagementFrame(frame);
}
```

Clients send data frames to the GO

```
void Ieee80211MgmtSTAWifiDirect::sendData()
{
    if(status == P2PClient){
        EV << "Client sends the data frame \n";

        std::string name="Exchange Data frame";

        Ieee80211DataFrame *frame = new Ieee80211DataFrame(name.↩
            c_str());
        Ieee80211DataFrameBody& body = frame->getBody();
```

```
        host = getContainingNode(this);
        IMobility *mobility = check_and_cast<IMobility *>(host->↩
            getSubmodule("mobility"));
        if(mobility == NULL){
            opp_error("mobility module not found");
        }

        EV << "location =" << mobility->getCurrentPosition() << "↩
            speed =" << mobility->getCurrentSpeed() << "\n";

        body.getBSM().DeviceID = myIface->getMacAddress().str();
        body.getBSM().Location = mobility->getCurrentPosition().↩
            info();
        body.getBSM().Speed = mobility->getCurrentSpeed().info();

        frame->setFromDS(false);
        frame->setToDS(true);

        frame->setTransmitterAddress(myIface->getMacAddress());
        frame->setReceiverAddress(assocAP.address);
        frame->setAddress3(MACAddress::BROADCAST_ADDRESS);

        sendDown(frame);
        EV << "frame sending down to 'macout' \n ";

    }
}
```

Uplink

```
void Ieee80211MgmtSTAWifiDirect::handleDataFrame(↩
    Ieee80211DataFrame *frame)
{
    if(status==P2PClient || status==P2PGroupOwner)
        {
        // Only send the Data frame up to the higher layer if the ↩
            STA is associated with an AP,
        // else delete the frame
            if(status==P2PClient)
            {
                if (isAssociated){
```

```
            EV << "Decapsulate and send to the upper layer↩
                ";
            sendUp(decapsulate(frame));
        }else
        {
            EV << "Rejecting data frame as STA is not ↩
                associated with an AP" << endl;
            delete frame;
         }
    }else
    {
        // check toDS bit
        if (!frame->getToDS()) {
            EV << "Frame is not for us (toDS=false) — ↩
                discarding\n";
            delete frame;
            return;
        }
        // handle broadcast/multicast frames
        if (frame->getAddress3().isMulticast() || frame->↩
            getAddress3().isBroadcast()) {
            EV << "Handling multicast frame or Handling ↩
                broadcast frame\n";
            MACAddress txAddress = frame->↩
                getTransmitterAddress();
            // look up transmitter address in the client ↩
                list
            auto it = clientList.find(txAddress);
            if (it == clientList.end()) {
                // frame transmitter address not in the ↩
                    clientList
                delete frame;
                return;
            }
            if(dataQueue.empty()){
            // create DATA entry
            EV << "dataQueue is empty" << frame->↩
                getTransmitterAddress();
            DATAInfo *data;
            data = &dataQueue[txAddress];  // this create ↩
                a new entry
            data->address = txAddress;
            data->dframe = frame->getBody();
        }else
```

```
        {
            auto it = dataQueue.find(txAddress);
            if(it == dataQueue.end()){
              // txaddress is not in the dataQueue
              // create new entry
              EV << "tx address is not in the dataQueue " ↩
                 << frame->getTransmitterAddress();
              DATAInfo *data;
              data = &dataQueue[txAddress];
              data->address = txAddress;
              data->dframe = frame->getBody();
            }
            else {
              DATAInfo *data1;
              EV << "tx is already send the data" << frame↩
                 ->getTransmitterAddress();
              it->second.dframe = frame->getBody();
            }
        }
        // duplicate the frame a
        frame2=frame->dup();
        ASSERT(!frame->getAddress3().isUnspecified());
        frame->setReceiverAddress(frame->getAddress3());
        frame->setAddress3(frame->getTransmitterAddress())↩
            ;
        frame->setType(ST_DATA_WITH_QOS);
        frame->addBitLength(QOSCONTROL_BITS);
        sendUp(decapsulate(frame));
        EV << " check whether every client send data to ↩
            initiate the downlink transmission \n ";
        if(staList.size() == dataQueue.size())
        {
            cancelEvent(Timer);
            GOSendData();
        }
        return;
    }
}
if(frame->getAddress3().equals(myIface->getMacAddress()))
{
    frame->setToDS(false);
    frame->setFromDS(true);
    // move destination address to address1 (receiver ↩
        address),
```

```
            // and fill address3 with original source address;
            // sender address (address2) will be filled in by MAC
            ASSERT(!frame->getAddress3().isUnspecified());
            frame->setReceiverAddress(frame->getAddress3());
            frame->setAddress3(frame->getTransmitterAddress());
            sendUp(decapsulate(frame));
            return;
        }
        // look up destination address in the client list
        auto it = clientList.find(frame->getAddress3());
        if (it == clientList.end()) {
            EV << "Frame's destination address is not in the ↩
                client list — dropping frame\n";
            delete frame;
        }
        else {
            // dest address is our clientList, but is it already ↩
                associated?
            if (it->second.status == ASSOCIATED)
            distributeReceivedDataFrame(frame); // send it out ↩
                to the destination STA
            else {
                EV << "Frame's destination STA is not in associated↩
                    state — dropping frame\n";
                delete frame;
            }
         }
    }else  {
    EV << "Rejecting data frame as STA is not associated with an ↩
        AP" << endl;
    delete frame;
    }
}
```

Downlink

```
void Ieee80211MgmtSTAWifiDirect::GOSendData()
{
    EV << "P2P model \n";
    int i = 0;
    while(i < clientList.size())
    {
```

```
      cframe = dataQueue.pop();
      for(auto it=clientList.begin();it!=clientList.end();it++)
     {
        if(it->first != cframe->getTransmitterAddress())
        {
            std::string name="Exchange Data frame";
            Ieee80211DataFrame *frame = new Ieee80211DataFrame(↩
               name.c_str());
            Ieee80211DataFrameBody& body = cframe->getBody();

            frame->setFromDS(true);
            frame->setTransmitterAddress(myIface->getMacAddress↩
               ());
            frame->setReceiverAddress(it->first);
            sendDown(frame);
        }
    }
    i++;
}

 EV << "GO send its data in P2P manner \n";
host = getContainingNode(this);
IMobility *mobility = check_and_cast<IMobility *>(host->↩
    getSubmodule("mobility"));
if(mobility == NULL){
    opp_error("mobility module not found");
}
for(auto it=clientList.begin();it!=clientList.end();it++)
{
    std::string name="Exchange Data frame";
    Ieee80211DataFrame *frame = new Ieee80211DataFrame(name.↩
       c_str());
    Ieee80211DataFrameBody& body = frame->getBody();

    body.getBSM().DeviceID = myIface->getMacAddress().str();
    body.getBSM().Location = mobility->getCurrentPosition().info↩
       ();
    body.getBSM().Speed = mobility->getCurrentSpeed().info();

    frame->setFromDS(true);
    frame->setTransmitterAddress(myIface->getMacAddress()); ↩
              ///(this->myAddress);
    frame->setReceiverAddress(it->first);
```

```
        EV << "GO send data frame to RX address : " << it->first << ↩
            "\n";
        dataQueue.insert(frame);

        if(clientList.size()==1)
        {
            Ieee80211DataFrame *frame = check_and_cast<↩
                Ieee80211DataFrame *>(dataQueue.pop());
            sendDown(frame);
        }
    }
}
```

```
void Ieee80211MgmtSTAWifiDirect::GOSendData()
{
    EV  << "GOB model \n";
    std::string name="Exchange Broadcast Data frame";
    Ieee80211DataFrame *frame = new Ieee80211DataFrame(name.c_str↩
        ());
    Ieee80211DataFrameBody& body = frame->getBody();

    host = getContainingNode(this);
    IMobility *mobility = check_and_cast<IMobility *>(host->↩
        getSubmodule("mobility"));
    if(mobility == NULL){
        opp_error("mobility module not found");
    }

    frame->setFromDS(true);

    body.getBSM().DeviceID = myIface->getMacAddress().str();
    body.getBSM().Location = mobility->getCurrentPosition().info()↩
        ;
    body.getBSM().Speed = mobility->getCurrentSpeed().info();

    while(!dataQueue.empty())
    {
        frame.aggregate(dataQueue.pop());
    }

    frame->setTransmitterAddress(myIface->getMacAddress());
    // Set frame RX address to broadcast
    frame->setReceiverAddress(MACAddress::BROADCAST_ADDRESS);
```

```
    dataQueue.clear();

    // make it a QoS frame
    frame->setType(ST_DATA_WITH_QOS);

    sendDown(frame);

}
```

References

[1] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289 – 299, Aug. 2014.

[2] F. Bai and B. Krishnamachari, "Exploiting the wisdom of the crowd: localized, distributed information-centric VANETs," *IEEE Commun. Magazine*, vol. 48, no. 5, pp. 138 – 146, May 2010.

[3] T. H. Luan, X. S. Shen, and F. Bai, "Integrity-oriented content transmission in highway vehicular ad hoc networks," in *Proc. IEEE International Conference on Computer Commun.*, Apr. 2013, pp. 2562 – 2570.

[4] G. Karagiannis, O. Altintas, 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 Commun. Surveys Tuts.*, vol. 13, no. 4, pp. 584 – 616, Jul. 2011.

[5] S. Zeadally, R. Hunt, Y.-S. Chen, A. Irwin, and A. Hassan, "Vehicular ad hoc networks VANETS: status, results, and challenges," *Telecommunication Systems*, vol. 50, no. 4, pp. 217 – 241, Aug. 2012.

[6] C. Olaverri-Monreal, P. Gomes, R. Fernandes, F. Vieira, and M. Ferreira, "The See-Through System: A VANET-enabled assistant for overtaking maneuvers," in *Proc. IEEE Intelligent Vehicles Symposium*, Jun. 2010, pp. 123 – 128.

[7] K.-Y. Ho, P.-C. Kang, C.-H. Hsu, and C.-H. Lin, "Implementation of WAVE/DSRC devices for vehicular communications," in *Proc. IEEE International Symposium on Computer Commun. Control and Automation*, May 2010, pp. 522 – 525.

[8] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich, "Design of 5.9 GHz DSRC-based vehicular safety communication," *IEEE Wireless Commun.*, vol. 13, no. 5, pp. 1536 – 1284, Nov. 2006.

[9] C. Satish, *Inter-vehicular communication for collision avoidance using Wi-Fi direct.* Thesis. Rochester Institute of Technology, 2014.

[10] A. Balasundram, T. Samarasinghe, and D. Dias, "Performance analysis of Wi-Fi Direct for vehicular ad-hoc networks," in *Proc. IEEE International Conference on Advanced Networks and Telecommunications Systems*, Nov. 2016, pp. 1 – 6.

[11] Michigan Department of Transportation, "Connected Vehicle Technology Industry Delphi Study," https://www.michigan.gov/documents/mdot/09-27-2012_Connected_Vehicle_Technology_-_Industry_Delphi_Study_401329_7.pdf, 2012.

[12] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich, "Device-to-device communications with Wi-Fi Direct: Overview and experimentation," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 96 – 104, Jun. 2013.

[13] H. Zhang, Y. Wang, and C. C. Tan, "WD2: An improved Wi-Fi Direct group formation protocol," in *Proc. ACM MobiCom Workshop on Challenged networks*, Sep. 2014, pp. 55 – 60.

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

[15] —, "OMNeT++ simulation manual, version 5.2," [online] https://www.omnetpp.org/doc/omnetpp/manual, 2016.

[16] S. Iskounen, T. M. T. Nguyen, and S. Monnet, "WiFi-Direct Simulation for INET in OMNeT++," *arXiv preprint arXiv:1609.04604*, 2016.

[17] —, "INET framework, version 3.5," [online] https://inet.omnetpp.org, 2016.

[18] Wi-Fi Alliance, "Wi-Fi Peer-to-Peer (P2P) Technical Specification, version 1.7," [online] https://www.wi-fi.org/discover-wi-fi/specifications, 2010.

[19] https://developer.android.com/training/connect-devices-wirelessly/wifi-direct.html, 2016.

[20] A. Varga, *OMNeT++ In Modeling and Tools for Network Simulation.* Springer-Verlag Berlin Heidelberg, 2010.

[21] C. Jin, J.-W. Choi, W.-S. Kang, and S. Yun, "Wi-Fi Direct data transmission for wireless medical devices," in *Proc. IEEE International Symposium on Consumer Electronics*, Jun. 2014, pp. 1 – 2.

[22] N. I. Shuhaimi, H. 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.

[23] W. Sun, C. Yang, S. Jin, and S. Choi, "Listen channel randomization for faster Wi-Fi direct device discovery," in *Proc. IEEE International Conference on Computer Commun.*, Apr. 2016, pp. 1 – 9.

[24] A. J. Monarrez, "Extending Wi-Fi Direct for automated operations," [online] https://calhoun.nps.edu/handle/10945/45229, 2015.

[25] M. I. Hassan, H. L. Vu, T. Sakurai, and L. L. Andrew, "Effect of retransmissions on the performance of the IEEE 802.11 MAC protocol for DSRC," *IEEE Trans. Veh. Technol.*, vol. 61, no. 1, pp. 22 – 34, Jan. 2012.

[26] J. Karedal, N. Czink, A. Paier, F. Tufvesson, and A. F. Molisch, "Path loss modeling for vehicle-to-vehicle communications," *IEEE Trans. Veh. Technol.*, vol. 60, no. 1, pp. 323 – 328, Jan. 2011.

[27] O. Onubogu, K. Ziri-Castro, D. Jayalath, K. Ansari, and H. Suzuki, "Empirical vehicle-to-vehicle pathloss modeling in highway, suburban and urban environments at 5.8 GHz," in *Proc. IEEE International Conference on Signal Processing and Commun. Systems*, Dec. 2014, pp. 1 – 6.

[28] M. Torrent-Moreno, F. Schmidt-Eisenlohr, H. Fussler, and H. Hartenstein, "Effects of a realistic channel model on packet forwarding in vehicular ad hoc networks," in *Proc. IEEE Wireless Commun. and Networking Conference*, Apr. 2006, pp. 385 – 391.

[29] P. Paschalidis, K. Mahler, A. Kortke, M. Peter, and W. Keusgen, "Pathloss and multipath power decay of the wideband car-to-car channel at 5.7 GHz," in *Proc. IEEE Vehicular Technology Conference*, May 2011, pp. 1 – 5.

[30] TG Road Safety, "Safe distance between vehicles," http://www.cedr.eu/download/Publications/2010/e_Distance_between_vehicles.pdf, 2010.

[31] A. F. Molisch, F. Tufvesson, J. Karedal, and C. F. Mecklenbrauker, "A survey on vehicle-to-vehicle propagation channels," *IEEE Wireless Commun.*, vol. 16, no. 6, pp. 12 – 22, Dec. 2009.

[32] H. Fernandez, L. Rubio, J. Reig, V. M. Rodrigo-Penarrocha, and A. Valero, "Path loss modeling for vehicular system performance and communication protocols evaluation," *Mobile Networks and Applications*, vol. 18, no. 6, pp. 755 – 765, Dec. 2013.

[33] H. Fernández, L. Rubio, V. M. Rodrigo-Peñarrocha, and J. Reig, "Path loss characterization for vehicular communications at 700 MHz and 5.9 GHz under LOS and NLOS conditions," *IEEE Antennas and Wireless Propagation Letters*, vol. 13, pp. 931 – 934, May 2014.

[34] M. Boban, T. T. Vinhoza, M. Ferreira, J. Barros, and O. K. Tonguz, "Impact of vehicles as obstacles in vehicular ad hoc networks," *IEEE Journal on Selected Areas in Commun.*, vol. 29, no. 1, pp. 15 – 28, Jan. 2011.

[35] A. Goldsmith, "Wireless communications," Cambridge university press, 2005.

[36] M. Torrent-Moreno, D. Jiang, and H. Hartenstein, "Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks," in *Proc. International Workshop on Vehicular Ad hoc Networks*, Oct. 2004, pp. 10 – 18.

[37] J. Yin, G. Holland, T. Elbatt, F. Bai, and H. Krishnan, "DSRC channel fading analysis from empirical measurement," in *Proc. IEEE International Conference on Commun. and Networking*, Oct. 2006, pp. 1 – 5.

[38] L. Rubio, J. Reig, and N. Cardona, "Evaluation of nakagami fading behaviour based on measurements in urban scenarios," *AEU-International Journal of Electronics and Commun.*, vol. 61, no. 2, pp. 135 – 138, Feb. 2007.

[39] V. Taliwal, D. Jiang, H. Mangold, C. Chen, and R. Sengupta, "Empirical determination of channel characteristics for DSRC vehicle-to-vehicle communication," in *Proc. ACM International Workshop on Vehicular ad hoc networks*, Oct. 2004, pp. 88 – 88.

[40] 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 mobile channels," in *Proc. IEEE Vehicular Technology Conference*, Sep. 1999, pp. 1460 – 1464.

[41] P. Chatzimisios, V. Vitsas, and A. C. Boucouvalas, "Throughput and delay analysis of IEEE 802.11 protocol," in *Proc. International Workshop on Networked Appliances*, Oct. 2002, pp. 168 – 174.

[42] T. K. Mak, K. P. Laberteaux, and R. Sengupta, "A multi-channel VANET providing concurrent safety and commercial services," in *Proc. International Workshop on Vehicular Ad hoc Networks*, Oct. 2005, pp. 168 – 174.