UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ANALYSIS OF VANET STANDARD IEEE 1609.4 MAC LAYER MULTI-CHANNEL OPERATIONS USING OMNET++ AND VEINS

A THESIS SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

SUSANA A. RODRIGUEZ CORIA
Norman, Oklahoma
2018

ANALYSIS OF VANET STANDARD IEEE 1609.4

MAC LAYER MULTI-CHANNEL OPERATIONS

USING OMNET++ AND VEINS

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Dr. Mohammed Atiquzzaman, Chair

Dr. Dean Hougen

Dr. Changwook Kim

To Alex.

To Octavio, Carmen, Josue, Noemi and Tavito Rodriguez.

To Dr. Atiquzzaman for allowing me to work under his mentorship.

Acknowledgements

It has been a long fulfilling journey. I am very happy to finally be a contributor in the research field, with the strong mentorship of Dr. Atiquzzaman. Thank you to Dr. Atiquzzaman for giving me the opportunity to pursue a graduate degree and for encouraging me to finish.

Thank you to Virginie Peres Woods and Sarah Vaughan for collaborating with me remotely to get the paperwork submitted. Thank you to Dr. Hougen, Dr. Kim, Dr. Irving and Dr. LaGreca for your support.Thank you to Dr. Sommers and Dr. Dressler for growing the software community in Vanets. Also for staying active and directly responding to questions about Veins.

Thank you to my friend Hernando, who introduced me the book “Deep Work” by Cal Newport. Thank you to other book authors like Umberto Eco, Jon Acuff, Chad Fowler, Cal Newport, Malcom Madwell, Alan Lakein, James Clear and Steven Pressfield. Their books were inspiring and helpful, to understand process oriented growth.

Thank you to my friend Nii for making suggestions over the phone when I was stuck in the abyss of linking errors. Thank you to Katy and Aria for reviewing the content. Thank you to my son who waited patiently in the mornings, evenings, nights and weekends while I worked on my research. Thanks to the Narciso family for keeping my son entertained on the weekends. Thanks to my parents and siblings for continuously watching my son and sending me positive messages from far away.

Table Of Contents

List of Figures

List of Tables

List of Acronyms

- **AC** - Access Category for a channel [15]
- **AP** - Access Point [30]
- **ATB** - Adaptive Traffic Beacon [22]
- **BSMs** - Basic Safety Messages also known as beacons [31]
- **CSMA/CA** - Carrier Sense Multiple Access protocol with Collision Avoidance
- **CW** - size of Contention Window
- **DSRC** - Dedicated Short Range Communication for vehicle-to-vehicle and vehicle to infrastructure, with a reserved frequency band for seven channels. A short to medium range wireless protocol for vehicle communication. [4] [15]
- **EDCA** - Enhanced Dedicated Channel Access [7/5]
- **ITS** - Intelligent Transportation Systems [31]
- **ITSA** - Intelligent Transportation Society of America [30]
- **IVC** - Inter Vehicle Communication [31]
- **MAC** - Medium Access Control
- **MANET**- Mobile Ad-Hoc Network [40]
- **NPD** - Tail drop policy , if no room in queue, new packet is dropped.
- **RSU** - Road Side Unit [37]
- **SNR** - Signal to Noise Ratio. (SNR > 1:1 is more signal than noise) [12]
- **SNIR** - Signal to Noise to Interference Ratio
- **TNS** - Time sensitive networking
- **VANETs** - Vehicle Ad-hoc Networks [37]
- **V2V** - Vehicle to Vehicle communication [37]
- **V2I -** Vehicle to Infrastructure communication [37]
- **WAVE** - Wireless Access in Vehicular Environments [22]
- **WSM** - Wave Short Message to represent data packet
- **BSM** - Basic Safety Message to represent a beacon
- **WSA** - Wave Service Advertisement to announce the next selected SCH.

List of Technical Terms

Multi-channel Communication	*A scenario using the control channel and service channels to receive safety and non-safety messages.*
Service Channel (SCH)	*The channel allocated to receive data messages, also known as non-safety messages. There is typically more than one service channel.*
Control Channel (CCH)	*A channel allocated to receive broadcasted safety messages also known as beacons. There is typically only one control channel.*
Dissemination Protocols	*Protocols that brake the packet/message into blocks, to be transmitted in consecutive intervals 14.*
Density Threshold	*The number of vehicles in a simulation. Some previous studies also define this as the maximum number of vehicles that the simulation can include. It does not mean that this number of vehicles was reached during the simulation period.*
Packet Delay	*Time duration from packet being sent to a packet being received.*
Throughput	*Packets delivered over Mbps (Megabytes per second).*
Capacity	*The maximum number of vehicles that can operate during the simulation scenario.*

Abstract

VANETS is an ad hoc network in vehicles with wireless communication capability. The network utilizes a system to relay data from one vehicle to another vehicle or to a Road Side Unit (RSU). This communication is also known as Vehicle to Vehicle (V2V) [31] and Vehicle to Infrastructure (V2I) [31]. The communication protocol for Wireless Access in Vehicular Environment (WAVE) [10], is the industry standard IEEE 802.11p to communicate between vehicles. This thesis examines the Medium Access Control (MAC) layer of this IEEE 1609.4 multi-channel communication protocol. In Dedicated Short Range Communications, the core of the WAVE protocol, there is an allocated spectrum in the frequency area of 5.9-GHz [20]. In the U.S, the allocated spectrum of 75 MHz was split into seven channels. A channel is defined as a frequency range of 10 MHz for a radio to tune into [28]. There is a control channel to relay safety messages and six service channels to relay non-safety messages, giving us two types of channels to choose from when in message transmission. Both the type and priority of the message are the factors considered. Many existing studies illustrate the impact of multi-channel and single-channel switching for non-safety and safety message transmissions. Most studies focus on optimizing the usability of the service channels. This thesis aims to determine the best use of the single radio in a vehicle i.e. to best utilize the Control Channel (CCH) and Service Channels (SCHs) in a Single Radio Multi-Channel (SR-MC) system [20]. We analyze the channel utilization, beacon transmission, and packet transmission of IEEE 1609.4 multi-channel operations in CCH and SCH. Some of the parameters used for comparison are the number of collisions, channel utilization, packet

transmissions, and beacon transmissions. We investigate the scenario with density of n vehicles in a real world map, using safety (beacons) and non-safety (data) messages. The technologies used are Instant Veins 4.6, OMNET++ 5.2.1, SUMO 0.30, Debian GNU/Linux 9 (stretch) 64-bit, VMware Fusion (Professional Version 10.1.4) and an open street map from Northampton. The advantage of using OMNeT++ and Simulation Urban Mobility (SUMO) framework is the thorough implementation of IEEE 1609.4 DSRC/ WAVE and IEEE 802.11p in the framework [29]. Additionally, important feature of realistic traffic along with factual map can be generated with SUMO [21]. The contributions provided in this thesis include the integration of the testing framework Catch, randomizing the SCH, adding beacon transmission to the MAC layer, tracking of vehicle neighbors, tracking of collisions, and channel utilization. Plus an analysis on multi-channel switching. In our results we found that the CCH is highly overloaded both with beacon and channel switching management, which has a strong impact on the switching operation with a high number of collisions. Furthermore we also found that as the number of beacons generated increased, there was an increase in lost frames independent of the channel . Lastly there was little fluctuation in the number of collisions with a higher “n” of vehicles.

Keywords

vehicular networks, multi-channel, IEEE 1609.4, channel utilization

CHAPTER 1: Introduction

The Vehicle Ad Hoc Network (VANET) is a new type of network infrastructure integrating wireless communication between vehicles. VANETs [37] construct the standards for sending, receiving and handling data messages via a single radio system in a vehicle jointly with the GPS application. The goal of a VANET protocol is to optimize packet transmissions by increasing delivery rate for all messages, resulting in reduced delays for safety messages [15]. Countries leading the automotive industry, have research emphasis on VANET protocols to enable the communication between vehicles. Therefore, the participating countries of Japan, U.S., U.K. and Germany have implemented standards to support multiple channel services with a Control Channel (CCH) and a set number of Service Channels (SCHs). For example, there is a total of seven 10 MHz channels in the U.S. and five in Europe. Followed by the communication protocol for Wireless Access in Vehicular Environment (WAVE) [10], which is the industry standard IEEE 802.11p to communicate between vehicles.

The remainder of this chapter is structured as follows: Section 1.1 (VANETs) defines the communication between vehicles, how it is used and the factors in the network. Section 1.2 (Background) covers VANET applications, communication protocols, protocol architecture, IEEE 802.11p ,DSRC Protocol, WAVE Protocol and IEEE 1609.4 Multi-Channel Protocol. Section 1.3 (Problem Statement) defines the

problem addressed in this thesis and our assumptions. Section 1.4 (Thesis Objective) defines the objective of this thesis and how it is related to the problem. Section 1.5 (Thesis Contributions) List the contributions of thesis and how they can be used in further research. For more information, use the list of acronyms and technical terms as a reference for the thesis.

1.1 VANETs

In VANETs, vehicles transmit messages Vehicle to Vehicle (V2V) [37] and Vehicle to Infrastructure (V2I) [37]. This communication can be used to create a decentralized network with the vehicles. The Intelligent Transportation Systems (ITS) [31] uses the communication systems to send high priority infotainment messages. Packet objects are categorized as safety and non-safety messages. Communications types are categorized by either infrastructure mode or ad-hoc mode. In the first mode, at least one member is the Access Point (AP) [30]. The member is a recipient of messages from other vehicles or Road Side Units (RSUs) [37]. In the ad-hoc mode, each member can be the access point (AP) for a new member. Some of the factors impacting the Time Sensitive Networking are the vehicle mobility, redirection, scalability, geographic location, and wireless interference from physical obstacles. To get a better idea on the reasons for this problem, the following section will lay out the context.

1.2 Background

VANETs have a strong influence in trafficking information. They construct a mobile network using vehicles as wireless connection nodes for transmitting data. The data is transmitted to Road Side Units (RSUs) [37] or other vehicles. RSUs are defined as

stationary access points to the network. In the network, data is transmitted by hopping between nodes to reach the destination. In this paper, we present an overview of VANETs focusing on the MAC layer and channel utilization. We elucidate the subject with a methodical approach. We begin with an overview of the VANET structure, applications, and challenges that it poses. We follow with current protocols and their classifications. We aim to answer the following questions:

- What are VANETS?
- What are are the current factors and challenges in this new type of network?
- What are the communication protocols?
- What is the VANET protocol architecture ?
- What are the standards used in this thesis?

1.1.1 VANET Applications

The leading application to VANETs is Active Road Safety. Road safety points to collision avoidance and traffic flow architecture. Ideally, with VANETs we want to minimize accidents by providing the driver with preemptive information. Generally, the driver receives rapid messages related to vehicles around it. For example, a driver will receive a message when the vehicle ahead of them comes to a sudden stop. Recently, the field of traffic flow research has gained much more attention. The capability of VANETs has motivated researchers to exploit traffic information to improve vehicle traffic control [2]. Other VANET application includes general use for personal communications and entertainment. This technology is capable of generating a network hotspot for internet access. This network facilitates rapid web access for police cars, UPS trucks, postal trucks, or any vehicle whose driver benefits from mobile access to perform their job.

1.1.2 Communication Protocols

In Vehicle Ad-hoc Networks, each vehicle is a wireless consumer as well as a provider for forwarding data to other transmitters. The VANET architecture applies the following three categories [5]:

- **WLAN/MANETs**[32]: Vehicle-to-Infrastructure (V2I) communication. The vehicles use base stations to access the wireless local network [40].
- **Ad-Hoc**: Vehicle-to-Vehicle (V2V) communication. The vehicles are the network forwarding data to other vehicles as well as sending message requests for themselves.
- **Hybrid**: Inter-vehicle communication (IVC) combining V2I and V2V. In this network, Vehicles can connect to RSUs as well as near vehicle nodes. This is beneficial because it offers constant accessibility to wireless connection, but its environment also produces a lot of routing challenges.

Part of the communication is the protocol architecture for the layers in the system. The following section will cover reasons behind the architecture and the protocol stack.

1.1.3 Protocol Architecture

VANET protocol design is identified for addressing unpredictable vehicle density, fast vehicle mobility, and rapid changes in network topology [11]. These issues are frequent contributors to wireless interference and link failure. Protocols use many techniques to reduce routing overhead and end to end delay while maintaining higher packet delivery ratio. These techniques include movement prediction, connection stability detection, channel selection based on stability, carry on forward hopping methods [24], geographic

vehicle trajectory statistics exploitation, and road system protocol adaptation. Behind the protocol design there is a general protocol architecture for the network.

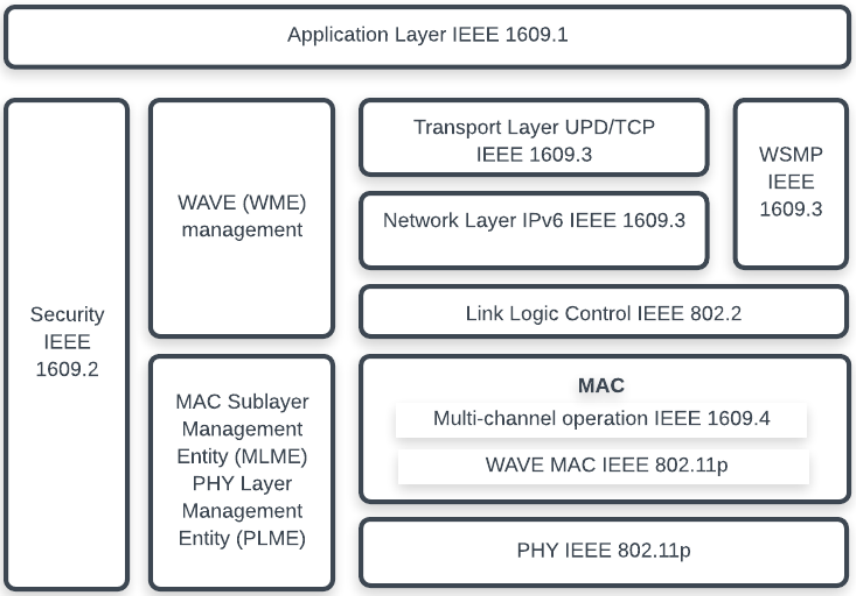

Figure 1.1 1609.4 WAVE architecture [30].

A network protocol architecture has a protocol stack. The protocol stack includes an application layer, transport layer, network layer, data link layer and a physical layer. Some newer additions are the security and medium access control (MAC) layer. Figure 1.1 above shows the protocol stack for VANETs. Standards have been defined by the industry of Inter Vehicle Communication (IVC) [31] making great progress in the last several years [22].

Some of the standard protocols displayed in Figure 1.1, are IEEE 802.11p and IEEE 1609.4 DSRC with WAVE [31] described in Secs 1.2.4-1.2.5. These compose the communication stack defining the physical and access layer in the inter vehicle communication (IVC) [31]. The following section, lays out the details of these standard.

1.1.3 IEEE 802.11p

The electromagnetic spectrum is a range of frequencies between 3Hz - 300 EHz as seen on the figure 1.2. As part of the electromagnetic spectrum, the radio spectrum of frequencies falls within the range of 3 kHz to 300 GHz [12]. The Intelligent Transportation Society of America (ITSA) [30] began designing IEEE 802.11p with the need to support Intelligent Transportation Systems (ITS) [31]. For example, existing applications once used to automate tolls had to share radio wave frequency with other systems. In 1997, ITSA proposed the Federal Communication Commission (FCC) to allocate 75 MHz within the range of 5.9-GHz for the communication in ITS applications. The European Telecommunications Standards Institute allocated a radio spectrum of 50 MHz for Europe. In 1999, the request for radio spectrum of a total of 75 MHz allocated in the frequency band of 5.85-5.925 GHz was granted by the U.S. Federal Communication Commission. That same year, the spectrum range was also granted to be divided into seven 10 MHz-wide channels.

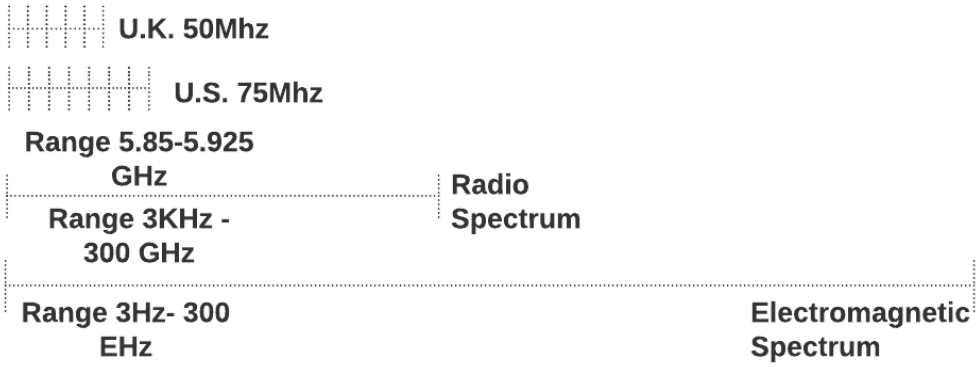

Figure 1.2 Frequency and Channel Allocation.

Operations like the multiple handshake in IEEE 802.11 carried too much overhead for vehicle communication. In 2010, the amendment IEEE 802.11p, also

known as WAVE Basic Service Set(BSS), was published to address the overhead. It specified the requirements in the physical and MAC layer for two VANET devices to communicate [37]. The IEEE 802.11p standard is implemented in the WAVE lower MAC and the physical layer of the communication stack. A main IEEE 802.11p feature provides the single-channel operation [31]. After having the availability to multiple dedicated channels, the next step was to standardize the most-efficient use of the frequency band [30]. The configuration was designed to implement the single-channel operation and simplify ad hoc BSS operations.

A Basic Safety Message (BSM) contains all the information it needs to be received, categorized, and transmitted among the Basic Service Set (BSS) of devices[37]. BSS VANET devices have parameters to operate at same rate to better communicate with each other [30]. To reach CA, it is assumed a 10 beacon message per second rate is adequate. [37]. The following list presents standards for IEEE 802.11p:

- **Beacon rate -** 10 beacons/second [37]
- **Max Communication range -** 1000 m [30]
- **Max Transmission Power -** 800 mW [30]

Along with IEEE 802.11p, the DRSC protocol was standardized to address the usability of the allocated spectrum. The next section will cover the details of this protocol.

1.1.4 DSRC Protocol

Dedicated Short Range Communications (DSRC) [37] was designed by the US Department of Transportation (US DOT) by a request of Congress in 1999. The job was to create a standard that would efficiently utilize the reserved spectrum [30]. The standards aggregation for DSRC included IEEE 802.11p and IEEE 1609.4 [22]. As in the figure below, the DSRC [37] spectrum was standardized for V2I communication and V2V communication.

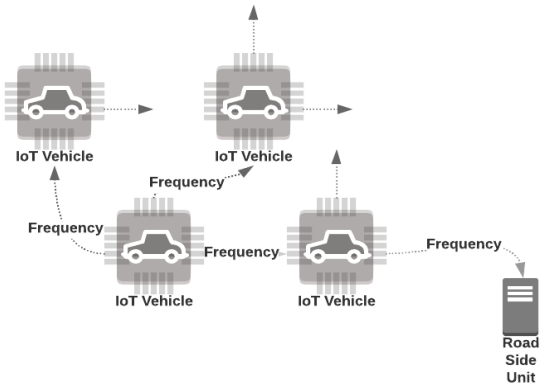

Figure 1.3 Dedicated Short Range Communication Protocol.

The channels are grouped by one Control Channel (CCH) for safety messages, identified as a default channel that all nodes can tune into, and six Service Channels (SCHs) for non-safety messages for any public use [30]. The infrastructure standard applied to the physical layer [31] is a modification of the IEEE 802.11a OFDM physical layer [30]. The standards design focuses on transmitting packets between V2I and V2V, with a reserved frequency band for seven channels. The DSRC research results from the American Society for Testing and Materials (ASTM) published in standard specification ASTM E2213 [30]. Due to the implementation complexity of E2213, a study group in 2004 published the amendment in 2010. It was later the IEEE 802.11p release of 2012.

1.1.5 WAVE Protocol

Because all vehicles must have the ability to communicate to each other, standards for the protocol communication have been set by the IEEE working group. One identifier among different types of networks is the frequency allocated for that specific network type. The frequencies are allocated for standard channels to be able to connect and communicate with other nodes. In vehicle networks, this is known as an Intelligent Transportation Systems (ITS) Radio. In order to guarantee communication across different vehicle systems, the working group released Quality of Services (QoS) requirements to regulate the Wireless Access in Vehicular Environment (WAVE). [15] The availability to switch between the CCH and the SCH was standardized as the WAVE Split Phase Protocol. This requires that vehicles synchronize every second via the GPS. To meet this requirement, the radios switch between channels in intervals of 50 ms [22]. Following is the definition of the IEEE 1609.4 Multi-Channel Protocol.

1.1.6 IEEE 1609.4 Multi-Channel Protocol

The IEEE 1609 working group, sets the higher layer criteria based on 802.11p [2]. The group is show in Table 1.1. It is composed of 1609.1 resource management services [2], 1609.2 security services, IEEE 1609.3 network services, and IEEE 1609.4 channel management services [30]. The 1609.4 service is used as the standard that defines the functions of multi-channel operations. The implementation is integrated as a layer on top of IEEE 802.11p [37]. The IEEE 1609.4 standards contribution is the channel switching operation applicable to the MAC layer of the infrastructure. [31] The figures below list the specification descriptions for each of the IEEE 1609 amendments.

Amendment	Description	Implementation
IEEE 1609.1	Defines the resource manager	**Application Layer**
IEEE 1609.2	Defines the format and processing for secure messages	**Security Layer**
IEEE 1609.3	Defines the addressing and routing of packets with secure data exchange	**Network Layer** **Transport Layer**
IEEE 1609.4	Defines enhancements of IEEE 802.11p multi-channel operation	**Mac Layer**

Table 1.1 IEEE 1609 Standards.

The standard designates the seven channels to specific roles. One channel is designated as the Control Channel (CCH) and the rest as Service Channels (SCHs). The multi-channel operation switches between the CCH and SCH in intervals of 50ms [37]. The intervals are separated by a short guard interval, to handle any unexpected asynchronization [22]. The channel switching operation requires radio systems to simultaneously tune back into the CCH using the GPS [30]. The figure below illustrates the channel intervals and synchronization to support channel switching.

The Control Channel in Figure 1.4 is responsible for receiving, sending, and transmitting safety and management information [30]. Safety messages are exchanged in format of Basic Short Messages (BSMs/beacons) [37]. To create a cooperative awareness between devices, beacons are sent and received using the Wave Short Message Protocol (WSMP) [37].

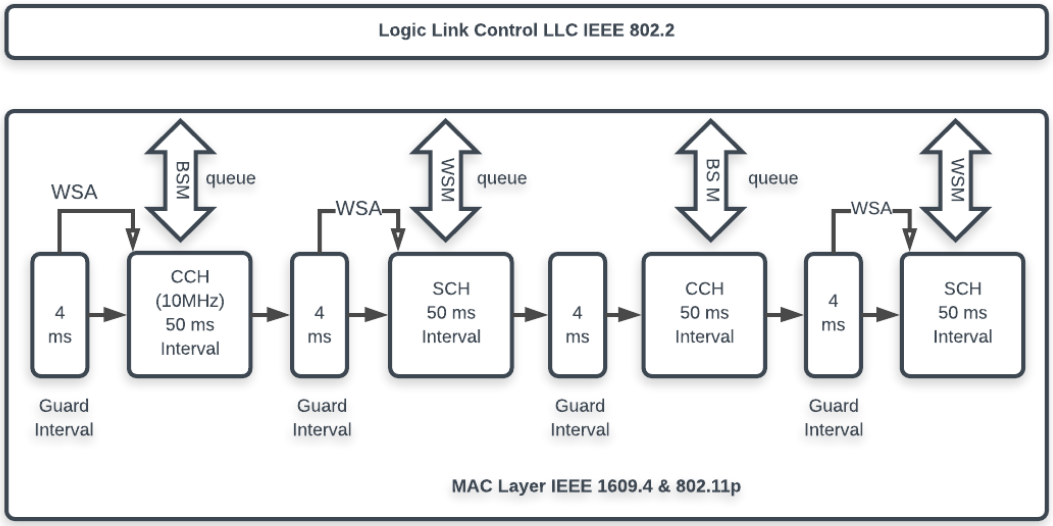

Figure 1.4. IEEE 1609.4 Multi-Channel Operations and synchronization[2].

The CCH sends Wave Short Announcement (WSAs) to announce the Service Channel selected to transmit the upcoming WAVE short messages (WSMs) [30]. A Service Channel's role is to manage other types of messages identified as data or non-safety messages. This includes any information that is low priority and unrelated to infotainments applications.

In [22], the following criteria were identified to consider the control channel operation:

- Given a vehicle system is in the beginning of the CCH, the CCH needs to calculate the time to send the channel selection Wave Short Announcement.
- Given a vehicle system is in the CCH, it must calculate up to the time period in the interval it should continue receiving BSMs.
- Given a vehicle system has received any request to use a specific SCH when it is in the CCH, then it must decide which SCH to use for the upcoming WSMs.
- Given a vehicle system is in the CCH, it should calculate a time target by when to

select the SCH.

- Given a vehicle system has selected the SCH to use, it should calculate time to send the WSA.
- Given a vehicle system has switched to the selected SCH, it should decide when to send data messages (WSMs).

Now that we have covered the background of VANETs, the standards and the multi-channel operation, it is important to look at how to best improve the VANET protocols. The next section will define the problem addressed in this thesis and why.

1.3 Problem Statement

Much attention was given to the unknown impact and efficiency of the multi-channel operations in 1609.4. In this thesis, we analyze the channel utilization, beacon transmission, and packet transmission of IEEE 1609.4 multi-channel operations in CCH and SCH to identify the points of impact in Vehicular ad-hoc Networks. The existing complexity of protocols presents a risk in this problem. Several trigger points for sending, receiving and generating collisions of messages also exist. The assumptions of the expected behavior were based on the implementation/configuration of the traffic simulation. This presented a compound risk of missing some of these trigger points due to the complexity of the existing codebase. Addressing this problem is important to the VANET research community. The objective behind it is laid out in the following section.

1.4 Thesis Objective

As indicated earlier, the goal of this thesis was to determine how to best use the single

radio in a vehicle i.e. to optimize Control Channel (CCH) and Service Channels (SCH's) in a Single Radio Multi-Channel (SR-MC) system with IEEE 1609.4. The parameters recorded to diagnose the operation include the following: collisions, channel utilization, packet transmission, and beacon transmission. We considered the scenario with density of n vehicles in a real world map, using safety (beacons) and non-safety (data) messages. We intended to define the performance benefits and drawbacks of IEEE 1609.4 multi-channel operations in safety and non-safety messages by evaluating : traffic against networks congestion, application layer against MAC layer message handling and channel utilization.

1.5 Thesis Contributions

The following summarizes the contributions of this thesis in the area of VANET:

- We added the following features to the simulation: Catch testing framework integration, SCH randomization, MAC layer beacon transmission, vehicle neighbors tracking and channel utilization tracking. These features were targeted to better judge the problem of performance, impact and efficiency of multi-channel operations. Collectively these features can be used when analyzing other problems in VANETs. Our implementation of these features can be found at *https://github.com/surod22/VeinsVanetSimulation-2018*. The repository includes the source code of the simulation, the framework veins used and documentation to run it.
- We proposed an analysis on traffic congestions versus network congestions, MAC layer message handling versus application layer message handling and the

MAC layer channel utilization for CCH/SCHs. We found the control channel to be highly overloaded by handling beacons and multi-channel operations. Many of the previous studies had schemes with a focus on the usability of the SCHs. In our study the CCH was the greatest point of impact and caused the most collisions. With our findings we recommend to put a stronger emphasis on reducing the responsibilities of the CCH. Putting this into effect, future proposed schemes will target a larger point of impact of multi-channel operations in IEEE 1609.4.

1.6 Thesis Outline

The remainder of this thesis is organized as follows:

- **Chapter 2 -** Introduces the basic components of VANET applications, communication protocol architectures, routing protocols for VANETs, protocol design, IEEE 802.11p, DSRC, WAVE, and IEEE 1609.4
- **Chapter 3 -** Presents the simulation scenario, with an overview of the software components, OMNET++ architecture, SUMO architecture and Veins architecture. Followed with the setup of the simulation.
- **Chapter 4 -** Covers the research features implemented. These features include Catch framework integration, MAC layer beacon transmission, vehicle neighbors tracking and channel utilization tracking.
- **Chapter 5 -** Presents an analysis of the multi-channel operation. It examines the application layer message delivery, the MAC layer message delivery, the MAC layer utilization and the traffic congestion against the the network congestion.
- **Chapter 6 -** Covers the results for network congestion against traffic congestions, message delivery in the application layer, message delivery in the mac layer and channel utilization.
- **Chapter 7 -** This section summarizes our conclusions and potential future work.

CHAPTER 2: Literature Review

VANET's area of research is fairly new compared to other networks line LAN, WLAN or MANETs. There is a large span of research topics considering the different network layers. To identify the most impactful trigger points in IEEE 1609.4 multi-channel operations, we carefully select a set of thirteen research papers. In this next section we will be covering the evaluation of these papers.

2.1 Previous Work

When investigating the definition of the connection between IEEE 802.11p and MAC 1609.4, particular parameters and metrics were used. These include IEEE 802.11p, MAC 1609.4, OMNET++, Veins simulators, multi-channel, safety messages, non-safety messages and beacons. We searched for references with proposed schemes, evaluations, published dates of 2010-2017 and a robust citations per published year. We aimed to identify the latest research trends in the field. We studied patterns in the number of channels used, determined their main objectives, and documented the problems addressed as well as any contributions, results, and evaluation metrics. The next section will cover the details of the patterns found previous work.

2.2 Number of Channels

Of the thirteen studies covered, studies [3], [30], [22], [37],[2], [14], [8], and [20] used one CCH and six SCHs. Studies [3], [21], and [22] used one CCH and four SCHs. In scenarios with a total of seven channels, the studies applied the U.S. standards rather than the U.K. standards which has five channels.

2.3 Range of Objectives

In [3], [7], [14], and [8], multi-channel operations in VANET IEEE 1609.4 were investigated and evaluated. The investigations of [14] and [8] focused on safety applications. Collectively, evaluation of [3] and [7] focused on both safety and non-safety applications. [20] and [7] studied how to increase performance by examining message delivery of frames. Studies [26], [21], [22], and [37], shared the common objective of analyzing and optimizing beacon delivery with multi-channel operations.

2.4 Overview of VANET Problems

Inquiries in this area of research address the efficiency of bandwidth use in the channel switching procedure [3] [21] [22] , communication delay on beacons [15] [26] [22], impact of 1609.4 channel scheduling [15], effective use of channel resources of technologies on a single radio [14] [8], best modeling for VANET mobility [28], impact of synchronous channel switching 1609.4 on delivery rate [37] [2], and collisions [23].

2.5 Scope of Previous Contributions

In the following, we introduce a set of model solutions previously evaluated. These are supported by the evaluation of efficiency of multi-channel operations. For example, we drew a comparison between multi-channel and single-channel performance with a realistic dataset [3] and analysis with a safety dissemination model. A closer look at the study of 1609.4 [15] showed that the tight channel synchronization issues foreseen by the protocol could dramatically impact the performance of safety-related applications with strict delivery ratio and delay requirements. Furthermore, it proposed two new

enhancements for the WAVE protocol stack to favor the dissemination of safety messages in multi-channel operations. The solutions were a group of dedicated solutions for message delivery and dissemination.

The WAVE-enhanced Safety message Delivery scheme (WSD) quickens the transmission of safety messages in multi-channel VANETs, while preserving compatibility with the IEEE 1609.4/802.11p standards [8]. The analytical model in [14] also proposes a WSD to enable accelerated dissemination of safety messages over multi-channel operations, whereas [9] provides a simulation scenario of varying traffic density using Flooding protocol in an urban scenario.

In order to evaluate the performance of dissemination protocols, a separate group of solutions analyzed [37] the performance of beaconing and showed existing solutions to minimize the impact of channel switching. Approach [26] uses a mathematical problem in a realistic simulation to calculate a Probability of Beacon Delivery (PBD) value. Study [21] explores the potentials of a multi-channel approach by extending a delay sensitive scheme. It extends the delay sensitive scheme and congestion aware Adaptive Traffic Beacon (ATB) protocol. This uses IEEE 802.11p/ 1609.4 DSRC/WAVE to its best potential. The scheme is then promoted in [22], with a Multi-Channel Beaconing Protocol (MCB). It is based on ATB, which is designed to better use the added SCHs of the DSRC band [22]. MCB implements new techniques to address the network topologies and extra delays in the split-phase network operation [22]. The investigations in [21] and [22] attest to the CCH operations. A new MAC contention control mechanism [7] is produced, adapting to the multi-channel operations of the 1609.4. In [20], two algorithms are proposed to optimize the timeshare of CCH

with non-safety applications. Lastly, in approach [4], a VANET Car Mobility Manager (VACaMobil) for OMNeT++ was evaluated. We will now examine the types of metrics used to evaluate the approaches mentioned.

Immediate and Extended Access [7]

Problem	Need to reduce the waste of remaining time in SCHservice channel
How it works	Defines how to use the remaining time in the interval in the CCH after transmission is complete
Pros	● Improves bandwidth usage for non-safety applications by switching for the remaining time.
Cons	● Does not address minimizing the impact of channel switching ● The scheme improves at the cost of CCH interval

Fragmentation Scheme [7]

Problem	Need to reduce bandwidth waste in the SCH
How it works	When there is not enough time remaining in SCH to transmit the next package, the packet is split into two fragments: one for the remaining SCH time and one for the next SCH interval.
Pros	● All service channel time is utilized
Cons	● Requires an extra header for fragmented packet

Best fit scheme

Problem	Need to use time remaining in SCH service interval
How it works	By reviewing the queue, a packet is chosen with less transmission time than the remaining SCH time
Pros	● More efficient performance than fragmentation when packets with different sizes are present in the queue
Cons	● Difficult implementation ● Need to track packet sizes (assuming packets are of different sizes) Constant changes on the following packet in the transmission queue creates challenges in determining the reason for contention and transmission

Exploitable node-assisted WBSS Broadcasting Mechanism (WBSS - Wave Basic Service Set)

Problem	Need to reduce collisions when nodes are broadcasting WBSS simultaneously due to CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance), a protocol that aims to prevent collisions
How it works	A exploitable node in the SCH broadcasts the WBSS to new vehicles and is acknowledged when other nodes are transmitting data to the RSU. The scheme defines a near area for higher priority and a far area for lower priority.
Pros	● None reported
Cons	● Does not address channel switching impact ● With CSMA/CA - when node is unable to send a packet if in the same channel and at the same time. ● The scheme is not standardized and has high complexity. ● Nodes must calculate their relative location to the RSY.

ATB [21]

Problem	Creates a beacon interval via the channel quality and priority of the message to prevent wireless collisions by sending beacons as often as possible without overloading.
How it works	Scenarios: 1. Artificial case with simple detours 2. Freeway Rural area, using Breitenbach am Inn/Tirol as a primary example Parameters: ● Vehicle rate: departing every 5 minutes ● Two lanes allowing vehicles to pass each other 1. Events artificial accident starting at a fixed point 2. Vehicle stops for a duration to broadcast warning 3. ATB or ATB-MCH transmission Scenario of beacon with high priority to specific channel, node without fingerprinting tunes in to receive message.
Pros	● Higher traffic increases beacon transmissions, leading to higher number of collisions

Cons	● Static traffic beaconing is not appropriate for every traffic scenario ● Does not focus on minimizing channel switching impact ● The scheme improves at the cost of CCH interval ● Low traffic may allow to large of a beacon interval

Table 1.2 Schemes Related to the Problem Bandwidth Wastage of SCH.

Table 1.2 compares several schemes illustrating the problem of bandwidth waste in the service channel. The selection criteria for the schemes used was in relationship to traffic Information System for IVC using beacons and approaches to multichannel scheduling for single and multi-radio environments [5,1,6]. The schemes address several targeted problems. Following the Pareto Principle, these trigger points assist in identifying the top 20% of variables and metrics to provide a stronger analysis.

2.6 Results of Previous Studies

Reviewing contributions in VANETs' area of research, we will categorize successful and unsuccessful results. This will illustrate immediate feedback on key trigger points of efficiency in the multi-channel operation of 1609.4. The evaluation of [3] found the delay of single-channel protocol IEEE 802.11p was reduced by 36% in comparison with its multi-channel protocol counterpart. We conclude that the parameters used in this study focused on packet/beacon delay, throughput, and lost packets [3].

Additionally, [15] shows that tight channel synchronization dramatically impacts the performance of safety-related applications with strict delivery ratio and delay requirements. Because the function of a safety application is to deliver beacons (safety messages), we see a pattern form between results from [15] and [26]. The impact of

safety applications is similar to the low Probability of Beacon Delivery (PBD) [26] in a large area with huge density of cars. Luckily, the proposed algorithm in [21] and [31] targets some of the tight channel synchronization by measuring the channel quality, message usability, and beacon interval [31]. ATB demonstrated the feasibility of its multi-channel approach, illustrating the reduction of channel utilization and successfully observed packet collisions without sacrificing throughput. The proposed algorithm MCB [22] focused on selecting the time to send coordination information. The algorithm reduced effects on the CCH while improving beacon delivery and reliability [22].

When comparing single-hop and multi-hop, the schemes recorded average delay, packet delivery ratio, packet loss, and throughput [2]. The proposed conclusion of these metrics was that the packet delivery rate and throughput of safety applications in multi-channel were enhanced [2]. Furthermore, both the scheme WSD [14] and the scheme of [8] effectively reduced the delayed delivery ratio and delivery delay accordingly. In [9], the simulation shows that higher traffic density results in an increase in the number of collisions. This negatively impacts the performance of dissemination protocols. After covering some of the findings in the literature, the next section will define the parameters/metrics used to evaluate the studies.

2.7 Spectrum of Performance Metrics

In the context of multi-channel switching, different parameters are reviewed to analyze the impact of 1609.4 multi-channel operations. Because most of the solutions covered explore the efficiency of the multi-channel operations, the comprehensive set of metrics vary.

Reference	Parameters and Metrics Used
[3]	1. Beacon ***Delay*** *for safety and non-safety applications* 2. Data ***Delay*** *for safety and non-safety applications* 3. Beacon ***Throughput*** *for safety and non-safety applications* 4. Data ***Throughput*** *for safety and non-safety applications* 5. ***Lost*** *Packets for safety and non-safety applications*
[15]	1. Delivery ***delay*** *per* ***number of vehicles*** 2. *Packet delivery ratio per distance (meter)*
[26]	1. ***Lost*** *Beacons per total* ***number of vehicles*** 2. *Probability of Beacon Delivery(PBD) Vs. Analytical Model* 3. *PBD per street for different types of streets* 4. *PBD per street different* ***number of vehicles*** 5. *PBD of straight streets vs Crossroads (2 region,100m area with 50 cars)*
[21]	1. Channel ***Utilization*** 2. Packet ***Collisions*** ***3. Beacon Interval*** 4. Relative ***Utilization*** *for benefit of fingerprinting* 5. Packet ***Collisions*** *for benefit of EDCA*
[22]	1. *Reliability* 2. *Channel load* 3. *Performance* ***4. 90 % vehicles and 10 % trucks*** 5. *Vehicle Density for medium utilized freeway* 6. *Vehicle Density for high road traffic jam* 7. ***Channel Utilization*** *vs (medium and high traffic jam)* 8. *Packet Success Rate vs (medium and high traffic jam)* 9. ***Beacon Interval*** *eCDF vs (medium and high traffic jam)* 10. *Informed vehicle in % vs (medium and high traffic jam)*
[37]	1. *CUmin* 2. *CWmax (Control Window Max)* 3. *AIFSN for VI (Video Traffic)* 4. *VO (Voice Traffic)* 5. *BE (best effort traffic)* 6. *BK (background traffic)* 7. Beacon ***generation rate*** 8. Data ***rate*** 9. *EDCA class* 10. *Beacon size* 11. *CWmin* 12. *Scheduling* ***13. Simulation time***

[2]	1. ***Simulation time*** 2. *Range transmission* 3. ***Number of vehicles*** 4. *Channel **data rate** ®* 5. *Number of channels* 6. *SCH interval* 7. *CCH interval* 8. *Guard interval*
[7]	1. ***Packet delivery ratio*** 2. ***Throughput*** 3. ***Channel utilization***
[14]	1. *Average Delay of Safety Messages(s)* 2. ***Packet Delivery Ratio*** 3. *Packet Un-Transmitted Risk Index* 4. ***Delay** percentile*
[8]	1. *Packets Un-transmitted Risk Index (PURI)* 2. *Average delay of safety messages* 3. *Delay percentile* 4. *Probability of Successful Delivery (PSD).*
[9]	1. ***Collisions** per traffic (vehicles/km2)* 2. *Packet loss per traffic* 3. ***Delay** per traffic* 4. ***Density*** 5. *Access Category* 6. ***Transmission Rate*** 7. *Size of the Data message* 8. ***Number of messages** produced*
[20]	1. *Successful **transmission rate*** 2. *Normalized Control Channel Interval*
[28]	1. *Average **number of vehicles*** 2. *Vehicle number* 3. *Standard deviation*

Table 1.3 Performance Parameters and Metrics for the State of the Art .

The common metrics used to analyze the area of research are summarized in Table 1.3. The metrics included beacon delay, data delay, lost packets, packet collisions, channel utilization, throughput, packet success rate, vehicle density (number of vehicles), packet delivery ratio, beacon interval, and beacon/data transmission rate. These common metrics are recognized in **bold** in the table. Based on the results of previous studies, we gathered that several of the metrics used included packet/beacon

delay, throughput and lost packets, strict delivery ratio, and delay requirements. This exhibits a clear pattern of parameters and metrics used, which are refined for inclusion in the analysis used in this thesis.

2.8 Topics in Future Work

This section combines the diverse set of recommendations of future work from the studies covered. In the comparison [3] of multi-channel and single-channel performance, the scenario evaluated vehicle densities with different speeds in the same lane. The authors wanted to further investigate a various number of vehicles in a different transmission range with varying speeds while taking lane change into account. A recommendation from [2] specified a scheme that could minimize the impact of the channel switching and improve the beaconing (i.e., CCH) performances. The proposed WAB scheme of [8] dynamically adapts the contention level of the control channel through a cooperative channel load estimator. The recommendation of future work is to further analyze the WAB's adaptation for event-driven safety applications and cover large-scale urban scenarios. Similarly, the authors in [20] extend the proposed paradigm to the multi-hop environment and introduce an analytical model for the WSD scheme. Considering the standards per country, the authors suggest simulating traffic of the city [20] using a different set of technologies/standards for comparison with 802.11p. For ATB in [22], a further extension would be investigating a beacon interval configured lower than one sync interval in WAVE. Lastly, in [13], they recommend further use of realistic scenarios and different dissemination for the simulation.

2.9 Simulation Engines

There are several simulation engines available for VANET protocols. Some of the existing engines are ns-2, ns-3, OMNET++ and Jist [30]. Ns-2, ns-3, and OMNET++ use C++ language while Jist uses Java. Each of the engines uses a network framework that contains the basic components to construct a network standard. Ns-2 and ns-3 use their own library. The main differentiation is that the ns-2 library is in Objective Tcl, while ns-3 is in Python. As a rewrite of ns-2, ns-3 improved scalability, extensibility, and modularity [30], OMNET++ uses INET for its network simulator. Ns-3 roots back to the NEST prototype from 1989 [30].

Reference	*1 [3]*	*2 [15]*	*3 [26]*	*4 [21]*	*5 [22]*	*6 [37]*	*7 [2]*	*8 [7]*	*9 [14]*	*10 [8]*	*11 [9]*	*12 [20]*	*13 [28]*
OMNET++ Sumo Veins 1	*x*		*x*	*x*	*x*	*x*					*X* *4.2.2*		*x*
Ns-2		*x*					*x*	*x*	*x*	*x*		*x*	

Table 1.4 Simulation engines and frameworks used in previous studies.

Name	**Tech stack**	**Usability**	**Requirements**
ns-2	**Network Simulator-2** C++ Network library: ns-2(Objective Tcl) Dependencies Tcl/Tk, Otcl and TclCL Open source 1989-2011	- High complexity - Limited functionality - Automated testing under development - Map-realistic based (can integrate with SUMO)	No IDE C++ compiler Unix (FreeBSD, Linux, SunOS, Solaris) Windows (Cygwin) Disk space: 320 MB to build 24MB for a 10K node [55]

Ns-3	**Network Simulator-3** C++ Network library Ns-3 (Python) Open source 2006 - now	- Scalable - Modular - New modules under development - Map-realistic based (can integrate with SUMO)	Ubuntu 18.04 (64 bit) with g++-7.3.0 and Python 2.7.15 Ubuntu 16.04 (64 bit) with g++-5.4.0 and Python 2.7.12/3.5.2 Fedora Core 28 (64 bit) with g++-8.1.1 and Python 2.7.15/3.7.0 Fedora Core 26 (64 bit) with g++-7.3.1 and Python 2.7.14/3.6.5 macOS High Sierra 10.13.5 with Xcode 9.4.1, Apple LLVM version 9.1.0, Python 2.7.10 [6]
JiST	**Java in Simulation Time** Java Network library SWANS (Java) Open source: 2005 -	- Transparent - Efficient - Standard - Object models - Scalable - Map-realistic based (undefined)	Virtual machine based Disk space: 20MB for 10K node Simulation [5]
OMNET++	**Objective Modular Network Testbed in C++ [27]** C++ Network library INET (C++) VANET library Veins with Sumo Open source 2008 - now	- Configurable parameters for simulation scenarios - Modular - Component-based - Complex to expand - Complex to automate tests - Map-realistic based with SUMO	Optional IDE base on top of Eclipse Graphical Interface Release 5.4.1 supports Windows 7 and 10 / 64-bit MacOS 10.12 Linux x86; 64-bit. Distributions: Ubuntu 16.04 LTS, Fedora Core 25, Red Hat Enterprise Linux Desktop Workstation 7.x and OpenSUSE 42 [27]

Table 1.5 Network simulation engines and frameworks [30].

To illustrate the frequency at which each simulator was used, Table 1.4 documents the simulator used for each study mentioned. Table 1.5 gives a detailed description of existing simulation engines in the area of VANET. It is important to note in the usability column, OMNET++ is identified as configurable. In this this a spike was done to identify the most ideal engine to use. Aiming to produce a realistic scenario with a concrete representation of the VANET standards, the pair of OMNET++ and Veins was selected. The following chapter will describe the components of the simulation scenario and how to set it up.

CHAPTER 3: Simulation

This chapter will cover a description of each of the simulators OMNET++, SUMO, and Veins. Each description will provide an overview of the architecture within the technology and its modularity. This section will also include the documentation on how to set up the simulation. Our simulation source code can be found at https://github.com/surod22/VeinsVanetSimulation-2018.

3.1 Software components

To simulate a vehicle's network scenario, the following software frameworks were used: Omnet++, SUMO and Veins. Omnet++ is an extensible and modular framework used to create different protocols. In the case for IEEE 1609.4 MAC layer in IEEE 802.11p, the extension is Veins. Veins is defined as a "fully-featured IEEE 802.11p and IEEE 1609.4 simulation model" [31]. SUMO is a realistic road traffic simulator that uses real street maps from Open Street Map to replicate the road system. Veins is the framework that connects Omnet++ and SUMO with a TCP socket. As each of the vehicles changes in position, direction, and speed in SUMO, the information is communicated to omnetpp through the socket. Omnetpp then replicates the traffic state and its effects on the network state. Inet 2.5 is a library that has a more extensive implementation of protocols and applications. These include IPv4, IPv6, TCP, SCTP, and UDP. The simulation was setup in a Debian virtual environment for the compatibility with the simulator versions stated. This section covers each of the frameworks involved in the simulation scenario. It will additionally provide a high-level summary of the physical modular architecture, communication layer architecture, and how to set up the simulation.

3.2 OMNET++ Architecture

OMNET++ is a framework that has the platform and taxonomy of a basic network layer. OMNET++ has set up the basic hierarchy and communication structure of a basic network layer. It is a source of infrastructure and tools to create network protocols [1]. The architecture of OMNET++ was designed to write network simulations [1]

3.2.1 Architecture

OMNET++ has a generic, event-driven architecture. It is composed of simple modules, compound modules, gates, links, messages, channels, collection classes, NED modules, parameters, and the configuration file omnetpp.ini. The modules are linked by gates, which are defined and connected in the NED file. The simulation is configured by parameters that can be initialized in the omnetpp.ini file.

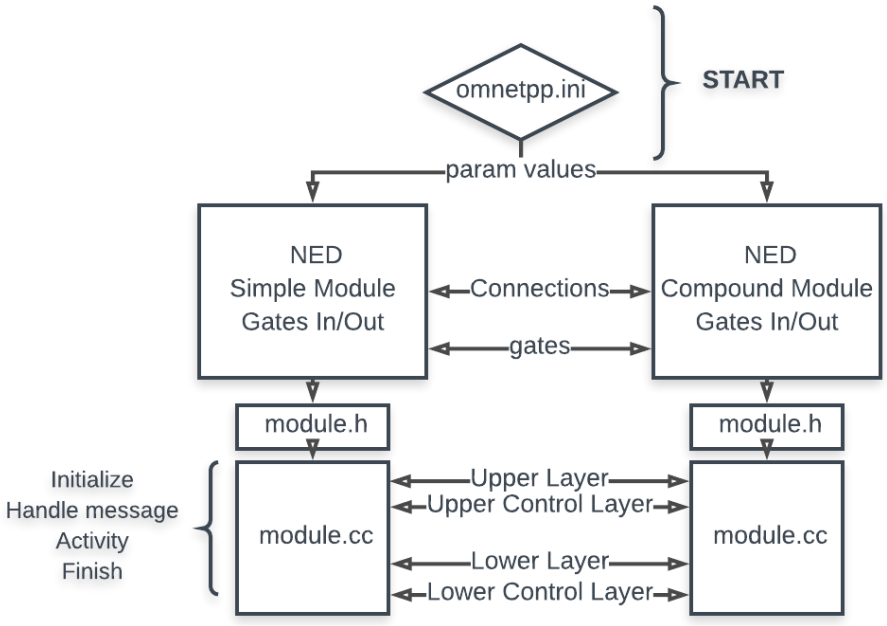

Figure 3.1 Basic layer taxonomy of OMNET++ architecture.

In Figure 3.1, we can see the base modules in OMNET++ and how they integrate. Modules are reusable components that derive more complex components of a network protocol [1]. A module can be used to create different components of the protocol like the transport, MAC and physical layer.

3.2.2 Modules

The base level of architecture is a simple module. The structure of a simple module involves the following inheritable virtual functions [19] :

- Initialize
- HandleMessage
- Activity
- Finish

The initialize function is called once the module is created. Following initialization, the module is triggered by receiving a message. This is where most of the business logic of getting information from the message, setting up for transmission to other layers, and scheduling the transmission happens. The activity function can also be used for the business logic instead of the HandleMessage. For complex model functionality, the HandleMessage is still preferable as the activity function makes it difficult to scale and debug. The finish function is typically used to clean up any instances and to finalize the recording of metric variables. All of these functions are the vanilla structure to use the basic modules. To use a simple or compound module type, the class must be subclassed, and these virtual methods must be implemented.

The next level of architecture covers compound modules. A compound module is a group of simple modules. The compound module represents the network of a vehicle incorporated in an application, Network Interface Card (NIC), and Veins mobility.

Modules are connected via ports identified at gates in the OMNET++ domain. A gate is an input or output of a module [1]. The gates in each module are connected via links. There is an input and output link for each layer within a module. The basic layers are used with the gates to communicate between two modules:

- Upper Layer
- Lower Layer
- Upper Control Layer
- Lower Control Layer

Modules communicate via messages. Messages are scheduled as an event to be sent with a time flag. When the time flag is triggered, the message is broadcasted to be received by other modules. As mentioned earlier, the message is received by the HandleMessage function. The message is handed down and dismantled to the lower layers. Once the message has been identified as received/ignored, another message is scheduled for acknowledgement when needed. Self messages can also be scheduled to trigger internal timers for certain functions to happen. An example may be when it is time to create and send a new packet.

A NED file is known as a network descriptor which contains all the gates, network components, connections, and parameters for a module. The NED descriptor only defines structure of the topology. The behavior is implemented in the C++ code. Configuration parameters are defined in the NED file as mentioned earlier. A default value can be initialized, and can be overwritten in the omnetpp.ini file.

3.2 SUMO Architecture

SUMO stands for Simulation Urban Mobility. It is an open source traffic simulator created by the Institute of transportation systems at the German Aerospace Center. According to the documentation, its focus is to produce and replicate large road networks [35]. SUMO has a feature to generate the road traffic network for the simulation from Open Street Maps. According to the Veins description, SUMO was integrated to “make simulations as real as possible, without sacrificing speed” [38].

Open Street Map is an open source map that depicts traffic infrastructure of the world[34]. These world maps are created from data input by people physically traversing roads. Some of the elements captured in this map include nodes as intersections and edges as streets [35]. An edge type is labeled as a streep type, which can be a highway, a railway, or a waterway. Some attributes that belong to the edge type include the number of lanes and the street speed. The map is in OSM (Open Street Map) XML form. Building objects are identified as polygon types. Some of these types include residential, industrial, commercial, parks, and military bases. More polygon types are identified within the road types link [36]. With the map osm xml file, the network xml file for SUMO can be generated.

```
$ netconvert --osm-files berlin.osm.xml -o berlin.net.xml
```

The SUMO network file is created with a network command as the one above [33]. The network convertor identifies default street types from the osm file. Provided a default street type exists in the osm file, a corresponding edge is generated in the network file. The main relationship between the osm file and the network file are the node IDs. The following section will cover the Veins architecture and modules.

3.4 Veins Architecture

Veins is an extension of OMNET++. It adds a top layer of protocol to represent the vehicle network architecture. It works to represent the communication of messages among nodes in the movement of vehicles. The following section will cover the underlying architecture of veins.

3.4.1 Architecture

At a high level, the vehicle network simulator is composed of three services. This includes the OMNET++ network simulator, a TraCI mobility simulator, SUMO as a traffic simulator and a SUMO-launched socket. The duties of the OMNET++ simulator include maintaining connections between nodes and wireless ad-hoc networks. The duties of the TraCI server are to control the steps of the of the vehicles. These include:

- Creating a vehicle
- Adding a vehicle to the queue
- Communicating when to execute the next step

The SUMO simulator is responsible for executing the next step in the traffic simulation. SUMO then returns the coordinates via the socket. Once TraCI receives the coordinates , OMNET++ updates the locations.

The TraCI manager couples OMNET++ and SUMO. The setup involves running a SUMO socket that listens for traffic networking instructions for the traffic scenario. As a vehicle position changes in the TraCI server, a command is sent via the socket. The SUMO socket listens on a specific port for the commands. When a TraCI command is received, SUMO is executed to simulate the step in the traffic interface. The SUMO

socket listens for connections on 127.0.0.1:9999. Once a connection is accepted, it reads a the launch configuration to extract the TraCI command. I then gets the sumo-launchd.launch.xml file to execute the SUMO command.

We now present an example of the SUMO communication with TraCI and OMNET++. First, the TCP connection is created. The TraCI client sends a message to set max speed. SUMO responds with status of max speed. TraCI sends a message to set simulation, and SUMO responds with a status after starting simulation. Then SUMO sends a message after each node is moved. An example of this scenario is when a tcp connection is created via a SUMO. The TraCI client will set the max speed using a command and subscribes to the command. SUMO returns an acknowledgement status. With the command subscription, the TraCI manager handles the response. It checks its time trigger and follows with a command to execute the next step. SUMO moves the appropriate vehicles to simulate the next time trigger and sends status response.

We will now present more detail for the execution tasks in the scenario above. The TraCIScenarioManager will initialize the command to send the launching file "sumo.launchd.launch.xml". Other things that the manager handles internally include subscribing to the list of vehicles and initializing mobility for position. It also takes care of executing a time step trigger, adding the mobility module, and updating the module position. For clean up, it will delete the managed module.

3.4.2 Vehicle Scenario

The scenario includes a controller for obstacles encountered in the SUMO simulation as well as the annotation manager, connection manager, a world utility, and a scenario TraCI manager. These modules are in the figure below. One thing to note: this module

does not allow all connections to be unconnected as opposed to others.

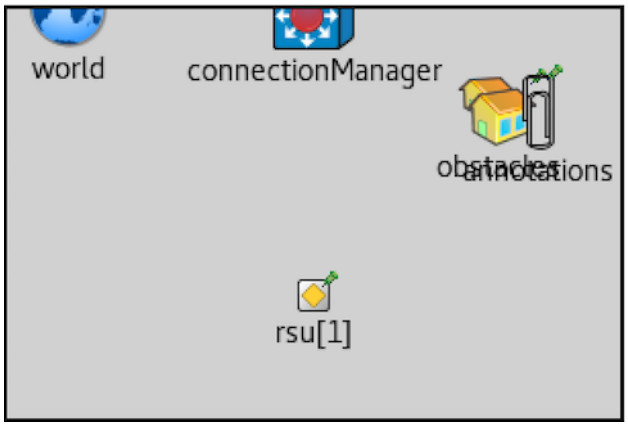

Figure 3.2 Transit and network modules for the vehicle scenario.

3.4.3 Obstacle Control

The obstacle controller administers objects that are blockers to simulate buildings and walls of the simulation [17]. An obstacle is represented by a type of polygon. A function to be done by the polygon includes calculating attenuation per car. It is responsible for adding an obstacle, adding by type and shape, and erasing an obstacle. As opposed to other modules, the controller does not handle any messages. This means that its responsibility is outside of transmitting packets. It represents a physical reaction

3.4.4 Annotation Manager

According to the description of the annotation manager, it handles the annotations on the OMNET++ canvas. It handles all UI components for the OMNET++ canvas and encapsulates the logic for actions on a canvas with objects like a point, line, group, and polygon. The existing actions available are creating, drawing, erasing, scheduling to erase, and showing and hiding the objects. The procedures for the annotation start by

initiating a group of figures also identified as annotations. It retrieves the canvas and adds the figures to it. The manager retrieves the annotations from a SUMO xml file that contains can tags like “line” and “poly”. In these simulations, the file that contains the poly annotations is northampton.poly.xml. More details are in the poly file we used at https://github.com/surod22/VeinsVanetSimulation-2018/tree/master/vanets/simulations/

As it extracts each tag, it will draw the object to the canvas. Similar to the obstacle control, the annotation manager does not handle messages. Its sole responsibility is to extract annotations from the SUMO polygons file and draw them on the canvas. [19]

3.4.5 Connection Manager

The connections manager is the central source that handles connections between nodes. According to its definition, the procedure includes creating gates and communicating with the mobility module in intervals [18]. It handles the grid by creating, transforming, and adding coordinates. Itt measures the existing interference distance and compares it to the max interference distance. The maximum value is provided by the user in the configuration file omnetpp.ini. It will create a world module and matrix to represent the grid on the canvas. The playground sized from the world is used to generate the grid dimensions. We did not find verification of the connection manager directly communicating with the channel access modules. It does, however, set the channel access to the NIC upon registry.

We will now discuss how the connections manager connects and tracks the communication between nodes. The manager keeps a list of all network interface cards available and their positions in the grid. The process is initiated when the base wave

application layer registers a NIC. The registration includes providing the NIC module type, the channel access, and its position. When a new network interface card is added to the grid, the module will update its connections. Given the module has the old and new position of the NIC interface, it will move the NIC to a new position. To update the connections, it will check the rest of the available NICs on the grid. When a NIC is in range of the NIC for that node, it will connect the network card to its own, given they have not already been connected. In the case where the current connected NIC is no longer in range, it will disconnect it. In relation to the range, the module tracks the gates of the NICS in that range. Wrap up features of the connections include unregistering and deleting all NICs at end of simulation. The unregistering is triggered by the wave application layer and the TraCI scenario manager.

3.4.6 TraCI Scenario Manager

The TraCI Scenario Manager initiates all the configuration variables to create a vehicle, insert a vehicle to the queue, and trigger its step. The total number of vehicles in the simulation is checked against the number of active vehicles and vehicles in the queue. It also subscribes to a vehicle. On a similar note, this module, like other managers, does not handle incoming messages. It tracks the network boundaries, arrived vehicles, departed vehicles, and vehicles in stages to teleport and park. Some general simulation subscriptions track the existing simulation time. This is also known as the time step of the simulation. The subscriptions are handled by the command query pattern described earlier as the TraCI command. Internally, it handles the execution of triggering a time step in the simulation. This is the trigger that is sent via the SUMO socket to update the traffic simulation to its next step. Every vehicle has a TraCI mobility submodule which is

initiated and updated by the TraCI manager. When a SUMO response is received from a TraCI subscription command, it will check for the response type from the list of variables it originally subscribed to. Given the module for the vehicle in the response ID exists, it will update the module's position. It updates the coordinates for the new position, edge, speed, angle, and vehicle signal.

3.4.7 Car Module

A vehicle component is made up of an application layer, an 802.11p network interface card also known as Nic. The layout is in Figure 3.3. For the mobility portion, it includes a mobility module and a radio in gate to send direct messages. The physical layer is the physical layer of the network interface card.

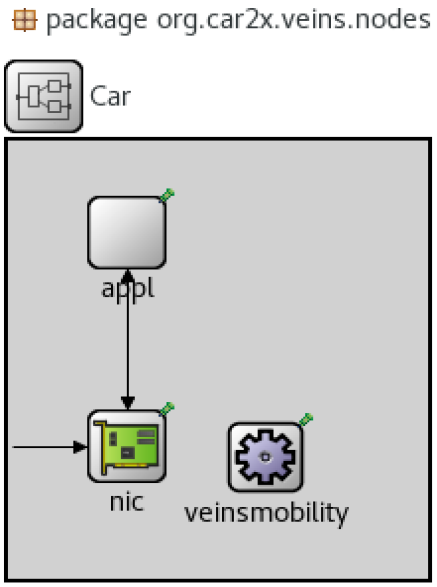

Figure 3.3 Architecture components for the Car module.

3.4.8 WAVE Application Module

The application layer has most of the configuration fields to address the header, beacon, and data. It initializes all the fields for the Wave Short Message, Basic Safety Message, and the Wave Service Advertisement. Internally, it creates the message types and schedule them to be sent. Based on the WAVE app layer description, the module handles all three types of messages when receiving a message from the lower layer. It will also handle the position and parking update. It then sends the message down to the applications lower layer and to the upper layer of the network interface card.

3.4.9 Nic Module

The network interface card module includes a radio in gate for direct connection, a physical layer, and a mac layer. The radio is directly connected to the radio in the physical layer.

3.4.9.1 Physical Layer

The physical module represents the physical layer of the 802.11p. It is responsible for handling the state of the packet by communicating with its upper control and upper layer to the lower control and lower layer of the mac service. According to the documentation, it sets the Clear Channel Assessment (CCA) [22] threshold for the bandwidth. According to the connection manager, this CAA is the threshold for the minimal receiving power. It also initializes the analogue models when the gradual loss of
received signals occurs. Some of the models used include the following: simple path loss, log normal shadowing, jakes fading, breakpoint path loss model, PER model, simple obstacle shadowing, two ray interference model, and nakagami fading. In terms of internal responsibilities, it determines when a transmission is over by initiating a radio switch to alert mac module. It also receives the airframe and the channel sense request. The physical layer communicates via messages to the mac layer with channels, and it also directly interacts with the decider.

3.4.9.2 MAC Layer 1609.4

In this section we discuss some of the responsibilities in the MAC layer, signal management, and channel transmissions. The MAC layer sends a frame, attaches a signal, creates a signal, sets the guard interval active, changes the service channel.

Additionally, itsets the transmission power and sets the mcs. According to the definition of the mac layer, the mcs is the standard for modulations and coding scheme. It helps attach a signal, create a signal, make the guard interval and change service channel. Internally, it sends the signal, creates the queue, and places the packet in the queue. After, it starts and stops content and can revoke a transmission. The mac layer also contains channel integration. It makes it possible to communicate with the service channel. It creates and attaches a signal while simultaneously managing the state of the channel, given it is busy internally or idle. It also manages the state of the acknowledgement, by receiving the transmission when the acknowledgment has timed out.

3.4.9.3 Decider Module 80211p

The decider module processes the new signal. First, it calculates the channel Sense RSSI. It then calculates the signal both with interface and without interface. It verifies that the signal power is strong to submit, and then proceeds to verify that the packet is in tact. Once the signal is verified, the decider module can calculate the CCA threshold. Once the packet is verified, it will activate the channel upon the signal's end. The Decider's documentation indicates that it is the Decider's responsibility to classify signal and transmission errors including he classified cases in which the signal has noise or no noise. It also classifies packet transmission errors.

3.5 Setup

In this section covers how to setup up Veinst with OMNET++. The instructions to set up SUMO, Catch and some extra options. The following list represents the tech stack used for our simulations:

- Debian GNU/Linux 9 (stretch) 64-bit
- Omnetpp-5.2.1
- Sumo 0.30
- Instant Veins 4.6
- VMWare fusion
- Catch (Unit Testing Framework)

3.5.1 Setup Veins with OMNET++

The fastest way to setup Veins is via the instant veins image which is an image of debian virtual machine. The image is then imported into a software to virtualize operating systems. The image includes the dependencies and setup up Veins, OMNET++ and SUMO along with any of the required dependencies. Because we used a mac operating system, we chose VMware fusion to run instant Veins.

- Source code : https://github.com/surod22/VeinsVanetSimulation-2018
- Configuration file : https://github.com/surod22/VeinsVanetSimulation-2018/blob/master/vanets/simulations/omnetpp.ini

In following section will describe how to setup Veins.

3.5.2 Setup Veins

We installed Veins a linux and a macOS . Both installations introduces many linking errors. After much investigation the setup used virtual machine image called Instant Veins.

1. Download Instant Veins
    a. http://veins.car2x.org/documentation/instant-veins/
    b. https://github.com/sommer/instant-veins
2. Mac - Install VMWare fusion or otherwise identify the best VM management software for your operating system.
3. Import the debian image to VMware fusion.
4. Expand partition on VM - The default partition is not enough. We recommend importing the image first, stopping the virtual machine, and expanding the disk space. Next, the volume must be expanded to enable use of the expanded partition on the disk. You can either do this via the command line using fdisk or with a partition manager. We found the gparted partition manager to be the fastest option. Tasks to do this are :
    a. Download gparted to VM: https://gparted.org/download.php
    b. Expand volume
    c. Restart VM

Note: To expand veins, use the simple or compound modules of the existing example. The corresponding classes must be subclassed, and the virtual modules must be implemented.

3.5.3 Setup SUMO

Because Instant Veins was used and it was already set up on the virtual machine, there is no further setup. If you would like to install SUMO separately, more information is provided here: http://veins.car2x.org/tutorial/. [39]

Directory Content

- vanets/ - Simulation tracking parameters as channel utilization, collisions, neighbors, packets (sent, received and dropped), beacons. This simulation is a model on top of veins that is implemented by subclassing veins.
- veins/ - Open source vehicular networks framework with logging for debugging.

Run SUMO Socket

Open terminal window

```
$ cd vanets/

$ ./sumo-launchd.py -vv -c sumo-gui
```

Run Veins Simulation with Omnet++

1. From omnet++ IDE (eclipse), select the file /vanets/src/omnetpp.ini
2. Select the green arrow icon to run.
3. From the pop up window, select the config file to run. The description is on title.
4. Click the run button on top left corner. (NOTE: It is usually faster.)
5. Stop the simulation by clicking the stop button.lick on stop button
6. Select the flag button to conclude simulation.
7. Once the simulation is finished, it will generate the result files under `/vanets/src/results/` .

In the next section will instructions to set up the Catch framework.

3.5.4 Setup Catch Test Framework

Theres is a couple steps to setup Catch. First the framework needs to be is installed and built. Once it is setup the user can run the tests.

Install Catch:

```
$ cd vanets/test/Catch2/
$ git clone https://github.com/catchorg/Catch2.git
$ cd Catch2
$ cmake -Bbuild -H. -DBUILD_TESTING=OFF
$ sudo cmake --build build/ --target install
```

Run Tests:

```
$ cd vanet/test/
```

Compile

```
$ g++ -std=c++11 -Wall -Icatch2/catch.h -I../src/
ChannelService.h -o ChannelServiceTest
ChannelServiceTest.cc && ChannelServiceTest –success
```

Run

```
$ gcc ChannelServiceTest.cc -o ChannelServiceTest
```

Note: Install Cmake if needed https://cmake.org/install

Optional Procedures:

- **Export your own Open Street Map with the instructions in the link below**

  https://www.openstreetmap.org/#map=5/37.666/-92.703.

- **Add additional metrics**
    a. Define variable in header file
    b. Initialize header in c++ file
    c. Find out where this method needs to be called to be recorded
    d. Call method
    e. Create method for functionality of metric
    f. Record variable in finish method
    g. Run to make sure it is recorded in the final .sca results file.

3.5.5 Simulation Network Parameters

In the figure below, you will find the network and traffic parameters used to configure the simulation. Some of the parameters were already the default configuration.

	My scenario	Source Location
Number of Vehicles in simulation	Low = 137 vehicles High = 285 vehicles	*.rou.xml
Vehicle mobility mode	Gaussian Kraus	*.rou.xml
Vehicle length	cars = 5m, Note: trucks not implemented yet	*.rou.xml
Min gap	2.5	*.rou.xml
Max acceleration	2.6 m/ms^2	*.rou.xml
Max speed	33.33m/ms	*.rou.xml

Table 3.1 Realistic Traffic and Map Parameters.

Parameter	My Scenario	Source location
Carrier frequency	5.890e9 MHz	Omnetpp.ini , config.xml
Data Rate	6 Mbps	Omnetpp.ini as bitrate
Number of channels	1CCH 4 SCH	omnetpp.ini *.nic.mac1609_4.useService Channel = true
Beacon Interval	1 s	Omnetpp.ini
Beacon length (B), Service Packet length (P)	B=3200 bits = 400 Bytes P=8000 bits = 1000 Bytes	Omnetpp.ini
Maximum Transmission power	100mw	Omnetpp.ini for phy80211p
Sensitivity	-89dBm	omnetpp.ini
CWmin - Minimum Content Window CWmax - Maximum Content Size	15 1023	Consts80211p.h used in the mac1609_4 class
Simulation length	150 sec	omnetpp.ini
Periodic switching every 50ms CH-> SCH	50 ms	
Header length	80 bit	
Accident time	Start : 10s End : 30s	

Table 3.2 Simulation network parameters based on standards.

In case there is further questions, feel free to reach me via the repository https://github.com/surod22/VeinsVanetSimulation-2018/. This chapter covered the setup used for the vehicle scenario. Chapter four will cover the research features we added.

CHAPTER 4: Research Features

This chapter covers the contributions made to the research topic: including integration of the testing framework Catch, SCH randomization, MAC layer beacon transmission, vehicle neighbors tracking and channel utilization tracking per CCH/SCHs.

4.1 Catch Framework Integration

This section will cover importance to adding tests to the codebase. Two of the testing frameworks integrated. Following with details on the selected framework Catch and why it is recommended.

4.1.1 Cxxtest

Integration tests are a strong source to validate the expected end to end behavior, when expanding a codebase . By the testing pyramid from Martin Fowler, it is recommended to have a small number of integration tests, followed by bigger set of functional tests and a larger set of unit tests. Ideally these should be added via test driven development. The veins framework used, included a set of integration tests. The integration test were complex due to extra layer of the ned modules. We wanted to expand the framework with low risk. Hence, we looked into integrating a testing framework for unit tests. Initially CxxTest was selected for its quick integration [13].

After the review, CxxTest was consequently complicated for expanding the codebase. It focused on using an integration tests. Once getting back into extracting the added features and testing the code. It turned out too redundant to create a test, as it

was using the output of the scalar files to verify outcome. We performed a further spike to identify the easiest unit testing framework to integrate. While reviewing the frameworks, another framework known as Catch was repeatedly recommended.

4.1.2 Catch

Benefits of catch include easy integration without extra dependencies and fast test setup. Catch was developed to have simpler assertions. To integrate the framework to the simulation, the package download is for a header file. After it is downloaded, it is built with a command. After its build is complete, the header is added to the testing class like so “#include <catch2/catch.hpp>” [16]. The test can then be compiled and executed. What made Catch so ideal, was the feature to add only the header file to the new class/ module, and extracting that single responsibility to this class [16]. There was no need to link to the original source files. This prevented any common linking errors to the resources. Having tested these features externally, there was assurance on the expected functionality.

This feature is highly important, as it makes it easier for other users to expand our simulation or Veins for their own research. We worked with OMNET++ and Veins several years. We have strong confidence that Catch was the best solution during this period. We encourage further users to use this testing framework to test their research functionality while using Veins. Next we will be covering how we randomized the SCH.

4.2 Service Channel Randomization

Before this scenario, the packets were only sent by default to one service channel, meaning the channel selection was static. The same single service channel was selected per every node on the simulation. The contributed feature was doing a random channel selection per node. The feature was added at the MAC layer. This happened when the vehicle internally handled the next channel switch.

```
int Mac16094Metrics::randomizeSCH( int min, int max) {

    srand((unsigned)time(NULL));
    int randomSCHNumber= rand()%( max - min) + min;
    int randomSCHEnum = 0;

    switch (randomSCHNumber) {
        case 1: randomSCHEnum = Channels::SCH1; break;
        case 2: randomSCHEnum = Channels::SCH2; break;
        case 3: randomSCHEnum = Channels::SCH3; break;
        case 4: randomSCHEnum = Channels::SCH4; break;
        default: throw cRuntimeError("Random Service Channel
must be between 1 and 4"); break;
    }
    return randomSCHEnum;
}
```

Figure 4.1 Randomizing the selection of the the service channel.

Figure 4.1 shows the functionality to randomize the service channel. After the feature was added, the data in Table 5.3 displayed that the SCH4 had zero packets for all vehicles. The logic below seemed to have the appropriate functionality. We looked at where the method was called and found the mySCH = randomizeSCH(1, 4). With this logic, It is not possible to generate a value of four with a min of one and max of four. This is the how the SCH4 was never used. We recognized this is a bug and can be fixed by replacing the max a value of five. This feature was added to simulate when

using different service channels. Some previous schemes focused on the optimized selection of the next service channel. This feature can used as a base case when comparing to such schemes. If someone plans to use this feature, the user must verify that the appropriate min and max values are set. The following section will cover the details of the beacon transmission to the MAC layer.

4.3 MAC Layer Beacon Transmission

The workflow of packets is where sending and receiving endpoints occur in each of the protocol layers. We summarize the lifetime of a packet as follows: When a step is triggered by the TraCI manager, a signal is received by the wave application layer. Upon signal receipt, the packet updates its state of the vehicle from the mobility module. The state includes information like the vehicle's position and speed. When triggered to step, the channel access module will register the vehicle's Network Interface Card (Nic) with the connection manager. Once the network interface card is registered, the list of other nics within range are updated. This creates a network of communication. The lifetime of a packet begins at the application layer. When the Wave application layer is initiated, it schedules a wave short acknowledgement and beacon to be sent. As the message gets triggered internally, it is sent down to the outgoing gate of the lower layer. By referencing the Car module, we can see that the lower layer out gate is connected to the upper layer in gate of the Nic module. In the Nic module, we can see its upper layer in gate is connected to the upper layer in of the mac layer. Once the mac receives the message, it can manage signal state, channel rotation, transmission intervals, and message state. During the transmission handling, the mac can communicate with the

physical layer to calculate the signal quality. The physical layer along with the decider will be the decision makers for whether the packet is identified as successfully transmitted or not.

In the Veins framework used, the WAVE application layer was recording when the beacons were received. Based on the measures of the Mac layer, the traffic related to beacon messages seemed much smaller than expected. While reviewing the code base, the findings where the wave short messages and basic safety messages were recorded on the Wave application layer of the simulation due to the application layer being the first layer component to receive a message from a node. At the same time, the beacons were not being sent down to the Nic and Mac layer. A functionality was added to send the beacon down to the Nic Mac layer and to be receive on the Mac layer end. This was verified by adding message metrics on the Mac layer. Looking at the channel utilization layer we can, before the feature, were getting CCH channel utilization values between 0.005597 to 0.380212. After the feature was added, the CCH channel utilization values were between 0.08774 and 11.409037. In addition, the time it took the the simulation to run increased. This feature was crucial to provide a better analysis.

4.4 Vehicle Neighbors Tracking

The number of neighbors is recorded when receiving a frame from the lower layer. This only happens when the packet is for the address of that node or if it is a broadcast message. It occurs at the receiving end of the simulation. Each time a frame is received, the frame contains the sender ID. The received frame is defragmented to a packet. Given the destination of the packet is the same as the existing node, the packet is

received and the packet sender ID is recorded into an array of unique values. Provided the sender ID does not exist in the array, the ID is added to the array. In the case where the sender ID does exist, the frequency is incremented for that index value. The goal is to have a list of the neighbors at the end of the simulation for that specific node with the possibility of being able to display the number messages received from that specific sender.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
n	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 4.1 Number of Neighbors per Vehicle ID.

After implementing this feature we found that each vehicle was only receiving messages from one vehicle. Table 4.1 displays the day for the number of vehicles. The first row is the vehicle ID. The second row is the number of vehicles found for the vehicle. To confirm that the number of neighbors feature worked properly, we looked addresses in the frame. For each frame received the source address was recorded for that vehicles mac address. Other variables recorded are displayed in the first column of Table 4.1. As in the Table 4.2, we can see that there was only one unique source address of -42537770 .

	Vehicles address	**WSM Sender Address**	**WSM Receiver Address**	**MAC Packet Source Address**	**MAC Packet Destination Address**	***BROADCAST Address***
Code	myMac Address	wsm->getSenderAddress()	wsm->getRecipientAddress()	macPkt->getSrcAddr()	macPkt->getDestAddr()	LAddress::*L2BROADCAST*()
19	2107226664	205	0	-42537770	-1	-1
16	1448096158	205	0	-42537770	-1	-1
17	1906850353	205	0	-42537770	-1	-1
18	-599277280	205	0	-42537770	-1	-1

Table 4.2 Address values for WSMs and Mac1609_4 frames.

As mentioned earlier, the tracking of neighbors was implemented when receiving a packet or broadcast for that node. The destinations address did not match the vehicles address and did match the BROADCAST's address. Therefore, the neighbor recorded was for broadcast messages and no packets were received at the Mac layer. This will explain the zero values in the column Received Unicast Packets at a later Table 5.5. Additionally, it was only one neighbor because all of the frames had a default source address of -42537770. A further investigation was done to find the cause of this.The existing code base only set the recipient address with a default value of zero. The recipient address value was set at the Wave Short Message via the populateWSM method shown below. The Wave Short Message(WSM) was populated in the TraCIDemo11p module (TraCIDemo11p.cc) and the BaseWayApplLayer module (BaseWaveApplLayer.cc).

```
void BaseWaveApplLayer::populateWSM(WaveShortMessage* wsm, int
rcvId, int serial) {
    wsm->setWsmVersion(1);
    wsm->setTimestamp(simTime());
    wsm->setSenderAddress(myId);
    wsm->setRecipientAddress(rcvId);
    wsm->setSerial(serial);
    wsm->setBitLength(headerLength);

    if (BasicSafetyMessage* bsm =
dynamic_cast<BasicSafetyMessage*>(wsm)) {
        bsm->setSenderPos(curPosition);
        bsm->setSenderPos(curPosition);
        bsm->setSenderSpeed(curSpeed);
        bsm->setPsid(-1);
        bsm->setChannelNumber(Channels::CCH);
        bsm->addBitLength(beaconLengthBits);
        wsm->setUserPriority(beaconUserPriority);
    }
    else if (WaveServiceAdvertisment* wsa =
dynamic_cast<WaveServiceAdvertisment*>(wsm)) {
        wsa->setChannelNumber(Channels::CCH);
        wsa->setTargetChannel(currentServiceChannel);
        wsa->setPsid(currentOfferedServiceId);
        wsa-
>setServiceDescription(currentServiceDescription.c_str());
    }else {
        if (dataOnSch) wsm->setChannelNumber(Channels::SCH1);
        //will be rewritten at Mac1609_4 to actual Service
Channel.
        //This is just so no controlInfo is needed
        else wsm->setChannelNumber(Channels::CCH);
        wsm->addBitLength(dataLengthBits);
        wsm->setUserPriority(dataUserPriority);
    }
}
```

Figure 4.2 Existing BaseWaveApplLayer method that populates WSM.

```
void TraCIDemo11p::handlePositionUpdate(cObject* obj) {

    BaseWaveApplLayer::handlePositionUpdate(obj);

    // stopped for for at least 10s?
    if (mobility->getSpeed() < 1) {

        if (simTime() - lastDroveAt >= 10 && sentMessage ==
false) {

            findHost()-
>getDisplayString().updateWith("r=16,red");
            sentMessage = true;
            WaveShortMessage* wsm = new WaveShortMessage();
            populateWSM(wsm);
            wsm->setWsmData(mobility->getRoadId().c_str());

            //host is standing still due to crash
            if (dataOnSch) {

                startService(Channels::SCH2, 42,
                                "Traffic Information
Service");

                //started service and server advertising,
                //schedule message to self to send later

scheduleAt(computeAsynchronousSendingTime(1,type_SCH),wsm);

            }
            else {

                //send right away on CCH,
                //because channel switching is disabled
                sendDown(wsm);
            }
        }
    }
    else {
        lastDroveAt = simTime();
    }
}
```

Figure 4.3 TraCIDemop location where WSM is populated.

```
void BaseWaveApplLayer::handleSelfMsg(cMessage* msg) {

    switch (msg->getKind()) {

      case SEND_BEACON_EVT: {
        BasicSafetyMessage* bsm = new BasicSafetyMessage();
        populateWSM(bsm);
        sendDown(bsm);
        scheduleAt(simTime() + beaconInterval, sendBeaconEvt);
        break;
      }

      case SEND_WSA_EVT:   {
        WaveServiceAdvertisment* wsa = new
WaveServiceAdvertisment();
        populateWSM(wsa);
        sendDown(wsa);
        scheduleAt(simTime() + wsaInterval, sendWSAEvt);
        break;
      }

      default: {
        if (msg)
            DBG_APP << "APP: Error: Got Self Message of
unknown kind! Name: " << msg->getName() << endl;
        break;
    }
  }
}
```

Figure 4.4 BaseWaveApplLayer location where WSM is populated.

We assumed this was done to simplify the IP address implementation. For future research we recommend an extension to this. It is important to understand and how many messages are received by each node. If this feature is added would be able to verify where the total of messages received are coming from. As well as have a more

realistic representation of the message traffic for analysis.

4.5 Channel Utilization Tracking

Since the module of a node has the features to send and receive, collisions can happen at the sender and the receiver side. Some existing variables to track the types of collisions included dropped packets, SNIR lost packets, and RXTX lost packets. Packets are considered dropped or lost during different scenarios. A packet qualified as a dropped packet in the scenario in which the packet was to be added to queue. Ifn the size of the queue was greater or equal to the max queue size, the packet was dropped. This happens when receiving a WSM from the upper layer. In the case where packets are lost, the collisions occur when receiving a message from the lower control. An SNIR lost packet scenario occured when the packet was identified as having bit errors or to have a collision. For a TXRX lost packet, this occurs as a packet was being received while simultaneously sending.

One thing to consider is how the base physical layer receives an airframe. It called the decider to process the signal. Next, it checked if the frame was detected by the radio card. If it was, when the radio is in transmission state, the packet was identified as receiving while sending. It followed by verifying that the signal if the packet had enough power to transmit.

The probability of a packet error was separated by a header and packet error. The success rate of each error is computed considering the bitrate, bandwidth, minimum signal interference, and payload bitrate. The values for each accordingly were as follows:

Frame category	COLLISION	NOT_DECODED	COLLISION	NOT_DECODED
Acceptance Criteria	Given **`collectCollisions`** is set, When the header error is due to interference, then it is a collision.	Give there was no interference, when the packet header had an error, then then unable to decoded the packet.	Given there was not a header error, when the error was due to interference, then the frame is a collision.	Given there was not header error, when the packet had an error, then unable to decode it.
Error Type Variable	**`headerNoError`** - header success rate with interference **`snirMin`**	**headerNoErrorSnr** - header success rate without interference **`snirMin`**	**`packetOkSinr`** - packet success rate with interference **`snirMin`**	**`packetOkSnr`** - packet success rate without interference **`snirMin`**
Bitrate	PHY_HDR_BITRATE = 3000000	PHY_HDR_BITRATE = 3000000	Packet bitrate = 6 Mbps	Packet bitrate = 6 Mbps
Bandwidth	*BW_OFDM_10_MHZ = 1*	BW_OFDM_10_MHZ = 1	BW_OFDM_10_MHZ = 1	BW_OFDM_10_MHZ = 1
Payload bitrate	PHY_HDR_PLCPSIGNAL_LENGTH = 24	PHY_HDR_PLCPSIGNAL_LENGTH = 24	lengthMPDU = PHY_SIGNAL_LENGTH + PHY_PSDU_HEADER + Payload	lengthMPDU = PHY_SIGNAL_LENGTH + PHY_PSDU_HEADER + Payload

Table 4.3 Variables to categorize a message in Decider80211p

Channel utilization was defined to be the compound time spent transmitting packets at each channel. To map this to the simulation, each Wave Short Message has a duration time. The duration time for a packet is the configured time for that component to simulate as transmission time. We recorded channel utilization by tracking the time it took for each message being transmitted in the CCH and SCHs, provided that the channel was active and the packet was successfully transmitted. The time per packet was verified by recording the duration times for different components of the message. For example the message body had a duration time of 0.000193 ms as below. There was also a duration time of 0.000032 ms for the PHY_HDR_PREAMBLE_DURATION.

Both the PHY_HDR_PLCSIGNAL_DURATION and the T_SYM_80211P had a duration time of 0.000008 ms. The times were verified at the application, physical, and mac layer. Based on the simulation code, the duration time for a message includes the duration time of header constants. Some of these headers are shown in Figure 4.3. Below is a list of some of the message components found.

- PHY_HDR_PREAMBLE_DURATION: 0.000032 ms
- PHY_HDR_PLCSIGNAL_DURATION: 0.000008 ms
- T_SYM_80211P: 0.000008 ms
- Sending Duration Time 0.000193 ms (Raw value for message body)

The sending duration time was recorded to confirm that the number of messages sent/received made sense. The control channel handles much of the load in the simulation scenario. We concluded that most of the service channels are underutilized. Collectively, the amount of beacons being generated was much greater then the amount of packets. Perhaps a further study would look into the beacon rate, to find the least beacon rate necessary to deliver safety messages.

CHAPTER 5: Multi-Channel Analysis

In this section, we analyze the patterns found on the results for the multi-channel operation. The scenario analyzed, mimics the traffic of 29 vehicles over a simulation time of 150 ms. A vehicle was triggered to simulate an accident starting at 10 ms and ending at 30 ms. The duration of 20 ms was not a direct representation for the time of a real accident. Figure 4.3 is part of the implementation for the accident. The vehicle is in crash mode for at least 10 ms, with the remaining as traffic jam. The idea was to replicate a network congestion in the scenario of a traffic congestion. The SimTime unit is an interval step in the simulation vs simulating real time. For example if the number of operation per simTIme step increased, the sim step would take longer than a real time ms. Both the time and the number of vehicles was generated by the slow overhead created after the feature of handing the beacons to the mac layer was added. We will be covering the message delivery for the application layer mac layer and road traffic. The MAC layer is where the multi-channel operations are integrated. The application layer hands the messages to the MAC layer. We analyse the delivery of these components to identify patterns and points of impact.

5.1 Single Vehicle Message Delivery

Before moving the specific evaluations in each of the layers, it is important to review the data for the base case scenario of one vehicle. Table 5.1 illustrates the data for frames generated, received and sent message. This was considered the base case as a set of vehicles is evaluated in other sections. This also simplified the basic patterns expected

within the larger set. This vehicle was selected as it used the majority of the service channels. This is information is in channel utilization data of Table 5.3.

Layer	Transmission type	WSMs (data)	BSMs (beacons)	WSAs (ack)
Application	Generated	3	46609	0
Application	Received	15	34864	0
Phy80211p	Collisions	N/A	46290	N/A

Layer	Transmission type	Packets (data)	Broadcasts (beacons)	Ack (ack)
Nic.Mac16094	Received	N/A	34879	N/A
Nic.Mac16094	Sent	13449	N/A	0
Nic.Mac16094	SNIRLost	235795	N/A	N/A
Nic.Mac16094	RXTXLost	44688	N/A	N/A
Nic.Mac16094	TotalLost	280483	N/A	N/A
Nic.Mac16094	Dropped	33163	N/A	N/A
Nic.Mac16094	Throughput Count	N/A	34879	N/A

Table 5.1 Beacon and packet delivery data for a single vehicle.

Figure 5.1 displays the raw data for vehicle ID 20. The data is separated by MAC layer, application layer and physical layer. The definitions for the metrics like SNIRLost, WSM and Packets can be found in Table 5.3. For example a Basic Safety Message is received and generated at the application layer and it is handed to the MAC layer as a Broadcast. A pattern in Vehicle 20 is the large number of beacons low number of packets generated. Also it is confirmed that the same number of beacons is received in the application and MAC layer.

```
scalar RSUScenario.node[20].appl generatedWSMs 3
scalar RSUScenario.node[20].appl receivedWSMs 15
scalar RSUScenario.node[20].appl generatedBSMs 46609
scalar RSUScenario.node[20].appl receivedBSMs 34864
scalar RSUScenario.node[20].appl generatedWSAs 0
scalar RSUScenario.node[20].appl receivedWSAs 0
scalar RSUScenario.node[20].appl beaconInterval 0.1
scalar RSUScenario.node[20].nic.phy80211p busyTime 1.006611487278
scalar RSUScenario.node[20].nic.phy80211p ncollisions 46290
scalar RSUScenario.node[20].nic.mac1609_4 throughputMetricMac 233.10214965581
scalar RSUScenario.node[20].nic.mac1609_4 throughputMbps 0.20515234713503
scalar RSUScenario.node[20].nic.mac1609_4 throughputControlMbps 0
scalar RSUScenario.node[20].nic.mac1609_4 receivedFramesLowerMsg 34879
scalar RSUScenario.node[20].nic.mac1609_4 receivedBitsLowerPackets 0
scalar RSUScenario.node[20].nic.mac1609_4 receivedBitsLoserWsm 0
scalar RSUScenario.node[20].nic.mac1609_4 packetsNotForMe 0
scalar RSUScenario.node[20].nic.mac1609_4 receivedTotalBits 30696880
scalar RSUScenario.node[20].nic.mac1609_4 collisionsPktNonDecoded 0
scalar RSUScenario.node[20].nic.mac1609_4 chUtilizationCCH 3.240966
scalar RSUScenario.node[20].nic.mac1609_4 chUtilizationSCH1 0.000465
scalar RSUScenario.node[20].nic.mac1609_4 chUtilizationSCH2 0.000465
scalar RSUScenario.node[20].nic.mac1609_4 chUtilizationSCH3 0.000465
scalar RSUScenario.node[20].nic.mac1609_4 chUtilizationSCH4 0
scalar RSUScenario.node[20].nic.mac1609_4 chUtilizationHPPS 0
scalar RSUScenario.node[20].nic.mac1609_4 chUtilizationCRIT_SOL 0
scalar RSUScenario.node[20].nic.mac1609_4 previousSignalQualityCCH 100
scalar RSUScenario.node[20].nic.mac1609_4 previousSignalQualitySCH1 100
scalar RSUScenario.node[20].nic.mac1609_4 previousSignalQualitySCH2 100
scalar RSUScenario.node[20].nic.mac1609_4 previousSignalQualitySCH3 100
scalar RSUScenario.node[20].nic.mac1609_4 previousSignalQualitySCH4 0
scalar RSUScenario.node[20].nic.mac1609_4 chPacketsCCH 26547
scalar RSUScenario.node[20].nic.mac1609_4 chPacketsSCH1 2
scalar RSUScenario.node[20].nic.mac1609_4 chPacketsSCH2 2
scalar RSUScenario.node[20].nic.mac1609_4 chPacketsSCH3 2
scalar RSUScenario.node[20].nic.mac1609_4 chPacketsSCH4 0
scalar RSUScenario.node[20].nic.mac1609_4 numberOfNeighbors 1
scalar RSUScenario.node[20].nic.mac1609_4 ReceivedUnicastPackets 0
scalar RSUScenario.node[20].nic.mac1609_4 ReceivedBroadcasts 34879
scalar RSUScenario.node[20].nic.mac1609_4 SentPackets 13449
scalar RSUScenario.node[20].nic.mac1609_4 SentAcknowledgements 0
scalar RSUScenario.node[20].nic.mac1609_4 SNIRLostPackets 235795
scalar RSUScenario.node[20].nic.mac1609_4 RXTXLostPackets 44688
scalar RSUScenario.node[20].nic.mac1609_4 TotalLostPackets 280483
scalar RSUScenario.node[20].nic.mac1609_4 DroppedPacketsInMac 33163
scalar RSUScenario.node[20].nic.mac1609_4 TooLittleTime 496
scalar RSUScenario.node[20].nic.mac1609_4 TimesIntoBackoff 22293
scalar RSUScenario.node[20].nic.mac1609_4 SlotsBackoff 32938
scalar RSUScenario.node[20].nic.mac1609_4 NumInternalContention 0
scalar RSUScenario.node[20].nic.mac1609_4 totalBusyTime 18.330632601611
scalar RSUScenario.node[20].nic.mac1609_4 throughputMetricMac:last
233.10214965581
attr source throughputSignalMac
attr title "throughputMetricMac, last"
scalar RSUScenario.node[20].nic.mac1609_4 throughputMetricMac:count 34879
```

```
attr source throughputSignalMac
attr title "throughputMetricMac, count"
Scalar RSUScenario.node[20].nic.mac1609_4 throughputMetricMac:sum
4698399.7681026
attr source throughputSignalMac
attr title "throughputMetricMac, sum"
scalar RSUScenario.node[20].nic.mac1609_4 throughputMetricMac:mean
134.70569018901
attr source throughputSignalMac
attr title "throughputMetricMac, mean"
scalar RSUScenario.node[20].veinsmobility startTime 98000
scalar RSUScenario.node[20].veinsmobility totalTime 51000
scalar RSUScenario.node[20].veinsmobility stopTime 149629
scalar RSUScenario.node[20].veinsmobility minSpeed 0
scalar RSUScenario.node[20].veinsmobility maxSpeed 9.8554973309629
scalar RSUScenario.node[20].veinsmobility totalDistance 80.260005883134
scalar RSUScenario.node[20].veinsmobility totalCO2Emission 67.284878665944
```

Figure 5.1 Raw data recorded for vehicle # 20.

One thing to consider in Figure 5.1 is the busy time for each of the layers. The physical layer had a busy time of 1.006611487278 ms, and the mac layer had a busy time for 18.330632601611 ms, while the vehicle was active for a total time of 51 ms. Also, although throughput (mbps) was recorded, it was more important to focus on the message delivery because it indicated a better metric to identify the state of the protocol. The following section will cover an analysis of message delivery in the application layer against the MAC layer.

5.2 Analysis: Application Layer Message Delivery

The message delivery metrics recorded both occurred at the application and mac layer due to the fragmentation of the message type. In the application layer, messages are categorized as Wave Short Messages (WSMs) , Basic Safety Messages (BSMs) and Wave Service Advertisement (WSA). In the MAC layer, they are identified as packets data type or broadcast. To identify where the messages drop, we

investigated the values of the messages received at each layer. Table 5.2 is sorted by the simulation time that each vehicle started. When the simulation number of vehicles was calculated, we found that all vehicles had one neighbor. The neighbor is for broadcast messages as mentioned in sec. 4.3, near Table 4.2 we explain the single neighbor is because the receiver address is not set when populating a WSM or BSM. This missing feature is important to understand the relationship between neighbors and message delivery. We highly recommended the implementation of this feature.

Vehicle start time (ms)	Vehicle ID	Generated WSMs	Received WSMs	Generated BSMs	Received BSMs	Generated and WSAs	Sent Packets	Received Frames LowerMsg
1	3	3	24	82197	60235	0	26772	60259
3	1	1	34	176243	132304	0	46094	132338
6	28	1	34	176377	132490	0	45810	132524
11	29	1	34	174907	131727	0	45275	131761
12	2	1	34	180091	136031	0	45749	136065
17	4	1	34	163860	124711	0	42720	124745
22	15	1	34	145210	110476	0	39050	110510
27	21	1	29	134245	101742	0	36096	101771
31	22	1	29	122764	93260	0	34132	93289
36	23	1	29	112233	84745	0	31637	84774
41	24	3	27	106640	80386	0	30077	80413
46	25	3	27	98929	74432	0	28429	74459
50	26	3	27	95028	71622	0	26962	71649
83	27	1	17	57372	42961	0	16590	42978
86	5	1	18	53937	40752	0	15967	40770
88	6	1	18	52288	39401	0	15521	39419
90	7	1	17	51197	38395	0	14917	38412
92	8	1	18	49138	36995	0	14471	37013

94	9	1	17	46725	35104	0	13984	35121
96	10	1	18	42451	31892	0	12888	31910
98	11	3	15	46609	34864	0	13449	34879
101	12	3	15	47120	35239	0	13161	35254
103	13	1	17	45417	33976	0	12801	33993
106	14	1	17	43439	32523	0	12071	32540
109	16	1	17	40230	30219	0	11158	30236
140	17	0	0	7884	6580	0	2299	6580
143	18	0	0	4701	4287	0	1443	4287

Table 5.2 Message handling metrics for application and MAC layer.

In terms of message delivery at the application layer. In the data of Table 5.2, the network for the accident was noticed as the number of beacons increases for vehicle ID 3. The peak of beacon generation is at start time 12 ms. We observed the received BSMs in the application layer came fairly close to the Received Frames in the Mac layer. Also, of all the set of vehicles 1, 28 , 29 , 2, 4, 15 and 21 that became active around the time of the accident had a higher number of received packets and BSMs. We will now review some of the patterns found in the message delivery for the Mac layer.

5.3 Analysis: MAC Layer Message Delivery

The mac layer passes down the message to the physical layer, which is later handed to the decider to check the signal and state of the air frame. The decider categorizes the package into either a DECODED, NOT_DECODED, or COLLISON [code, Decider]. Decoded is defined as packet was received. Not decoded indicates that there was low signal (power), causing the packet to have bit errors. When a packet is categorized as

NOT_DECODED, the MAC layer labels it as lost. Table 5.3 shows the values for the lost packets under SNIR lost packets and TXRX lost packets. The collision category is a packet having bit errors due to a collision.

VANETs	Message successfully delivered	Message un-successfully delivered	Message un-successfully delivered	Safety message	Non-safety message
MAC	Received	RXTX Lost or SNIR Lost	Collision	Broadcast	Packet
Decider	Frame DECODED	Frame NOT_DECODED due to bit errors	Frame with bit errors due to a collision	Frame of type beacon	Frame of type data

Table 5.3 Message type definitions for VANETS, MAC and Decider.

The goal of the metrics in Table 5.3 was to visualize the delivery scope of beacons (broadcasts) and data messages (packets) for the Mac layer with the multi-channel operation. As well as the relationship of categorizing frames in the decider.

Vehicle Start Time	ID	Received Broadcasts	Received Unicast Packets	Sent Packets	Dropped Broadcasts	RXTX Lost Packets	SNIR Lost Packets	Total Lost Packets	N collisions
1	3	60259	0	26772	55428	27461	159452	186913	59596
3	1	132338	0	46094	130149	94739	369737	464476	72203
6	28	132524	0	45810	130568	93692	370911	464603	72194
11	29	131761	0	45275	129633	95682	369202	464884	72111
12	2	136065	0	45749	134343	96912	362436	459348	66483
17	4	124745	0	42720	121141	93644	371093	464737	73327
22	15	110510	0	39050	106161	88545	382531	471076	82780
27	21	101771	0	36096	98150	85295	385109	470404	85692
31	22	93289	0	34132	88633	79600	389652	469252	88782
36	23	84774	0	31637	80597	77883	387513	465396	91039
41	24	80413	0	30077	76566	75960	381010	456970	87797

46	25	74459	0	28429	70503	71827	374946	446773	86307
50	26	71649	0	26962	68069	70018	366337	436355	83517
83	27	42978	0	16590	40782	51177	288277	339454	62453
86	5	40770	0	15967	37970	49879	279575	329454	59811
88	6	39419	0	15521	36768	49459	273018	322477	57958
90	7	38412	0	14917	36281	47679	267109	314788	55837
92	8	37013	0	14471	34668	47110	260391	307501	54043
94	9	35121	0	13984	32742	45773	254567	300340	52833
96	10	31910	0	12888	29564	42669	251912	294581	53480
98	11	34879	0	13449	33163	44688	235795	280483	46290
101	12	35254	0	13161	33962	44022	219828	263850	40712
103	13	33993	0	12801	32617	43168	210958	254126	38673
106	14	32540	0	12071	31369	41073	197430	238503	35298
109	16	30236	0	11158	29073	37665	185888	223553	33090
140	17	6580	0	2299	5585	8187	48123	56310	8792
143	18	4287	0	1443	3258	5228	34476	39704	6450
145	19	2877	0	890	1932	3178	25111	28289	4757
147	20	1570	0	380	767	1409	15292	16701	2915

Table 5.4 Beacon delivery vs Packet delivery at the MAC layer.

Table 5.4 shows the beacon and packet delivery metrics data for the MAC layer and the decider. Once again, there was a pattern between the accident time and the number of packets lost. For vehicles with a start time between 3 and 27, there was increase in lost packets.

An unexpected data in this table was the zero Received Unicast Packets received in comparison to the received WSM in Table 5.2. Earlier, section 4.3 described the correlation of one neighbor found per vehicle to the default address of a broadcast. The receiver address missing when populating the WSM, BSM and macPkt, and the default address was used. This explained the zero values in the column Received

Unicast Packets for Table 5.4. Since there was no receiver address set at the TraCi and application layer, zero packets were received at the MAC layer and decider.

It is important to identify that the values for a type of lost packets are in the same range of beacons generated. They are not similar to the small number of packets generated. For example lost packets are in the thousands similar to beacons. Lost packets should have been named lost beacons. Interestingly, as the number of packets lost and beacons dropped increases, the number of collisions decreased. Now that we have a more robust understanding on frame delivery between layers, it is import to analyze the usability of each channel. The following section will analyze the data for channel switching and utilization.

5.4 Analysis: MAC Layer Channel Utilization

Channel utilization was defined as the total time handling successful packets/beacons, given that channel is active. The other data analyzed in this section is number of packets received per channel. The number of packets received was defined as the total packets received successfully by the channel, given the channel was active. It was unexpected to see that the SCH4 in Table 5.5. had zero utilization. The investigation of channel randomization in section 4.2, explained the bug on the feature. The (1 min , 4 max) values were used to generate the random channel. Values of (1min, 5 max) should have been used to include SCH4 .

Vehicle Start Time	CCH before feature	CCH + MAC Layer Beacon Transmission Feature	SCH1	SCH2	SCH3	SCH4	MAC Total BusyTime
1	0.380212	7.288209	0	0	0.000465	0	49.1062241
3	0.362842	11.409037	0	0.000465	0	0	48.03534764
6	0.384072	11.334449	0.000465	0.00093	0	0	34.20923655

11	0.353192	11.169178	0	0.000465	0	0	46.58979752
12	0.219957	11.23858	0.000233	0	0	0	22.56146924
17	0.216162	10.466207	0.000233	0	0	0	21.8530523
22	0.212237	9.506569	0.000233	0	0	0	21.10403171
27	0.208377	8.801663	0.000233	0	0	0	20.4124089
31	0.204517	8.259603	0.000233	0	0	0	19.76368752
36	0.200657	7.672468	0.000233	0	0	0	19.03450713
41	0.196862	7.278842	0.000465	0.000465	0.000465	0	18.3306326
46	0.191007	6.88609	0.000465	0.000465	0.000465	0	17.25281308
50	0.187212	6.517519	0.000233	0	0	0	16.53584827
83	0.181357	4.026125	0.000465	0	0	0	15.44728718
86	0.343542	3.893598	0	0.000233	0	0	45.05650719
88	0.175567	3.755872	0.000465	0	0	0	14.39944335
90	0.115802	3.583236	0	0	0	0	3.414398978
92	0.110012	3.474502	0	0	0	0	2.35298223
94	0.106152	3.388575	0	0	0	0	1.646960498
96	0.102292	3.093527	0	0	0	0	0.941155623
98	0.096502	3.240966	0	0	0.000465	0	43.32506311
101	0.092642	3.173862	0	0	0.000233	0	41.90391659
103	0.088782	3.145856	0	0.000233	0	0	40.33319019
106	0.084922	2.913382	0.000465	0.000465	0.000465	0	38.47815587
109	0.081062	2.688597	0	0.000465	0.00093	0	36.58516915
140	0.333892	0.556411	0	0.000465	0.00093	0	35.31265076
143	0.075272	0.352803	0.000465	0	0	0	23.64613068
145	0.069482	0.214394	0	0.000233	0	0	48.79108756
147	0.057902	0.08774	0	0	0.000233	0	48.28244715

Table 5.5 Channel utilization for CCH and SCH (ms) at the MAC layer.

A positive outcome of the beacon feature explained in section 4.2, is shown in the utilization for columns CCH before feature and CCH + MAC Layer Beacon Transmission Feature. On average the CCH utilization increased twenty nine times the

original value. Most importantly the new values the linear pattern of beacons received in Figure 6.5. This pattern will be discussed in chapter 6, when comparing Figure 6.6 and Figure 6.5. Adding this functionality was a solid contribution to more accurate data for the multi-channel operation. To verify the channel utilization, the number of frames received was also recorded.

Vehicle ID	CCH	SCH1	SCH2	SCH3	SCH4
1	91315	0	0	2	0
2	90713	0	2	0	0
3	52794	2	4	0	0
4	84689	0	2	0	0
5	31514	1	0	0	0
6	30603	1	0	0	0
7	29432	1	0	0	0
8	28560	1	0	0	0
9	27592	1	0	0	0
10	25422	1	0	0	0
11	26547	2	2	2	0
12	26018	2	2	2	0
13	25313	1	0	0	0
14	23870	2	0	0	0
15	77330	0	1	0	0
16	22068	2	0	0	0
17	4535	0	0	0	0
18	2847	0	0	0	0
19	1753	0	0	0	0
20	748	0	0	0	0
21	71404	0	0	2	0
22	67511	0	0	1	0
23	62545	0	1	0	0
24	59455	2	2	2	0

25	56189	0	2	4	0
26	53257	0	2	4	0
27	32731	2	0	0	0
28	90674	0	1	0	0

Table 5.6 Packets received per channel at the MAC Layer.

Table 5.6 has the packets received per channel at the MAC layer. Many previous studies reflected a great deal of focus on optimizing the SCH utilization. In Table 5.6, we see that the majority of the traffic happened at the CCH. This supports that our recommendation of optimizing the usability of the control channel due to the control channel managing the communication and decisions for the upcoming channel switch.

At the same time, Table 5.2 confirmed that there was way more beacons generated than packets. Another recommendation is to analyze a scenario with a larger number of packets generated. With the much effort of using a realistic simulation, it is important to analyze the relationship of the network and traffic congestion. This is covered in the next section.

5.5 Analysis: Traffic congestion vs the network congestion

With Veins as the realistic traffic simulation, we wanted to evaluate the relationships between the traffic jam and the number of collisions. Looking at the relationship of vehicle density and the network, the first column of Active Vehicles is compared to the number of collisions. Assuming that each vehicle sent similar average of beacons and packets. The expected behavior was for the collisions to increase as the number of vehicles increases. As seen from start time 41 to 143, vehicle density increases by 16 vehicles. During this period, the number of collisions does not fluctuate much. However,

when there is 26 vehicles in the simulation, the collisions drop from 33090 to 8792. Also, the busy time in the MAC layer decreases. It is recommended to to study the pattern between number of vehicles and collisions more in a larger number of simulation runs. Lastly a bias to consider is the threshold towards the end of the simulation. As a vehicle with lower total time has lower number of collisions.

Active Vehicles	Start Time	Total Time	ID	Total Distance	Min Speed	Max Speed	collisions	MAC Total Busy Time
1	1	148	3	1487.06866	0	13.99812867	59596	34.20923655
2	3	146	1	197.531656	1.320179877	1.388696462	72203	49.1062241
3	6	143	28	192.2461384	0.8838096125	1.393291953	72194	48.79108756
4	11	138	29	185.7802316	0.9856438959	1.399275053	72111	48.28244715
5	12	137	2	183.5658233	0.8243705354	1.388072967	66483	48.03534764
6	17	132	4	177.2467625	0.9325684591	1.398566944	73327	46.58979752
7	22	127	15	170.7850123	0.9809413182	1.391785811	82780	45.05650719
8	27	122	21	164.2077012	1.061905699	1.397178956	85692	43.32506311
9	31	118	22	158.1509357	0.8453348706	1.399181308	88782	41.90391659
10	36	113	23	151.6924971	0.9362882634	1.405918627	91039	40.33319019
11	41	108	24	145.1831635	0	10.53132752	87797	38.47815587
12	46	103	25	138.5649047	0	8.735856198	86307	36.58516915
13	50	99	26	132.3925057	0	7.410544899	83517	35.31265076
14	83	66	27	121.4132993	0.0009649957	7.18002694	62453	23.64613068
15	86	63	5	115.0587852	0.4497240707	7.133446207	59811	22.56146924
16	88	61	6	109.7951971	0.6035547639	6.820338504	57958	21.8530523
17	90	59	7	104.0418463	0.7590023523	6.301196164	55837	21.10403171
18	92	57	8	98.1522499	0.8900745093	5.465737862	54043	20.4124089
19	94	55	9	92.21676985	0.9638579363	5.009624307	52833	19.76368752
20	96	53	10	86.37638398	1.001522262	4.653284283	53480	19.03450713

Table 5.7 Traffic vs Network congestion.

In this section, the data of message delivery in the application layer, MAC layer and the decider was analyzed. This also includes an analysis of channel utilization, traffic congestion and network congestion. The base case scenario was the data from vehicle ID 20. The vehicle was selected for its usability in in the control channel and three of the service channels.

The objective was to find any pain points related to the multi-channel operation. There was an average of 81420 total beacons generated per vehicle in comparison to an average of 1 packet generated. With the larger number of beacons generated and switch operations, the control channel was overloaded. For collisions there was a total average of 56386. We also found that only broadcasts were received at the MAC layer and decider. No packets were received beyond the application layer caused by the receiver's address not allocated. This also leads to the only one neighbor found per vehicle. Lastly there was also little correlation between the density of vehicles and the number of collisions. This chapter covered any analysis of the raw data. The results chapter compares and identifies the patterns for the data in graphical form.

CHAPTER 6: Results

This chapter will compare the results in graphical form with a linear regression. It reports the confidence of the underlying data in chapter 5. The findings were in the areas of traffic congestion against network congestion, application layer message delivery, MAC layer message delivery and MAC layer channel utilization. The applied computation, aimed at identifying some of the patterns found in the analysis of chapter 5 and any additional information. The first section will cover the traffic and network congestion.

6.1 Traffic and Network Congestion

Network congestion is the quality reduction of a system when the vehicles are handling more data then their capability. The graph in Figure 6.1 compares the number of collisions against the start time per vehicle. It was expected for the collisions to increase during the accident and after the traffic jam. The activity happened at the incremented curve starting at 12 ms, with a peak at 36 ms. We assumed the drop between 11 ms and 12 ms start time was due to the simulation initiating the generation of the accident.

The last drop starting at start time 140 ms, was due to the low time the vehicles spent in the simulation. For example the vehicle before start time 140 ms had a start time of 109 ms. All the collisions during 31 ms interval and before, were not accumulated in the collisions of vehicles after 140 ms. It explains the huge drop of collisions at start time 140 ms.

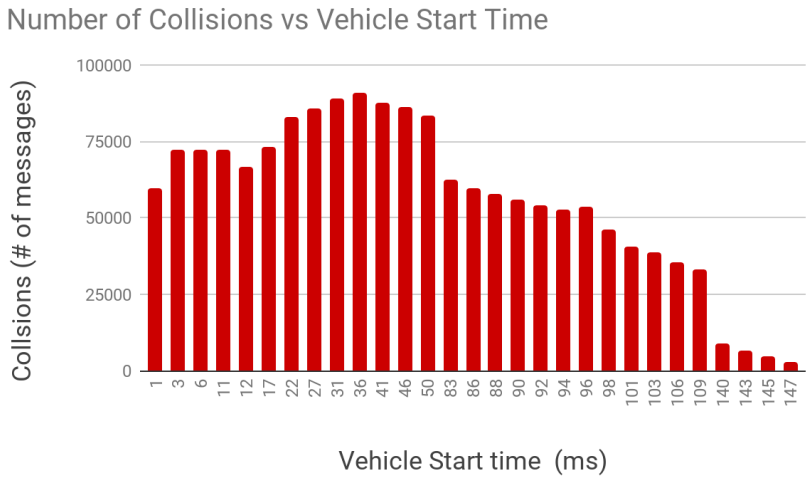

Figure 6.1 Number of Collisions VS Start Time.

As a relationship was derived between the number of collisions and vehicle start time, other points of traffic data were considered.

Some essential traffic data metrics recorded where vehicle density, max speed per vehicle, min speed per vehicle. Figure 6.2 compared the traffic variables against the total busy time for the MAC layer. We were assessing the likelihood of a vehicle density and speed affecting the outcome of the MAC busy time. Our intuition expected the number the mac busy time to increase as the vehicle density increased. However, the empirical evidence in Figure 6.2 supported a decrease of MAC busy time as the number of vehicles increased. Considering survivorship bias, we recommend a more thorough density scenario to formalized this outcome. Moving on, the speed of vehicles was also considered. The average min speed for the vehicles was 0.7896875877 m/ms. The max average speed was 5.352483742 m/ms. For this scenario there was no meaningful pattern found against the MAC busy time.

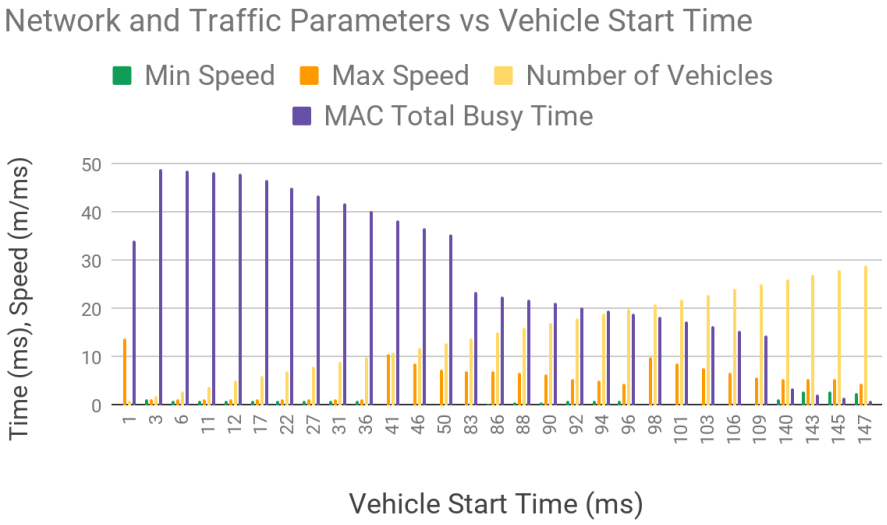

Figure 6.2 Network and Traffic Parameters vs Vehicle Start Time.

Beyond the external metrics, we looked at the relation of the message handling and vehicle start time. The following section will cover the results for message delivery at the application layer.

6.2 Application Layer Message Delivery

The applications layer is the initial point of entry for a message into the network. The metrics of generating, sending and receiving were recorded. The horizontal axis in Figure 6.3 was the starting time per vehicle, the vertical axis represented the number of data packets created and delivered. Near the left edge of the graph, the WSMs's received were at its highest of 34 packets, in the range 6 ms and 27 ms. The interval was very near the time range of the accident. On the right edged of the graph, the percent of WSMs generated and received was 0. It is the activity of the framework initiating a simulation stop.The middle interval had the drop of received messages after the accident. There was much greater fluctuation during this period.

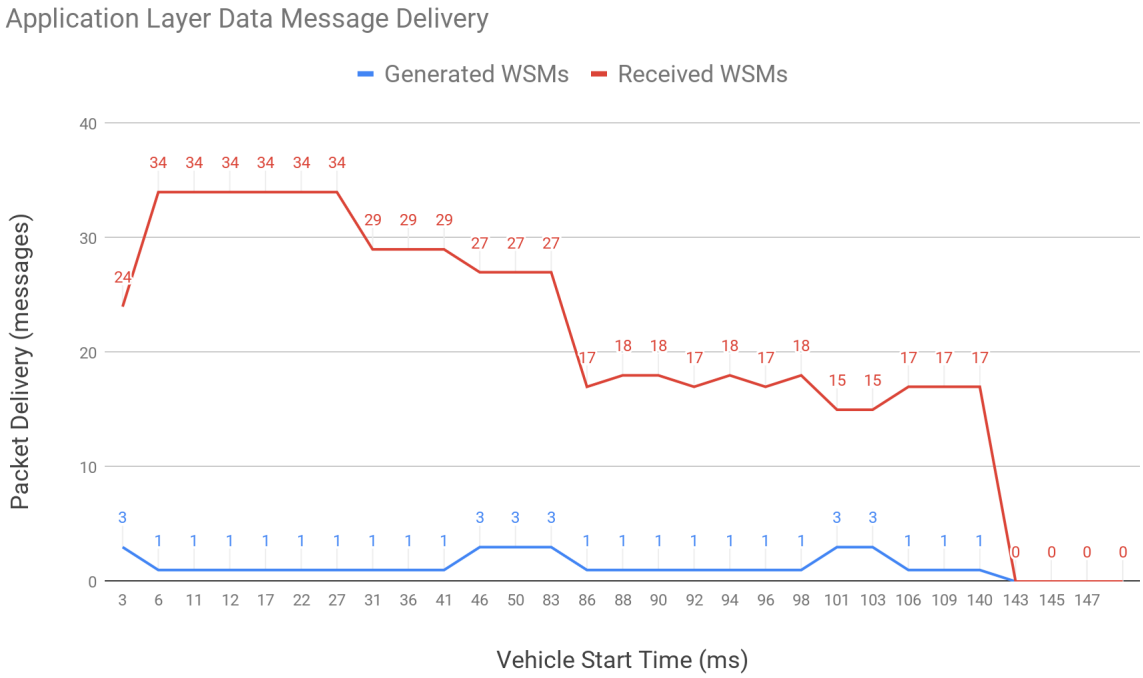

Figure 6.3 Application Layer Data Message Handling.

The averaged of generated data packets was 1 WSM/vehicle. Leading to an average of 20 WSM's received. These results were quite lower than the high average of 81420 BSMs generated and the 61441 BSMs received. The linear graph in Figure 6.4 has a profound comparison of delivery of BSMs against WSMs. Most importantly it compacted the scope having a large congestion of beacons in the scenario. The area difference between the yellow line of generated beacons and the green line of received beacons was much greater at the in the interval of the accident.

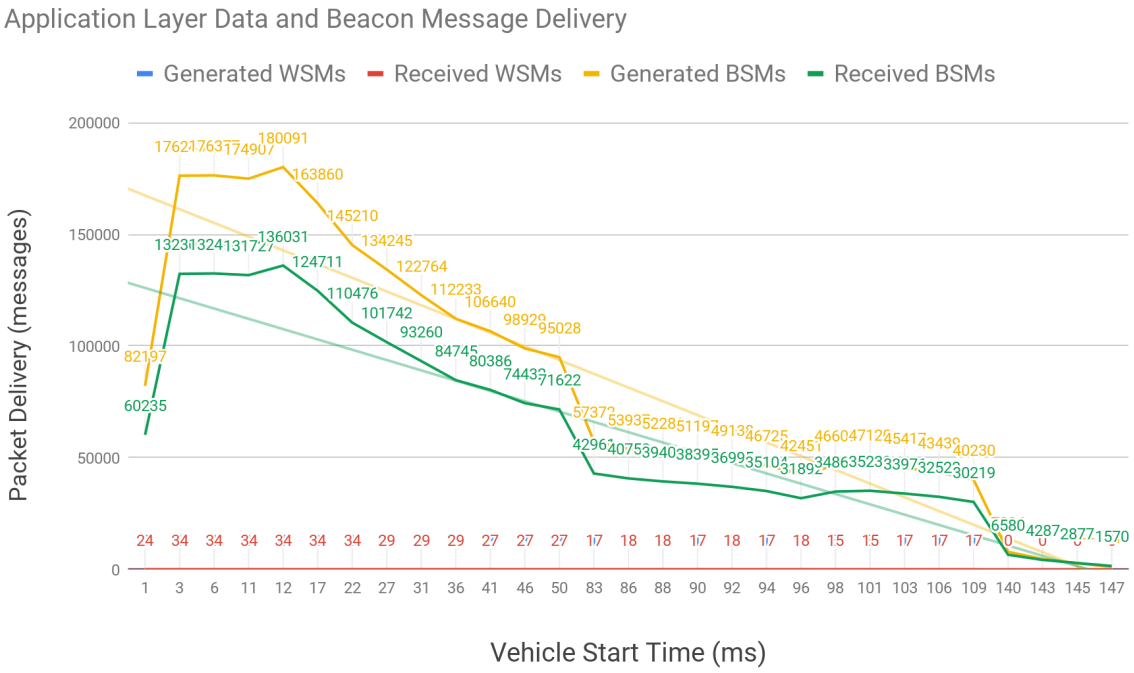

Figure 6.4 MAC Layer Packets Received per Channel Type.

Judging the increase of generated BSMs, the shift of increased collisions, found it was not beneficial to increase the beacons during the accident. For a future study, the generation rate of WSMs should be set much greater to reduce the survivorship bias. The mainstream of the multi-channel operation happens in the MAC layer. The following section will cover message delivery for the MAC layer.

6.3 MAC Layer Message Delivery

The results of the MAC layer in Figure 6.4, showed a similar graph line to Figure 6.5. The horizontal axis was the start time per vehicle. The vertical axis was the number of packets sent and the number of beacons delivered for Received Frames Lower Msg. The Received Frames Lower Msg represented beacons. Beacons were the purple line in MAC layer mapping to the yellow line in figure 6.4. In both figures, the beacons had a

decreasing slope targeting the bottom right corner. As mentioned earlier in sections 4.3 and 5.2, the MAC layer had no packets received. The problem was produced by the not attaching the receiving address to the WSM and the MacPkt. Revealing a default value of -1 in the MacPkt, which is the BROADCAST address. An unexpected result was the number of packets sent by the MAC layer. On average, 22441 packets were sent by the MAC in comparison to the average of one packet sent in application layer.

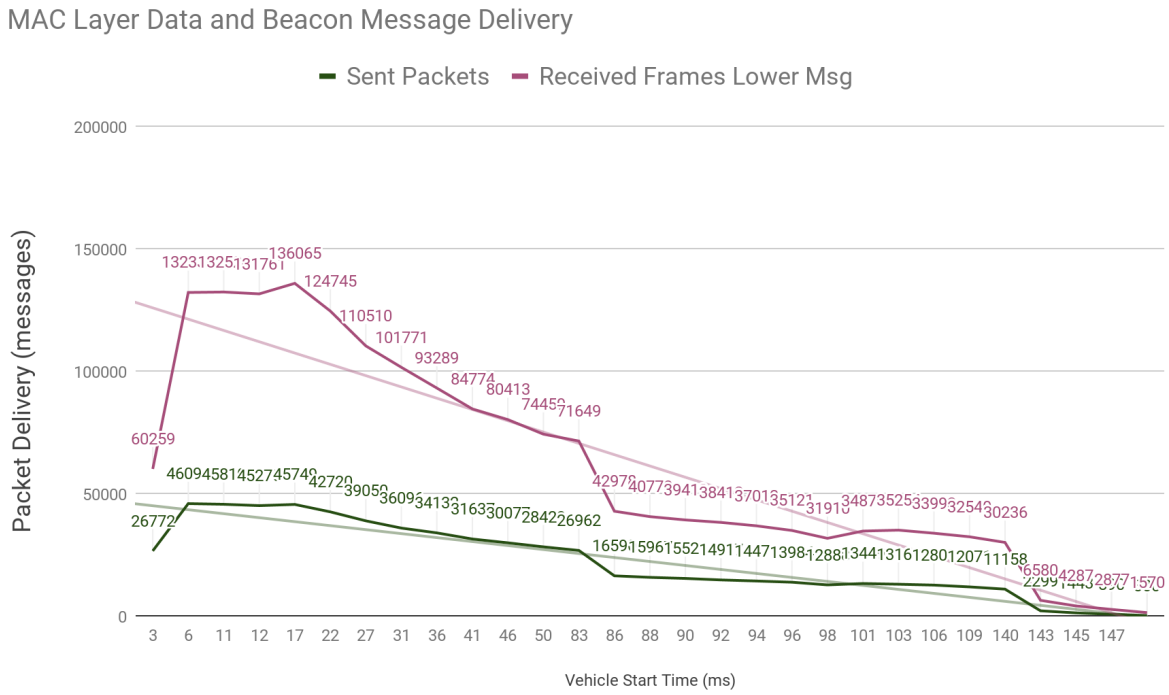

Figure 6.5 MAC Layer Data and Beacon Message Delivery.

Incidentally the MAC layer did send messages to it self to handle any internal scheduling. This could have been the cause of the extra packets or possibly a bug. There is definitely some uncertainty in this number. Moving on to how the traffic, application layer and MAC layer related to the multi-channel operation. The following section will cover the results of channel delivery and channel utilization.

21	98	51	11	80.26000588	0	9.855497331	46290	18.3306326
22	101	48	12	72.4172962	0	8.544792526	40712	17.25281308
23	103	46	13	64.98961882	0	7.626548871	38673	16.53584827
24	106	43	14	55.66143908	0	6.740322333	35298	15.44728718
25	109	40	16	46.67888	0	5.9430106	33090	14.39944335
26	140	9	17	37.11117662	1.261208888	5.421882829	8792	3.414398978
27	143	6	18	26.86569073	3.014944729	5.390808185	6450	2.35298223
28	145	4	19	16.25409363	2.761359579	5.504373607	4757	1.646960498

6.4 Channel Utilization Message Delivery

The break point of results happened at the number of messages delivered to each channel. Tying back to the channel utilization, we were expecting some high volume of beacons in the CCH. In figure 6.6 the received beacons in turquoise line depicts to the received beacons in figure 6.4 (purple line) and 6.4.(green line). The pattern is fairly close. There was an average difference of 17078 beacons between the lines. The largest difference of 45352 packets happened at vehicle start time of 12 ms.

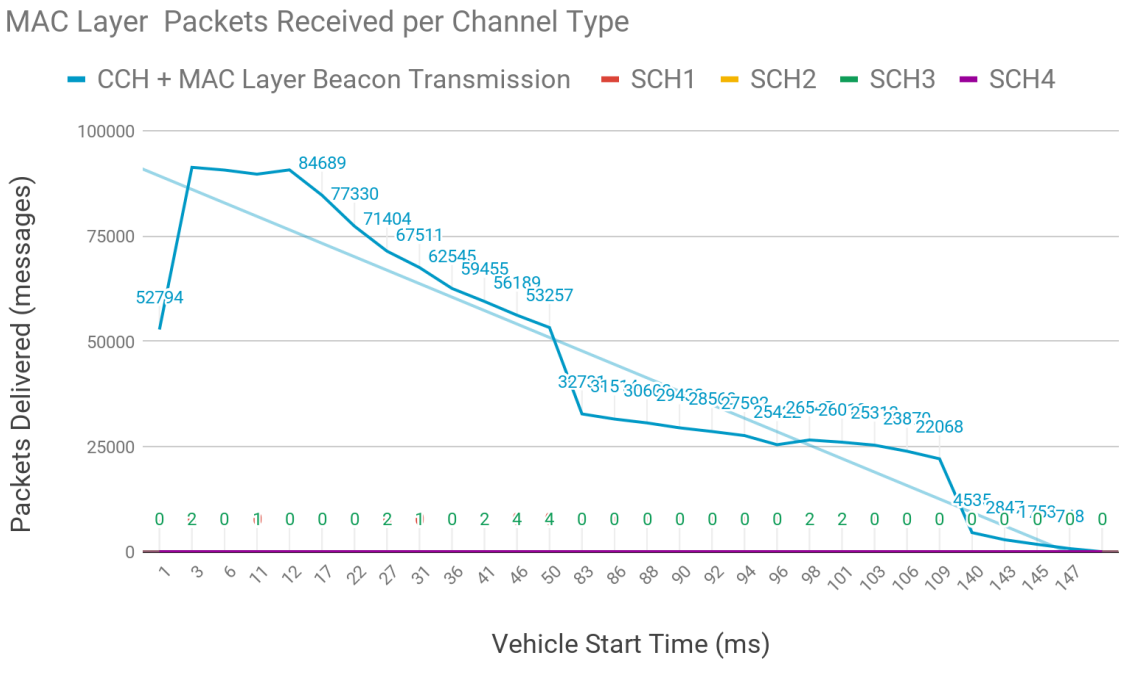

Figure 6.6 MAC Layer Packets Received per Channel Type.

We did expect a low number of packets to be received in the service channels with the low number of WSM packets generated in Figure 6.4. Note that the MAC layer had no data packets recorded as received and the SCH did have packets recorded. The implementation of the metric in the MAC layer checked for the destination and receiver id. The implementation for the packets in the service channel checked for the type of message received before decapsulation to a packet. For example, WSM categorized as a data packet and BSM as beacon. Overall, the outcome of these results was the overload of the CCH. This was further confirmed on the time used per channel in Figure 6.7. The horizontal axis plotted the vehicle start time. The vertical axis plotted the amount of time used per channel to transmit its messages, while active.

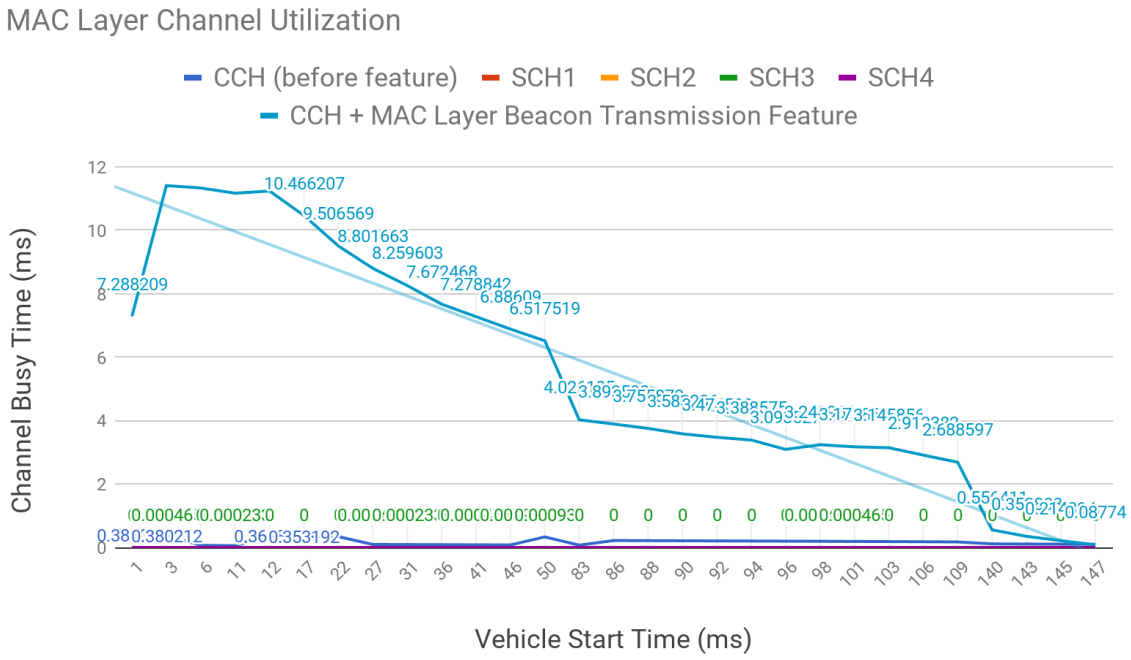

Figure 6.7 MAC Layer Channel Utilization.

Furthermore, the dark blue line in the graph represented the usability of the CCH before the beacon transmission feature in the MAC layer was contributed. The turquoise line was the usability with the feature. The results of the CCH with the MAC layer beacon transmission feature, expose congruent data aligned with the beacon activity. The main contributions of this thesis are the findings in the high usability of the CCH, and the beacon transmission feature to map the channel utilization to the beacons received in the application and MAC layer. We return to these thoughts with a summary in the conclusions chapter.

CHAPTER 7 : Conclusions and Future Work

The communication protocol for Wireless Access in Vehicular Environment (WAVE) [10], is the industry standard IEEE 802.11p to communicate between vehicles. This thesis examined the MAC layer of this IEEE 1609.4 multi-channel communication protocol. We considered both the type and priority of the message. We evaluated recent studies to illustrate problems covered for the impact of multi-channel and single-channel switching for non-safety and safety message transmissions. This thesis aimed to determine factors to the best utilize the Control Channel (CCH) and Service Channels (SCHs) in a Single Radio Multi-Channel (SR-MC) system [20 with IEEE 1609.4]. We analyze the channel utilization, beacon transmission, and packet transmission of IEEE 1609.4 multi-channel operations in CCH and SCH. Some of the parameters used for comparison are the number of collisions, channel utilization, packet transmissions, and beacon transmissions.

We investigate the scenario with density of n vehicles in a real world map, using safety (beacons) and non-safety (data) messages. The technologies used are Instant Veins 4.6, OMNET++ 5.2.1, SUMO 0.30, Debian GNU/Linux 9 (stretch) 64-bit, VMware Fusion (Professional Version 10.1.4) and an open street map from Northampton. The advantage of using OMNeT++ and Simulation Urban Mobility (SUMO) framework is the thorough implementation of IEEE 1609.4 DSRC/WAVE and IEEE 802.11p in the framework [29]

The prime contributions of this thesis included research features and a multi-channel analysis. The features were projected to the problem of establishing the impacts of multi-channel switching. They were derived by recent points of impact, IEEE standards and the existing Veins architecture. These features were Catch framework integration, MAC Layer Beacon Transmission, vehicle neighbors tracking and channel utilization tracking. The Catch framework enabled testing to the simulation. The prime contribution of the adding beacon transmissions to MAC layer, aided in accurate correlation of the beacon delivery in the application/MAC layer to the CCH. With the tracking of neighbors we found the receiving id of messages was not set for sent packets. Leading to zero received packets in the MAC layer. With the channel utilization feature we were able to see the message delivery activity in the channels. Additionally all the features are transitive to be used in future Vanet studies.

In the analysis, we examined the beacon and packet delivery for a single vehicle. Following with n vehicles in the application layer, n vehicle in the MAC layer and n vehicles in the multi-channel operation. We also looked at the numbers for traffic and network congestion. Aiming to find any significant relationships with the realistic model. We were assessing the effects of the IEEE 1609.4 multi-channel operations on beacon transmission and packet transmission, both CCH and SCH. Many research professionals focused on optimizing the usability of the service channels. Looking at the channel usability, the CCH undertook a high volume of beacons while managing the channel switching responsibilities. With the evidence, we concluded a higher significant impact in the CCH. Judging the increase of generated BSMs, a shift of increased collisions, inferred no benefit of increased beacons during the accident.

For future work, it is recommended to replicate the study with a higher number of packets generated and using generated beacons as the dependent variable. Setting the receiving address for the MAC layer will be needed. The analysis should have non-linear curves with the horizontal axis for generated beacons and the vertical axis for packet delivery. The highest point of the line segment should expose the maximum message delivery. This point can be used to select the ideal interval to generate beacons. This should include moving channel switching responsibilities out of the CCH.

References

[1] Varga, A. (2018). OMNeT++ - Simulation Manual. [online] Doc.omnetpp.org. Available at: https://doc.omnetpp.org/omnetpp/manual/ [Accessed 27 Nov. 2018].

[2] Ahyar, M. and Sari, R.F., 2013. Performance evaluation of multi-channel operation for safety and non-safety application on vehicular ad hoc network IEEE 1609.4. International Journal of Simulation--Systems, Science & Technology, 14(1), pp.16-22.

[3] Alaa, M., Abdala, M.A. and Al-Sherbaz, A., 2014. Evaluation study of IEEE 1609.4 performance for safety and non-safety messages dissemination. International Journal of Enhanced Research in Science Technology & Engineering, 3(11), pp.29-36.

[4] Báguena, M., Tornell, S.M., Torres, Á., Calafate, C.T., Cano, J.C. and Manzoni, P., 2013, June. VaCaMobil: VANET car Mobility Manager for OMNET++. In Communications Workshops (ICC), 2013 IEEE International Conference on (pp. 1057-1061). IEEE.

[5] Barr, R. (2018). JiST - Java in Simulation Time / SWANS - Scalable Wireless Ad hoc Network Simulator. [online] Jist.ece.cornell.edu. Available at: http://jist.ece.cornell.edu/docs.html [Accessed 27 Nov. 2018].

[6] Code.nsnam.org. (2018). ns-3.29: Summary. [online] Available at: http://code.nsnam.org/ns-3.29 [Accessed 27 Nov. 2018].

[7] Di Felice, M., Ghandour, A.J., Artail, H. and Bononi, L., 2012, July. On the impact of multi-channel technology on safety-message delivery in IEEE 802.11 p/1609.4 vehicular networks. In Computer Communications and Networks (ICCCN), 2012 21st International Conference on (pp. 1-8). IEEE.

[8] Di Felice, M., Ghandour, A.J., Artail, H. and Bononi, L., 2012, June. Enhancing the performance of safety applications in IEEE 802.11 p/ WAVE Vehicular Networks. In World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a (pp. 1-9). IEEE.

[9] Donato, E., Madeira, E. and Villas, L., 2015, July. Impact of desynchronization problem in 1609.4/WAVE multi-channel operation. In New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on (pp. 1-5). IEEE.

[10] Dressler, F., Kargly, F., Ottz, J., Tonguzx, O.K. and Wischh, L., 2010. Research Challenges in Inter-Vehicular Communication. In Lessons of the 2010 Dagstuhl Seminar, Institute of Computer Science, University of Innsbruck, Austria.

[11] Eiza, M.H. and Ni, Q., 2012, June. A reliability-based routing scheme for vehicular ad hoc networks (VANETs) on highways. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on (pp. 1578-1585).

[12] En.wikipedia.org. (2018). Radio spectrum. [online] Available at: https://en.wikipedia.org/wiki/Radio_spectrum [Accessed 27 Nov. 2018].

[13] Gamesfromwithin.com. (2004). Exploring the C++ Unit Testing Framework Jungle – Games from Within. [online] Available at: http://gamesfromwithin.com/exploring-the-c-unit-testing-framework-jungle [Accessed 27 Nov. 2018].

[14] Ghandour, A.J., Di Felice, M., Artail, H. and Bononi, L., 2014. Dissemination of safety messages in IEEE 802.11 p/WAVE vehicular network: Analytical study and protocol enhancements. Pervasive and mobile computing, 11, pp.3-18.

[15] Ghandour, A.J., Di Felice, M., Bononi, L. and Artail, H., 2012, March. Modeling and simulation of WAVE 1609.4-based multi-channel vehicular ad hoc networks. In Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (pp. 148-156). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[16] GitHub. (2018). catchorg/Catch2. [online] Available at: https://github.com/catchorg/Catch2/blob/master/docs/why-catch.md#top [Accessed 27 Nov. 2018].

[17] GitHub. (2018). namnatulco/veins. [online] Available at: https://github.com/namnatulco/veins/blob/master/src/veins/modules/obstacle/ObstacleControl.h [Accessed 27 Nov. 2018].

[18] GitHub. (2018). sommer/veins. [online] Available at: https://github.com/sommer/veins/blob/6f07401c04b4074212d94b646b06f5d1b53fb778/src/veins/base/connectionManager/ConnectionManager.ned [Accessed 27 Nov. 2018].

[19] GitHub. (2018). sommer/veins. [online] Available at: https://github.com/sommer/veins/blob/6f07401c04b4074212d94b646b06f5d1b53fb778/src/veins/modules/world/annotations/AnnotationManager.cc [Accessed 27 Nov. 2018].

[20] Hafeez, K.A., Anpalagan, A. and Zhao, L., 2016. Optimizing the control channel interval of the DSRC for vehicular safety applications. IEEE Transactions on Vehicular Technology, 65(5), pp.3377-3388.

[21] Klingler, F., Dressler, F., Cao, J. and Sommer, C., 2013, March. Use both lanes: Multi-channel beaconing for message dissemination in vehicular networks. In Wireless On-demand Network Systems and Services (WONS), 2013 10th Annual Conference on (pp. 162-169). IEEE.

[22] Klingler, F., Dressler, F., Cao, J. and Sommer, C., 2016. MCB–A multi-channel beaconing protocol. Ad Hoc Networks, 36, pp.258-269.

[23] Li, F. and Wang, Y., 2007. Routing in vehicular ad hoc networks: A survey. IEEE Vehicular technology magazine, 2(2).

[24] Lo, S.C. and Lu, W.K., 2009, April. Design of data forwarding strategies in vehicular ad hoc networks. In Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th(pp. 1-5). IEEE.

[25] Menouar, H., Lenardi, M. and Filali, F., 2007, September. Movement prediction-based routing (MOPR) concept for position-based routing in vehicular networks. In Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th (pp. 2101-2105). IEEE.

[26] Noori, H. and Olyaei, B.B., 2013, June. A novel study on beaconing for VANET-based vehicle to vehicle communication: Probability of beacon delivery in realistic large-scale urban area using 802.11 p. In Smart Communications in Network Technologies (SaCoNeT), 2013 International Conference on (Vol. 1, pp. 1-6). IEEE.

[27] Omnetpp.org. (2018). OMNeT++ Discrete Event Simulator. [online] Available at: https://omnetpp.org/ [Accessed 27 Nov. 2018].

[28] Sahu, P.K., 2012. Destination Discovery based Geographic Routing Protocol in VANET's Highway and City Scenarios. 中央大學資訊工程學系學位論文,pp.1-122.

[29] Sichitiu, M.L. and Kihl, M., 2008. Inter-vehicle communication systems: a survey - IEEE Journals & Magazine. [Accessed 27 Nov. 2018].

[30] Sommer, C. and Dressler, F. (2014) Vehicular Networking. Cambridge: Cambridge University Press. doi: 10.1017/ CBO9781107110649.

[31] Sommer, C., Tonguz, O.K. and Dressler, F., 2011. Traffic information systems: efficient message dissemination via adaptive beaconing. IEEE Communications Magazine, 49(5), pp.173-179.

[32] Standards.ieee.org. (2018). IEEE 802.11p-2010 - IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments.

[33] Sumo.dlr.de. (2018). Networks/Import/OpenStreetMap - Sumo. [online] Available at: http://sumo.dlr.de/wiki/Networks/Import/OpenStreetMap#Road_Types [Accessed 27 Nov. 2018].

[34] Sumo.dlr.de. (2018). OpenStreetMap file - Sumo. [online] Available at: http://sumo.dlr.de/wiki/OpenStreetMap_file [Accessed 27 Nov. 2018].

[35] Sumo.sourceforge.net. (2018). SUMO_User_Documentation - SUMO - Simulation of Urban Mobility. [online] Available at: http://sumo.sourceforge.net/userdoc/ [Accessed 27 Nov. 2018].

[36] Suthaputchakun, C. and Sun, Z., 2011. Routing protocol in inter vehicle communication systems: a survey. IEEE Communications Magazine, 49(12)

[37] van Eenennaam, M., van de Venis, A. and Karagiannis, G., 2012, November. Impact of IEEE 1609.4 channel switching on the IEEE 802.11 p beaconing performance. In Wireless Days (WD), 2012 IFIP (pp. 1-8). IEEE.

[38] Veins.car2x.org. (2018). Veins. [online] Available at: https://veins.car2x.org/ [Accessed 27 Nov. 2018].

[39] Vinayakray-Jani, P. and Sanyal, S., 2012. Routing protocols for mobile and vehicular Ad-Hoc networks: A Comparative Analysis. [online] Arxiv.org. Available at: https://arxiv.org/abs/1206.1918 [Accessed 27 Nov. 2018].

[40] Willke, T.L., Tientrakool, P. and Maxemchuk, N.F., 2009. A survey of inter-vehicle communication protocols and their applications. IEEE Communications Surveys & Tutorials, 11(2).