# On the Reliability of VANET Safety Applications for Diverse Traffic Participants

A Dissertation
Presented in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
with a
Major in Computer Science
in the
College of Graduate Studies
University of Idaho
by
Mohammad Mokhtar Baqer

Major Professor: Axel Krings, Ph.D.
Committee Members:
Michael Lowry, Ph.D.;
Robert Rinker, Ph.D.;
Terence Soule, Ph.D.
Department Administrator: Terence Soule, Ph.D.

May 2020

## Authorization to Submit Dissertation

This dissertation of Mohammad Mokhtar Baqer, submitted for the degree of Doctor of Philosophy with a Major in Computer Science and titled "On The Reliability of VANET Safety Applications for Diverse Traffic Participants", has been reviewed in final form. Permission, as indicated by the signatures and dates below, is now granted to submit final copies to the College of Graduate Studies for approval.

| | | |
|---|---|---|
| Major Professor: | Axel Krings, Ph.D. | Date |
| Committee Members: | Michael Lowry, Ph.D. | Date |
| | Robert Rinker, Ph.D. | Date |
| | Terence Soule, Ph.D. | Date |
| Department Administrator: | Terence Soule, Ph.D. | Date |

## Abstract

Dedicated Short Range Communication (DSRC) is used for wireless communication in the 5.9 GHz band that is assigned for Intelligent Transportation Systems (ITS). It allows vehicles to communicate with other vehicles (V2V) and with the infrastructure (V2I) to aid in traffic management and to reduce or avoid collisions, thereby increasing road safety. DSRC enables the establishment of Vehicular Ad-Hoc Network (VANET), which is designed for very quick message exchanges used for critical safety applications and services.

In the context of connected vehicles, the focus of safety applications has mainly been on collision avoidance of motor vehicles, and little attention has been given to the safety of other participants, like bicycles. This research presents safety applications for diverse participants in connected vehicles. First a bicycle safety application is introduced that aims to reduce the so-called right hook conflict, a common collision scenario where a right-turning vehicle causes a crash with an adjacent bicycle. The information exchanged during periodic beacon messages, called Basic Safety Messages (BSMs), emitted by vehicles is used by the safety application to alert drivers of potential collisions with bicycles. The goal is to achieve this without introducing addition message overhead or deviating from existing standards.

Wireless vehicle-to-vehicle communication is vulnerable to malicious jamming attacks not only for motorized vehicles, but also for any VANET participants, such as bicycles. This has potential impact on the reliability of a bicycle safety application since this application also depends on a timely reception of BSMs, that are sent periodically every 100ms. The bicycle safety application uses an algorithm that has the capability to mitigate against such attacks, as well as message loss due to natural phenomena. The algorithm was analyzed in terms of accuracy and safety application reliability. Its effectiveness was evaluated based on real-world field experiments using commercial equipment installed in the vehicles and bicycle.

The research so far considered a mobility model characterized by relatively fast moving vehicles, e.g., motorized vehicles traveling at the speed limit or fast moving bicycles. Now, the traffic participants are expanded to include visually impaired persons and wheelchair users. These two new traffic participants introduce unique properties that require an investigation of Safety Applications for slow moving traffic participants. These properties are relatively low speed and the potential to turn over very short distances. We will describe scenarios using visually impaired people and wheelchair users to investigate accuracy and timeliness of OBUs heading information based on real-world field experiments.

Finally VANET safety applications for scooters are investigated, which consider their unique mobility model, characterized by possible radical changes in heading over short distance at speed higher than that of pedestrians, but possibly lower than motorized vehicles. Furthermore, scooters are likely to behave rather unpredictable. They may drive close to or between vehicles, in-between lanes, or between moving traffic and parked vehicles. As this could have implications on GPS coordinate accuracy the reliability of GPS information related to the mounting point of GPS antennas on scooters in different scenarios is investigated.

## Acknowledgements

First and foremost, I would like to thank GOD the beneficent the merciful for his support to have this opportunity to complete my research. Next, I want to express my sincere and deepest gratitude for my Major Professor, Axel Krings, for his caring, patience and support for accomplishing this research.

Special thanks for my committee members for their support and guidance: Dr. Michael Lowry, Dr. Robert Rinker and Dr. Terence Soule. Another special thanks goes to Firas Assaad, Salahaldeen Duraibi, Mubarak Al-Mubarak and Ali Al-Mosaqlab who are true friends with unlimited kindness and support and who are always willing to help.

Also, I would like to thank the Department Chair Dr. Terence Soul, the Department Manager Sue Branting, the Administrative Coordinator Arleen Furedy and the System Manager Victor House for all the help and support. Moreover, I am thankful to the Ministry of Information and the Civil Service Commission in Kuwait for facilitating the process to study abroad. Finally, I am grateful to the University of Idaho faculty, staff, students and friends for providing me a healthy and energetic environment to succeed and accomplish my educational journey leading to this dissertation.

## Dedication

This dissertation is dedicated to the memory of my father,

**Mokhtar Baqer**

who inspired me with science;

To my mother,

**Zeinab**

who provided me the endless care and love;

To my wife,

**Mona**

for her truly patience, and heartening to pursue my research;

Finally to my kids,

**Abullah, Noor & Fajer**

who make it all fruitful.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| RHC | Right Hook Conflict |
| BSM | Basic Safety Message |
| CCH | Control Channel |
| CSMA/CA | Carrier Sense Multiple Access with Collision Avoidance |
| DNPW | Do Not Pass Warning |
| DoS | Denial of Service attack |
| DSRC | Dedicated Short Range Communication |
| EEBL | Emergency Electronic Brake Lights |
| FCC | Federal Communication Commission |
| FCW | Forward Collision Warning |
| GPS | Global Positioning System |
| HV | Host Vehicle |
| I2V | Infrastructure-to-Vehicle |
| ICA | Intersection Collision Avoidance |
| ITS | Intelligent Transportation Systems |
| LCW | Lane Change Warning |
| MAC | Media Access Control |
| MANET | Mobile Ad-Hoc Networks |
| NHTSA | National Highway Traffic Safety Administration |
| OBU | On-Board Unit |
| PDR | Packet Delivery Ratio |
| PER | Packet Error Rate |
| RSU | Road Side Unit |
| RV | Remote Vehicle |
| SCH | Service Channel |
| USDOT | United States Department of Transportation |
| V2I | Vehicle-to-Infrastructure |
| V2V | Vehicle-to-Vehicle |

| | |
|---|---|
| V2P | Vehicle-to-Pedestrian |
| V2P | Vehicle-to-Bicycle |
| V2E-s | Vehicle-to-Escooter |
| VANET | Vehicular ad hoc Networks |
| WAVE | Wireless Access in Vehicular Environments |
| WSMP | WAVE Short Message Protocol |
| WSM | WAVE Short Messages |
| MIPS | Microprocessor without Interlocked Pipeline Stages |

CHAPTER 1

# Introduction

One of the essential goals of Intelligent Transportation Systems (ITS) is to increase safety and reduce collisions. A report by the National Highway Traffic Safety Administration (NHTSA) showed that 6.1 million collisions were reported in 2014 [1]. Lately, in the context of connected vehicles, Safety Applications (SA) were introduced that rely on communication between vehicles and with the infrastructure, such as traffic lights in an intersection. These safety applications are expected to reduce road collisions by up to 82% and eventually will save thousands of lives in the United States [2]. In the past SA were mainly discussed in the context of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I), however, little consideration has been given to other traffic participants like bicycles and their special needs. A particular common source of bicycle collisions is the so-called Right Hook, where a turning vehicle crashes with a bicycle to its right, while performing the turn. The study in [3] showed that vehicles were almost unaware of the adjacent bicycle when performing the right turn.

This research introduces a bicycle safety application that uses the same basic communication capabilities as vehicles, thus allowing vehicle-to-bicycle communication. We simply assume that bicycles are vehicles capable of V2V and V2I communication.

The main two technologies that facilitate communications are Dedicated Short Range Communication (DSRC) [2], which was assigned by the Federal Communications Commission (FCC) in 1999 to be used for vehicular communication [4], and cellular vehicle-to-everything (C-V2X), which has been driven by the Third Generation Partnership Project (3GPP) [5].

Whereas in general cellular networks communication includes base stations, C-V2X communication can be directly between vehicles in a Device-to-Device (D2D) fashion [6]. In this paper we will focus on DSRC, however, the general issues discussed are expected to have similar implication is C-V2X as well.

This research does not consider line-of-sight sensor technology such as radar or lidar, but rather focuses on non-line-of-sight communication based on DSRC.

## 1.1 PRELIMINARY BACKGROUND INFORMATION

The communicating nodes in V2V and V2I implement a Vehicular Ad Hoc Network (VANET). VANETs are similar to Mobile Ad Hoc Networks (MANET), but they consider short message exchanges and a fast changing network topology. Communication is assumed to be based on DSRC, which operates in a dedicated bandwidth of 75 MHz at 5.9 GHz [9]. One of the seven DSRC channels, i.e., Channel 172, is assigned to safety applications. It is used to broadcast the Basic Safety Message (BSM), which is the most important beacon message broadcast by each vehicle every 100ms [2]. Each BSM contains information related to the sender's Global Positioning System (GPS) position, and additional information like position accuracy, speed, heading, steering wheel angle, transmission and break status. This information is used by the safety applications executing in each vehiclel's OBU.

In [16] safety applications and their associated crash scenarios are identified. We will describe SA from the viewpoint of a Host Vehicle (HV), which receives BSMs from Remote Vehicles (RV). When specific event information extracted from a received BSM of an RV suggests a critical situation, the driver of the HV is issued an alert.

## 1.2 RESEARCH MOTIVATION AND OBJECTIVES

The focus of the first part of the dissertation is on the reliability of the Bicycle Safety Application for the so-called the Right Hook Conflict, where a truck driver is planning to make a right turn and a bicyclist to the truck's right-side is overlooked, resulting in a collision. Moreover, this dissertation will present a detection mechanism to allow a vehicle within a VANET to detect a bicycle within a very short distance to avoid a possible collision.

Since VANET is vulnerable to DoS attacks, where a malicious node jams the communication for the DSRC safety applications, the dissertation will address the potential vulnerability of safety applications as the result of such attack. Then a solution is given to mitigate the attack, which otherwise can lead the safety application to fail, potentially resulting in injury or loss of life.

The proposed safety applications were implemented using commercial equipment, installed in the vehicle and bicycle, and the effectiveness was evaluated based on real-world field experiments.

## 1.3 SUMMARY OF CONTRIBUTIONS

The main contributions of the dissertation are as follows:

- A Safety Application for Bicycles is proposed for connected vehicles. The bicycle safety application aims to reduce the so-called right hook conflict, which is the collision scenario where a right-turning vehicle causes a crash with an adjacent bicycle. The information exchanged during normal beacon messages of vehicles is used by the application to alert drivers of potential collisions with bicycles without introducing addition message overhead or deviating from current standards.

- A VANET Bicycle Safety Application considering Malicious Environments is proposed, with special focus on jamming attacks. A bicycle safety application algorithm is presented that is capable of mitigating against such attacks, as well as message loss due to natural phenomena. The algorithm was analyzed in terms of accuracy and application reliability.

- As a third and final contribution, safety applications for different VANET participants that have unique mobility models were investigated. Mainly, micro-mobility models such as applicable for visually impaired persons and wheelchair users were considered. In the absence of gyros, the safety application calculates the heading from the information given by the OBUs. In addition, E-scooters were studied as they share certain properties of the micro-mobility model used for pedestrians and wheelchairs. Lastly, an algorithm is introduced to mitigate traffic collisions for different VANET participants.

## 1.4 DISSERTATION OUTLINE

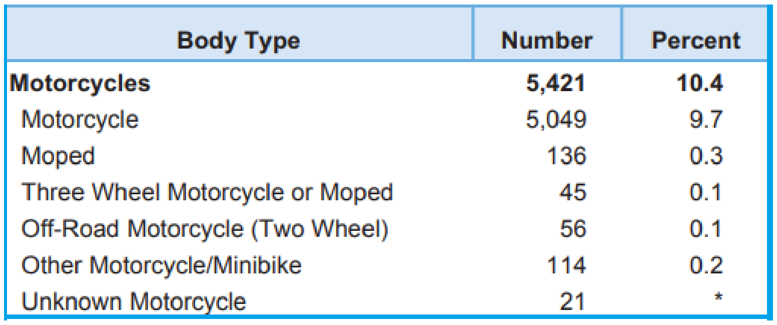
The rest of the dissertation will be organized as follows: Chapter 2 will cover the relevant background information. VANET Safety Applications for Bicycles is discussed in detail in Chapter 3. The Reliability of VANET Bicycle Safety Applications in non-malicious and malicious environments is presented in detail in Chapter 4 and Chapter 5 respectively. Safety Applications for VANETs with Diverse Traffic Participants are introduced in Chapter 6. Lastly, conclusions are presented in Chapter 7.

CHAPTER 2

# Background

## 2.1 INTELLIGENT TRANSPORTATION SYSTEMS

A key objective of Intelligent Transportation Systems is to increase safety and reduce collisions. One of the most recent considerations for collision reduction are safety applications (SA) in the context of connected vehicles. Figure 2.1 is an example of fatal crash data. The SAs are implemented in a device called On Board Unit (OBU),



| Body Type | Number | Percent |
|---|---|---|
| **Motorcycles** | **5,421** | **10.4** |
| Motorcycle | 5,049 | 9.7 |
| Moped | 136 | 0.3 |
| Three Wheel Motorcycle or Moped | 45 | 0.1 |
| Off-Road Motorcycle (Two Wheel) | 56 | 0.1 |
| Other Motorcycle/Minibike | 114 | 0.2 |
| Unknown Motorcycle | 21 | * |

FIGURE 2.1: Vehicles Involved in Fatal Crashes, [modified from] [1]

which are mounted in the vehicles. OBUs are responsible for communications between vehicles (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P) and Vehicle-to-Bicycle (V2B). In addition, the OBUs can operate with the vehicle's CAN-BUS to have more information about the vehicular system [7, 8]. Road Side Units (RSU) on the other hand are fixed units installed in the infrastructure, e.g., intersection or along a highway. They are responsible for V2I communication, or other RSUs and back-end servers for ITS to process data [7]. The communication architecture in VANET enables traffic participants to use V2V, V2I, V2P and V2B, collectively also called V2X, to distribute status information to surrounding vehicles by using OBUs and RSUs as illustrated in Figure 2.2. The safety applications use the information to predict and alert the driver to potential hazards.

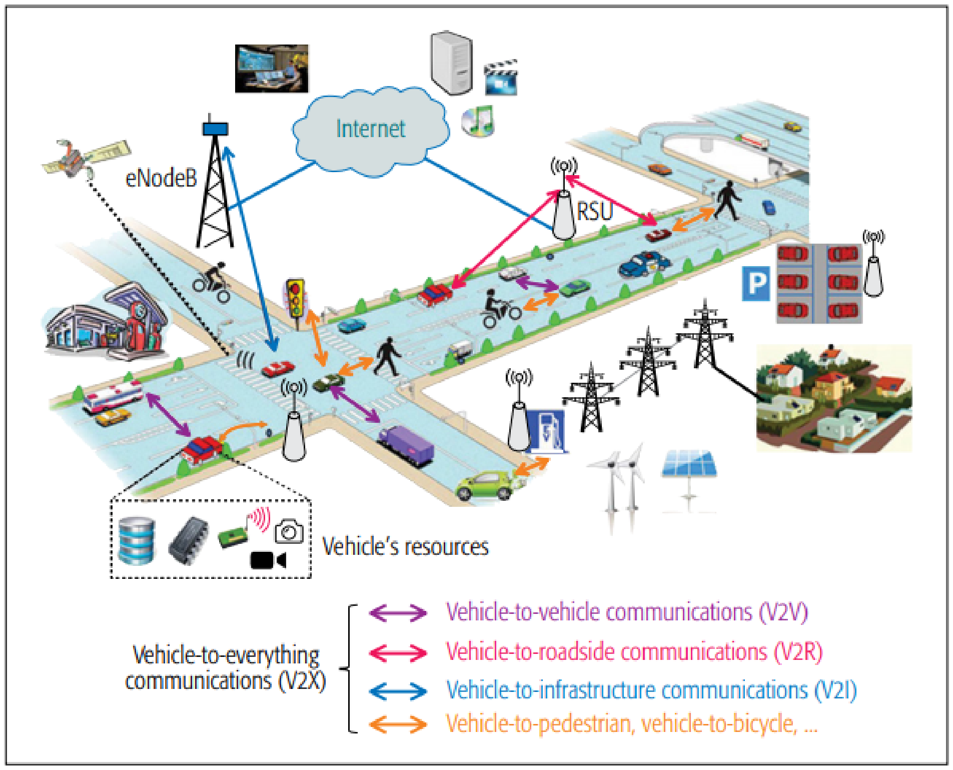

Internet
eNodeB
RSU
Vehicle's resources
Vehicle-to-everything communications (V2X)
Vehicle-to-vehicle communications (V2V)
Vehicle-to-roadside communications (V2R)
Vehicle-to-infrastructure communications (V2I)
Vehicle-to-pedestrian, vehicle-to-bicycle, ...

FIGURE 2.2: V2V, V2R, V2I, V2P and V2B Communications, [8]

### 2.1.1 *DSRC, IEEE 802.11P and WAVE Protocol*

The Federal Communications Commission (FCC) together with the US Department of Transportation (USDoT) assigned 75MHz of dedicated bandwidth at 5.9GHz to be used for DSRC communication in 1999 [9]. The DSRC spectrum is shown in Figure 2.3. Within this spectrum of 5.850-5.925GHz, six service channels are denoted by CH172, CH174, CH176, CH180, CH182, and CH184, in addition with a control channel CH178 [21]. Channels can be combined to allow a larger bandwidth of 20 MHz, e.g., CH174 with CH176 and CH180 with CH182.

The WAVE in VANET is used for low latency communication to deliver a high bit-rate speed among OBUs. These units can communicate under DSRC with different protocol types depending on the application. For example, the protocol WAVE is used for safety applications while TCP, UDP, and IPv6 are used for a non safety application as shown in Figure 2.4.

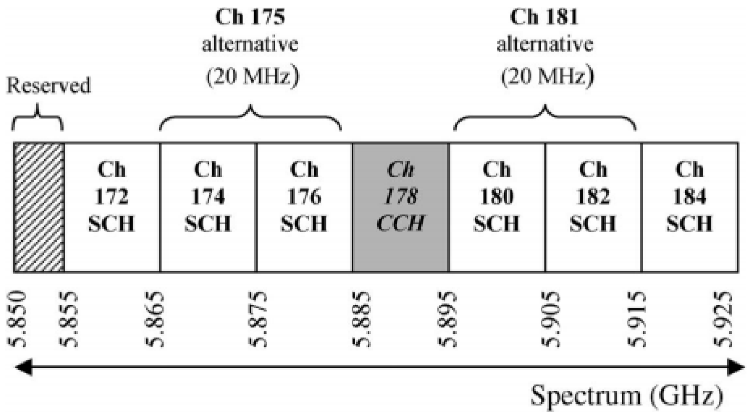


Figure 2.3: DSRC Spectrum, [2]

The IEEE 802.11p protocol is an enhanced version of IEEE 802.11 used to serve WAVE communications. Furthermore, the 802.11p protocol is used in DSRC Medium Access Control layer (MAC). In addition, the WAVE protocol has Enhanced Distributed Channel Access (EDCA) implementation [23, 24].

The IEEE 802.11p protocol uses some features of the previous IEEE 802.11a and IEEE 802.11e versions. For example, the IEEE 802.11a uses the OFDM modulation technique, which is beneficial for VANET applications. Also, the quality of service that IEEE 802.11e offers for message priority is implemented in IEEE 802.11p for VANET applications [19].

The IEEE 802.11p standard spans two layers, the physical layer and the MAC layer of the DSRC protocol. The 1609 protocol series are used as follows; The 1609.4 is used for channel switching. For example, the communications in DSRC may occur in the control channel or a service channel depending on the type of the application. The second standard of 1609 series is 1609.3. It is used for services while the 1609.2 standard is used for security purposes. In the application layer of the DSRC stack, the SEA J2735 standard is used for Basic safety message structure [7] as shown in Figure 2.5 More about IEEE 208.11p performance can be found in [25].

In this research, the most important channel that we are going to focus on is the safety channel CH172, which has a band of 10MHz, and is assigned to safety applications for vehicle to vehicle communication. In line with the standard, the CH172 will also be used in the proposed Diverse Traffic Participants safety application.

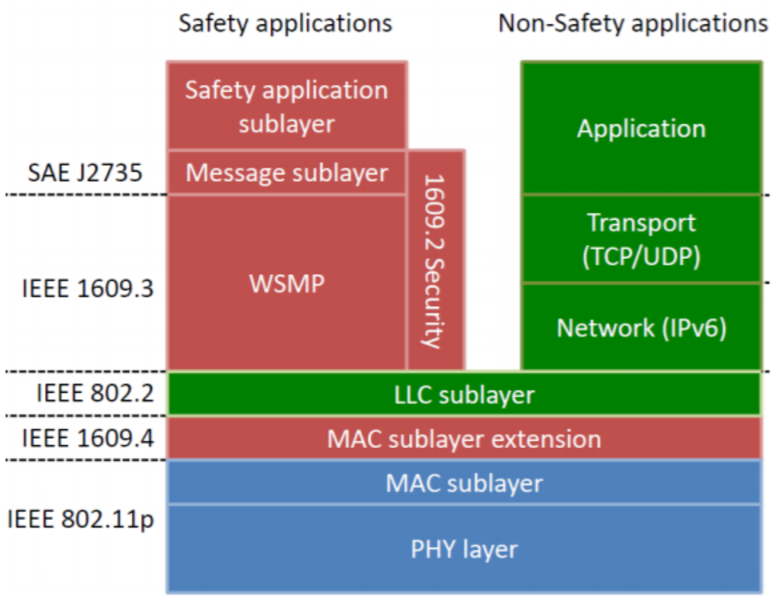


Figure 2.4: DSRC Architecture, [7]

### 2.1.2 *Basic Safety Message*

The most important message related to safety applications is the Basic Safety Message (BSM) shown in Figure 2.6. It is a beacon message broadcast by each vehicle every 100ms [2]. According to standard SAE J2735, the BSM has a mandatory part 1, and an optional part 2. The mandatory part consist of fourteen fields as described in [22]: *MessageID* is a one byte fields used to indicate the message type, so the receiver knows how to interpret the remaining bytes. *MsgCount* is a one byte fields, which is a sequence number of successive BSMs sent by a specific vehicle. *TemporaryID* is a four byte field, which is a temporary id of a sender. *DSecond* is a two byte field that encodes the current time.*Latitude* and *Longitude* are four byte each, and hold the geographic latitude and longitude. *Elevation* is a two byte field used to indicate the geographic position above or below sea level. *PositionalAccuracy* is a four byte field used to indicate the position error along different axis. *TransmissionAndSpeed* is a two byte field indicating the transmission's gear and the speed in meters per second. *Heading* is a two byte field showing the current heading of the vehicle's motion. *SteeringwheelAngle* is a one byte field indicating the angle of the steering wheel. *AccelerationSet4Way* is a four byte field providing longitudinal, lateral, and

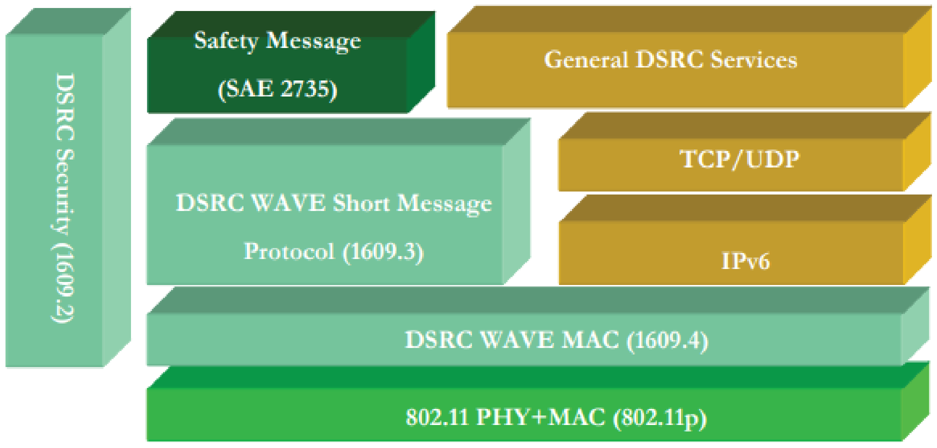


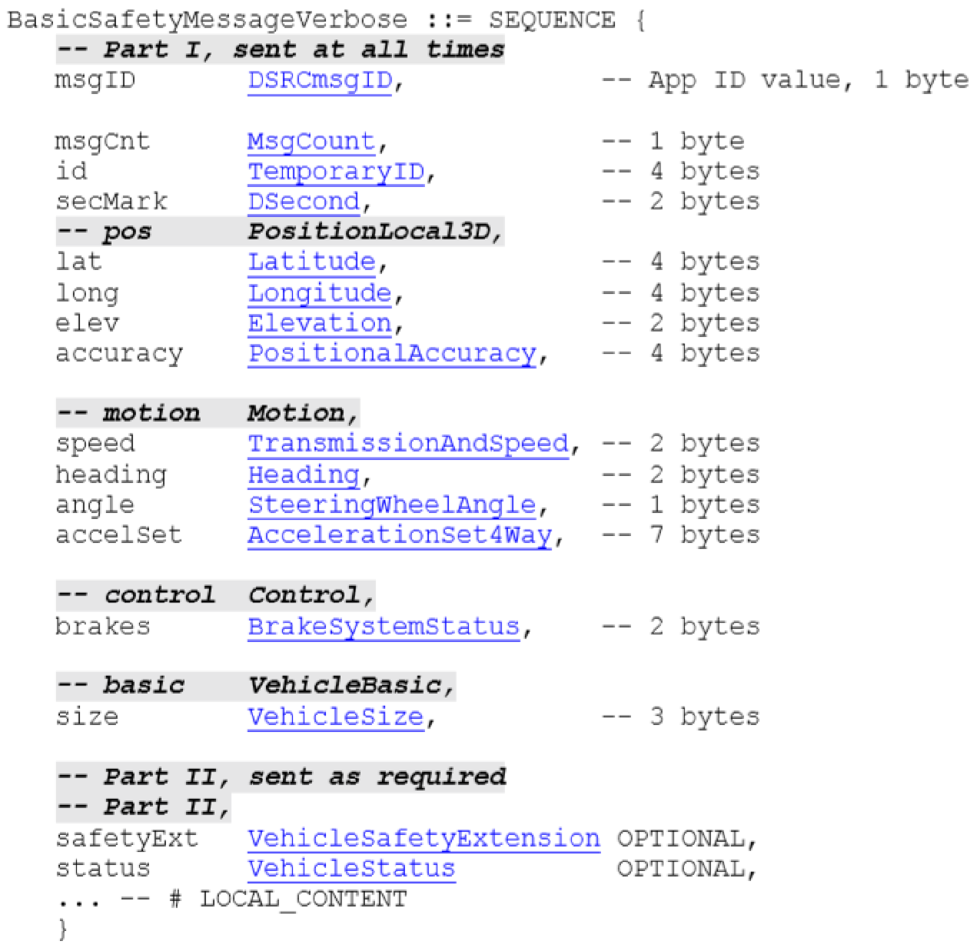
FIGURE 2.5: DSRC Layered Architecture, [16]

vertical acceleration, in addition to the yaw rate. *BrakeSystemStatus* is a two byte field used to indicate information about the current brake system status, such as brake usage, or anti-lock brake status. Lastly, *VehicleSize* is a three byte field used to provide the vehicle length and width.

The optional BSM part 2 is used to provide additional information for specific applications. The most significant BSM fields used in this research are the GPS position fields, the speed and steering wheel angle. Other fields can be used to filter out vehicles not relevant to the safety application, e.g., using Heading to filter out vehicles in opposite direction on a divided multi-lane highway.

### 2.1.3 *DSRC Safety Applications*

In general there are many applications in VANET, which consist of safety applications and non-safety applications discussed in [19]. This research focuses on safety applications in VANET since they are more critical than none-safety applications. In a report by the USDoT and the Crash Avoidance Metrics Partnership on behave of the Vehicle Safety Communications 2 Consortium [16] several crash scenarios and safety applications were identified. The goal of the safety applications is collision prevention and hazard avoidance. The safety applications use information from the periodically exchanged BSMs, such as GPS coordinates or vehicle status information, to issue alerts to drivers in case of hazards. Safety applications will be described



```
BasicSafetyMessageVerbose ::= SEQUENCE {
   -- Part I, sent at all times
   msgID        DSRCmsgID,              -- App ID value, 1 byte

   msgCnt       MsgCount,               -- 1 byte
   id           TemporaryID,            -- 4 bytes
   secMark      DSecond,                -- 2 bytes
   -- pos       PositionLocal3D,
   lat          Latitude,               -- 4 bytes
   long         Longitude,              -- 4 bytes
   elev         Elevation,              -- 2 bytes
   accuracy     PositionalAccuracy,     -- 4 bytes

   -- motion    Motion,
   speed        TransmissionAndSpeed,   -- 2 bytes
   heading      Heading,                -- 2 bytes
   angle        SteeringWheelAngle,     -- 1 bytes
   accelSet     AccelerationSet4Way,    -- 7 bytes

   -- control   Control,
   brakes       BrakeSystemStatus,      -- 2 bytes

   -- basic     VehicleBasic,
   size         VehicleSize,            -- 3 bytes

   -- Part II, sent as required
   -- Part II,
   safetyExt    VehicleSafetyExtension OPTIONAL,
   status       VehicleStatus          OPTIONAL,
   ... -- # LOCAL_CONTENT
   }
```

Figure 2.6: Basic Safety Message, [22]

from the viewpoint of a Host Vehicle (HV), which receives beacon messages from Remote Vehicles (RV). When specific event information received in the BSM from an RV suggests a critical situation, the driver of the HV is issued an alert. The report, [16], identified seven safety applications.

Emergency Electronic Brake Lights (EEBL) shown in Figure 2.7 refers to the situation where a vehicle is subjected to a potential rear-end collision. When a vehicle brakes hard, the so-called hard-braking event is broadcasted in its BSMs. When an HV receives such event from the RV's BSM, it can issue an alert to the driver. This is particularly helpful when the driver's line of sight to the initiating RV is obstructed.

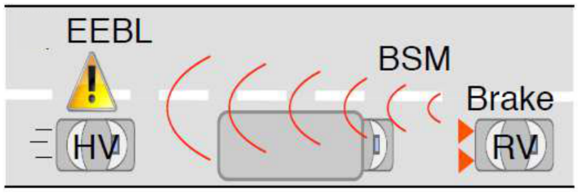


Figure 2.7: Emergency Electronic Brake Lights (EEBL)

Forward Collision Warning (FCW) shown in Figure 2.8 is similar, in that it warns the driver of the HV of a potential rear-end-collision with a vehicle in the same lane and direction of travel.

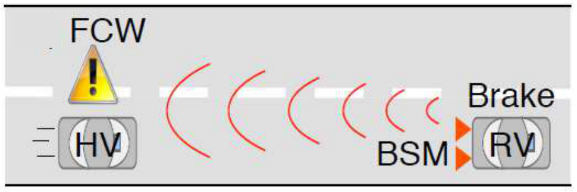


Figure 2.8: Forward Collision Warning (FCW)

Blind Spot Warning+Lane Change Warning (BSW+LCW) shown in Figure 2.9 addresses situations related to lane changes, when vehicles are hidden in the blind spot.

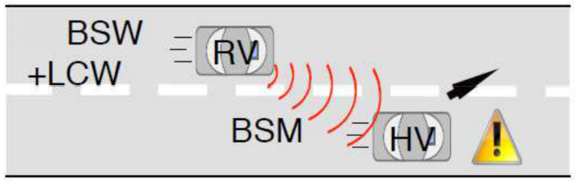


Figure 2.9: Blind Spot Warning+Lane Change Warning (BSW+LCW)

Do Not Pass Warning (DNPW) shown in Figure 2.10 warns the driver of the HV during a passing maneuver attempt that it is unsafe, due to an oncoming vehicle in the passing zone.

Intersection Movement Assist (IMA) shown in Figure 2.11 warns the driver of the HV that it is not safe to enter an intersection due to a potential crash with an RV in an intersection.

Lastly, Control Loss Warning (CLW) allows to warn a driver in response to a *control loss event* broadcast from an RV that has lost control.

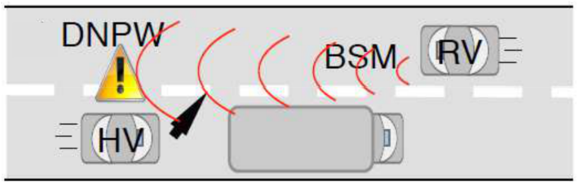


Figure 2.10: Do Not Pass Warning (DNPW)

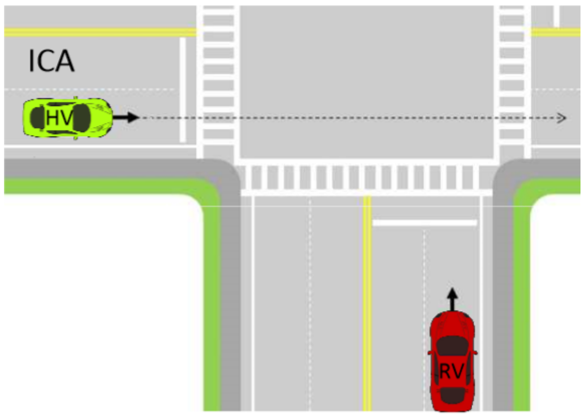


Figure 2.11: Intersection Movement Assist (IMA)

In the aforementioned safety applications bicycles play only a peripheral, limited role. For example, bicycles often drive at lower speeds, they may occupy limited space in the right lane, are often overlooked by the drivers of vehicles, and the riders are much more vulnerable and susceptible to injuries in an collision, e.g., a right hook collision.

Approaching Emergency Vehicle Application is also an essential application in VANET to give a high traffic priority for the emergency vehicle to pass safely through alerting other vehicles of its approach [20]

### 2.1.4 *EDCA Channel Access Rules*

The media access control (MAC) in VANET uses IEEE 802.11p, an amendment of IEEE 802.11. It plays a big role to form the communicating between the On-Board Unit (OBU) and Road Side Unit (RSU) in VANET. The channel access in VANET, shown in Figure 2.12, is formed by the help of the MAC protocol coordinated channel access. The IEEE 802.11p uses the same mechanism implementation that IEEE 802.11 uses, which is the Distributed Coordination Function (DCF) to facilitate the contention of multiple access request for a channel through the help of Carrier Sense Multiple Access with Collision Avoidance(CSMA/CA). Figure 2.12 illustrates the access mechanism. Assume a node in a network, e.g., a bicycle equipped with a Locomate Me, wants to transmit a packet to a truck's OBU. First it will check if the channel is idle for a time known as Distributed Inter-Frame Space (DIFS) before any transmission and if channel is busy the access is deferred and if the channel is idle a back-off time is selected before any transmission to avoid a packet collision [23].

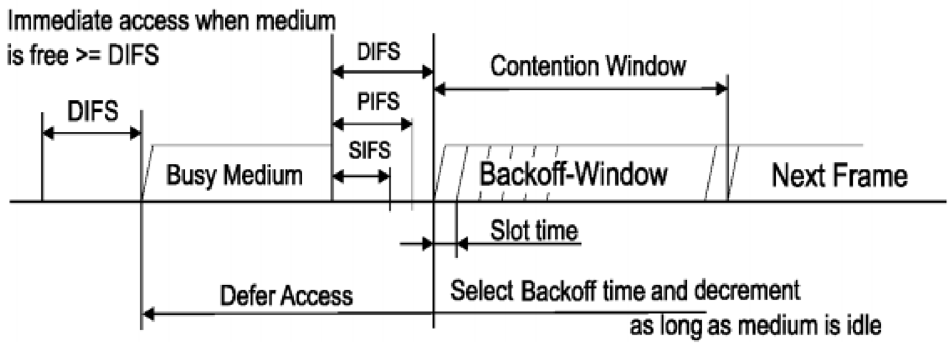

Immediate access when medium is free >= DIFS
DIFS
DIFS
PIFS
SIFS
Contention Window
Busy Medium
Backoff-Window
Next Frame
Slot time
Defer Access
Select Backoff time and decrement as long as medium is idle

Figure 2.12: Basic Access Method, [23]

### 2.1.5 *Arada Systems Locomate OBU and Locomate ME*

Arada Systems offers OBUs and RSUs, which are available in our laboratory as shown in Figure 2.13. These unites can be used on vehicles and road infrastructures to enable communication between the nodes to form what is known as Vehicular Ad Hoc Network (VANET). The block digram of the OBU from the Arada's user guide manual [49] is shown in Figure 2.14. As indicated in the Arada's user guide manual, the LocoMate uses a MIPS processor running at 680MHz. It should be noted that in this research we focus on the Arada Locomate classic OBU and the Arada portable Locomate ME as shown in Figures 2.15 to conduct experiments for all of the presented data. The data was extracted from the Basic Safety Messages.

### 2.1.6 *Possible Attacks in VANET*

VANET is prone to different possible network attacks since the communication between vehicles is wireless.

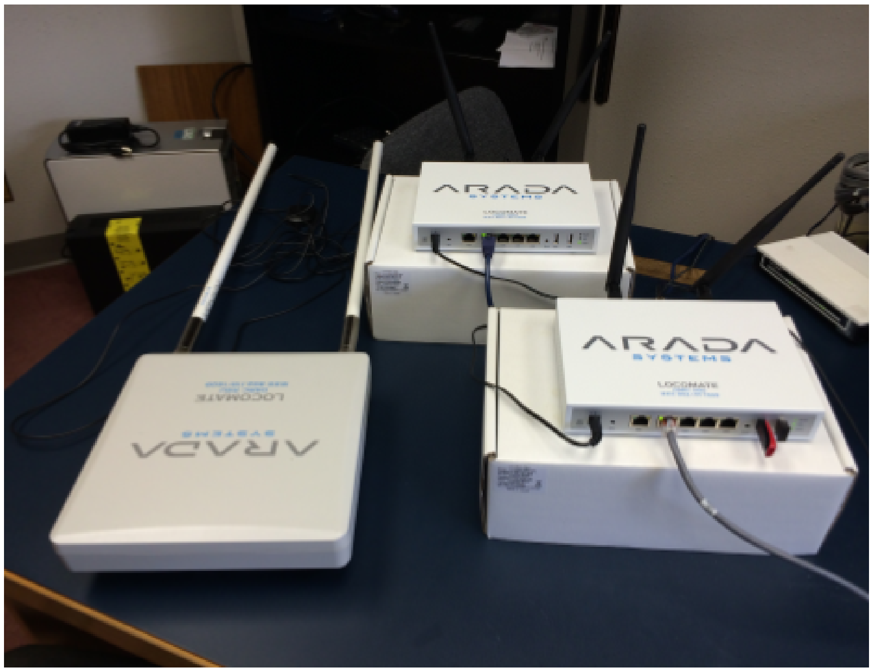


Figure 2.13: Arada Locomate OBU and RSU

One possible attack in VANET is called Sybil Attack. Here a malicious node sends BSMs referring to different non-existing nodes at different physical locations. Each of these spoofed nodes may be difficult to detect, as they may even use stolen or mobile OBUs located in the attackers vehicle. Sybil Attack can affect traffic in different ways. For example, letting other vehicles think the road is congested to cause them to change routes. A more dangerous scenarios would project a Sybil vehicle in the middle of dense traffic, reporting fake events, like the hard braking event, potentially causing rear-end collisions [11, 12].

GPS spoofing is another known attack in VANET. A special device transmits false GPS information by overpowering the original GPS signal. As a result, the victim's OBU can be injected with falsified information, which in turn may cause safety applications to fail [18, 12].

In this research we consider one of the Denial of Service (DoS) attacks known as jamming attack. Jamming can disturb any wireless communication and since VANET uses IEEE 802.11p, VANET is subjected to such an attack. Moreover, jamming has an affect on the receiver's signal-to-noise ratio (SNR) as stated: "disruption of existing wireless communications by decreasing the signal-to-noise ratio at

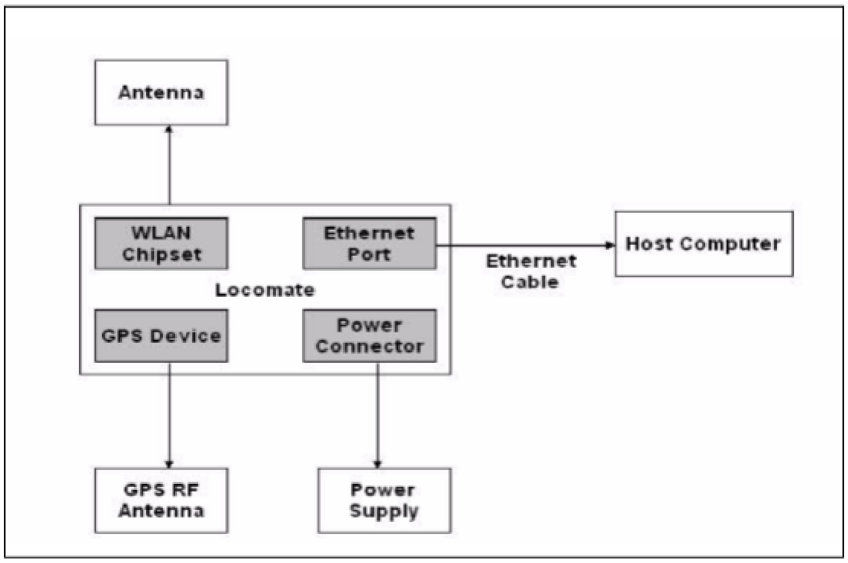


Figure 2.14: Block-diagram-of-Arada-LocoMate-OBU, [49]

Figure 2.15: Arada Locomate OBU and Locomate ME, [10]

receiver sides through the transmission of interfering wireless signals" [13]. As a result, a receiver might not receive a BSM or might receive an outdated BSM. The consequence may be late decisions, not allowing a driver to react to avoid a collision [15, 14]. The technique in channel jamming uses a high power rate to paralyze or interfere with communication among VANET participants and might cause the OBUs at the end to drop packets [17, 14]. There are different jamming models, ranging from simple jamming to intelligent jamming [14, 35]. These jamming models includes different sub-categories of jammers such as a constant jammer, deceptive jammer, random jammer and reactive jammer. In this research we consider a constant Jammer, which does not follow the rule of the MAC protocol by emitting constant random bits to prevent other nodes from accessing the medium, thereby preventing other VANET nodes from transmitting packets.

# CHAPTER 3
# BICYCLE SAFETY APPLICATIONS IN VANET

A very common source of bicycle collisions is the aforementioned Right Hook [3]. In this scenario a vehicle turns right into an adjacent bicyclist. Consider Figure 3.1, which shows the scenario leading to the right hook situation depicted in Figure 3.2. The bicycle is traveling in the right lane, e.g., a bicycle lane. Assume that the truck in the left lane has the intention of turning right. Several areas are of interest. The right hook conflict zone, RHC Zone, is the area where potential right hook collisions may occur. To avoid such collision, a driver needs to be alerted to the potential collision before it is too late to react. Let $T_{react}$ denote a reaction time. The reaction time of a bicyclist is approximately 1 second [28], however the combined perception and brake reaction time is 2.5 seconds [29]. The reaction time of a truck driver, described as the driver's time initial steering, which is the duration of time until the driver starts steering to avoid an collision, is about 1.7 seconds. This was based on field tests and simulation results in [30]. The bicycle safety application (BSA) needs to alert drivers about a potential collision, based on BSM information acquired in the decision area, before it is too late to react. Thus, the alert has to be given before $t_{react}$.

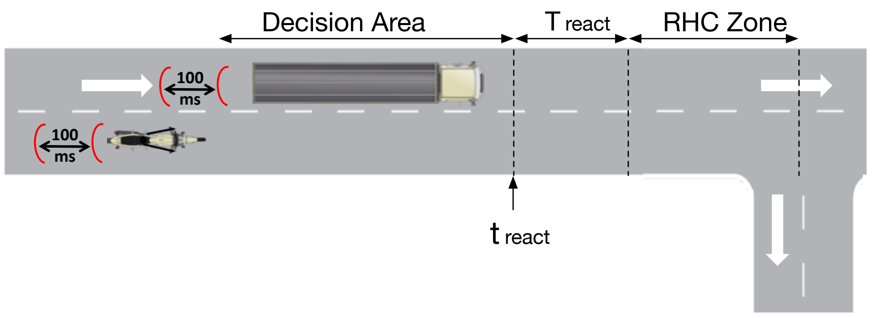


FIGURE 3.1: Scenario leading up to potential Right Hook Conflict

In the RHC Zone timing is critical, as distances between the bicycle and the truck may be short. In fact, the bicycle and truck may be next to each other, as shown in Figure 3.2. This will leave little time for both drivers to react.

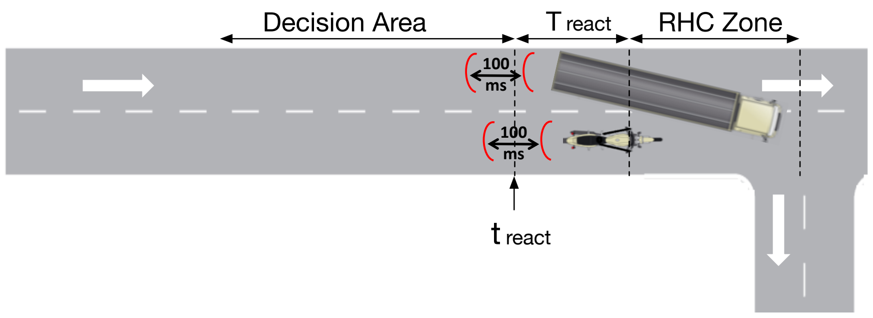


Figure 3.2: Right Hook Conflict

In the discussion of the safety applications in Subsection 2.1.3, it was clear which vehicle was the HV and which was the RV. For example, in the EEBL safety application it was the vehicle following the hard-braking vehicle that needed to be alerted, in order to avoid a potential rear-end collision with the hard-braking vehicle. In the context of the BSA, both the cyclist and the truck driver have the potential to react in order to avoid collision. We will describe the BSA from the viewpoint of the truck for right-hand driving roads. For left-hand driving roads the logic has to be reversed due to the mirrored geometry. Since vehicle behavior in the RHC zone, and especially time is very critical, tracking a right turn of the truck based on GPS information alone may be too slow. It takes multiple BSM's to be able to detect the turn based on the differences between consecutive BSMs to detect a right turn trajectory. Furthermore, the accuracy of GPS coordinates depends heavily on the number of satellites locked with the OBU. Rather than considering differences in GPS coordinates, it may be better to use the steering wheel angle to detect the turn. This information can be provided from the vehicle via its CAN bus, to be used in the BSM's SteeringwheelAngle field.

## 3.1 Safety Applications Reliability

In the general field of dependability, reliability $R(t)$ is the probability that the system functions up to specifications during the entire time interval $[0, t]$ [26]. In the context of safety applications, this means that at least one BSM indicating an event generated from the RV is received by the HV, to be able to generate an alert before it is too late to react.

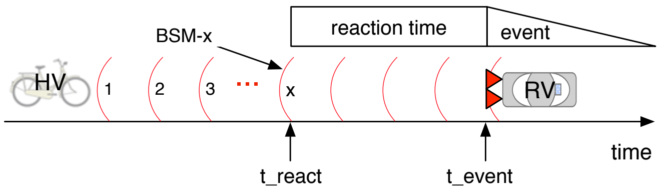


FIGURE 3.3: BSM Propagation and Timing

Consider Figure 3.3, where the RV broadcasts BSMs every 100ms indicating an event starting at $t_{event}$. The HV needs to receive at least one BSM to warn the driver timely, before $t_{react}$. The safety application fails only if no BSM is received by the HV in time. This is the case when all $x$ BSMs, i.e., BSM$_1$, ..., BSM$_x$, were lost. Receiving a BSM at or after $t_{react}$ will not help, as the driver will not have enough time to react to the event.

Let $Q(t) = 1 - R(t)$ be the safety application unreliability. Under the assumption that BSM packet delivery is independent of that of another BSM, the probability that all $x$ messages are lost is

$$Q(t) = \prod_{i=1}^{x} Q_i(t_i) \tag{3.1}$$

where $Q_i(t_i)$ is the probability that BSM$_i$ was not received by the HV, and $t_i$ is the time it should have been received. In [27] $Q_i$ was computed based on packet error rates and packet delivery ratio.

## 3.2 BSA DETECTION MECHANISM

The positions of participating nodes in VANET are determined by GPS coordinates, broadcast in the BSMs. The distance between two vehicles is therefore determined by the relative distance of two sets of coordinates. Let $Lat(B)$, $Long(B)$, $Lat(T)$, and $Long(T)$ be the geographical coordinates for the bicycle and truck respectively. The differences in longitudes and latitude between the two are denoted by $\Delta_{Long(TB)}$ and $\Delta_{Lat(TB)}$. Multiple methods for determining distances between coordinates have been used, e.g., Law of Cosine, the Polar Coordinate Flat-Earth formula, and the haversine formula. Since the bicycle (RV) and the truck (HV) may be very close, a method capable of calculating accurately even for small distances is desirable. An accuracy comparison of the aforementioned methods is shown in Figure 3.4. For a given angular differences from the GPS data, the corresponding distances in meters are calculated. As can be seen, haversine has the most accurate results, especially for short distances. The computational error in very small angular differences was also described in [31], where the authors suggested to use haversine for such situations.

The differences are in the order of decimeters in the worst case. It is this accuracy that was experienced in the experiments described in Section 4.1 Figure 4.6 below.

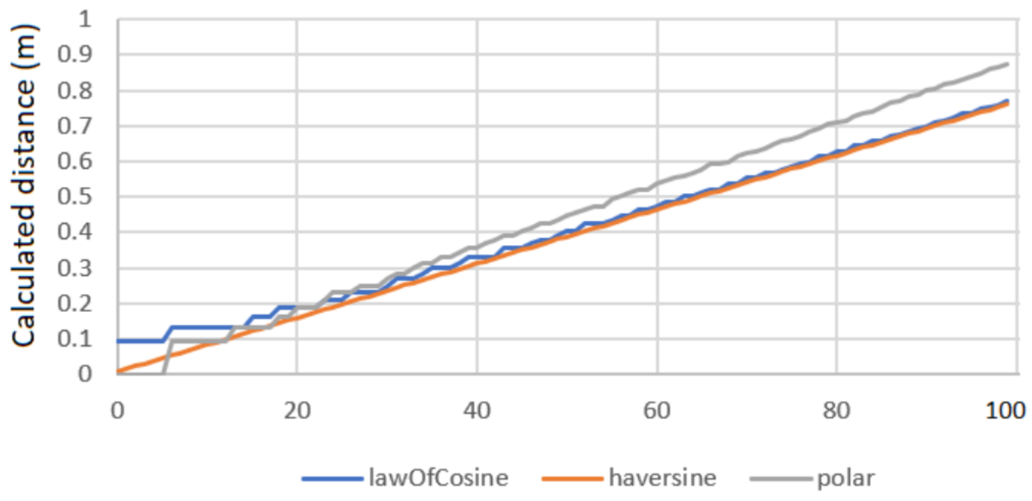


FIGURE 3.4: Calculated results for different methods The x-axis is the distance in multiples of degree 0.000001, which corresponds to 1.11cm.

Since the calculations are using polar coordinates and the coordinate points from the OBUs are in geographical degree form, one needs to convert the coordinates from degree to radian.

The Haversine Formula in Equation 5.2 [32] is used to calculate the distance $d_{TB}$ between the truck and the bicycle as

$$d_{TB} = 2r_{earth} \sin^{-1}\left\{\sin^2\left(\frac{\Delta_{Long(TB)}}{2}\right) + cos(Lat(B))\, cos(Lat(T))\, \sin^2\left(\frac{\Delta_{Lat(TB)}}{2}\right)\right\}^{1/2} \tag{3.2}$$

where $r_{earth}$ is the earth's radius in meters. To find the bicycle's stopping distance $S$ [in meters] under consideration of the combined perception and brake reaction time, the Minimum Stopping Sight Distance Equation from [33] is used:

$$S = \left( \frac{V^2}{254(f \pm G)} + \frac{V}{1.4} \right) \tag{3.3}$$

where $V$ is the velocity [in km/h], $f$ is the coefficient of friction (which is 0.32 for dry condition [33]), and 1.4 is the distance of the bicyclist's eye above the pavement. $G$ is the grade.

Later in Section 4.0.1 and 5.2, the distance $d_{TB}$ from Equation 3.2 is compared with stopping distance $S$ from Equation 3.3. Only if $S < d_{TB}$ is not met will the truck driver be alerted. Note: in our implementation we increased the $S$ value by 10% to be on the conservative side.

CHAPTER 4

# On the Reliability of VANET Safety Applications for Bicycles

This chapter focuses on the reliability of bicycle Safety application for the so-called right-hook-conflict, a common collision scenario where a right-turning vehicle causes a crash with an adjacent bicycle. The proposed detection mechanism improves the resilience of the safety application in non-malicious environments. The proposed approach uses the haversine method, which showed an accuracy of sub-meter distance. This was proven by real world field test.

### 4.0.1 *Bicycle Safety Application Algorithm for Truck*

The algorithm of the BSA, as implemented in the truck's OBU, is shown in Figure 4.1. When a BSM is received from a bicycle that has not been seen before, it is registered. Then a time stamp is recorded. To reduce the number of false alerts to the truck driver a mechanism is needed to enable alerts within the BSA only when it is relevant. In our implementation we assume the truck's blinker has to be engaged. At this time the truck starts including a right-blinker-flag indicating the intention to turn in its BSM, e.g., in its optional BSM Part 2. This can be used by the bicycle to start its BSA. Next, the bicycle's coordinates and speed are extracted from the BSM to calculate the distance between the bicycle and the truck, as well as the bicycle's Minimum Stopping Sight Distance $S$. If $S < d_{TB}$ it is safe to the truck to turn. However, if $S \geq d_{TB}$ the truck driver needs to be alerted of a possible collision with the bicycle.

A less effective alternative to using the blinker as a means to indicate that the truck turns could be the steering wheel angle. This should be available in the truck and it is a BSM field. However, timing is much more critical in this option, as it implies that the turn is already in progress. Whether it is useful to include both, blinker and the steering wheel angle, is not the scope of this paper.

The algorithm in Figure 4.1 registers bicycles, but there is no explicit mechanism to unregister them. To avoid keeping track of bicycles that are out of range, we

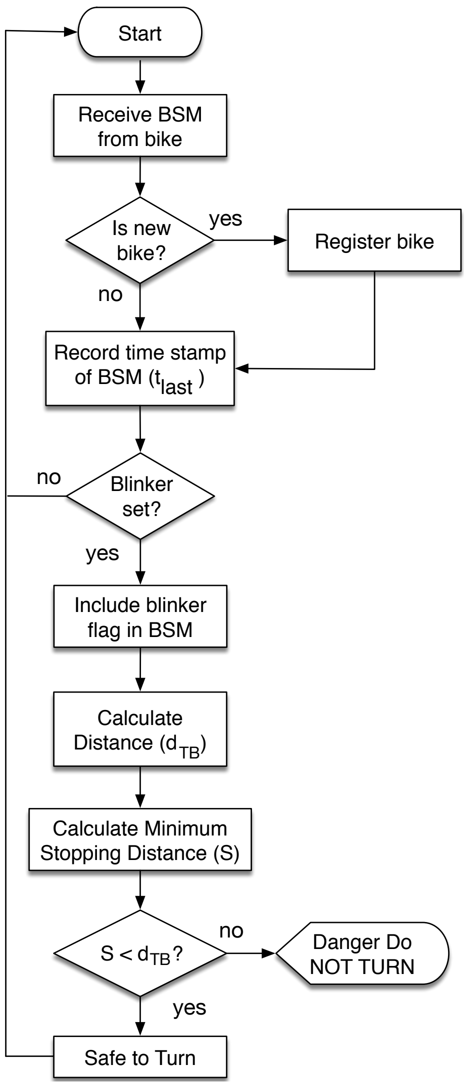

Start
Receive BSM from bike
Is new bike?
yes
Register bike
no
Record time stamp of BSM ($t_{last}$)
no
Blinker set?
yes
Include blinker flag in BSM
Calculate Distance ($d_{TB}$)
Calculate Minimum Stopping Distance (S)
$S < d_{TB}$?
no
Danger Do NOT TURN
yes
Safe to Turn

FIGURE 4.1: BSA Algorithm executed in truck's OBU

execute a periodic cleanup thread. Specifically, the recorded time stamp $T_{last}$ of each registered bicycle is compared to the current time. If the values differ by more than some threshold $T_{max}$, the bicycle is considered no more relevant, and it is unregistered. In our application $T_{max}$ was set to 10 seconds, which for consecutive BSM omissions would account for 100 missed BSMs.

### 4.0.2 *Bicycle Safety Application Algorithm for Bicycle*

The BSA algorithm executing on the OBU of the bicycle is simpler. It is engaged when a BSM with a right-blinker-flag set is received. Now, just as in the truck's BSA, $d_{TB}$ and $S$ are computed and an alert is issued if $S \geq d_{TB}$.

## 4.1 FIELD EXPERIMENTS AND RESULTS

### 4.1.1 *A note on experiments and assumptions about data presented*

The results presented here were not based on simulations, but on real field tests using off-the-shelf equipment. When using a simulator it is straight forward to simulate a large number of scenarios. When conducting real field tests, the efforts associated with every single test for a scenario are rather high. A typical test requires personnel and equipment. In some cases this included a vehicle with a driver and an extra person to operate the vehicle's OBU, in addition to a bicycle driver or anther equipped vehicle, and a person to control a jammer acting as an attacker. The setup time for a single iteration of an experiment could take hours. Moreover, experiments were often affected by delays due to weather conditions, often requiring postponement by days and weeks. Asking volunteers to help repeatedly during all this time was a real test of friendship. This is very different from setting the number of experiments using a variable in a simulator. As a result, the number of scenarios conducted in each experiment was in the order of tens rather than hundreds. Whereas often metrics like average, medium, or standard deviation are of interest, we were looking for worst-case or best-case scenarios. In many cases however the results were simply "typical", meaning cases differed, but no consistent pattern could be extracted. When this occurred the data from representative cases were used. In other cases we could conclude outcomes with only few test scenarios based on similar observations during years of extensive field testing. Thus, given the amount of effort, only a small number of field tests was conducted. For example, certain results were coherent with the outcomes from similar experiments conducted for other research projects.

### 4.1.2 *Experiment setup and results*

The BSA was implemented using an ARADA LocoMate Classic OBU for the vehicle and an ARADA LocoMate ME, which is a battery powered small OBU, mounted on the bicycle. Experiments were conducted in open space and in close proximity, and in-between buildings of the university campus. Both OBUs used the standard transmission rate of 10 BSMs per second and a transmission power of 23 dBm, using Safety Channel CH172. A summary of the field test parameters is given in Table 4.1.

TABLE 4.1: Field Experiment configuration Parameters

| | |
|---|---|
| Truck OBU Model | Arada Systems LocoMate Classic |
| Bicycle OBU Model | Arada Systems LocoMate Mobile ME |
| Test range | Open space road, and building block |
| Speed open space | Varying between 1-7 m/s |
| Speed between building block | Approximately 3 m/s fixed |
| None Fixed and Fixed distance between OBUs | 4 meters |
| BSM generation | 10 BSM/s |
| Channel | Safety Channel 172 |
| Transmitter power | 23 dBm |
| Data rate | 3Mbps |

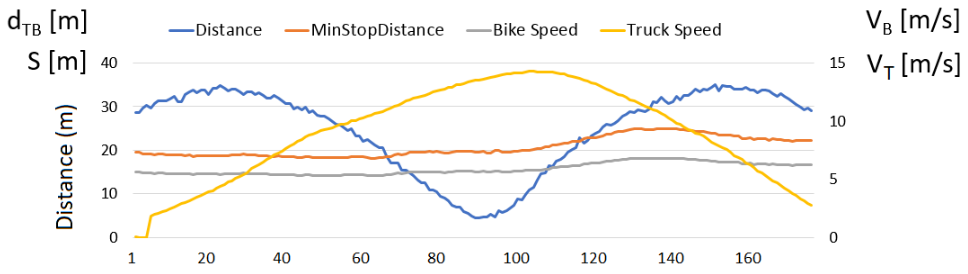


FIGURE 4.2: Experiment distance between vehicle and bicycle. The x-axis indicates BSMs.

Figure 4.2 shows the result of an experiment conducted in an open space where a bicycle driving a head of a vehicle. The mounted OBUs on the vehicle and the bicycle can be seen in Figure 4.3. The experiment shown in Figure 4.2 span over a time period of about 17 seconds. At first, the distance between the vehicle and the bicycle was about 28 meters and as the vehicle's speed increased (yellow plot) from 0 to 6 m/s seen on BSM 35, the distance between the two vehicles decreases. At BSM$_{67}$, the distance was approximately 17 meters where the minimum stopping distance $S$ (orange plot) also increased as the bicycle's speed increased where $S \geq d_{TB}$. Thereby, an alert was issued for the vehicle driver. At BSM$_{119}$, the distance increased over the minimum stopping distance $S$ where the alert went off which indicate a safe even for the vehicle to make a right turn. Next, we wanted to examine the accuracy for the calculated distance given by the two OBUs. Therefor, to find out, the GPS reliability must be examined. Hence, the two unites were installed at a fixed distance as shown in Figure 4.7.

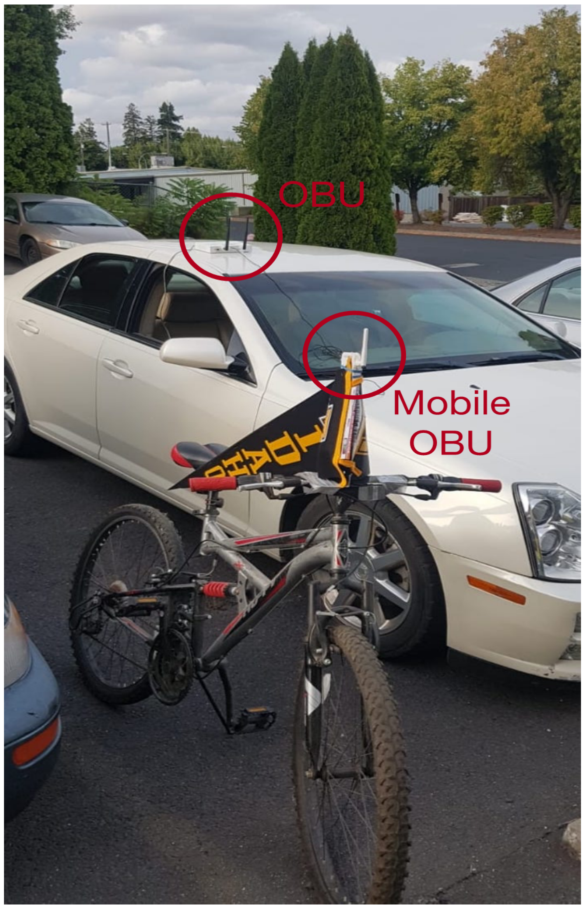


Figure 4.3: OBUs are mounted on the bicycle and vehicle

The GPS antennas of the vehicle and bicycle were spaced at a distance of 4 meters, i.e., the vehicle and the bicycle were driving next to each other at an exact distance of 4m. Figure 4.4 shows the results of a typical experiment conducted in open space. The plots shown in the figure span over a time period of about 9 seconds, during which over ninety BSMs were received by each OBU. The blue plot shows the calculated distance between both antennas, $d_{TB}$, using Equation 5.2.

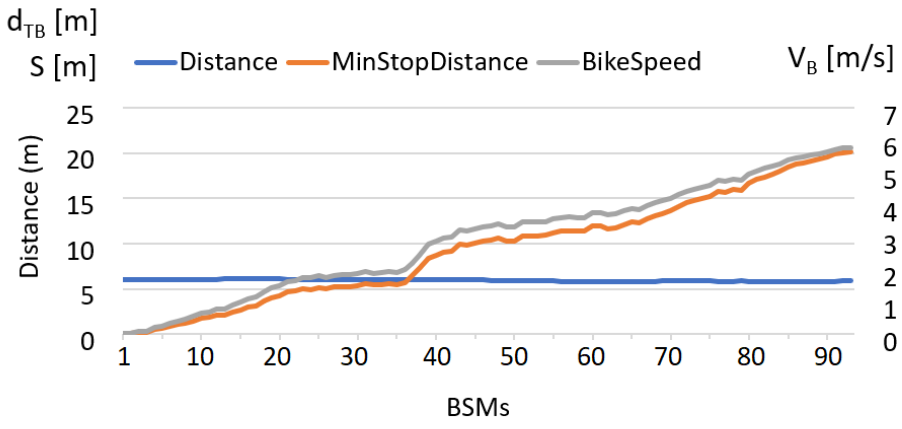


FIGURE 4.4: Experiment with 4m fixed distance between vehicle and bicycle

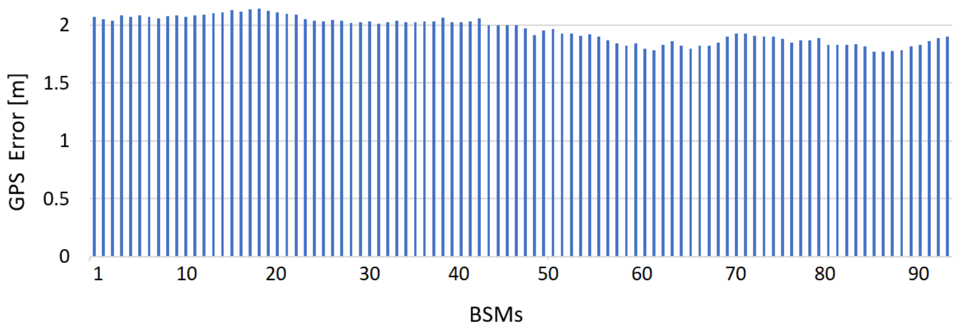


FIGURE 4.5: Error due to GPS inaccuracies

As can be seen, the calculated distances are slightly larger than the actual distance of 4m, as GPS inaccuracies of up to 2m were observed.

The exact distance errors produced due to GPS inaccuracy can be seen in Figure 4.5 for each BSM in Figure 4.4. This error was calculated as $d_{TB}$ minus the actual distance, which was precisely know during the experiment.

As the vehicle and bicycle increased their speeds (grey plot) from 0 to 6 m/s, the minimum stopping distance $S$ (orange plot) also increased. The driver alert is issued when $S \geq d_{TB}$, which occurred starting with the BSM37 in the figure.

Whereas the GPS inaccuracies of the field test described above was rather stable around 2m. It should be noted that this GPS inaccuracies for such given scenario is for the worst representative case for open space experiment. Other field tests

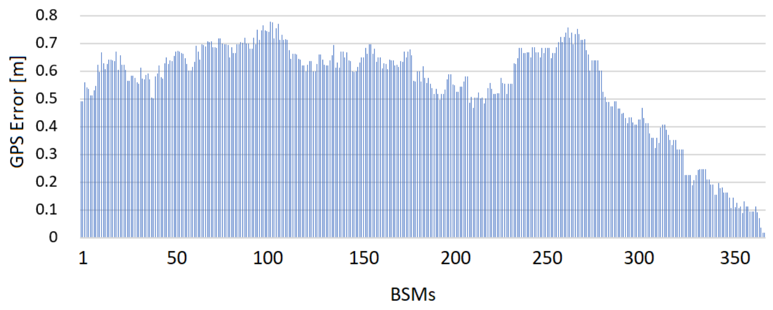


Figure 4.6: Error due to GPS inaccuracies, stationary test

showed much better accuracy. Figure 4.6 is such an example, where mostly sub-meter accuracy was observed when the vehicle and bicycle were stationary.

To test the BSA in extreme situations a test was conducted on the University of Idaho campus location shown in Figure 4.9. As in the previous experiments the distance between the OBU antennas was fixed at 4m as shown in Figure 4.7

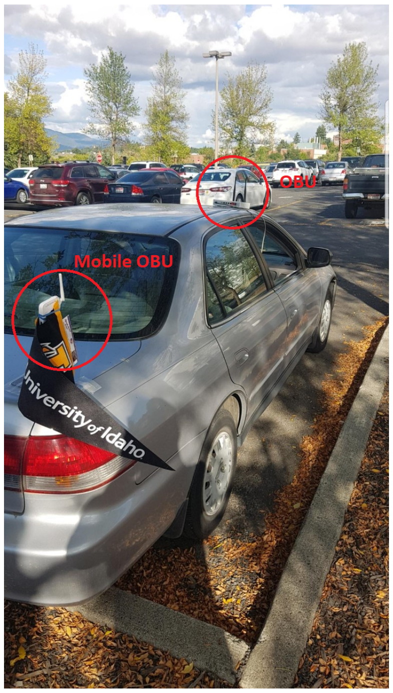


Figure 4.7: OBUs are fixed by 4 meters

The accuracy of $d_{TB}$ from the starting point all around the circular path indicated is given in Figure 4.8. For the first 4s of the southbound test area, i.e., $BSM_1$ through $BSM_{40}$ the accuracy was in the sub-meter range. However, once the GPS antennas entered the constricted area between buildings, before and after turning west into

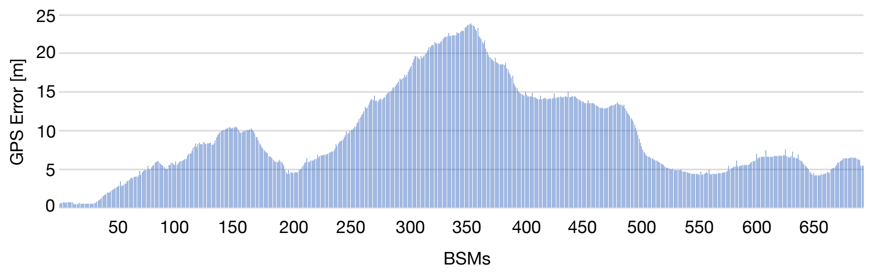


Figure 4.8: Error due to GPS inaccuracies, block drive

the narrow area between buildings, the accuracy was greatly reduced. Even after turning north, on S Line St., errors within 5m were achieved only starting with $BSM_{525}$. We attributed this behavior, experienced in many test runs, to the time required by the OBUs to acquire reliable GPS data once space opened up, e.g., going east-bound on W 6th St., back to the starting point.

In most field tests positive errors were observed, i.e, the calculated distance minus the actual measured distance resulted in a positive number. Only in rare cases was the error negative, which, given our short antenna distance of 4m, comes to no surprise. The most significant impact of the error is that it affects $d_{TB}$, and thus the alert criteria, i.e., when $S \geq d_{TB}$. The errors have no impact on $S$, which is based on parameters such as bicycle speed and reaction time. This means that in areas with low GPS accuracy, e.g., the narrow corridor in Figure 4.9, the probability of *false*

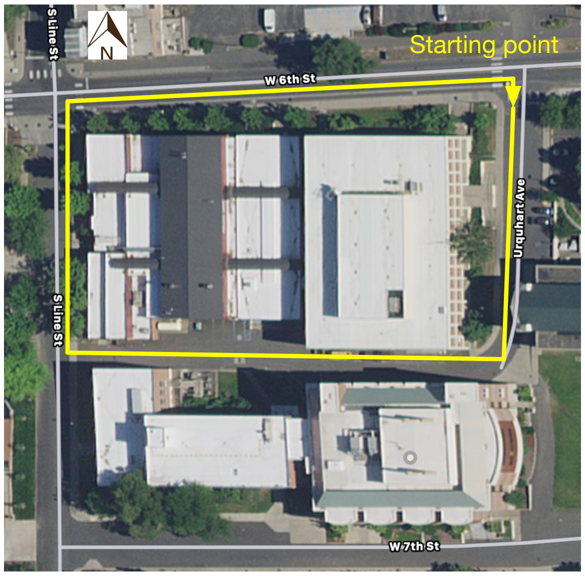


Figure 4.9: UI campus test area

*negatives* is higher. Here the term false negative implies that an alert is not issued, when in fact it should have been. High false negatives should be seen in the context of the physical space where they occur. One may argue that a bicyclist riding in a narrow constricted area is assumed to be more aware of potential right hook conflicts. Thus, a false negative, i.e., a missing alert when in fact the alert criteria would be met given the physical distances, could be of less consequence given the drivers are paying more attentions.

False positives, i.e., the safety application issued an alert when in fact the situation does not require it, may be less of an issue. This is because only in rare cases did we have errors that were negative, i.e., when $d_{TB} - 4m < 0$. In the vast majority of cases $d_{TB} - 4m > 0$. Thus, an error of more than 4m was much more likely than an error of less than 4m.

The experiment shown in Figure 4.9 was for the worst case scenario observed. Another representative experiment shows the best case for the same scenario with the same configuration parameters shown in Figure 4.10. Here the highest GPS error in narrow area between the buildings is about 10 meters unlike the previous experiment shown in Figure 4.8 where the highest GPS error was approximately 24 meters.

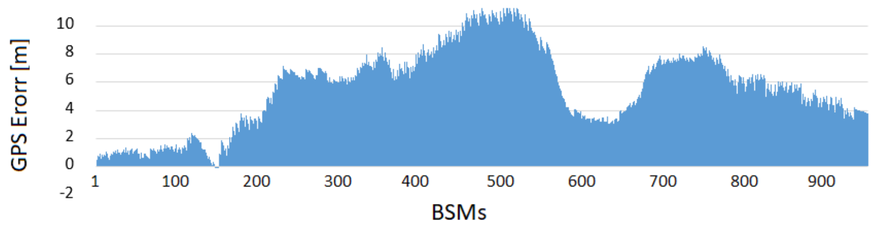


FIGURE 4.10: Error due to GPS inaccuracies, block drive, best case

## 4.2 CONCLUSIONS

A bicycle safety application was introduced that uses Basic Safety Messages of vehicle-to-vehicle communication in connected vehicles. The BSMs provided information like speed and geographic locations, which was then used to alert drivers of possible right hook crash scenarios. The safety application has different algorithms for vehicles and bicycles, however, both issue alerts when the minimum stopping sight distance of the bicycle is greater than or equal to the distance between them. However, since this distance is calculated from the GPS coordinates broadcast in the BSMs, it is affected by GPS inaccuracies. Field tests showed that in the absence of large buildings effective right hook alerts could be issued. Only when the safety application operated in very narrow confined areas was the GPS inaccuracy large enough to greatly reduce its effectiveness.

CHAPTER 5

# Reliability of VANET Bicycle Safety Applications in Malicious Environments

The loss of a BSM can be due to benign reasons like environmental signal degradation, or the so-called shadowing effect. However, we additionally assume malicious act as the source of message omissions. One way to attack wireless communication is by using jamming. Different types of jammers, ranging from constant jammers to intelligent jammers, were address in [35], and a hybrid jammer, immune to Packet-Delivery-Ratio (PDR) detection mechanisms, was shown in [36]. This research does not restrict itself to any specific jammer type, but simply assumes that message loss due to jamming will occur.

## 5.1 BICYCLE SAFETY APPLICATION

The aforementioned Right Hook [3] is shown in Figure 5.1. In this scenario the

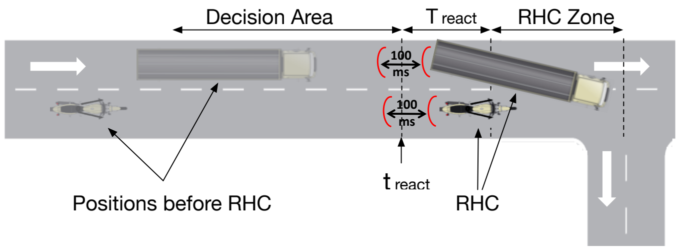


FIGURE 5.1: Typical Right Hook Conflict

bicycle is traveling in the right lane, e.g., a bicycle lane. Assume the truck in the left lane has the intention of turning right. The right hook conflict zone, RHC Zone, is the area where potential right hook collisions may occur. To avoid such collision, a driver needs to receive an alert to the potential collision before time $t_{react}$. Given the reaction time $T_{react}$, any alert after $t_{react}$ comes too late. As summarized in [37], the reaction time of a bicyclist is approximately 1 second [28], the combined perception and brake reaction time is 2.5 seconds [29], and the reaction time of a truck driver, the driver's time to initial steering, is about 1.7 seconds [30]. In the RHC Zone timing is critical due to the fact that distances between the bicycle and the truck may be extremely short, as both may even be right next to each other.

### 5.1.1 *Bicycle Safety Appliaction Prediction Algorithm*

The bicycle safety application uses a prediction algorithm that is capable of mitigating against jamming attacks. The description to follow will be from the viewpoint of the truck.

When BSMs from a bicycle, known to the truck from previous BSMs, are not received in a timely manner, the position of the bicycle needs to be estimated. This projection will be based on Dead Reckoning [38], which calculates the estimated position of the bicycle based on the last known position. This requires information like the speed and the last recorded coordinates, available from the last received BSM, and computed last recorded bearing. The time elapsed since the last received BSM and the bearing are computed locally.

Let $Lat(B)$, $Long(B)$ and $Lat(T)$, $Long(T)$ denote the geographical coordinates for the bicycle and truck respectively, and $\Delta_{Long(TB)}$ $\Delta_{Lat(TB)}$ or $\Delta_{Long(BT)}$ $\Delta_{Lat(BT)}$ their respective differences in longitude and latitude. When it is necessary to indicate whether coordinates are in degree or radian, a $d$ or $r$ will be added in parenthesis, e.g., $Lat(B[d])$ indicates the latitude of a bicycle in degree, and $Lat(T[r])$ latitude of a truck in radian.

Since the calculations are using polar equations and the coordinate points from the OBUs are in geographical degree form, one needs to convert from degree to radian to get the polar coordinates. The Bearing (Azimuth) [39] starts from north clockwise $0° - 360°$. It is denoted by $\beta_{TB[d]}$ and is determined using the truck and bike coordinates as shown in Equation 5.1, which was derived from [44]

$$\beta_{TB[d]} = \tan^{-1}\left\{ \frac{\sin(\Delta_{Long(TB)}) \cos(Lat(B))}{\cos(Lat(T)) \sin(Lat(B)) - \gamma} \right\} \tag{5.1}$$

with $\gamma = \sin(Lat(T)) \cos(Lat(B)) \cos(\Delta_{Long(TB)})$.

Next, the Haversine Formula of [32] is used to calculate the distance, $d_{TB}$, between the truck and the bike:

$$d_{TB} = \\ 2r_{earth}\sin^{-1}\left\{\sin^2\left(\frac{\Delta_{Long(TB)}}{2}\right) + cos(Lat(B))\,cos(Lat(T))\,\sin^2\left(\frac{\Delta_{Lat(TB)}}{2}\right)\right\}^{1/2} \tag{5.2}$$

where $r_{earth}$ is the earth's radius in meters. Let $C_T(t)$ be the clock value of the truck at real time $t$ [in ms]. Furthermore, let $C_T(t_{rec(B)})$ be the recorded time of the truck's clock when the last BSM of the bike was received. Based on the bicycle's velocity $v_B$ from its last BSM, the truck can estimate the bike's distance, $d'_B$, traveled in any direction since the last BSM was recorded. If the speed of the truck $v_T$ is less than or equal to the average approaching right-turn speed, i.e., there is no deceleration, $d'_B$ is calculated using

$$d'_B = v_B\,[C_T(t) - C_T(t_{rec(B)})] \tag{5.3}$$

One needs to find the time the truck will take to reach a speed less than or equal to that of an average truck about to make a right turn. Based on [41] $v_{RT}$ was determined as 10 m/s. We use the maximum truck deceleration, denoted by $a_T^{-1}$, which is $0.8m/s^2$ [42]. The difference in speed between the truck and the average truck's speed on approaching to right-turn, $\Delta v_T$, is $\Delta v_T = v_T - v_{RT}$.

How much will the bicycle have moved by the time the truck will have reached its right-turn-approaching speed? The truck's estimated time to reach this turning speed is $T_{ToReachTurnSpeed} = \Delta v_T / a_T^{-1}$. The time the bicycles is moving unobserved by the truck (due to jamming) is the time that has passed since the truck received the bicycle's last BSM, $C_T(t_{rec(B)})$, plus $T_{ToReachTurnSpeed}$. Thus the bicycle will move for a duration of

$$T_{BikeMoving} = [C_T(t) - C_T(t_{rec(B)})] + \frac{\Delta v_T}{a_T^{-1}} \tag{5.4}$$

and its projected distance covered is

$$d'_B = v_B \, T_{BikeMoving} \tag{5.5}$$

To find the bike's angular distance ratio, $\alpha_B$, under consideration of the earth curvature, $d'_B$ is divided by the earth radius [in km], $\alpha_B = \frac{d'_B}{6371}$. The equations from [44] are used to find the new estimated latitude and longitude of the bicycle:

$$EstLat(B[r]) = \sin^{-1}\left\{ \sin(Lat(T))\cos(\alpha_B) + \cos(Lat(T))\sin(\alpha_B)\ \cos(\beta_{TB}) \right\} \tag{5.6}$$

$$EstLong(B[r]) = Long(T) \, + \, \tan^{-1}\left\{ \frac{\sin(\beta_{TB})\, sine(\alpha_B)\ \cos(Lat(T))}{\cos(\alpha_B) - \sin(Lat(T))\sin(Lat(B))} \right\} \tag{5.7}$$

The latitude and longitude of the truck are calculated analogously, except its time base is $T_{ToReachTurnSpeed}$ rather than $T_{BikeMoving}$. The Minimum Stopping Sight Distance from [33] is used to find the bike's stopping distance as

$$S = \frac{V^2}{254(f \pm G)} + \frac{V}{1.4} \tag{5.8}$$

where $V$ is its velocity [in km/h], $f$ is the coefficient of friction (which is 0.32 for dry condition), 1.4 is the distance of the bicyclist's eye above the pavement, and $G$ is the grade. Note: since a flat road is assumed, $G$ can be neglected. Now, $d_{TB}$ from Equation 5.2 is compared with $S$ from Equation 5.8. A driver alert should be issued if $S \geq d_{TB}$.

### 5.1.2 *Basic BSA Algorithm for Malicious Environments*

The bicycle safety application that implements dead reckoning is shown in Figure 5.2. It overcomes the limitations in the algorithm described in [37], which was only suitable for a benign environment. Now we consider DoS attacks, such as jamming. The shaded area to the right shows the algorithm's behavior if a BSM is received. It is

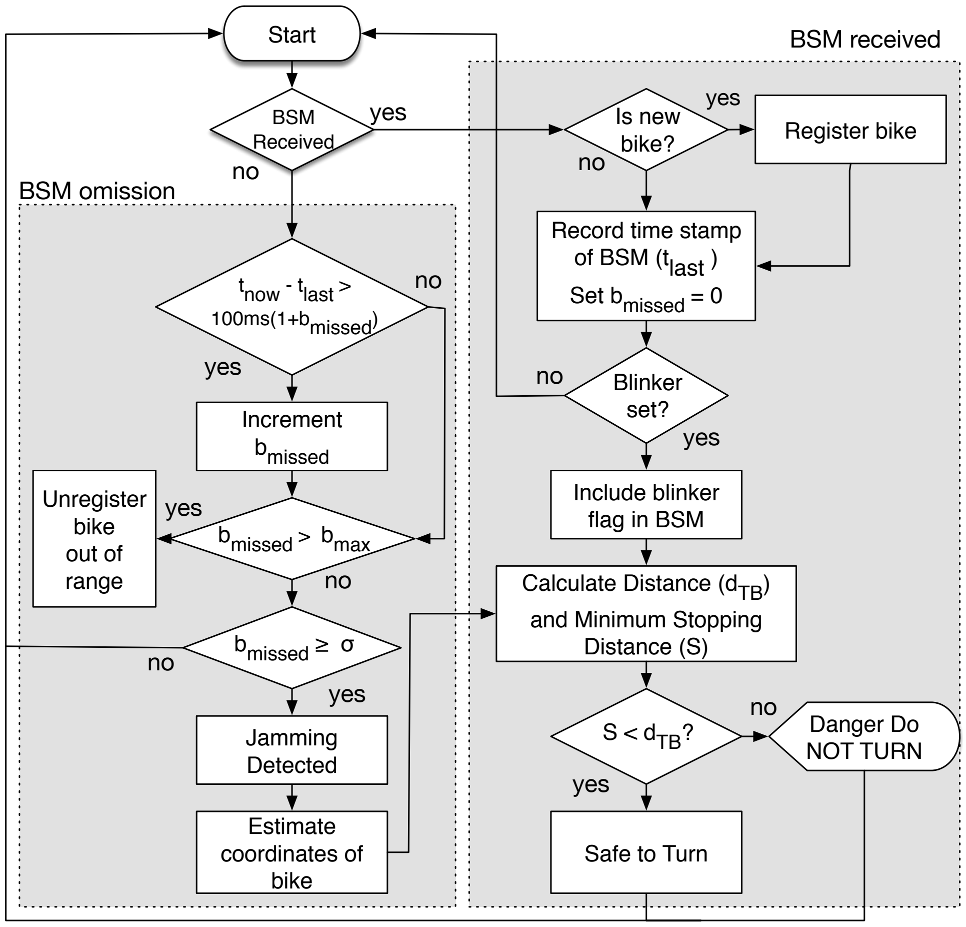


FIGURE 5.2: BSA Flowchart from the viewpoint of one bicycle

similar to the benign-case algorithm in [37]. If a BSM from a new bicycle is receive, this bike is registered. Next the OBU's time of BSM reception, $t_{last}$, is recorded. This time serves as a reference for bicycle BSM omissions, e.g., due to jamming or shadowing. It is used for dead reckoning when messages are not received. If the trucks blinker is set, indicating the intention to turn right, a blinker flag is included in its BSMs, which is used by the bicycle safety application. Next the distance between the two vehicles, $d_{TB}$, and the minimum stopping distance $S$ are calculated. If $S$ is less than $d_{TB}$, then it is safe to turn. Otherwise an alert needs to be issued.

The case when no BSM was received is shown in the left area of Figure 5.2. An omission is detected if no BSM is received within the BSM inter-arrival time of approximately 100ms. Omission counter $b_{missed}$ keeps track of the number of consecutively missed BSMs. A predetermined $b_{max}$ specifies the threshold of omissions before the bicycle should be unregistered. This avoids tracking bicycles that are no longer relevant, e.g., they are out of range or the units have been shut down. When a BSM is received from a bicycle, the counter $b_{missed}$ is reset.

In [43] it was argued that BSM's older than 500ms, 5 BSM intervals, should be considered outdated. We assume that if the number of missed BSM's has not reached this threshold $\sigma$, i.e., if $b_{missed} < \sigma = 5$, then the omissions do not pose immediate threats. Otherwise, we assume a DoS is ongoing. Given the knowledge of the bicycle's last position and velocity, as well as the time that has expired since then, the bicycle's coordinates can be estimated as shown in Subsection 5.1.1. This initiates the transition to the part of the algorithm that determines if the bicycle's position could pose a danger in the RHC-zone, i.e., if $S \geq d_{TB}$, in which case an alert should be issued.

### 5.1.3 *Experimental Results*

The algorithm of Figure 5.2 was implemented using an ARADA LocoMate Classic OBU for the truck, and an ARADA LocoMate ME, a battery powered small OBU mounted on the bicycle. Experiments were conducted using a data rate of 3Mbps, 23 dBm transmitter power, and 100ms BSM spacing, in open space and close proximity of OBUs. A summary of the field test parameters is given in Table 5.1.

TABLE 5.1: Field Experiment configuration Parameters

| | |
|---|---|
| Truck OBU Model | Arada Systems LocoMate Classic |
| Bicycle OBU Model | Arada Systems LocoMate Mobile ME |
| Test range | Open space road |
| Speed open space | 7 m/s |
| BSM generation | 10 BSM/s |
| Channel | Safety Channel 172 |
| Transmitter power | 23 dBm |
| Data rate | 3Mbps |

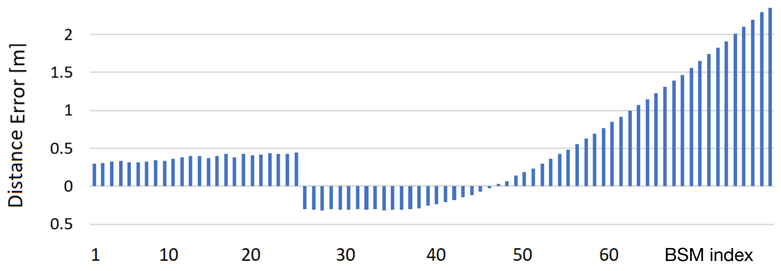


FIGURE 5.3: GPS error with 4m fixed distance between vehicle and bicycle. Truck speed is less than turning speed.

Many experiments were conducted with the truck's speed approaching the right turn less and more than the turning speed of $v_{RT} = 10$ m/s from [41]. Due to space limitations we only present one typical experiment conducted in open space, as per discussion in Subsection 4.1.1. Figure 5.3 shows GPS errors, which is the calculated distance between both antennas, $d_{TB}$, using Equation 5.2 minus the actual known OBU distance. The GPS antennas of the vehicle and bicycle were spaced at a distance of 4 meters, i.e., the vehicle and the bicycle were driving next to each other at that exact distance. The x-axis represents BSM time slots, here referred to as BSM indices. Jamming started at 29, i.e., after 2.9s. The prediction algorithm started when $b_{missed}$ reached $\sigma$. A sub-meter GPS error was observed most of the time. Only several seconds after jamming started did the error slightly grow, as expected due to dead reckoning errors.

Figure 5.4 shows the calculated distance between OBUs, and the minimum stopping distance from Equation 5.8, as it relates to the bicycle speed, which in this case was equal to the truck's speed. The blue graph shows the distance calculated by the algorithm up to jamming, and dead reckoning after its detection. The yellow

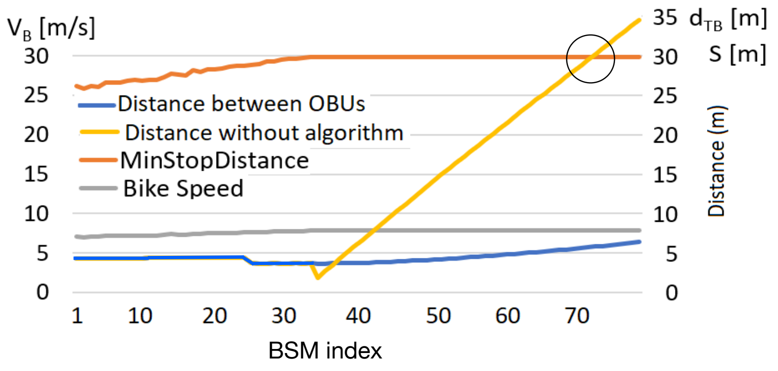


Figure 5.4: Graphs for speed less than turning speed.

line indicates what would happen without the algorithm, in which case the safety application would fail when the calculated OBU distance is falsely interpreted to be greater than the minimum stopping distance. This is the case when the two graphs cross, as marked by the circle. Thus, without the algorithm a jammer could cause the safety application to fail, potentially giving an attacker the power to cause an collision.

## 5.2 CONCLUSIONS

This research presented a bicycle safety application to address right hook conflicts. The underlying algorithm can overcome the impact of BSM omissions, as the result of natural phenomena or malicious act, by applying dead reckoning. Using commercial OBUs, it was demonstrated that jamming attacks could be mitigated by the proposed algorithm, thereby avoiding potential dangerous scenarios where attackers could produce collisions. Field experiments in open space showed that sub-meter GPS accuracy was achieved.

# CHAPTER 6
# APPLICATIONS FOR VANETS WITH DIVERSE TRAFFIC PARTICIPANTS

## 6.1 CONSIDERING PEOPLE WITH DISABILITIES

So far only motorized vehicles and bicycles have been considered. Now we extend the traffic participants to include visually impaired persons and wheelchair users.

The research that has been described up to this point considered a mobility model characterized by relatively fast moving vehicles, e.g., motorized vehicles traveling at the speed limit or fast moving bicycles. However, the two new traffic participants require an investigation of Safety Applications for slow moving traffic participants. For example, wheelchair users or visually impaired people who intend to cross a street will move at much lower speed. Both have one property in common in that they may change their heading over very short distances, e.g., a person may turn on the spot. This is very different from motorized vehicles, which have a much larger turning radius. We will describe scenarios using visually impaired people and wheelchair users. The terms heading, trajectory, and bearing will be used interchangeably to describe the direction of movement.

The motivation behind this research is from the number of incidents reported. According to a NHTSA of July 2015 [46] there were an average of 28 wheelchair users who died in traffic every year from 2007-2013. In a later report [47] it was estimated that in 2017 a pedestrian crash causing death occurred every 88 minutes and according to [48] 75% of the pedestrian crashes happened in the dark. To emphasize the latter, it should be noted that VANET safety applications are especially suitable for such scenarios of low visibility or no line of sight.

Figure 6.1 and Figure 6.2 illustrate two interesting scenarios. Figure 6.1 shows a vehicle and a visually impaired person with trajectories that could result in a collision. Whereas a vehicle is assumed to have a fairly stable trajectory, the visually impaired person could change his/her trajectory on the spot. This opens the ques-

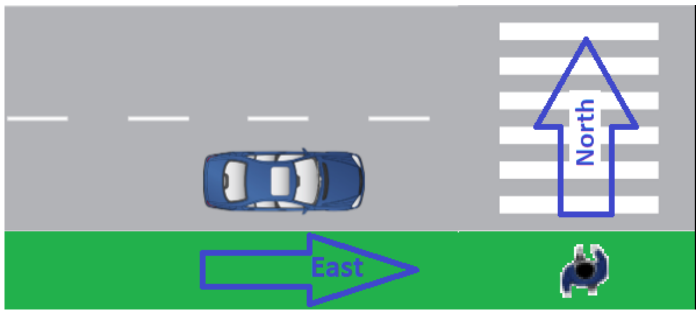


Figure 6.1: Detection of visually impaired person crossing the street at foggy night

tion as to how sensitive the OBUs are to changes in trajectory, especially changes over extreme short distances.

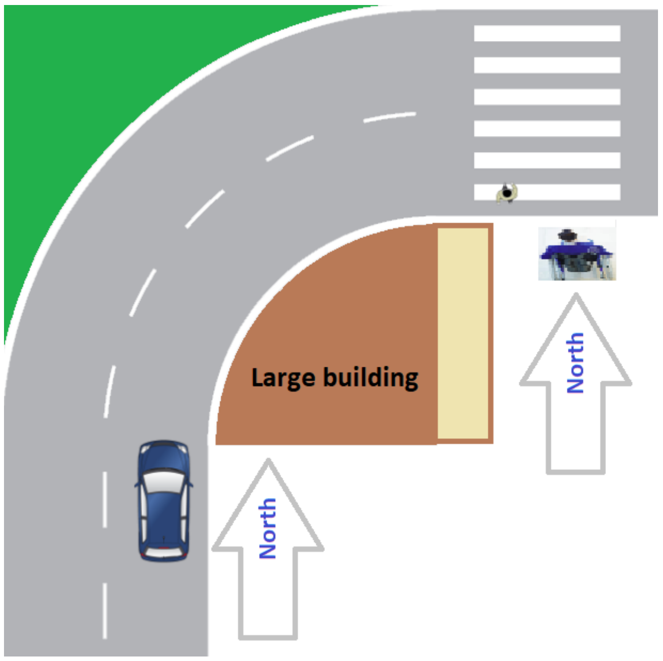


Figure 6.2: Graph for wheelchair and visually impaired person on crossing a street

A scenario that initially does not suggest potential collisions is shown in Figure 6.2. Both, the vehicle and the wheelchair user are moving in the same direction, in this case both are north-bound. Their line of sight is blocked by a large building so that the driver of the vehicle cannot see the wheelchair. The vehicle is approaching a turn, which will result in a change of bearing, ultimately resulting in intersecting trajectories.

### 6.1.1 *Establish that errors will be small for long distances*

In our previous work, we have considered close distance cases, e.g., a bicycle moving within a close distance to a vehicle [37, 45]. In this research we wanted to examine the reliability of the safety application for a large distance to see how reliable are the OBUs to provide parameters for a low mobility application such as a wheelchair. The experiment configuration parameters are shown in Table 6.1, and the result of this experiment is shown in Figure 6.3 where the calculated distance of two stationary OBUs apart from each other by 30 meters. To zoom into the GPS error for the calculated distance and to find the diviation from the actual distance, which is 30 meters fixed, Figure 6.4 shows a relatively small GPS error, which in the worst case was about 2.5 meters.

Table 6.1: Field Experiment configuration Parameters

| | |
|---|---|
| Truck OBU Model | Arada Systems LocoMate Classic |
| Bicycle OBU Model | Arada Systems LocoMate Mobile ME |
| Test range | Residential area (Levick Street) |
| Speed | Stationary |
| BSM generation | 10 BSM/s |
| Channel | Safety Channel 172 |
| Transmitter power | 23 dBm |
| Data rate | 3Mbps |

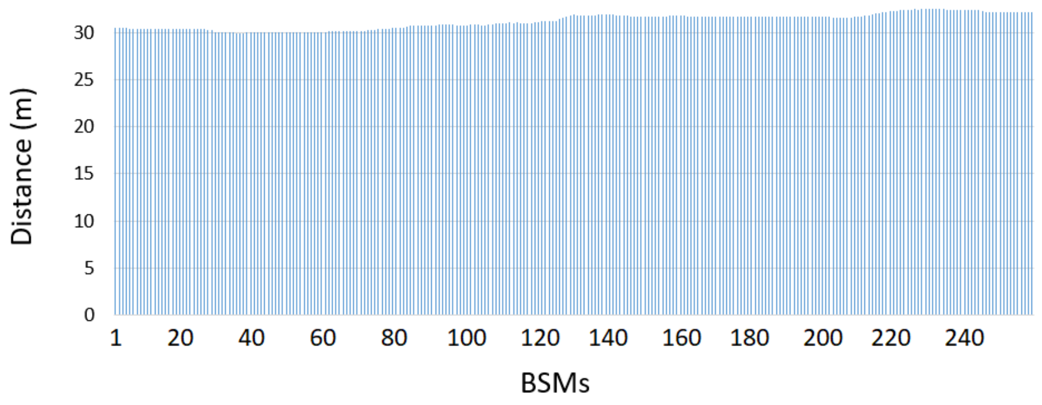


Figure 6.3: Calculated distance when two stationary OBUs were placed 30 meters apart

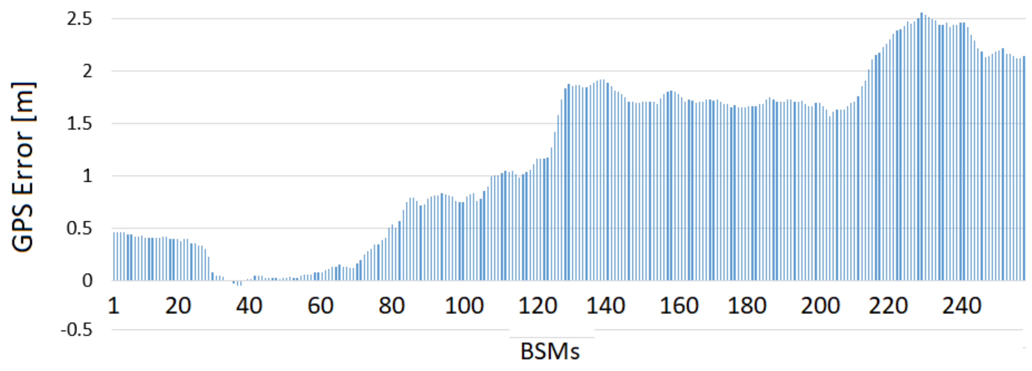


Figure 6.4: Graph for GPS error of two stationary OBUs at 30 meter distance

### 6.1.2 *Investigation of accuracy and timeliness of OBU heading information*

Next, we wanted to examine the OBUs to see if there is a built-in mechanism to detect the trajectory change on the spot. Our findings were that Arada OBUs do not have a mechanism to detect changes in heading on the spot. For example, there is no gyro like in most contemporary smart phones. Hence, an investigation is needed to find out how accurately and timely the OBUs can supply useful values for headings. Several experiments were conducted: First, the OBU was initially calibrated to indicate the west-bond heading with a sudden change, on the spot, to east-bound, e.g. a person turns from west to moves east. Our goal for this experiment was to find how long it takes (at what distance) to detect the correct new heading, which now is east. The result of two typical experiments will be shown next. The configuration parameters used for the experiments are shown in Table 6.2. A sophisticated GPS device, the GeoExplorer 3 from Trimble shown in

Table 6.2: Field Experiment configuration Parameters

| | |
|---|---|
| Truck OBU Model | Arada Systems LocoMate Classic |
| Bicycle OBU Model | Arada Systems LocoMate Mobile ME |
| GPS device | GeoExplorer 3 Trimble |
| Test range | Kibbie Dome area |
| Speed | 1 m/s |
| BSM generation | 10 BSM/s |
| Channel | Safety Channel 172 |
| Transmitter power | 23 dBm |
| Data rate | 3Mbps |

Figure 6.5, was used as a reference to find the actual heading in degree.

FIGURE 6.5: Trimble GPS device GeoExplorer 3

The result of the first experiment is shown in Table 6.3, where each row indicates the heading information of a BSM. In this experiment, which lasted 1 second, hence 10 BSMs given the 100ms spacing, the OBU could not detect the correct heading over 10 received BSMs for a moving distance of 2 feet. Specifically, 10 BSM messages were sent, but all of them had the initial heading, the one before the turn. The results of another experiment is shown in Table 6.4. Here the OBU could detect the change of the heading as reflected in the 9th BSM, but with an error of +6 degrees.

In general, it was observed that it took about one meter for reliable heading information. These are only our observations from using the MobileMe units in a sequence of tests, two of which have been shown above, and we claim no responsibility for general accuracy of this result. It would require further test to establish the

Table 6.3: Observed heading for field experiment over 2 feet

| BSM | Pedestrian OBU GPS Heading | Actual Heading | Deviation |
|---|---|---|---|
| BSM1 | 275 | 90 | +185 |
| BSM2 | 275 | 90 | +185 |
| BSM3 | 275 | 90 | +185 |
| BSM4 | 275 | 90 | +185 |
| BSM5 | 275 | 90 | +185 |
| BSM6 | 275 | 90 | +185 |
| BSM7 | 275 | 90 | +185 |
| BSM8 | 275 | 90 | +185 |
| BSM9 | 275 | 90 | +185 |
| BSM10 | 275 | 90 | +185 |

Table 6.4: Observed heading for field experiment over 3 feet

| BSM | Pedestrian OBU GPS Heading | Actual Heading | Deviation |
|---|---|---|---|
| BSM1 | 274 | 90 | +184 |
| BSM2 | 274 | 90 | +184 |
| BSM3 | 274 | 90 | +184 |
| BSM4 | 274 | 90 | +184 |
| BSM5 | 274 | 90 | +184 |
| BSM6 | 274 | 90 | +184 |
| BSM7 | 274 | 90 | +184 |
| BSM8 | 274 | 90 | +184 |
| BSM9 | 96 | 90 | +6 |
| BSM10 | 96 | 90 | +6 |

accuracy of the distance, as well as how it may differ between OBUs from different vendors.

### 6.1.3 *Disability Safety App*

The safety application for people with disabilities, e.g., visually impaired person or wheelchair users, aims to reduce traffic accidents through alerting vehicle derives of potential crashes. These VANET participants have micro-mobility and they can turn over very short distance. The effectiveness of the proposed algorithm for a safety application for these traffic participants can be describe using the scenario seen in Figure 6.1 and Figure 6.2. We will simply relate to a "participant", P, when we talk about the visually impaired person/wheelchair user.

In Figure 6.2 assuming it is dark or foggy so that both VANET participants have restricted vision. There is therefore a higher risk that this scenario may lead to a potential collision. The SA will examine the heading of both participants to see if there will be a heading conflict within a specified radius indicated by the algorithm. In Figure 6.2 the vehicle is driving north with no line-of-sight, due to the large building, with a wheelchair user that is also moving north. Here, the vehicle will turn, eventually heading east, so that the headings of both participants will intersect potentially leading to a crash.

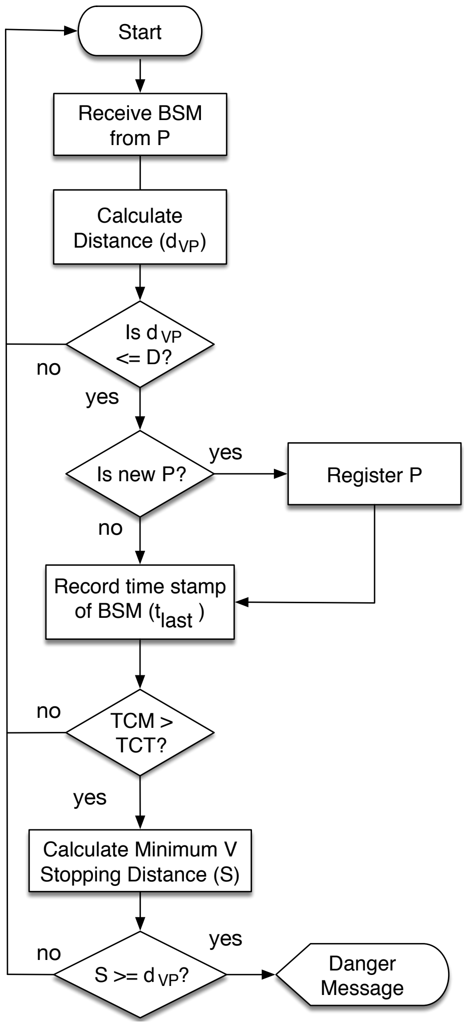


FIGURE 6.6: FlowChart for safety application for people with disabilities

The algorithm for the SA is shown in Figure 6.6. First, after receiving BSM from a participant $P$ calculate the distance $d_{VP}$ between the two VANET participants using question 5.2. It should be noted that only those within the vicinity, of distance $\mathcal{D}$, are of interest. Thus $\mathcal{D}$ is the distance threshold, which was set at 30m in the field tests. Next, participants in the vicinity are registered. The reason for registration is to remember $P$ in case of BSM omissions, e.g., due to OBU malfunction, natural phenomena like shadowing, or jamming. One needs to investigate if the vehicle and $P$ are on potential collision course. For this purpose a Trajectory Criticality Metric (TCM) and a Trajectory Criticality Threshold (TCT) are introduced. The TCM is a function that can have multiple variables and returns a numeric value indicating how likely there could be an intersection of the participant's trajectory in area of concern, e.g., a pedestrian crossing. Examples of TCM variables are:

1) Heading difference, which is the angle between two headings.

2) Speed and rate of change of the angle.

In our implementation option 1) was used. Specifically, the TCM used is the relative trajectory intersection angle of the two headings. The TCT was set to 20 degrees. The area of interest, i.e., of potential collision area, is the pedestrian crossing. This is similar to the RHC-zone in Section 3. If a collision is possible, the minimum stopping distance of the vehicle is calculated using Equation 6.1. If this stopping distance is greater than or equal to $d_{VP}$ an alert is issued.

The registration of $P$ in the algorithm is used to predict the location of $P$ in case of a DoS attack. In our previous work in Section 5, we considered a jamming scenario for the bicycle safety application for malicious environments. When a number of BSMs were lost, e.g., due to jamming, the bicycle position had to be estimated. For slow moving participants this is relatively simple, since the position during a brief jamming period is expected to be rather stable compared to that of a fast moving vehicle. For example, during a 1s jamming period a person is expected to move approximately 1m. This was very different in the case of bicycles, where speeds were much faster, e.g., 5-10 m/s. Hence, the algorithm in Fig 6.6 uses the last recorded BSM information to be used in case of BSM omissions.

## 6.2 ELECTRIC SCOOTER SAFETY APPLICATION

One of the latest addition to the traffic landscape is the electric scooter (E-scooter), such as shown in Figure 6.7. E-scooter share programs are expected to grow fast [50]. An E-scooter has a micro-mobility model that does not yet follow regulated infrastructure rules, and there is no established global standard yet on how to manage such mobility model. Moreover, the mobility model of E-scooters is quite different from other participants. Scooters may drive close to, or between vehicles, in-between two lanes or between moving traffic and parked vehicles. As stated in [51], "We believe that a lack of infrastructure dedicated to E-scooters increased the difficulty in regulating their operation". Therefor, it makes it hard for motorized vehicles to cope. The result is that "news reports of e-scooter crashes and fatalities have started to accumulate" [52]. Hence, we introduce a VANET safety application for E-scooter to increase traffic safety and to mitigate collisions.

FIGURE 6.7: Electric scooters

What is the similarity between the behavior of scooters and the visually impaired or wheelchair users? One defining behavior is the possible radical change in heading over short distances at faster speed than for example pedestrians. But scooter drivers are also known to change between the pedestrian and vehicle infrastructure. They may ride on a side walk, a bicycle lane or the road, and they may change rapidly between them, potentially behaving rather unpredictable and sporadic.

Two typical scenarios involving E-scooters are discussed. Figure 6.8 depicts an E-scooter traveling north, with no line of sight to a vehicle moving toward east. This situation could result in a possible collision.

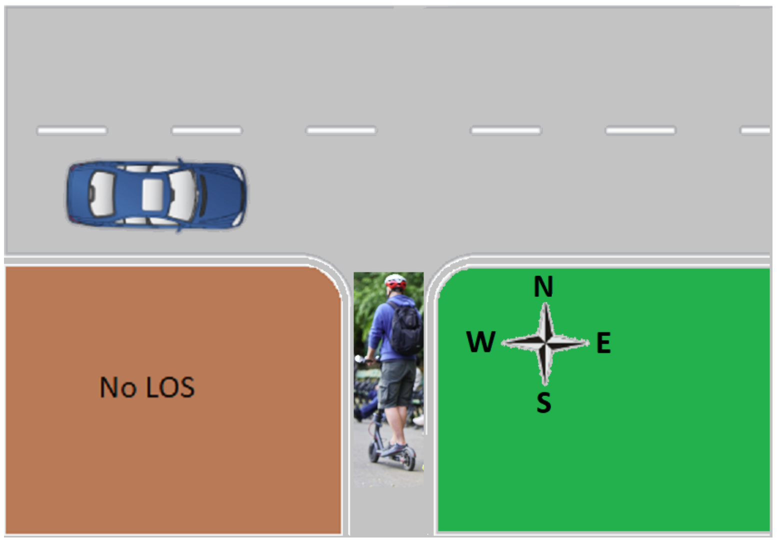


Figure 6.8: North-bound scooter traveling at low speed. Potential collision with eastbound vehicle

Similarly, Figure 6.9 illustrates a command crash scenario where an E-scooter is traveling west-bound, believing that the vehicle driver is aware of him/her approaching from the right side. However, the driver's attention is focusing on the left since the street is one way. This type of scenario can result in a crash since the driver's attention is being focused in the direction of vehicle traffic flow, possibly failing to also focus on the right-hand side when exiting the narrow street.

As referred to above, scooters are likely to travel close to other participants, e.g., during lane splitting. To investigate this rather unique mobility model we examined the uniqueness of E-scooters with GPS accuracy in confined areas. Due to the closeness of the vehicles in the lanes GPS accuracy could be affected by phenomena like signal fading, scattering, reflection, or shadowing.

The first experiment conducted attempted to shed light on the impact of the antenna mounting point on the scooter in a scenario such as shown in Figure 6.10.

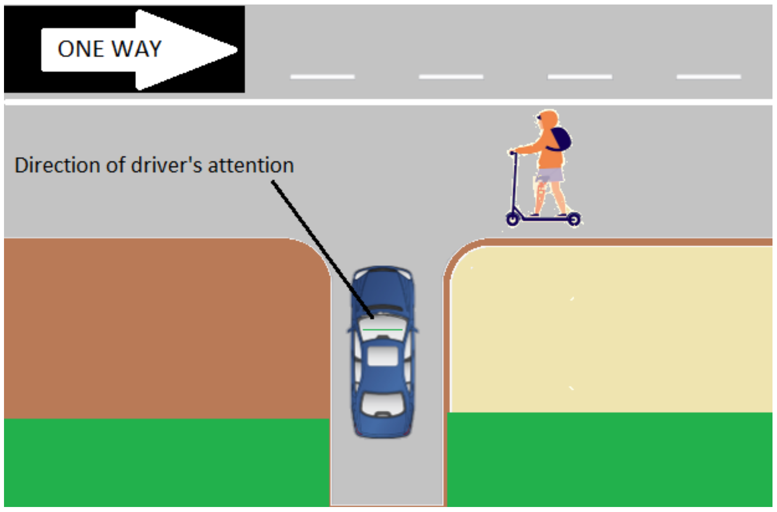


FIGURE 6.9: West-bound scooter traveling in one-way street. Potential collision with north-bound vehicle

This setup emulates a scooter driving between vehicles in adjacent lanes, such as would occur during lane splitting.

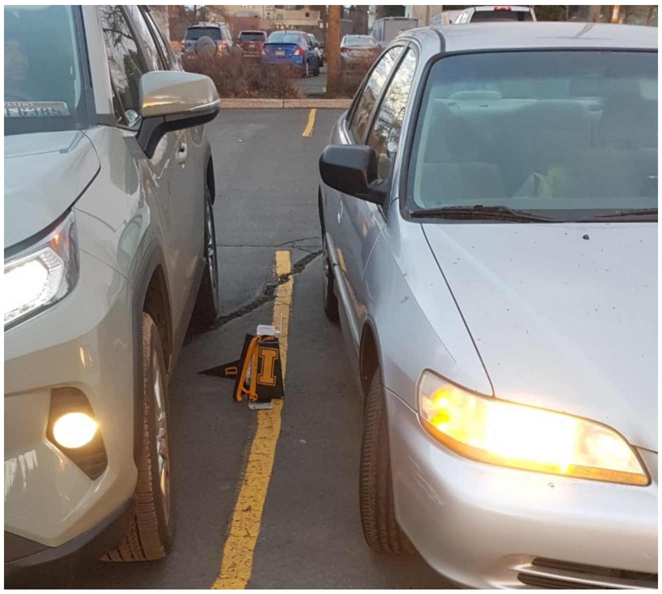


FIGURE 6.10: Low-mounted Locomate ME blocked in between two vehicles

Two experiments were conducted, one with a low mounted GPS antenna at 0.39m height, and another with a higher mounted setup at 1.3m. The complete list of parameters used in the field tests is shown in Table 6.5.

Table 6.5: Lane Splitting Field Experiment configuration parameters

| | |
|---|---|
| Truck OBU Model | Arada Systems LocoMate Classic |
| Bicycle OBU Model | Arada Systems LocoMate Mobile ME |
| Test range | Kibbie Dome area |
| Speed | stationary |
| Height of Locomate ME OBU's antenna from ground | 0.39m and 1.3 m |
| Distance between OBUs' antennas | 0.97m |
| BSM generation | 10 BSM/s |
| Channel | Safety Channel 172 |
| Transmitter power | 23 dBm |
| Data rate | 3Mbps |

### 6.2.1 *Low mounted antenna*

This experiment examined the GPS accuracy when an OBU antenna is mounted low, as shown in Figure 6.10. When talking about low or high mounted OBUs we actually refer to the GPS antenna, as the actual mounting of the OBU hardware is irrelevant. It is the position of the antenna that matters. The height from the ground to the top of the Locomate ME OBU's antenna was 0.39m, and the distance between the two OBU's GPS antennas was 0.97m. The findings of a representative experiment (see Subsection 4.1.1 for explanation about experiments) is shown in Figure 6.11. The calculated distance was about 3 meters, whereas in reality the actual distance between the two OBUs was about 1m.

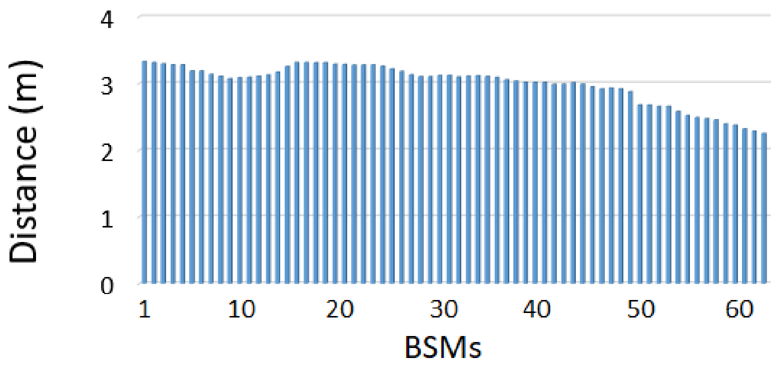


Figure 6.11: Calculated distance of stationary OBUs low mounted 0.97m apart

The GPS error of the same experiment is shown in Figure 6.12. The GPS error is the calculated distance minus the actual distance, as measured during the setup of the experiment.

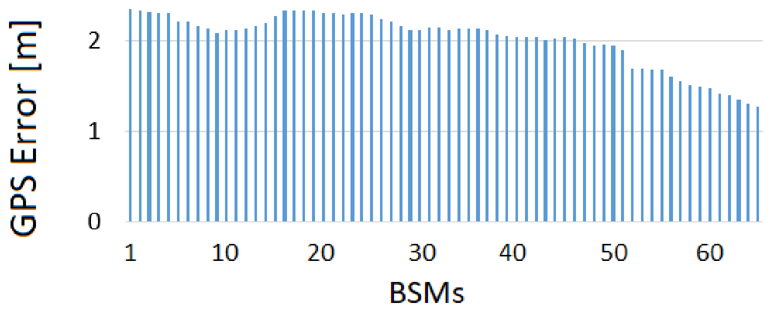


Figure 6.12: GPS error for low mounted stationary OBU 0.97 apart

Another representative result for the calculated distance of a low-mounted OBU is show in Figure 6.14. Here the experiment has a longer run than the previous one to find if there is any difference. The findings result has the approximately error as shown in Figure 6.14.

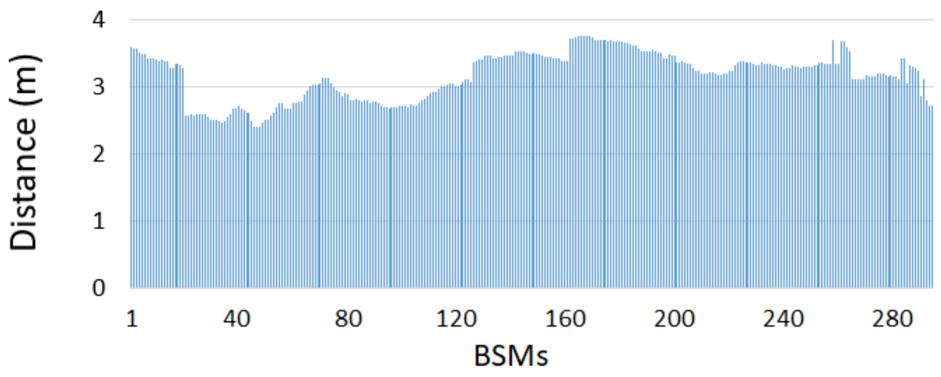


Figure 6.13: Calculated distance of stationary low-mounted OBUs, mounted 0.97m apart for long run

The results shown are representative for a larger number of test. As seen in the two figures, the GPS errors were about approximately 2 meters. This implies if the OBU antenna is mounted-low in an E-scooter, the GPS error will be around 2 meters. This prompted an investigation into the accuracy of high-mounted OBU antennas.

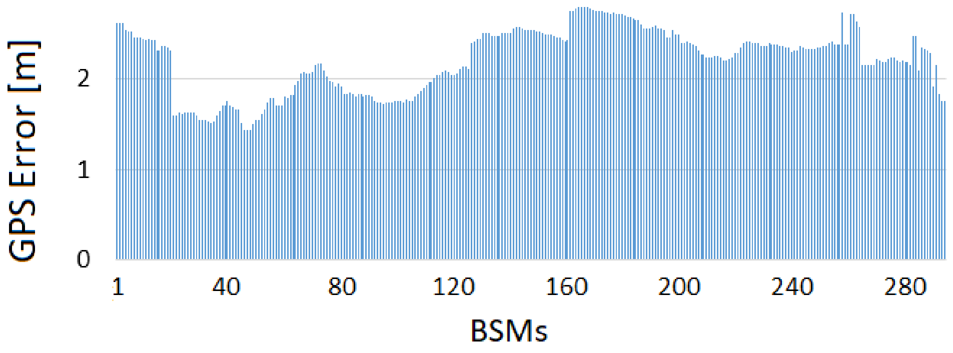


Figure 6.14: GPS error for low-mounted stationary OBU, 0.97 apart, long run

### 6.2.2 *High mounted antenna*

To investigate the GPS accuracy when the OBU antenna is mounted higher an experiment was contacted as shown in Figure 6.15. The height of the antenna from the ground to the top of the Locomate ME OBU's antenna was 1.3m, and the distance between the two OBU's GPS antennas was 0.97m. As before, the experiment was conducted between two parked cars, exaggerating somewhat a car splitting scenario. The results from two representative tests are shown next. The findings of the first

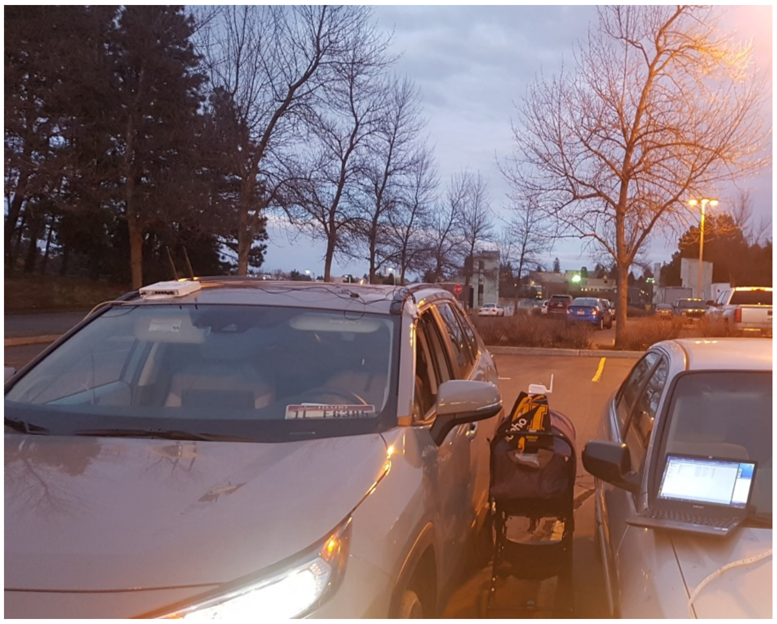


Figure 6.15: Locomate ME blocked in between two vehicles

experiment is shown in Figure 6.16. The experiment showed sub-meter accuracy, and the GPS error Figure 6.17 were mostly negative. Again, the GPS error is the calculated distance minus the actual measured distance.

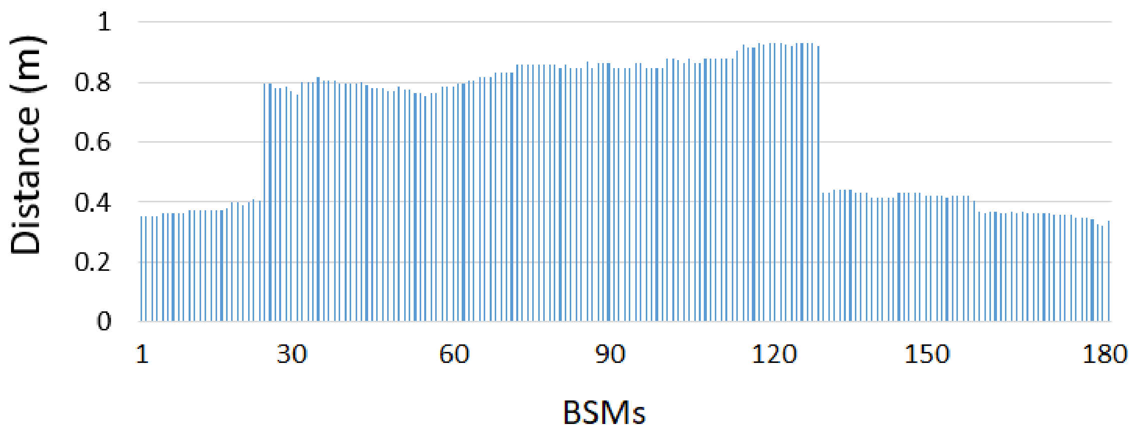


Figure 6.16: Calculated distance of high mounted stationary OBUs 0.97 apart. Height was 1.3m

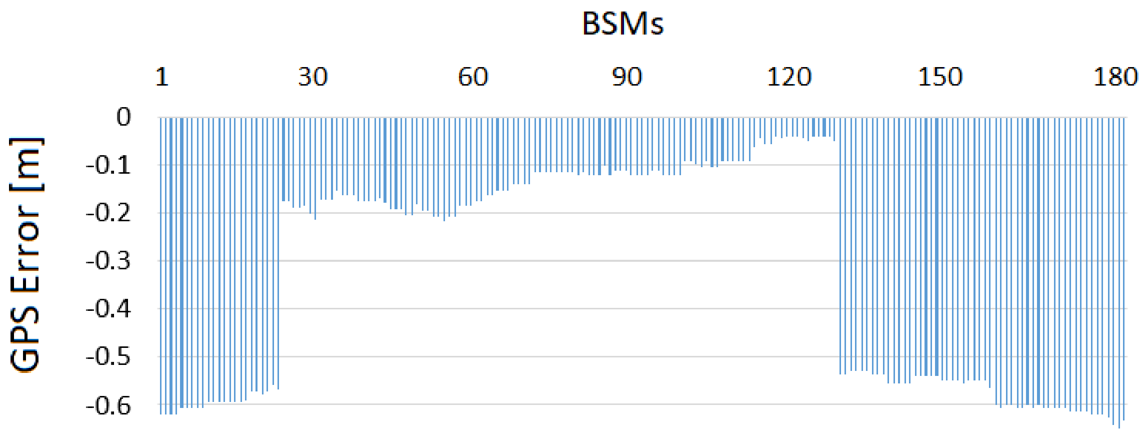


Figure 6.17: GPS error of high mounted stationary OBUs 0.97 apart. Height was 1.3m

A second experiment was conducted for a high-mounted OBU to see if a longer run will show noticeable differences than the previous high-mounted OBU experiment. A representative outcome for this scenario was selected to be shown in Figure 6.18, and the corresponding GPS error is shown in Figure 6.19.

The results for the high-mounted OBU showed sub-meter GPS accuracy. We therefore suggest that the OBU's GPS antenna should to be high-mounted on E-

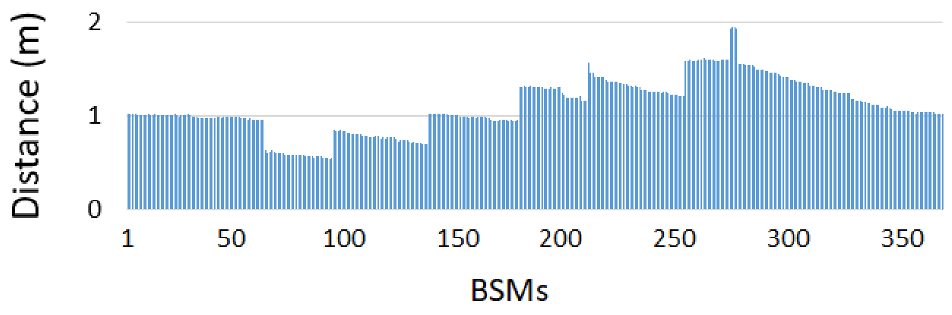


Figure 6.18: Calculated distance of high mounted stationary OBUs 0.97 apart. Height was 1.3m

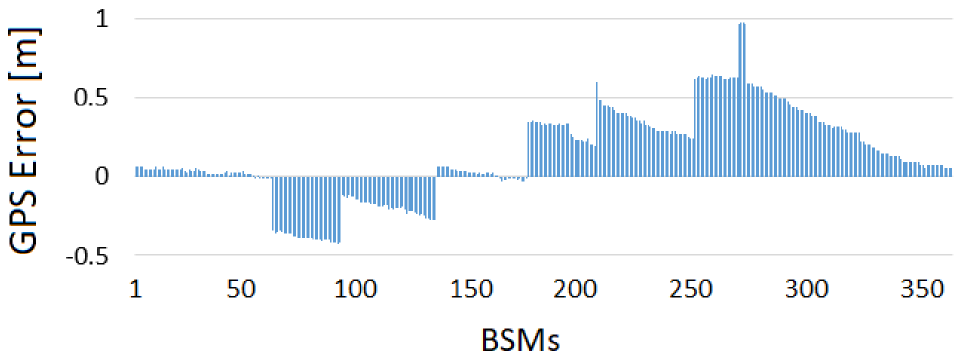


Figure 6.19: GPS error of high mounted stationary OBUs 0.97 apart. Height was 1.3m

scooter to have more reliable GPS accuracy compared with low-mounted GPS antenna.

### 6.2.3 *Car - Scooter crash scenario*

In this section two representative experiments related to E-scooter safety applications are discussed. These two tests were conducted in reference to the scenario shown in Figure 6.8. Specifically, we assume an E-scooter is first traveling west-bound and then suddenly changes its heading over a short distance to the north, resulting in the exact scenario shown in the figure. The experiment was conducted at the University of Idaho in the Kibbie Dome area, with the configuration parameters shown in Table 6.6.

Table 6.6: Field Experiment configuration Parameters

| | |
|---|---|
| Truck OBU Model | Arada Systems LocoMate Classic |
| Bicycle OBU Model | Arada Systems LocoMate Mobile ME |
| Test range | Kibbie Dome area |
| Speed | 6-13 m/s |
| BSM generation | 10 BSM/s |
| Channel | Safety Channel 172 |
| Transmitter power | 23 dBm |
| Data rate | 3Mbps |

Three math equations are used for the safety application. First, the Haversine formula from Equation 3.2 is used to find the distance between the diverse traffic participants. Second, the bearing formula from Equation 5.1 is used to find the heading. Lastly, to find the vehicle's stopping sight distance $V_S$ with a combined brake reaction time, the Vehicle Minimum Stopping Sight Distance Equation6.1 from [34] is used.

$$V_S = \left(0.278Vt + \frac{V^2}{254(f \pm G)}\right) \tag{6.1}$$

where $V$ is the velocity [in km/h], $t$ = reaction time 2.5 in second and $f$ is the coefficient of friction, which is 0.21 for dry condition [34], and $G$ is the grade.

The result of the first experiment where the E-scooter moves at low speed is show in Figure 6.20. In this experiment, the distance between the vehicle and the E-scooter was about 80 meters; the heading for a vehicle was east, and the E-scooter heading was north. As the vehicle's speed increases the distance between the two participant decreases. An alarm is issued when the vehicle Minimum Stopping Distance is greater than the distance between the two vehicles. In the figure this can be seen at BSM 69, where the Truck_MSD (yellow) and Distance (blue) plots intersect.

Next, we wanted to conduct a more realistic experiment where the time is minimal and the speed of the E-scooter is faster. The time for the experiment was about 2 seconds. Initially, the distance was 20 meters, the east-bound vehicle's speed was 8 m/s , and the north-bound scooter's speed was 13 m/s. The result of a typical experiment is show in Figure 6.21. An alarm is issued at BSM 4, as the vehicle's

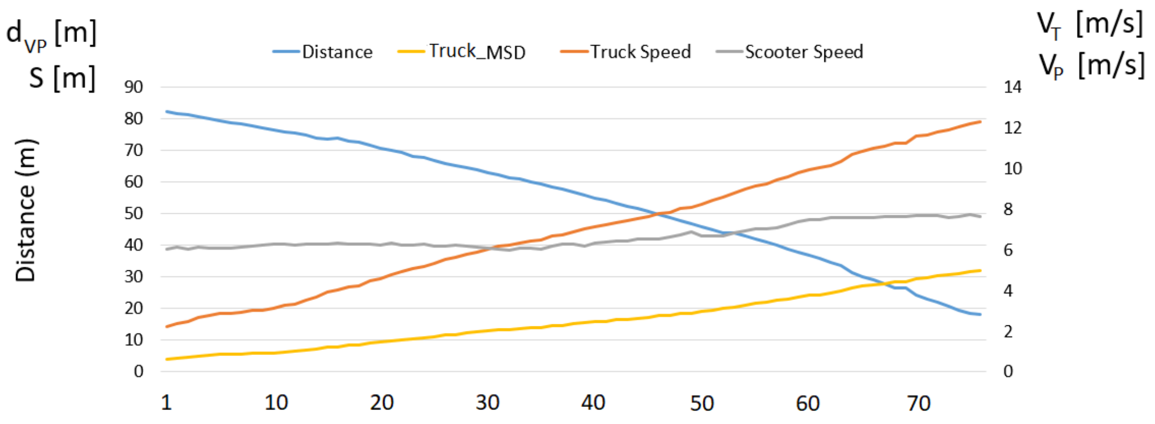


Figure 6.20: Experiment result of North-bound scooter traveling at low speed. Potential collision with eastbound vehicle. X-axis is the index of the BSM

Minimum Stopping Distance was greater than the distance between the motorized vehicle and the scooter.

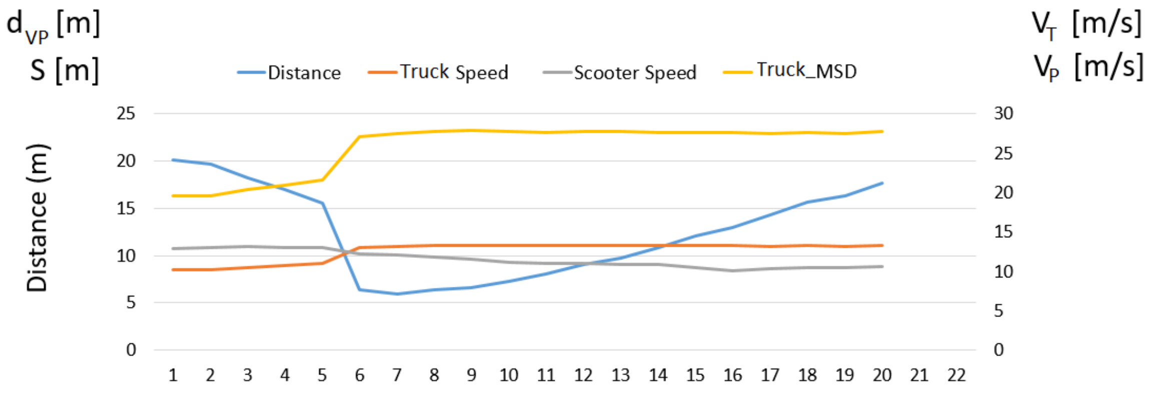


Figure 6.21: Experiment result of North-bound scooter traveling at high speed. Potential collision with eastbound vehicle

## 6.3 Conclusions

In conclusion, the safety application was introduced to mitigate traffic collision for different VANET participants such as visually impaired person, wheelchair users and E-scooters that have a unique mobility model characterized as micro-mobility model. We investigated how reliable are the OBUs to produce accuracy for large range calculated distances. The experiment results showed small GPS errors for large calculated distances from sub-meter of accuracy to at most 2.5 meters. Next, we examined Arada OBUs to see how reliable their heading information is while changing directions over very small distances, e.g., on the spot. The OBUs did not provide a mechanism to determine the heading in such situation. Therefore, we conducted experiments to find out how much distance it will take an OBU to move to produce a reliable heading accuracy. It turns out that the OBUs need at least 3 feet, about 1 meter, to produce a reliable trajectory. This is important as E-scooter behavior could possibly have radical changes in heading over short distances at faster speed, unlike other VANET traffic participant such as a wheelchair or visually impaired person. In addition, E-Scooter users are known to change between different infrastructures such as the pedestrian infrastructure or vehicle infrastructure with a hasty speed. The are also know for lane splitting and to be moving between lanes and parked vehicles, which has an effect on GPS accuracy. Hence, the mounting point of the GPS antennas was investigates, i.e., low-mounted and height-mounted antennas, to see if the height of the GPS antenna on the E-scooter would have an impact on the GPS accuracy. The experiments conducted showed that high mounted antennas had a better accuracy than those mounted low. The proposed algorithm will get triggered when the distance between the vehicle and other VANET participant, e.g., an E-scooter, is less than or equal to 30 meters, in order to eliminate false positive alerts. The algorithm checks if there is a possible trajectory intersection between the two VANET participants. If the calculated vehicle minimum stopping distance is greater than or equal to the calculated distance between the two traffic participants an alert will be issued. All results presented were based on real field test using commercial equipment.

CHAPTER 7

# Conclusions and Future Work

The main focus of this research was on the reliability of VANET safety applications involving bicycles and different VANET participants such as wheelchair users, visually impaired persons, and E-scooters. The proposed solutions included counter measures for communication disruptions in VANET safety applications aiming to reduce road collisions, considering benign and malicious environments. The proposed strategies were tested in the field using commercial communication equipment and the efficiency and reliability issues for the proposed methods were analyzed. The introduced algorithms didn't require any extra hardware or message overhead. Furthermore, none of the solutions required any changes to existing standards.

As a first contribution, a bicycle safety application was introduced that uses Basic Safety Messages emitted by every VANET node. The bicycle safety application extracts information from BSMs such as speed and geographic locations to alert vehicle driver of possible right hook collision scenarios. The bicycle safety application has different algorithms for vehicles and bicycles, yet, both issue alerts when the minimum stopping sight distance of the bicycle is greater than or equal to the distance between them. This calculated distance uses coordinates provided periodically by BSMs, but it can be effected by GPS inaccuracies. During extensive field experiments we could observe that in the absence of large buildings the safety application could issue a reliable right hook alert. Whereas in a constricted areas, e.g., between two large buildings, the safety application will be effected by the GPS inaccuracy, and as a result the effectiveness of the safety application was reduced.

As a second contribution, an algorithm for a bicycle safety application was proposed to address the right hook conflicts that can overcome the impact of BSM omissions as the result of natural phenomena or malicious attack. The algorithm uses dead reckoning during the time spanned by the omissions. The solutions were tested using commercial OBUs in experiments conducted in open space. The results showed that sub-meter GPS accuracy could be achieved, even during jamming attacks. This suggests that the algorithm can mitigate the impact of jamming attacks and avoid possible crash scenarios.

As a third and final contribution, safety applications for VANET participants that have different mobility models were investigated. Specifically, micro-mobility models such as applicable for visually impaired persons, wheelchair users and E-scooters were considered. First, the ability of OBUs to calculate accurate distances for vehicles at larger separation was investigated. The findings showed that distances were accurately computed with only small GPS errors, mostly ranging from sub-meter accuracy to a maximum of 2.5 meters. Second, the OBUs capability of recognizing trajectory changes over very short distances, such as turning on the spot, was investigated. In the absence of gyros, bearing had to be calculated from OBU data, which required an examination of how much travel an OBU required to provide reliable bearing information. Field tests showed that the OBUs needed at least 3 feet to be able to produce an accurate heading. Next, E-scooter were studied as they share certain properties of the micro-mobility model used for pedestrians and wheelchairs. Specifically, E-scooters can make radical changes in bearing over very short distances at relatively high speed compared to that of visually impaired persons or wheelchairs. Furthermore, E-scooters do not necessarily follow a static infrastructure like pedestrians or vehicles, e.g., they may drive between lanes of moving or stationary vehicles, which can affect GPS accuracy. Field tests to determine GPS inaccuracy showed that the mounting point of the antennas of the E-scooters had great impact. The result confirmed that high-mounted GPS antennas produce more reliable GPS accuracy compared to that of low-mounted antennas. The proposed algorithm starts when the calculated distance between VANET nodes is less than or equal to 30m. In that case the algorithm examines if there was a potential conflict in heading, in which case an alert is issued to the driver if the minimum stopping distance is less than or equal to the calculated distance.

## 7.1 FUTURE WORK

The technology used in research work was mainly focusing on DSRC communications, which operates on this dedicated spectrum. Since DSRC is a radio communication it is very suitable for non-line-of-sight scenarios. It will be interesting to see how the safety application technologies presented here can be integrated with sensory like cameras, LIDAR and radar, in autonomous vehicles. These classic sensor technologies will not be able to cope with scenarios such as shown in Figure 6.2. With the synergies of different layers of combined technologies such as DSRC, cameras, and LIDAR, the advantages of line-of-sight, such as the standard sensors in autonomous vehicles, and non-line-of-sight, such as DSRS, can be exploited to increase reliability of safety applications, Lastly, it will be necessary to explore safety applications that use 5G device-to-device (D2D) technology, rather than DSRC. Whereas most fundamental issues will be the same, as both are wireless technologies, the attack vectors are expected to have variations for 5G and DSRC.

# Bibliography

[1] *Traffic Safety Facts: Crash Stats, U.S. Department of Transportation*, National Highway Traffic Safety Administration, DOT HS 812 326, August 2016.

[2] J. B. Kenney, *Dedicated Short-Range Communications (DSRC) Standards in the United States*, Proceedings of the IEEE, vol. 99, no. 7, pp. 1162-1182, 2011.

[3] M. Jannat, D. S. Hurwitz, C. Monsere, K. H. Funk II, *The role of driver's situational awareness on right-hook bicycle-motor vehicle crashes*, Safety Science, vol 110, pp 92-101, 2018.

[4] J. Yin, T. Eibatt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, and T. Talty, *Performance evaluation of safety applications over DSRC vehicular ad hoc networks*, in Proc. VANET, Oct. 2004, pp. 1-9.

[5] Yinigmin Tsai, Guodong Zhang, Donald Grieco, Fatih Ozluturk, *Cell Search in 3GPP Long Term Evolution Systems*, IEEE Vehicular Technology Magazine, vol. 2, issue 2, June 2007.

[6] F. Jameel, et. al., *A Survey of Device-to-Device Communications: Research Issues and Challenges*, IEEE Communications Surveys & Tutorials, Vol. 20, No. 3, Third Quarter 2018.

[7] Jerry Daniel J, Lijo Thomas, Senju Thomas Panicker, Shalu R, Jacob T Mathew, Berlin Vince Joe V S *Safety Alert Systems using Dedicated Short Range Communication for on Road Vehicles*, International CET Conference on Control, Communication, and Computing (IC4), Thiruvananthapuram, India, July, 2018.

[8] M. Amadeo, C. Campolo, and A. Molinaro, *Information-centric networking for connected vehicles: A survey and future perspectives*, IEEE Commun. Mag., vol. 54, no. 2, pp. 98–104, Feb. 2016.

[9] *Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems - 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Spec.*, ASTM E2213-03, 2010.

[10] Arada Systems, *www.aradasystems.com*

[11] Aravendra K. Sharma, Sushil K. Saroj, Sanjeev K. Chauhan and Sachin K. Saini, *Sybil attack prevention and detection in vehicular ad hoc network*, IEEE International Conference on Computing, Communication and Automation (ICCCA), 2016.

[12] Mohamed S. Mohamed, P. Dandekhya and Axel Krings, *Beyond Passive Detection of Sybil Attacks in VANET*, The 6th IEEE International Conference on Reliability, Infocom Technologies and Optimization (ICRITO), 2017

[13] K. Grover; A. Lim; Q. Yang, *Jamming and Anti-jamming Techniques in Wireless Networks: A Survey* Int. J. Ad Hoc and Ubiquitous Computing, Vol. 17, No. 4, 2014.

[14] Xu W, Trappe W, Zhang Y, Wood T, *The feasibility of launching and detecting jamming attacks in wireless networks*, In: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2005, pp 46-57.

[15] I. K. Azogu, M. T. Ferreira, J. A. Larcom, and H. Liu, *A new antijamming strategy for VANET metrics-directed security defense*, in Proc. IEEE Global Commun. Conf.,pp. 1344-1349, Atlanta, GA, USA, Dec. 2013.

[16] *Vehicle Safety Communications-Applications (VSC-A) Final Report*, DOT HS 811 492 A. U.S. DoT, NHTSA. September 2011.

[17] I.A. Sumra, I. Ahmad, H. Hasbullah and J. L. bin Ab Manan, *Behavior of attacker and some new possible attacks in Vehiculer Ad hoc Network (VANET)* , 3rd International Congress on Ultra Modern Telecmmunication and Control Systems and Workshops, pp. 1-8, 5-7 Oct, 2011.

[18] Mohammed Saeed Al-kahtani*Survey on Security Attacks in Vehicular Ad hoc Networks (VANETs)*. 6th International Conference on Signal Processing and Communication Systems 12-14 Dec. 2012.

[19] Mostafa M. I. Taha, Yassin M. Y. Hasan, *VANET-DSRC Protocol for Reliable Broadcasting of Life Safety Messages*, IEEE International Symposium on Signal Processing and Information Technology 2007.

[20] A. Buchenscheit, F. Schaub, F. Kargl, and M. Weber, *A VANETbased emergency vehicle warning system*, presented at the Vehicular Networking Conference (VNC), 2009 IEEE, 2009, pp. 1-8.

[21] *Amendment of the Commission's Rules Regarding Dedicated Short-Range Communication Services in the 5.850-5.925 GHz Band (5.9 GHz Band)*, Federal Communications Commission FCC 03-324, 2004.

[22] *Dedicated Short Range Communications (DSRC) Message Set Dictionary. Society of Automotive Engineers* SAE J2735, November 2009.

[23] *802.11-2007 - IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications* IEEE Std 802.11, 12 June 2007.

[24] *IEEE Standard for Information technology–Telecommunications and information exchange between systems–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*, IEEE Std 802.11p, 2010.

[25] M. Noor-A-Rahim, G. G. M. N. Ali, H. Nguyen, and Y. L. Guan, *Performance analysis of IEEE 802.11p safety message broadcast with and without relaying at road intersection* ,IEEE Access, vol. 6, pp. 23786–23799, Apr, 2018.

[26] W. B. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley Publishing Company, New York, 1989.

[27] A. Serageldin, H. Alturkostani and A. Krings, *On the Reliability of DSRC Safety Applications: A Case of Jamming*, in IEEE International Conference on Connected Vehicles and Expo (ICCVE), Las Vegas, NV, pp. 501-506, 2013.

[28] M. Figliozzi, N. Wheeler, and C. M. Monsere, *Methodology for estimating bicyclist acceleration and speed distributions at intersections*, Transportation Research Record:Journal of the Transportation Research Board, vol. 2387, pp. 66-75, 2013.

[29] *Wisconsin Bicycle Facility Design Handbook. Wisconsin Department of Transportation*, Madison, January 2004.

[30] McGhee D. V., Mazzae E. N., Baldwin G.H.S, *Driver Reaction Time in Crash Avoidance Research: Validation of a Driving Simulator Study on a Test Track*. Proc. 14th Triennial Congress of the International Ergonomics Association and the 44th Annual Meeting of the Human Factors and Ergonomics, pp. 320-323, Santa Monica, CA, 2000.

[31] Smith, M. J., Longley, P. A., Goodchild, M. F. *Geospatial analysis: A comprehensive guide to principles, techniques and software tools*, Winchelsea Press 2007.

[32] Chopde, N., Nichat, M. *Landmark Based Shortest Path Detection by Using A* and Haversine Formula*. International Journal of Innovative Research in Computer and Communication Engineering. Vol.1, Issue 2: 298-302. 2013.

[33] *General design factors, Mn/DOT Bikeway Facility Design Manual*, Chapter 5, March 2007, pp.123-136.

[34] R. Layton and K. Dixon, *Stopping sight distance*, Kiewit Center Infrastruct. Transp., Oregon Dept. Transp., Salem, OR, USA, 2012.

[35] K. Pelechrinis, et. al., *Denial of Service Attacks in Wireless Networks: The Case of Jammers* Communications Surveys & Tutorials, IEEE, Vol.13, No.2, pp.245-257, 2$^{nd}$ Quarter 2011.

[36] S. Hussein, et.al.,*A New Hybrid Jammer and its Impact on DSRC Safety Application Reliability*, in proc. 7th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, pp. 1-7, 2016.

[37] Mohammad Baqer, and Axel Krings, *On the Reliability of VANET Safety Applications for Bicycles*, IEEE International Conference on Connected Vehicle and Expo (ICCVE), Graz, Austria, 2019.

[38] Randell C., Djiallis C., Henk L.*Personal Position Measurement Using Dead Reckoning*,Proceedings of the Seventh International Symposium on Wearable Computers, Oct. 2003, pp 166-173, New York, USA, 2003.

[39] Monmonier, M. *Viewing azimuth and map clarity* Annals of the Assocation of American Geographers Vol 68(2): pp 180-195 Jun 1978.

[40] Ben Gardiner, Waseem Ahmad, Travis Cooper, *Collision Avoidance Techniques for unmanned Aerial Vehicles*, Auburn University, National Science Foundation, 08-07-2011.

[41] Schattler K, Hanson T, Maillacheruvu K.*Effectiveness Evaluation of a Modivied Rightturn Land Densign at Intersections*. Civil Engineering Studies,2016;(16013).

[42] A. K. Maurya and P. S. Bokare, *Study of deceleration behavior of different vehicle types*, International Journal for Traffic and Transport Engineering, vol. 2, no. 3, 2012.

[43] X. Ma, X. Yin, and K.S. Trivedi, *On the Reliability of Safety Applications in VANETs*, Invited paper, International Journal of Performability Engineering Special Issue on Dependability of Wireless Systems and Networks, 8(2), March 2012.

[44] Ben Gardiner, Waseem Ahmad, Travis Cooper, Matthew Haveard, James Holt, and Saad Biaz, *Collision Avoidance Techniques for unmanned Aerial Vehicles Technical Report #CSSE11-01*, National Science Foundation, Technical Report, Auburn University, 2011.

[45] Mohammad Baqer, and Axel Krings, *Reliability of VANET Bicycle Safety Applications in Malicious Environments.*, 27th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2019.

[46] *Traffic Safety Fact Sheet: Fatalities of Vehicle Nonoccupants in Wheelchairs Struck by Motor Vehicles* , U.S. Department of Transportation National Highway Traffic Safety Administration, DOT HS 812 185, July 2015.

[47] *Traffic Safety Fact Sheet: 2017 SUMMARY OF MOTOR VEHICLE CRASHES* , U.S. Department of Transportation National Highway Traffic Safety Administration, DOT HS 812 794, September 2019.

[48] *Traffic Safety Fact Sheet: 2017 PEDESTRIANS* , U.S. Department of Transportation National Highway Traffic Safety Administration, DOT HS 812 681, March 2019.

[49] *Arada LocoMate User's Guide* Version 1.23, 2012.

[50] *National Association of City Transportation Officials (NACTO). Bike Share in the US: 2017. New York, NY.*

[51] Junfeng Jiao, and Shunhua Bai, *Understanding the Shared E-scooter Travels in Austin, TX* Urban Information Lab, The University of Texas at Austin, Austin, TX , January, 2020.

[52] Allen, J.-P., and Majmundar, A, *Are electric scooters promoted on social media with safety in mind? A case study on Bird's Instagram*, US National Library of Medicine: NIH, 13, 62-64; doi 10.1016/j.pmedr.2018.11.013, November, 2018.

# APPENDIX A
# Arada LocoMate Classic OBU and Locomate ME Commands

## A.1 Arada Locomate OBU commands for the default built-in application

The Arada OBUs come with a built-in application named *getwbsstxrxencdec* for Locomate ME and Locomate Classic OBU. This application enables both OBUs to transmit and receive BSMs. The *getwbsstxrxencdec* has a configurable feature to provide different BSM information. This configuration can be enabled through different given parameters. An example of these parameters to to run the default application: $getwbsstxrxencdec\ -s\ 172\ -t\ BSM\ -o\ TXRX\ -X\ TXRXLOG\ -r\ 6.0\ -j\ 24\ -d\ 100$

Option *-s* flag indicate a service channel followed by the channel number e.g. 172 used for safety application. Option *-t* flag indicate BSM message. Option *-o* flag indicate transmission mode. Option*TXRX* flag indicate transmission and reception. Option *-X* flage is used for logging BSMs. e.g. TXRXLOG is used to log BSMs for transmission and reception. Option *-r* flag is used for the data rate. Option *-j* flag is used to indicate the transmission power. Option *-d* flag is used indicates packet delay in ms e.g. 100 ms is the default delay. Tables A.1, A.2 and A.3 shows the Arada locomate application parameters and values.

Table A.1: Common options [49]

| **Parameter** | **Description** |
|---|---|
| -m | Mac Address [xx:xx:xx:xx:xx:xx] |
| -s | Service Channel |
| -b | TxPkt Channel |
| -w | Service Type [Provider/User] |

continued . . .

. . . continued

| **Parameter** | **Description** |
|---|---|
| -t | Message Type [BSM/ PVD/ RSA/ ICA/ SPAT/ MAP/ TIM] |
| -e | Security Type [Plain/Sign/Encrypt] |
| -D | Certificate Attach Interval in millisec should be in multiple of packet delay |
| -l | Output log filename, (specify path ending with / for pcap format) |
| -P | Prefex of certificate files) |
| -o | Tx/Rx Options [TXRX/ NOTX/ NORXALL/ NORX/ TXRXUDP/ NOTXRX] |
| -X | Logging Options [TXRXLOG/ TXLOG/ RXLOG/ NOLOG] |
| -g | sign certificate type [certificate/digest_224/digest_256/certificate_chain] |
| -p | BSM Part II Packet interval (n BSM Part I messages) |
| -v | Path history number [2 represents BSM-PH-2, 5 represents BSM-PH-5] |
| - k | Vehicle_Type (value as per DE_VehicleType) |
| -y | psid value (any decimal value) |
| -d | packet delay in millisec |
| -q | User Priority 0/1/2/3/4/5/6/7 |
| -j | txpower in dBm |
| -M | Model Deployment Device ID |
| -T | Temporary ID control (1 = random, 0 = fixed upper two bytes) |
| -S | Safety Supplement (wsmp-s) <0:disable / 1:enable> |
| -L | Vehicle Length in cm |

continued . . .

. . . continued

| **Parameter** | **Description** |
|---|---|
| -W | Vehicle Width in cm |
| -r | data rate 0.0, 3.0, 4.5, 6.0, 9.0, 12.0, 18.0, 24.0, 27.0, 36.0, 48.0,54.0mbps |
| -n | no argument, and selects no gps device available |
| -f | Type xml or csv for logging in XML or CSV format. Type pcaphdr for only pcap header logging and pcap for full packet logging |
| -F | frameType for TIM Packet 0-unknown(default) 1-advisory 2-roadSignage 3-commercialSignage |
| -A | Active Message Status |
| -B | Port Address for RSU receive from UDP Server |
| -R | Repeat rate for WSA frame (Number of WSA per 5 seconds) Repeatrate is included in WSA-Header only if enabled from /proc/wsa_repeatrate_enable |
| -G | Repeat rate for TA frame (Number of TA per 5 seconds) TA is available only if TA channel [-c option] is given |
| -I | IP service Enable 1= enable 0 = disable |
| -O | Timeout for receiving udp data = 10 seconds |

Table A.2: Provider options [49]

| Parameter | Description |
|---|---|
| -z | Service Priority |
| -a | Service Channel Access [1:Alternating, 0:Continuous] |
| -c | Specify Channel Number to Transmit TA |
| -i | TA Channel Interval [1:cch int, 2:sch int] |

Table A.3: User options [49]

| Parameter | Description |
|---|---|
| -u | User Request Type [1:auto, 2:unconditional(not wait for WSA from provider), 3:none] |
| -x | Extended Access <0:alternate /1:continuous> |

## A.2 ARADA CLASSIC LOCOMATE OBU JAMMER COMMAND

Instructions to start the jammer:

*Start_tx*99 ˘*f* 5860 ˘*m* 1 ˘*r* 6000 ˘*p* 18 ˘*c* 0

Option *-f* flag indicates the frequency of the jammer for the saftey channel, CH172. Option *-m* flag indicates the service channel access mode. Option *-r* flag indicates the data rate. Option *-p* flage indicates the transmission power. Option *-c* flage is used to indicates data mode and single carrier.

The command to stop the jammer application is: *stop_tx*99

APPENDIX B

# Receiver Code

## B.1 BSM RECEPTION

Listing B.1: Safety Application Sample code

```
/*
*
* This code is Modified from: Sherif Hussein's Dissertation
    transmitter and receiver code.
* Computer Sceince Department/University of Idaho.
* This code is the Receiving thread for the Diverse Traffic
    Participants
* Safety Application.
* This version is a sample code and not a complete implimintation
    code.
* This version is used to show e.g. the functions
* that are being used for such safetyp application,
* The Diverse Traffic Participants safety application
* relies on the BSM information such as
* speed, longitude and latitude.
* These information are used in this
* Safety application.
*
*/


/* Function declaration */

/*
* The function is used to check if a bicycle
* exists within the safety application record
*/
void IsNewBike();
```

```

/*
* The function is used to check the number of
* missing BSMs from a registered bicycle
*/
void CheckMissingBSM(bool);

/*
* The function is used to print BSM information
* also it checks if the number of missing BSMS
* are greater than a certain threshold
*/
void PrintAndCheckForJamming(int);

/*
* The functuion calculates the Minimum Stopping
* Distance for a bicycle
*/
double MinimumStoppingDistance(double, double);

/*
* The functuion calculates the Minimum Stopping
* Distance for a vehicle
*/
double MinimumStoppingDistance_Vehicle(double, double);

/*
* The function takes longitutes and latitudes
* received by the On-board-unites of a vehicle
* and a bicycle and calculates the distances
* between the two unites in meter
*/
double DistanceTo(double, double, double, double);

/*
* The function talke the longitudes and latitudes
* of a vehicle, bicycle or wheelchair to calculate
* the bearing
*/
double BearingTo(double , double , double , double);

/*
*  The function takes the known longitude, latitude
*  time, velocity and bearing to calculate the new estimated
*  longitude and latitude.
*/
void NewPosition(double , double , int , double , double , int);


/* Safety Application global Variables */

#define NumOfVanetParticipants 2;
int Time_ms = 0;
int mySpeed = 0;
int CyclistTimeToReact = 1;
double DvriverTimeToReact = 1.66;
uint64_t TimeStamp = 0;
double bike_speed = 0;
uint8_t PrintForJamming = false;
double  truckLat = 0;
double  truckLng = 0;
double newLat = 0;
double newLng = 0;
int bike = 0;
int True = 1;
int received = 0;
int To_Switch_Between_Current_Old_BSM = 0;
int i = 0;
int index = 0;




/*
```

```
*  The data structure for the Safety Application
*  to hold related BSM information for bicycle,
*  vehicle, wheelchair and E-scooter.
*/

struct Node {
  uint64_t time;
  double latitude;
  double longitude;
  double latitude1;
  double longitude1;
  double latitude2;
  double longitude2;
  double Truck_latitude1;
  double Truck_longitude1;
  double Truck_latitude2;
  double Truck_longitude2;
  double speed; /* used for a fixed distance when unites are
      mounted on the same vehicle*/
  double Bike_speed;
  double WheelChair_Speed;
  double bearing;
  double bearing_Of_WheelChair;
  double bearing_Of_Truck;
  double E-scooter_speed;
};


struct Node VanetParticipants[NumOfVanetParticipants];

while (1) {


  /* while receiving BSM*/

  if (Received)
  {

    /* Record my speed */
    mySpeed = (double)wsmgps.speed;

    int  templatitude = 0;
    int  templongitude = 0;
    short tempspeed = 0;
    double  speed;


    /* Record the received latitude */
    templatitude = recived_templatitude;

    /* Record the received longitude */
    templongitude = recived_templongitude;

    /* Record the received speed */
    tempspeed = received_tempspeed;

    /*recording a timestamp */
    VanetParticipants[index].time = ((tv.tv_sec + (uint64_t)tv.
       tv_usec / 1000000.0) * 1000);

    /* Record GPS information */
    VanetParticipants[index].latitude = (double)templatitude;
    VanetParticipants[index].longitude = (double)templongitude;
    VanetParticipants[index].speed = (double)wsmgps.speed;

    /* recording the Bike speed */
    VanetParticipants[index].Bike_speed = (double) tempspeed;

    /*recording WheelChair Speed*/
    VanetParticipants[index].WheelChair_Speed = (double)tempspeed;

    /*recording E-scooter Speed*/
    VanetParticipants[index].E-scooter_speed = (double)tempspeed;

    /* Recored the last received time */
    VNTLastPktSec[index] = tv.tv_sec;
    VNTLastPktuSec[index] = tv.tv_usec;


    /*
    * Updating the VNT table
    * for the first packet only
    */

    VNTLastPktSec[Vindex] = tv.tv_sec;
    VNTLastPktuSec[Vindex] = tv.tv_usec;
    VNTPktReceived[Vindex]++;


  }
  else {

    CheckMissingBSM(false);
  }

}
}
//////////////////////////////////

/*
* Fanction definition for IsNewBike
* to check if a bycicle exists in
* our record
*/

int IsNewBike() {

  int i = 0;
  for (i = 0; i < VNTCounter; i++)
  {
    if (VehicleID == VNT[i])
    {

      printf("BIke ID is registered\n");

    }
  }

}

/////////////////////////////////////

/*
*
* CheckMissingBSM Fanction definition
* the function check the numbers of missing
* BSMs for a registered bycicle
*
*/

void CheckMissingBSM(bool isReceived)
{
  int i = 0;

  if (isReceived) {
    for (i = 0; i < VNTCounter; i++)
    {
      PrintAndCheckForJamming(i);
    }
  }

  int ii = 0;
  if (VNTCounter > 0)
  {
    /* Record the current time */

    CurrentPktSec = tv.tv_sec;
    CurrentPktuSec = tv.tv_usec;
    for (ii = 0; ii < VNTCounter; ii++)
    {
      /* Find the time differance between now and the last received
           BSM */

      TimeDiffSec = CurrentPktSec - VNTLastPktSec[ii];
      TimeDiffuSec = CurrentPktuSec - VNTLastPktuSec[ii];


      if ((TimeDiffSec * 1000000 + TimeDiffuSec) >(100000 +
          VNTMissingBSMs[i] * 100000))
      {
        /* Increment the counter for the number of missed BSMs */

        VNTMissingBSMs[ii]++;
        PrintAndCheckForJamming(ii);
        if (VNTMissingBSMs[ii] >= 5)
        {

          printf("<Number of Missing BSM from Bike 0X%08X is %03d>\
             n", VNT[ii], VNTMissingBSMs[ii]);

          PrintForJamming = true;

        }
        if (VNTMissingBSMs[ii] > 80)
        {
          DeleteBicycle(ii);


        }
      }
    }
  }
}


```

```c
/*
* The function is used to print BSM information
* also it checks if the number of missing BSMS
* are greater than a certain threshold to calculate
* the estimated position the function calls other
* functuions the Minimum Stopping Distance function
* for the bicycle, the Minimum Stopping Distance
* function for the vehicle, the DistanceTo function
* to calculate the distance between the two VANET nodes,
* the NewPosition function to find the new estimated
* coordinates and the BearingTo function
*/

void PrintAndCheckForJamming(int vntCounterIndex) {

  /* Get the GPS coordinates of myself (on-board-unit OBU) */

  truckLat = wsmgps.latitude, truckLng = wsmgps.longitude;

  int i = vntCounterIndex;
  printf("\nTimeStamp: seconds = %llu, microseconds = %06d> \n", (
     uint64_t)tv.tv_sec, (uint32_t)tv.tv_usec);
  TimeStamp = ((tv.tv_sec + (uint64_t)tv.tv_usec / 1000000.0) *
     1000);

  /* Print out the last recorded bike latitude information */

  fflush(stdin);
  printf("\nTimeStamp: ms: = %llu \n", TimeStamp);
  printf("Bike Latitude: ");
  printf(" %f", VanetParticipants[i].latitude);

  /* Print out the last recorded bike longitdue information */
  fflush(stdin);
  printf("\nBike Longitude: ");
  printf(" %f", VanetParticipants[i].longitude);

```

```
/* Print out the current OBU latitude information */
fflush(stdin);
printf("\nTruck Latitude: ");
printf(" %f", truckLat);

/* Print out the current OBU longitude information */
fflush(stdin);
printf("\nTruck Longitude: ");
printf(" %f", truckLng);

/* Print out the current OBU speed information */
fflush(stdin);
printf("\nTruck Speed:   ");
printf(" %f", wsmgps.speed);

/* Print out the recorded wheelchair speed information */
fflush(stdin);
printf("\nWheelChair Speed:   ");
printf(" %f", VanetParticipants[i].WheelChair_Speed);

/* Print out the last recorded bike speed */
fflush(stdin);
printf("\nBike Speed:    ");
printf("%f  ", VanetParticipants[index].Bike_speed);

/* Print out the last recorded E-scooter speed */
fflush(stdin);
printf("\nE-scooter Speed:    ");
printf("%f  ", VanetParticipants[index].E-scooter_speed);




/*
* While BSM message is received calculate
* the bearing and record it
*/
if (Received) {
    double b;
    b = BearingTo(VanetParticipants[i].latitude, VanetParticipants[
       i].longitude, truckLat, truckLng);
    printf("\n%f The calculated bearing: ", b);
    VanetParticipants[i].bearing = b;
  }

  /* Print out the last calculated bearing in case of BSM omission
     */
  if (Received) {

    printf("\n%f", VanetParticipants[i].bearing);

  }

  /*
  * If the number of BSM omissions meets the threashold
  * which indacates a possible jamming
  * call NewPosition function to estimate the new coordinates
  * the latitude and longitude to find the new estimated
  * distance between the two VANET nodes
  */

  if (VNTMissingBSMs[i] >= 5 && wsmgps.speed < 10) {
    printf("\nPossible Jamming Detected or TX GPS signal loss.
       Going into position estimation mode to prevent possible
       collision\n");
    printf("\nCalculating The Estimated Coordinates Position");

    uint64_t currentTimeInMillisecond = ((tv.tv_sec + (uint64_t)tv.
       tv_usec / 1000000.0) * 1000);
    int64_t millisecondTimeDiffBetweenLastMsgAndNow =
       currentTimeInMillisecond - VanetParticipants[i].time;

    printf("\n millisecondTimeDiffBetweenLastMsgAndNow =  %llu and
       currentTimeInMillisecond = %llu",
        millisecondTimeDiffBetweenLastMsgAndNow, currentTimeInMillisecond);

      NewPosition(VanetParticipants[i].latitude, VanetParticipants[i].longitude, millisecondTimeDiffBetweenLastMsgAndNow,
        VanetParticipants[i].speed*3.6, VanetParticipants[i].bearing, bike);
  }

  if (VNTMissingBSMs[i] > 5 && wsmgps.speed > 10) {
    printf("\nPossible Jamming Detected or TX GPS signal loss. Going into position estimation mode to prevent possible collision\n");
    printf("\nCalculating The Estimated Coordinates Position");
    double averageDeceleration = 0.80;
    double speedDiffToTurnSpeed = (wsmgps.speed - 10.0);

    uint64_t currentTimeInMillisecond2 = ((tv.tv_sec + (uint64_t)tv.tv_usec / 1000000.0) * 1000);//-----------------------------------Timestamp-----------------
    uint64_t millisecondTimeDiffBetweenLastMsgAndNow2 = currentTimeInMillisecond2 - VanetParticipants[i].time;

    int timeToReachTurnSpeedBike = (millisecondTimeDiffBetweenLastMsgAndNow2 + ((speedDiffToTurnSpeed / averageDeceleration) * 1000)); //how many ms to turn
    int timeToReachTurnSpeedTruck = ((speedDiffToTurnSpeed / averageDeceleration) * 1000);

    printf("\n truckLat = %f   truckLng = %f   VNTbsmCoords[i].bearing =%f", truckLat, truckLng, VanetParticipants[i].bearing);

    NewPosition(truckLat, truckLng, timeToReachTurnSpeedTruck, (10)*3.6, VanetParticipants[i].bearing, truck);
    printf("\n VNTbsmCoords[i].latitude = %f  VNTbsmCoords[i].
       longitude = %f  VNTbsmCoords[i].speed*3.6 =%f  VNTbsmCoords[
       i].bearing =%f bike=%d", VanetParticipants[i].latitude,
       VanetParticipants[i].longitude, VanetParticipants[i].speed
       *3.6, VanetParticipants[i].bearing, bike);
    printf("\n");

    NewPosition(VanetParticipants[i].latitude, VanetParticipants[i
       ].longitude, timeToReachTurnSpeedBike, (VanetParticipants[i
       ].speed*3.6)*0.75,
      VanetParticipants[i].bearing, bike);

  }

  /*
  * d is the variable used to store
  * the distance between the two VANET nodes
  */
  double d = 0;

  /*
  * If BSM is received record the bicycle
  * or a wheelchair coordinates
  *
  */

  if (Received == True) {
    newLat = VanetParticipants[i].latitude;
    newLng = VanetParticipants[i].longitude;

    /*
    * To calculate the bearing of the vehicle and
    * the wheelchair, the To_Switch_Between_Current_Old_BSM
    * variable is used, so when its value is equal to 1
    * the coordinates of the first BSM packed are recorded
    * and when its value is 2 for the consecutive BSM
    * coordinates of the second BSM packed are recorded
  */


  /* calculate the distance between the two VANET nodes a vehicle
      and (a bicycle, a wheelchair or a scooter) */
  d = DistanceTo(newLat, newLng, truckLat, truckLng);

  To_Switch_Between_Current_Old_BSM = (
      To_Switch_Between_Current_Old_BSM % 2) + 1;

  if (To_Switch_Between_Current_Old_BSM == 1) {

    /* For the WheelChair */
    VanetParticipants[i].latitude1 = VanetParticipants[i].
        latitude;
    VanetParticipants[i].longitude1 = VanetParticipants[i].
        longitude;

    /* For the vehicle */
    VanetParticipants[i].Truck_latitude1 = wsmgps.latitude;
    VanetParticipants[i].Truck_longitude1 = wsmgps.longitude;


    if ((VanetParticipants[i].latitude2 == VanetParticipants[i].
        latitude1) && (VanetParticipants[i].longitude2 =
        VanetParticipants[i].longitude1)) {


      NewPosition(VanetParticipants[i].latitude2,
          VanetParticipants[i].longitude2, 100, VanetParticipants[
          Vindex].Bike_speed * 3.6, b, bike);

      VanetParticipants[i].latitude2 = newLat;
      VanetParticipants[i].longitude2 = newLng;
    }


```

```
      if ((VanetParticipants[i].Truck_latitude1 ==
         VanetParticipants[i].Truck_latitude2) && (
         VanetParticipants[i].Truck_longitude1 == VanetParticipants
         [i].Truck_longitude2)) {

        NewPosition(VanetParticipants[i].Truck_latitude2,
            VanetParticipants[i].Truck_longitude2, 100, wsmgps.speed
             * 3.6, b, truck);

        VanetParticipants[i].Truck_latitude2 = truckLat;
        VanetParticipants[i].Truck_longitude2 = truckLng;

      }

      if (((VanetParticipants[i].latitude2 == VanetParticipants[i].
         latitude1) && (VanetParticipants[i].longitude2 =
         VanetParticipants[i].longitude1)) || (VanetParticipants[i
         ].Truck_latitude1 == VanetParticipants[i].Truck_latitude2)
          && (VanetParticipants[i].Truck_longitude1 ==
         VanetParticipants[i].Truck_longitude2)) {

        d = DistanceTo(newLat, newLng, truckLat, truckLng);

        printf("\n\n New Estimated Distance Between the Truck and
            the WheelChair Position");
        fflush(stdin);
        printf("\n%f", d);

      }

      /* Find the bearing for the wheelchair */

      VanetParticipants[i].bearing_Of_WheelChair = BearingTo(
         VanetParticipants[i].latitude1, VanetParticipants[i].
         longitude1, VanetParticipants[i].latitude2,
         VanetParticipants[i].longitude2);
      printf("\n%f bearing_Of_WheelChair", VanetParticipants[i].
         bearing_Of_WheelChair);

      /* Find the bearing for the vehicle */
      VanetParticipants[i].bearing_Of_Truck = BearingTo(
         VanetParticipants[i].Truck_latitude1, VanetParticipants[i
         ].Truck_longitude1, VanetParticipants[i].Truck_latitude2,
         VanetParticipants[i].Truck_longitude2);
      printf("\n%f bearing_Of_Truck", VanetParticipants[i].
         bearing_Of_Truck);
      printf("\n");

      /* 30 is the radius of detection to eliminate false positive
         for the saftey application */

      if (d <= 30) {

        if ((VanetParticipants[i].bearing_Of_WheelChair >= 80 &&
            VanetParticipants[i].bearing_Of_WheelChair <= 100) &&
          (VanetParticipants[i].bearing_Of_Truck >= 350 ||
             VanetParticipants[i].bearing_Of_Truck <= 10)) {

          printf("\n\n VEHICLE DIRECTION TO THE NORTH DANGER =>
             WATHC POSSIBLE INTERSECTION WITH A WHEELCHAIR MOVING
             TOWARD THE EAST DIRECTION! \n\n");

        }


        if ((VanetParticipants[i].bearing_Of_WheelChair >= 350 ||
            VanetParticipants[i].bearing_Of_WheelChair <= 10) &&
          (VanetParticipants[i].bearing_Of_Truck >= 80 &&
             VanetParticipants[i].bearing_Of_Truck <= 100)) {

          printf("\n\n VEHICLE DIRECTION TO THE EAST DANGER =>
             WATHC POSSIBLE INTERSECTION WITH A WHEELCHAIR MOVING
             TOWARD THE NORTH DIRECTION! \n\n");

        }


        if ((VanetParticipants[i].bearing_Of_WheelChair <= 280 &&
            VanetParticipants[i].bearing_Of_WheelChair >= 260) &&
          (VanetParticipants[i].bearing_Of_Truck >= 350 ||
              VanetParticipants[i].bearing_Of_Truck <= 10)) {

          printf("\n\n VEHICLE DIRECTION TO THE NORTH DANGER =>
              WATHC POSSIBLE INTERSECTION WITH A WHEELCHAIR MOVING
              TOWARD THE WEST DIRECTION! \n\n");

        }

        if ((VanetParticipants[i].bearing_Of_WheelChair >= 170 &&
            VanetParticipants[i].bearing_Of_WheelChair <= 190) &&
          (VanetParticipants[i].bearing_Of_Truck >= 80 &&
              VanetParticipants[i].bearing_Of_Truck <= 100) ||
          (VanetParticipants[i].bearing_Of_Truck <= 280 &&
              VanetParticipants[i].bearing_Of_Truck >= 260)) {

          printf("\n\n DANGER => WATHC POSSIBLE INTERSECTION WITH A
               WHEELCHAIR MOVING TOWARD THE SOUTH DIRECTION! \n\n");

        }

        // Works for turning right or left

        if ((VanetParticipants[i].bearing_Of_WheelChair >= 350 ||
            VanetParticipants[i].bearing_Of_WheelChair <= 10) &&
          (VanetParticipants[i].bearing_Of_Truck >= 350 ||
              VanetParticipants[i].bearing_Of_Truck <= 10)) {

          printf("\n\n VEHICLE DIRECTION TO THE NORTH DANGER =>
              WATCH CAREFULLY FOR THE WHEELCHAIR WHILE TURNING! \n\n
              ");

        }

        // Works for turning right or left

        if ((VanetParticipants[i].bearing_Of_WheelChair >= 170 &&
            VanetParticipants[i].bearing_Of_WheelChair <= 190) &&
          (VanetParticipants[i].bearing_Of_Truck >= 170 &&
              VanetParticipants[i].bearing_Of_Truck <= 190)) {

          printf("\n\n VEHICLE DIRECTION TO THE SOUTH DANGER =>
              WATCH CAREFULLY FOR THE WHEELCHAIR WHILE TURNING! \n\n
              ");

        }


      }

    }


    if (To_Switch_Between_Current_Old_BSM == 2) {

      /* For the vehicle */

      VanetParticipants[i].Truck_latitude2 = wsmgps.latitude;
      VanetParticipants[i].Truck_longitude2 = wsmgps.longitude;

      /* For the WheelChair */

      VanetParticipants[i].latitude2 = VanetParticipants[i].
          latitude;
      VanetParticipants[i].longitude2 = VanetParticipants[i].
          longitude;

      if ((VanetParticipants[i].latitude2 == VanetParticipants[i].
          latitude1) && (VanetParticipants[i].longitude2 =
          VanetParticipants[i].longitude1)) {

        NewPosition(VanetParticipants[i].latitude2,
            VanetParticipants[i].longitude2, 100, VanetParticipants[
            Vindex].Bike_speed * 3.6, b, bike);

        VanetParticipants[i].latitude1 = newLat;
        VanetParticipants[i].longitude1 = newLng;

      }

      if ((VanetParticipants[i].Truck_latitude1 ==
          VanetParticipants[i].Truck_latitude2) && (
          VanetParticipants[i].Truck_longitude1 == VanetParticipants
          [i].Truck_longitude2)) {

        NewPosition(VanetParticipants[i].Truck_latitude2,
            VanetParticipants[i].Truck_longitude2, 100, wsmgps.speed
             * 3.6, b, truck);

        VanetParticipants[i].Truck_latitude1 = truckLat;
        VanetParticipants[i].Truck_longitude1 = truckLng;
      }


      if (((VanetParticipants[i].latitude2 == VanetParticipants[i].
          latitude1) && (VanetParticipants[i].longitude2 =
          VanetParticipants[i].longitude1)) || (VanetParticipants[i
          ].Truck_latitude1 == VanetParticipants[i].Truck_latitude2)
           && (VanetParticipants[i].Truck_longitude1 ==
          VanetParticipants[i].Truck_longitude2)) {

        d = DistanceTo(newLat, newLng, truckLat, truckLng);

```

```
            printf("\n\n New Estimated Distance Between the Truck and
                the WheelChair Position");
            fflush(stdin);
            printf("\n%f", d);

        }



        /* Find the bearing for the wheelchair */

        VanetParticipants[i].bearing_Of_WheelChair = BearingTo(
            VanetParticipants[i].latitude2, VanetParticipants[i].
            longitude2, VanetParticipants[i].latitude1,
            VanetParticipants[i].longitude1);
        printf("\n%f bearing_Of_WheelChair", VanetParticipants[i].
            bearing_Of_WheelChair);

        /* Find the bearing for the vehicle */

        VanetParticipants[i].bearing_Of_Truck = BearingTo(
            VanetParticipants[i].Truck_latitude2, VanetParticipants[i
            ].Truck_longitude2, VanetParticipants[i].Truck_latitude1,
            VanetParticipants[i].Truck_longitude1);
        printf("\n%f bearing_Of_Truck", VanetParticipants[i].
            bearing_Of_Truck);
        printf("\n");

        /* 30 is the radius of detection to eliminate false positive
            for the saftey application */

        if (d <= 30) {

            if ((VanetParticipants[i].bearing_Of_WheelChair >= 80 &&
                VanetParticipants[i].bearing_Of_WheelChair <= 100) &&
              (VanetParticipants[i].bearing_Of_Truck >= 350 &&
                VanetParticipants[i].bearing_Of_Truck <= 10)) {
```

```

      printf("\n\n VEHICLE DIRECTION TO THE NORTH DANGER =>
         WATHC POSSIBLE INTERSECTION WITH A WHEELCHAIR MOVING
         TOWARD THE EAST DIRECTION! \n\n");

    }

    if ((VanetParticipants[i].bearing_Of_WheelChair <= 280 &&
        VanetParticipants[i].bearing_Of_WheelChair >= 260) &&
      (VanetParticipants[i].bearing_Of_Truck >= 350 ||
         VanetParticipants[i].bearing_Of_Truck <= 10)) {

      printf("\n\n VEHICLE DIRECTION TO THE NORTH DANGER =>
         WATHC POSSIBLE INTERSECTION WITH A WHEELCHAIR MOVING
         TOWARD THE WEST DIRECTION! \n\n");

    }

    if ((VanetParticipants[i].bearing_Of_WheelChair >= 170 &&
        VanetParticipants[i].bearing_Of_WheelChair <= 190) &&
      (VanetParticipants[i].bearing_Of_Truck >= 80 &&
         VanetParticipants[i].bearing_Of_Truck <= 100) ||
      (VanetParticipants[i].bearing_Of_Truck <= 280 &&
         VanetParticipants[i].bearing_Of_Truck >= 260)) {

      printf("\n\n DANGER => WATHC POSSIBLE INTERSECTION WITH A
          WHEELCHAIR MOVING TOWARD THE SOUTH DIRECTION! \n\n");

    }

    // Works for turning right or left
    if ((VanetParticipants[i].bearing_Of_WheelChair >= 350 &&
        VanetParticipants[i].bearing_Of_WheelChair <= 10) &&
      (VanetParticipants[i].bearing_Of_Truck >= 350 ||
         VanetParticipants[i].bearing_Of_Truck <= 10)) {

          printf("\n\n VEHICLE DIRECTION TO THE NORTH DANGER =>
             WATCH CAREFULLY FOR THE WHEELCHAIR WHILE TURNING! \n\n
             ");

        }


        if ((VanetParticipants[i].bearing_Of_WheelChair >= 350 ||
            VanetParticipants[i].bearing_Of_WheelChair <= 10) &&
          (VanetParticipants[i].bearing_Of_Truck >= 80 &&
             VanetParticipants[i].bearing_Of_Truck <= 100)) {

          printf("\n\n VEHICLE DIRECTION TO THE EAST DANGER =>
             WATHC POSSIBLE INTERSECTION WITH A WHEELCHAIR MOVING
             TOWARD THE NORTH DIRECTION! \n\n");

        }

        // Works for turning right or left

        if ((VanetParticipants[i].bearing_Of_WheelChair >= 170 &&
            VanetParticipants[i].bearing_Of_WheelChair <= 190) &&
          (VanetParticipants[i].bearing_Of_Truck >= 170 &&
             VanetParticipants[i].bearing_Of_Truck <= 190)) {

          printf("\n\n VEHICLE DIRECTION TO THE SOUTH DANGER =>
             WATCH CAREFULLY FOR THE WHEELCHAIR WHILE TURNING! \n\n
             ");

        }


      }
    }


  }
```

```

/* Print out the latitudes and logitudes of the vehicle and  (a
    bicycle or a wheelchair) */

printf("\n newLat = %f  newLng = %f truckLat = %f  truckLng =%f
    ", newLat, newLng, truckLat, truckLng);
  printf("\n\n Estimated Distance Between the Truck and the
      WheelChair Position");
fflush(stdin);
printf("\n%f", d);

/*
*  The MinimumStoppingDistance function takes two arguments
*  the speed of the bicycle in kilometre and the coefficient
    friction
*  for the dry condition road for a bicycle
*  the minStopDist is used in case of a fixed distance between
    the
*  OBUs that are mounted on the same vehicle
*
*/

double  minStopDist = MinimumStoppingDistance(wsmgps.speed * 3.6,
     0.32);

/*
*  The MinimumStoppingDistance function takes two arguments
*  the speed of the bicycle in kilometre and the coefficient
    friction
*  for the dry condition road for a bicycle
*  the minStopDist2 is used for normal case where the distance
    between the
*  OBUs are not fixed
*
*/
double  minStopDist2 = MinimumStoppingDistance(VanetParticipants[
    i].Bike_speed * 3.6, 0.32);
```

```


/*
* MinimumStoppingDistance_Vehicle function takes two arguments
*  the speed of the vehicle in kilometre and the the coefficient
   friction
*  for the dry condition road for a vehicle
*/

double  minStopDist3 = MinimumStoppingDistance_Vehicle(wsmgps.
   speed * 3.6, 0.7);

minStopDist = minStopDist * 1.1;

minStopDist2 = minStopDist2 * 1.1;

printf("\n\n Min Stopping Distance: ");

printf("\n%f", minStopDist);

/*---Used for fixed distance-----*/
/*---When the two OUBs are mounted on the same vehicle*/

if (minStopDist < d) {

  printf("\n\n Safe to turn! \n\n");
}
else {

  printf("\n\n DANGER => DO NOT TURN! \n\n");
}


printf("\n\n Not-fixed Min Stopping Distance: ");

printf("\n%f", minStopDist2);

```

```
  /*--------Normal case not fixed distance------*/

  if (minStopDist2 < d) {

    printf("\n\n Not-fixed Safe to turn! \n\n");
  }
  else {

    printf("\n\n Not-fixed DANGER => DO NOT TURN! \n\n");
  }

  printf("\n\n Vehicle Min Stopping Distance: ");

  printf("\n%f", minStopDist3);

  /*--------MSD for the Vehicle-----*/

  if (minStopDist3 < d) {

    printf("\n\n Vehicle Safe to turn! \n\n");
  }
  else {

    printf("\n\n Vehicle DANGER => DO NOT TURN! \n\n");
  }

}
/////////////////////////////////////////////

double pi = 3.14159265359;
double EarthRadiusInKilometers = 6378.0;

/*
* the coordinates are given by the OBUs in degree
* so  coordinates need to be converted to radians
*/

```

```
double DegreeToRadian(double angle) { return pi * angle / 180.0; }
double RadianToDegree(double angle) { return 180.0 * angle / pi; }

/*
* The function receives the longitudes and latitudes
* of the vehicle, the bicycle and the wheelchair to find
* the bearing
*/

double BearingTo(double lat, double lng, double latitude, double
    longitude)
{

  double lat1 = DegreeToRadian(latitude);
  double lat2 = DegreeToRadian(lat);
  double dLon = DegreeToRadian(lng) - DegreeToRadian(longitude);

  double y = sin(dLon) * cos(lat2);
  double x = cos(lat1) * sin(lat2) - sin(lat1) * cos(lat2) * cos(
      dLon);
  double brng = atan2(y, x);

  return fmod((RadianToDegree(brng) + 360.0), 360.0);

} /* end BearingTo */


  /*
  *  The function receives the known longitude, latitude
  *  time, velocity, bearing and the type of the vehicle
  *  to calculate the new estimated longitude and latitude.
  */

void NewPosition(double lat, double lng, int timeMs, double
    velocityKm, double bearingDegrees, int type) {

  double bearingRad = (bearingDegrees * pi) / 180;
  double kmDistance = velocityKm * ((timeMs / 1000.0) / 3600.0);
  double distRatio = (kmDistance) / (EarthRadiusInKilometers);
  double distRatioSine = sin(distRatio);
  double distRatioCosine = cos(distRatio);
  double startLatRad = DegreeToRadian(lat);
  double startLonRad = DegreeToRadian(lng);
  double startLatCos = cos(startLatRad);
  double startLatSin = sin(startLatRad);
  double endLatRads = asin((startLatSin * distRatioCosine) + (
      startLatCos * distRatioSine * cos(bearingRad)));
  double endLonRads = startLonRad + atan2(sin(bearingRad) *
      distRatioSine * startLatCos, distRatioCosine - startLatSin *
      sin(endLatRads));

  if (type == 1) {

    truckLat = RadianToDegree(endLatRads);
    truckLng = RadianToDegree(endLonRads);
    fflush(stdin);
    printf("\n truckLat = %f truckLng = %f ", truckLat, truckLng);
    fflush(stdin);

  }

  else {
    newLat = RadianToDegree(endLatRads);
    newLng = RadianToDegree(endLonRads);
    fflush(stdin);
    printf("\n newLat = %f  newLng = %f", newLat, newLng);
    fflush(stdin);
  }

}

/*
* The function receives the longitutes and latitudes
* given by the On-board-unites of vehicles.
* These VANET participants can be a bicycle, a motorized vehicle,
* a wheelchair or E-scooter.
* The received information is passed to be calculated to find the
    distance
* between the two unites in meter
*/

double DistanceTo(double lat, double lng, double  latitude, double
    longitude)
{
  double R = EarthRadiusInKilometers * 1000;
  double dLat = DegreeToRadian(lat) - DegreeToRadian(latitude);
  double dLon = DegreeToRadian(lng) - DegreeToRadian(longitude);
  double a = sin(dLat / 2) * sin(dLat / 2) + cos(DegreeToRadian(
      latitude)) * cos(DegreeToRadian(lat)) * sin(dLon / 2) * sin(
      dLon / 2);
  double c = 2 * atan2(sqrt(a), sqrt(1 - a));
  double distance = c * R;
  return distance;

}

/*
*  The minimum stopping distance function for a bicycle.
*  This function takes two arguments,
*  the speed of the bicycle in kilometre and the coefficient
    friction
*  for the dry condition road for a bicycle
*/

double MinimumStoppingDistance(double velocityKm, double
    frictionCooeficient) {

  /*
  *  Here in the commented section, the minimum stopping distance
      is modified with
  *  to include a reaction time used for some experements
  *  to find out if there is any differeces between the
  *  standard minimum stopping distance used in this section.
  *  return (((pow(velocityKm, 2) / (254.0*(frictionCooeficient)))
  *  + (velocityKm / 1.4)) + (velocityKm * (2.5 / 3600)));
  *
  */

  /* Minimum stopping distance */

  return (((pow(velocityKm, 2) / (254.0*(frictionCooeficient))) + (
     velocityKm / 1.4)));
}

/*
*  The motorized vehicle mininumum stopping
*  distance function.
*  The function receives two arguments
*  the speed of the vehicle in kilometre
*  and the the coefficient friction
*  for the dry condition road for a vehicle
*/

double MinimumStoppingDistance_Vehicle(double velocityKm, double
    frictionCooeficient) {

  int G = 0; /* assumed flat road for G */
  double t_react = 1.66;
  return ((0.278 * t_react * velocityKm) + pow(velocityKm, 2) /
     (254 * (frictionCooeficient + G)));
}
```